

O‘ZBEKISTON RESPUBLIKASI
OLIY TA’LIM, FAN VA INNOVATSIYALAR
VAZIRLIGI

«SUN’IY INTELLEKT ASOSLARI»

(darslik)

60610700- “Sun’iy intellekt”, 60611000-“Telekommunikatsiya texnologiyalari: Telekommunikatsiyalar” bakalavriat, 70710903- “Intellektual muhandislik tizimlari” magistraturra ta’lim yo‘nalishi talabalari uchun darslik sifatida tavsiya etiladi

UO‘K: 004.8

KBK: 32.813

Abdumanonov, Quchqarov

A13, Q95 Sun’iy intellekt asoslari / Darslik / A.A.Abdumanonov, A.A.Quchqarov – Farg‘ona: “Sunrise-pro” nashriyoti, 2025. – 344 b.

Darslik 60610700 - “Sun’iy intellekt”, 60611000 - “Telekommunikatsiya texnologiyalari: Telekommunikatsiyalar” bakalavriat yo‘nalishi, 70710903-“Intellectual muhandislik tizimlari” magistratura ta’lim yo‘nalishida tahsil olayotgan talabalarga o‘zining ta’lim sohasi bo‘yicha talab etilgan ma’lumotlarni o‘z ichiga olgan. Darslik su’niy intellekt asoslari bo‘yicha umumiy tushunchalar, fan rivojlanishining asosiy ta’riflari va asosiy yo‘nalishlari, su’niy intellekt tizimlari va ularni loyihalash usular, qo‘llaniladigan dasturlash tillari va qo‘shimcha kutubxonalar, vositalari yordamida intellektual tizimlar yaratish bo‘yicha misollari keltirilgan. Darslikda mavjud boblar va paragraflar sun’iy intellekt asoslari, bilim va uni modelari, bilimni hosil qilishda o‘qituvchili va o‘qituvchisiz o‘rgatish algoritmlari, mantiqiy dasturlash, sun’iy neyron tarmoqlar va ularga o‘qitish algoritmlari va ularni dasturlashda qo‘llash Python dasturlash tili misolida amaliy ko‘nikmalarga ega bo‘lish, ular asosida mustaqil ta’limni tashkil etish hamda sohaga oid muammolarga optimal yechimlar ishlab chiqish bo‘yicha amaliy misollar keltirilgan.

Darslik majburiy fan asosida o‘qitilayotgan «Sun’iy intellekt asoslari» fanini o‘rganishda 60610700 - “Sun’iy intellekt”, 60611000-Telekommunikatsiya texnologiyalari: Telekommunikatsiyalar bakalavriat, 70710903-Intellectual muhandislik tizimlari magistraturra ta’lim yo‘nalishi talabalari uchun asosiy adabiyot sifatida tavsiya etiladi.

Mas’ul

muharrir: **A.A.Quchqarov** – “Elektronika va asbobsozlik” kafedراسи dotsenti

Taqrizchilar:

L.K.Mamadaliyeva – “Intellectual muhandislik tizimlari” kafedراسи professori

D.A.Xalilov – Toshkent axborot-texnologiyalari universiteti Farg‘ona filiali “Axborot texnologiyalari” kafedراسи professori

*O‘zbekiston Respublikasi Oliy ta’lim va innovatsiyalar vazirligining 2025 yil 09 iyuldagi
258-sonli buyrug‘iga asosan nashr etishga ruxsat berilgan
(Ruxsatnoma №902988)*

978-9910-574-66-5

© **A.A.Abdumanonov, A.A.Quchqarov, 2025**

© **"Sunrise-pro" nashriyoti, 2025**

MUNDARIJA

KIRISH.....	9
I-BOB. SUN'IY INTELLEKT ASOSLARI.....	10
1.1-§. SUN'IY INTELLEKT	10
SI tizimlarining rivojlanish tarixi.....	16
Ekspert tizimlari	26
Sun'iy intellektni rivojlantirish yo'nalishlari.....	29
Sun'iy intellektni o'rganish zarurati	32
Aql-idrok nima?	33
Aql-idrok nimadan iborat?	34
Ta'lim olish (o'rganish)	35
Bilimlarni taqdim etish modellari	36
Sun'iy intellekt bilan nima bog'liq?.....	47
Sun'iy intellektni qo'llash.....	48
Agent va muhit.....	49
O'z-o'zini nazorat qilish uchun savollar.....	50
1.2-§. PYTHON YORDAMIDA SUN'IY INTELLEKT – BOSHLANG'ICH KO'NIKMALAR.....	52
Nega aynan Python sun'iy intellekt uchun?.....	52
Python ishga tushirish	56
Integratsiyalashgan ishlab chiqish muhiti.....	57
NumPy kutubxonasi, ko'p o'lchovli massiv	58
Grafiklar qurish uchun Matplotlib kutubxonasida foydalanish	72
Matplotlib kutubxonasini o'rnatish va foydalanish	74
Pandas kutubxonasi.....	81
Pandasni o'rnatish, import qilish av ishlash	83
O'z-o'zini nazorat qilish uchun savollar.....	88
II-BOB. SUN'IY INTELLEKTADA O'RGATISH ALGORITIMLARI.	90
2.1-§. MASHINALI O'RGANISH - MACHINE LEARNING.....	90
Mashinali o'rganish turlari (ML)	90
O'qituvchisiz mashinani o'rganish algoritmlari	91
Mustaxkamlash bilan mashinani o'rganish algoritmlari.....	92
Mashinani o'rganishning eng keng tarqalgan algoritmlari	92
Sodda Bayes klassifikatori.	94

Kerakli kutubxonalarni import qilish	101
Modellarni o'qitish va natijalarni baholash	102
Sodda Bayes klassifikatorining afzalliklari va kamchiliklari	103
O'z-o'zini nazorat qilish uchun savollar	105
2.2-§. MA'LUMOTLAR TAYYORLASH.....	106
Ma'lumotlarni oldindan qayta ishlash bosqichlari.....	106
Ma'lumotlarni oldindan qayta ishlash usullari.....	107
O'z-o'zini nazorat qilish uchun savollar	113
2.3-§. KO'RSATMA BO'YISH O'RGATISH: KLASIFIKATSIYA.....	114
Python-da klassifikator yaratish bosqichlar	114
Tayanch vektorlar mashinasi (SVM)	118
Logistik regressiya	121
Qaror daraxti klassifikatori.....	123
Tasodifiy o'rmon klassifikatori.....	126
Klassifikatorning ishlashi.....	127
Chalkashlik matritsasi	127
Sinf nomutanosibligi muammosi	129
Ansambl usullari	131
O'z-o'zini nazorat qilish uchun savollar	132
2.4-§. KO'RSATMA BO'YISH O'RGANISH: REGRESSIYA	133
Python da regressorlarni yaratish	133
Ko'p o'zgaruvchan regressor	136
O'z-o'zini nazorat qilish uchun savollar	138
2.5-§. MANTIQUIY DASTURLASH.....	139
Mantiqiy dasturlash yordamida qanday muammolarni hal qilish mumkin	139
Foydali paketlarni o'rnatish	140
Mantiqiy dasturlashga misollari	141
Odiy sonlarni tekshirish	142
O'z-o'zini nazorat qilish uchun savollar	145
2.6-§. NAZORATSIZ O'RGANISH: KLASSTERLASH	146
Klasterlash nima?	146
Ma'lumotlarni Klasterlash Algoritmnlari	146
Klasterlash samaradorligini o'lchash	152
Siluet ballini hisoblash	152

Eng yaqin qo'shnilarini qidirish.....	154
K-eng yaqin qo'shni tasniflagichi	157
O'z-o'zini nazorat qilish uchun savollar	160
2.7-§. TABIIY TILNI QAYTA ISHLASH.....	161
NLP komponentlari	161
NLU ni qiyinchiliklar	162
NLP terminologiyasi	162
NLP bosqichlari	163
NLTK paketi	164
NLTK-ni import qilish	164
NLTK ma'lumotlari yuklash.....	165
Boshqa kerakli paketlarni o'rnatish	165
Chanking: ma'lumotlarni bo'laklarga bo'lish.....	169
Parchalanish turlari.....	169
“So'zlar sumkasi” (BoW) modeli	171
Statistika tushunchasi	172
NLTKda so'zlar modelini yaratish	173
Muammoni hal qilish	174
Mavzuni modellashtirish: Matn ma'lumotlaridagi qonuniylikni aniqlash	178
O'z-o'zini nazorat qilish uchun savollar.....	180
2.8-§. VAQTLAR QATORI MA'LUMOTLARI TAHLIL QILISH.....	181
Foydali paketlarni o'rnatish	181
Pandas: Vaqt qatori ma'lumotlaridan statistik ma'lumotlarni qayta ishlash, kesish va olish	183
Vaqt qatorlari ma'lumotlaridan statistik ma'lumotlarni olish	185
Yashirin Markov modeli (HMM) yordamida ketma-ket ma'lumotlarni tahlil qilish	188
O'z-o'zini nazorat qilish uchun savollar.....	192
2.9-§. NUTQNI TANISH (SPEECH RECOGNITION).....	193
Nutqni tan tizimini yaratish.....	193
Ovozli signallarni vizualizatsiya qilish - fayldan o'qish va u bilan ishlash	195
Ovoz signalining xarakteristikasi: chastota domenini o'zgartirish.....	197
Monoton ovoz signalini yarating	199
Nutqdan funktsiyalarni ajratib olish.....	201
Og'zaki so'zni aniqlash.....	203

O‘z-o‘zini nazorat qilish uchun savollar	204
2.10-§. EVRISTIK QIDIRISH	205
Sun’iy intellektda evristik qidiruv tushunchasi.....	205
Malumotsiz va axborotli qidiruv o‘rtasidagi farq	205
Sehrli kvadrat	207
O‘z-o‘zini nazorat qilish uchun savollar	209
2.11-§. SUN’IY INTELLEKT BILAN PYTHON YORDAMIDA O‘YINLAR	210
Qidiruv algoritmlari.....	210
Kombinatsiyalangan qidiruv	210
Alfa-beta kesish	211
Negamax algoritmi	212
O‘yinlar uchun botlarni yaratish	212
Krestik va nollar o‘yni uchun bot.....	214
O‘z-o‘zini nazorat qilish uchun savollar	217
III-BOB. SUN’IY INTELLEKTDAN NEYRON TARMOQLAR.....	218
3.1-§. SUN’IY NEYRON TARMOQLAR.....	218
Sun’iy neyron tarmoqlar (Artificial Neural Networks - ANN) nima?.....	218
Ob’ektlar va ularning xususiyatlari	218
O‘qituvchi bilan o‘qitish	220
Neyron	221
Neyronlar tarmog‘i	222
Neyron giperplan sifatida	224
Neyron ishonchliligi	226
Neyronning foydaliligi	229
Bir nechta chiqish.....	230
Yashirin qatlam qachon kerak?	231
Neyronlar mantiqiy element sifatida.....	233
XOR to‘rtta varianti	235
$y=f(x)$ funksiyaning yaqinlashishi	235
Noravshan mantiq	236
Foydali paketlarni o‘rnatish	237
Neyron tarmoqlarni qurish	238
Perseptron asosidagi klassifikator	238
Bir qatlamli neyron tarmoqlari.....	239

Ko‘p qatlamli neyron tarmoqlar.....	242
Kohonen neyron tarmog‘i.	244
Kohonen neyron tarmog‘ining tuzilishi.	245
Kohonen neyron tarmog‘ini o‘rgatish.....	247
Kohonen xaritalari.....	250
Xopfild va Xemming neyron tarmoqlari.....	255
Xopfild neyron tarmog‘ini amalga oshirish.....	264
O‘z-o‘zini nazorat qilish uchun savollar.....	270
3.2-§. MUSTAXKAMLASH YORDAMIDA O‘RGANISH	271
Mustaxkamlash yordamida o‘rgatish asoslar.....	271
Qurilish bloklari: atrof-muhit va agent	271
Python yordamida muhit yaratish	273
Python yordamida o‘quv agentini yaratish	274
O‘z-o‘zini nazorat qilish uchun savollar.....	275
3.3-§. GENETIK ALGORITMLAR.....	276
Genetik algoritmlar nima?.....	276
Genetik algoritmlar yordamida yechimlarni amalga oshirish.....	277
O‘z-o‘zini nazorat qilish uchun savollar.....	284
3.4-§. KOMPYUTERLI KO‘RISHI.....	285
Kompyuter ko‘rishining asosiy vazifalari.....	287
Kompyuter ko‘rish texnologiyalari.....	287
Kompyuter ko‘rish modelini qo‘shimcha o‘qitish.....	291
Kompyuter ko‘rishi va tasvirlarni qayta ishlash	292
OpenCV nima.....	294
Kompyuter ko‘rish va OpenCV qaerda qo‘llaniladi.....	295
Rasmlarni o‘qish, yozish va ko‘rsatish	298
Rang maydonini aylantirish	299
Chetni aniqlash.....	301
Yuzni tanish.....	302
Ko‘zni aniqlash	304
Kompyuter ko‘rinishi uchun MEDIAPIPE.....	305
YOLO nima?	311
YOLO qanday o‘rnatiladi?.....	315
O‘z-o‘zini nazorat qilish uchun savollar.....	320

3.5-§. SUN'IY INTELLEKTDADA CHUQUR O'QITISH.....	321
Chuqur o'rganish (deep learning)	321
Chuqur o'qitish inqilobi	321
Chuqur o'rganish algoritmlari.....	323
Deep Learning qayerda qo'llaniladi?	324
Chuqur o'rganishsiz mashinani o'rganish.....	325
Konvolyutsion neyron tarmog'i (Convolutional Neural Network - CNN).....	326
Kerakli Python paketlarini o'rnatish	328
ANN yordamida chiziqli regressorni qurish	328
O'z-o'zini nazorat qilish uchun savollar	333
GLOSSARIY.....	334
ADABIYOTLAR	339

KIRISH

Sun'iy intellekt (SI) — bu mashinalarga insonlarning aqliy faoliyatlarini amalga oshirish imkoniyatini beruvchi ilmiy soha va texnologiya yo'nalishidir. Bu fan, kompyuterlar va tizimlarga o'rganish, tahlil qilish, qarorlar qabul qilish, mantiqiy fikrlash va hatto hissiy reaksiyalarni anglash kabi inson faoliyatiga xos ko'nikmalarni o'rgatish bilan shug'ullanadi. Sun'iy intellekt bugungi kunda butun dunyo bo'ylab texnologik rivojlanishning markaziy elementlaridan biri sifatida qaraladi va uning qo'llanilishi kundan-kunga kengayib bormoqda.

Darslikda, sun'iy intellekt asoslari, uning asosiy tamoyillari, metodlari, dasturlash yondoshuvlarining mohiyati, shuningdek, SI tizimlarining yaratish jarayonlari batafsil ko'rib chiqiladi. Ustuvor yo'nalishlardan biri sifatida mashinada o'rganish, tabiiy tilni qayta ishlash (NLP), tasvirni tanish va boshqa ko'plab zamonaviy texnologiyalar yoritiladi.

Sun'iy intellektning rivojlanishi bugungi kunda shaxsiy yordamchilar, avtonom transport vositalari, sog'liqni saqlash sohasidagi diagnostika tizimlari, biznesda avtomatlashtirilgan qarorlar qabul qilish va ko'plab boshqa sohalarda sezilarli o'zgarishlarga sabab bo'lmoqda. Shu bilan birga, SI texnologiyalari bilan bog'liq etik, xavfsizlik va iqtisodiy masalalar ham ahamiyat kasb etadi.

Ushbu darslik, sun'iy intellektning nazariy asoslarini, uning dasturlash yondoshuvlarini va amaliy qo'llanilishini chuqur o'rganishga yordam beradi. Darslik talabalarga, o'quvchilarga, va mutaxassislarga ushbu murakkab soha haqida kengaytirilgan bilimlarni taqdim etishga qaratilgan bo'lib, sun'iy intellektni o'rganishda ko'makchi vosita bo'lishni maqsad qiladi.

Hozirgi kunda sun'iy intellekt ilmiy tadqiqotlar, texnologik yangiliklar va biznes sohasidagi muvaffaqiyatlarning asosiy omiliga aylanishi bilan birga, kelajakda bu soha yanada kengayib, yangi imkoniyatlar yaratishga qaratilgan innovatsion yechimlar ishlab chiqilishiga olib keladi. Darslikning asosiy maqsadi — talabalarni sun'iy intellekt texnologiyalarining zamonaviy yo'nalishlari, ularning hayotimizdagi o'rni va kelajakdagi imkoniyatlari haqida chuqur tushunchaga ega bo'lishini ta'minlashdir.

I-BOB. SUN'IY INTELLEKT ASOSLARI.

1.1-§. SUN'IY INTELLEKT

Kompyuterlar ixtiro qilinganidan beri, ularning turli vazifalarni bajarish qobiliyati geometrik progressiyada o'sib bormoqda. Insonlar vaqt o'tishi bilan kompyuter tizimlarining ish sohalarini kengaytirish, tezligini oshirish va o'lchamlarini kichraytirish borasida ulkan rivojga erishdilar.

Informatikaning bir sohasi bo'lgan sun'iy intellekt esa, inson kabi aqlli kompyuterlar yoki mashinalar yaratishni maqsad qiladi. Sun'iy intellektning asosiy tushunchasi

Sun'iy intellekt (ong, aqil) - bu insonning intellektual faoliyatini modellashtirish bilan shug'ullanadigan fan sohasidir. 700 yildan ko'proq vaqt oldin o'rta asrlarda Ispaniyada paydo bo'lgan sun'iy intellekt XX asr o'rtalarida mustaqil fan sohaga aylandi.

Sun'iy intellect (SI) usullari muhandislik, tibbiyot, biologiya, zoologiya, sotsiologiya, madaniyatshunoslik, siyosatshunoslik, iqtisodiyot, biznes, sud ekspertizasi va boshqa inson faoliyatining turli sohalarida oldin farmalizatsiyalash va algoritmlashtirish inkoni yo'q deb hisoblangan masallarni echish uchun xilma-xil kompyuter dasturlarini yaratishga imkon berdi. Kompyuter dasturlarini o'qitish va o'z-o'zini o'rganish g'oyalari, bilimlarni to'plash, noaniq va mavhum bilim qayta ishlash orqali mo'ljizalar yarata oladiga dasturlarni yaratishga imkon berdi. Kompyuterlar shaxmat bo'yicha jahon chempioni unvoni uchun muvaffaqiyatli kurashmoqda, musiqiy va she'riy asarlar yaratish, tasvirlar va tasvirlarni tanib olish, nutqni - tabiiy inson tilidagi matnlarni tanib olish, tushunish va qayta ishlash orqali insonning ijodiy faoliyatini modellashtirmoqda. Inson miyasi foliyat va unga o'xshash usulda yaratilgan neyrokompyuterlar murakkab texnik ob'ektlarni boshqarish, inson kasalliklarini tashxislash, murakkab texnik qurilmalarning nosozliklarini muvaffaqiyatli xal qilishda; ob-havo va valyuta kurslarini, ovoz berish natijalarini bashorat qilish, xakerlar va potentsial bankrotlarni aniqlash; abituriyentlarga to'g'ri mutaxassislikni tanlashda yordam berish va boshqa bir qator masalalrni muvofaqiyatli echmoqda.

Biz allaqachon kompyuterlarning ko‘z o‘ngimizda “aqili” bo‘lib borayotganiga o‘rganib qolganmiz va kompyuter dasturlari tobora intellektuallashib bormoqda. Aql-idrok tushunchasining o‘zi ilm-fan va inson rivojlanishi bilan doimiy ravishda o‘zgarib turadi. Qo‘shish, ko‘paytirish, bo‘lish kabi arifmetik operatsiyalarini bajarishdan iborat vazifalar anchadan beri intellektual deb hisoblanmay qo‘ydi. Differentsial tenglamani integratsiyalash, agar u uchun qat’iy deterministik algoritm ma’lum bo‘lsa intellektual muammo deb hisobga olinmaydi. Hozirgi vaqtda algoritmlash mumkin bo‘lmagan vazifalarni intellektual deb hisoblash odatiy holdir. Bular noaniq, ishonchsiz, nomalum va hatto noan’anaviy bilimlarni manipulyatsiya qilishni talab qiladigan vazifalardir.

SI qoidalarini ko‘rib chiqishni atamalar va ta’riflardan boshlaymiz.

Aql (*intelligence*) atamasi lotincha *intellectus* — aql, ong, fikirlash degan ma’noni anglatadi (insonning fikrlash qobiliyati). Shunga ko‘ra, **sun’iy intellekt** (artificial intelligence) — SI (AI) odatda avtomatik tizimlarning inson intellektining individual funktsiyalarini o‘z zimmasiga olish xususiyati sifatida talqin etiladi, masalan, ilgari olingan tajriba va tashqi ta’sirlarni oqilona tahlil qilish asosida maqbul qarorlarni tanlash va qabul qilish.

“Aql” tushunchasi bugungi kunda ham texnikada, ham muhandislik fanlarda qo‘llaniladi, bu ongning psixologik va falsafiy tadqiqotlari kontekstida shakllangan ta’riflardan farq qiladi. **Aql deganda** biz fikrlash, voqealarni oldindan bilish, o‘z harakatlarining natijalarini oldindan bilish, ularning holati va atrof-muhitini tahlil qilish va baholash, atrofdagi dunyo haqidagi g‘oyalarimizga muvofiq qaror qabul qilish qobiliyatini tushunamiz. Akademik N.N. Moiseev tomonidan berilgan ta’rif intellektual faoliyatni informatika nuqtai nazaridan ko‘rib chiqadi. Ammo bu aql-idrokda eng muhim narsani ta’kidlaydi - bu mavhum fikrlash, mavhumlik qobiliyati, buning natijasida o‘z-o‘zini anglash va reflekslash paydo bo‘ladi.

Shunday qilib, **aql** – bu miyaning tajriba asosida o‘rganish jarayonida bilimlarni egallash, yodlash va maqsadli o‘zgartirish va turli xil sharoitlarga moslashish orqali (intellektual) muammolarni hal qilish qobiliyatidir.

Shu bilan birga, “**bilim**” atamasi nafaqat sezgi organlari orqali olinadigan ma’lumotni anglatadi. Ushbu turdagi bilimlar juda muhim, ammo intellektual faoliyat uchun etarli emas. Haqiqat shundaki, bizni o’rab turgan atrof-muhit ob’ektlari nafaqat sezgi organlariga ta’sir qilish, balki bir-biri bilan muayyan munosabatlarda bo’lish xususiyatiga ega. Atrof muhitda intellektual faoliyatni amalga oshirish (yoki hech bo’lmaganda mavjud bo’lish) uchun bilim tizimida bu dunyoning modeliga ega bo’lish kerak. Atrof-muhitning ushbu axborot modelida haqiqiy ob’ektlar, ularning xususiyatlari va ular o’rtasidagi munosabatlar nafaqat namoyish etiladi va esda qoladi, balki aqlning ushbu ta’rifida ta’kidlanganidek, aqliy ravishda “maqsadga muvofiq ravishda o’zgartirilishi” mumkin. Shu bilan birga, tashqi muhit modelining shakllanishi “tajriba asosida o’rganish va turli xil sharoitlarga moslashish jarayonida” sodir bo’lishi muhimdir.

Intellektual vazifa. Intellektual vazifa va shunchaki vazifa o’rtasidagi farqni tushuntirish uchun kibernetikaning asosiy shartlaridan biri bo’lgan “algoritm” atamasini kiritish kerak.

Algoritm deganda ma’lum bir sinf (to’plam) vazifalaridan har qanday muammoni hal qilish uchun operatsiyalar tizimini ma’lum bir tartibda bajarish to’g’risida aniq ko’rsatma tushuniladi. “Algoritm” atamasi 9-asrda eng oddiy arifmetik algoritmlarni taklif qilgan o’zbek matematiki Al-Xorazmiy nomidan kelib chiqqan. Matematika va kibernetikada ma’lum bir turdagi muammolar sinfi uni hal qilish uchun algoritm o’rnatilganda hal qilingan deb hisoblanadi. Algoritmlarni topish insonning turli xil vazifalar sinflarini echishda tabiiy maqsadidir. Ushbu turdagi ba’zi bir vazifalar uchun algoritmni topish juda zukkolik va yuqori malakani talab qiladigan nozik va murakkab mulohazalar bilan bog’liq. Muayyan turdagi muammolar sinfini hal qilish algoritmini topish bilan bog’liq **muammolar intellektual deb nomlanadi.**

Yechimi allaqachon aniqlangan algoritmlar kelsak, SI sohasidagi taniqli mutaxassis M. Minskiyning ta’kidlashicha, “ularga “intellektuallik” kabi mistik xususiyatlarni berish kerak emas”. Darhaqiqat, bunday algoritm allaqachon topilgandan so’ng, tegishli muammolarni hal qilish jarayoni shunday bo’ladiki, uni

odam, kompyuter (to'g'ri dasturlashtirilgan) yoki robot aniq bajarishi mumkin, biz esa buni qanday bajarilishi haqda hech qanday tasavurga ega bo'lmashimiz xam mumkin. Bu faqat hal qiluvchi vazifani bajaradigan shaxs jarayon shakllanadigan elementar operatsiyalarni bajarishga qodir bo'lishi va bundan tashqari, taklif qilingan algoritm tomonidan nozikli va ehtiyotkorlik bilan boshqarilishi kerak. Bunday shaxs, aytilganidek, faqat mashinal harakat qilib, ko'rib chiqilayotgan turdagi har qanday muammoni muvaffaqiyatli hal qilishi mumkin.

Shuning uchun ularning intellektual vazifalarni bajarishi standart usullari mavjud bo'lgan vazifalarni istisno qilish mutlaqo tabiiy ko'rinadi. Bunday muammolarga misol qilib turli arifmetik hisoblash muammolarini keltirish mumkin: chiziqli tenglamalarni echish, differentsial tenglamalarning raqamli integratsiyasi va boshqalari ushbu turdagi muammolarni hal qilish uchun hisoblash mashinasi dastur shaklida osongina amalga oshirilishi mumkin bo'lgan elementar operatsiyalarning ma'lum bir ketma-ketligi bo'lgan standart algoritmlar mavjud. Aksincha, tasvirni tanish, shaxmat o'ynash, teoremlarni isbotlash va boshqalar kabi intellektual muammolarning keng klassi uchun, aksincha, echimni topish jarayonini individual elementar bosqichlarga rasmiy ravishda ajratish ko'pincha juda qiyin bo'ladi, hatto ularni echish unchalik qiyin bo'lmasa ham. Shunday qilib, aqlning ta'rifini muayyan muammolarni hal qilish algoritmlarini yaratishga qodir bo'lgan universal superalgoritm sifatida ifodalash mumkin.

Bu erda yana bir qiziqarli eslatma shundaki, bizning ta'riflarimizga asoslanib, dastlash soxasi eng intellektuallardan biridir, chunki dasturchi faoliyatining mahsuli algoritimli dasturlar hisoblanadi. Shuning uchun hatto SI elementlarini yaratish uning mehnat unumdorligini sezilarli darajada oshirishi kerak.

Intellektual muammolarni hal qilishga qaratilgan miya faoliyati (aqlga ega) fikrlash yoki **intellektual faoliyat deb ataladi**. Aql - idrok va fikrlash teoremlarni isbotlash, mantiqiy tahlil qilish, vaziyatlarni aniqlash, xatti-harakatlarni rejalashtirish, o'yin o'ynash va noaniqlik sharoitida boshqarish kabi muammolarni hal qilish bilan uzviy bog'liqdir. Muammolarni hal qilish jarayonida namoyon bo'ladigan aqlning o'ziga xos xususiyatlari o'rganish, umumlashtirish, tajriba (bilim

va ko'nikmalar) to'plash va muammolarni hal qilish jarayonida o'zgaruvchan sharoitlarga moslashish qobiliyatidir. Aqlning ushbu fazilatlari tufayli miya turli xil muammolarni hal qila oladi, shuningdek, buning uchun standart, oldindan ma'lum bo'lgan echim usullari mavjud bo'lmagan muammoni hal qilish qobiliyatiga ega. Shunday qilib, aql-idrokka ega bo'lgan miya keng ko'lamli muammolarni (shu jumladan norasmiy) hal qilishning universal vositasidir.

Shuni yodda tutish kerakki, aqilni boshqa, faqat xulq-atvor (funktional) ta'riflari mavjud. Shunday qilib, A. N. Kolmogorovning so'zlariga ko'ra, fan, adabiyot va san'at muammolarini uzoq vaqt muhokama qilish mumkin bo'lgan har qanday moddiy tizim aqlga ega. Aqlning xulq-atvorini talqin qilishning yana bir misoli A. Turingning taniqli ta'rifi bo'lishi mumkin. Uning ma'nosi quyidagicha. Turli xonalarda odamlar va mashina bor. Ular bir-birlarini ko'ra olmaydilar, lekin ma'lumot almashish imkoniyatiga ega (masalan, elektron pochta orqali). Agar o'yin ishtirokchilari o'rtasidagi muloqot jarayonida odamlar ishtirokchilardan biri mashina ekanligini aniqlay olmasalar, unda bunday mashinani aqlga ega deb hisoblash mumkin.

A. Turing tomonidan taklif qilingan fikrlashga taqlid qilish rejasi quyidagicha. "Voyaga etgan odamning aql — idrokiga taqlid qilishga urinish - biz inson miyasining hozirgi holatiga etib borishi jarayoni haqida ko'p o'ylashga majbur bo'lamiz... nega biz kattalar aql-idrokiga taqlid qiladigan dasturni yaratishga urinishdan ko'ra, bolaning aql-idrokiga taqlid qiladigan dasturni yaratishga harakat qilmaymiz? Axir, agar bolaning aql-zakovati tegishli tarbiyani oladigan bo'lsa, u kattalarning aql-idrokiga aylanadi... bizning hisob-kitobimiz shundan iboratki, unga o'xshash qurilmani osongina dasturlash mumkin... shunday qilib, biz muammomizni ikki qismga ajratamiz: "bola dasturi" ni yaratish vazifasi va ushbu dasturni "tarbiyalash" vazifasi - deb yozadi Turing.

Yuqoridagilarni hisobga olgan xolda deyarli barcha SI tizimlari ushbu yo'ldan foydalanadi deb aytishimiz mumkin. Axir, barcha bilimlarni ancha murakkab tizimga kiritish deyarli mumkin emasligi aniq. Bundan tashqari, yuqorida sanab

o'tilgan intellektual faoliyat belgilari (tajriba to'plash, moslashish va boshqalar) faqat shu yo'lda namoyon bo'ladi.

"Sun'iy intellekt" atamasi 1956 yilda Massachusetts texnologiya instituti professori J. Makarti hisoblash jarayonlarini o'rganish nazariyasi va amaliyoti bilan bog'liq bo'lgan amerikalik fan mutaxassislari ilmiy anjumanida tilga olingan. Bu anjumanni amerikaliklar sun'iy intellekt bo'yicha birinchi konferentsiya deb hisoblashadi va Dortmut kollejidagi ushbu yig'ilishda yangi ilmiy-texnik sohadagi ikkita asosiy vazifa shakllantirildi: *inson fikrlash mexanizmini o'rganish va ushbu jarayonni taqlid qila oladigan elektron mashinani yaratish*.

Ushbu ilmiy sohani to'liq tavsiflovchi yagona ta'rif bugungi kungacha mavjud emas. Bugungi kunda ko'plab nuqtai nazarlar orasida uchta ustunlik mavjud. Birinchisiga ko'ra - SI sohasidagi tadqiqotlar fundamental tadqiqotlar bo'lib, unda an'anaviy ravishda intellektual deb hisoblangan va ilgari rasmiylashtirish va avtomatlashtirishga bo'ysunmagan muammolarni hal qilish modellari va usullari ishlab chiqishga yo'naltirilgan. Ikkinchi nuqtai nazarga ko'ra, yangi yo'nalish kompyuterda muammolarni hal qilishning yangi g'oyalari, tubdan farq qiladigan dasturlash texnologiyasini ishlab chiqish, kompyuter arxitekturasiga o'tish, klassik arxitekturani rad etish bilan bog'liq bo'lib masalalarni echishga bag'ishlangan va nihoyat, uchinchi nuqtai nazar, ehtimol, eng pragmatik bo'lib, sun'iy intellekt sohasidagi ishlar natijasida yaratilgan tizimlar yaroqsiz bo'lgan muammolarni hal qiladigan ko'plab amaliy tizimlar tushiniladi.

Albatta, bu uchta nuqtai nazarning barchasi bir-biri bilan bog'liq, SI sohasida fundamental tadqiqotlar, yangi dasturlash texnologiyasi, yangi texnik vositalar arxitekturasi rivojlanmoqda va bularning barchasi turli sohalarda ishlashga mo'ljallangan amaliy tizimlarni yaratish uchun ishlatiladi.

Sun'iy intellekt deganda biz an'anaviy ravishda intellektual deb hisoblangan va rasmiylashtirish va avtomatlashtirish mumkin bo'lgan muammolarni hal qilish uchun modellar, usullar, texnik va dasturiy vositalar ishlab chiqilgan ilmiy tadqiqotlar sohasini tushunamiz.

Aqlli tizimlar deganda maqsadli xatti-harakatlar qobiliyatini ko'rsatadigan har qanday biologik, sun'iy yoki rasmiy tizimlar tushuniladi. Ikkinchisiga aloqa xususiyatlari (namoyon bo'lishi), bilimlarni to'plash, qaror qabul qilish, o'rganish, moslashish va boshqalar kiradi.

SI tizimlari - bu kompyuterda bunday amaliy vazifalarni bajarish uchun mo'ljallangan tizimlar, agar ular odamlar tomonidan bajarilsa, intellektual deb ataladi. AI nazariyasida ko'pincha AI tizimlari aqlli tizimlar deb ataladi.

Sun'iy intellektda "intellektual tizim" tushunchasining yana bir ta'rifi D. A. Pospelov tomonidan taklif qilingan. Unga ko'ra tizim quyidagi uchta asosiy funktsiyani amalga oshirgan taqdirda intellektual hisoblanadi:

1) Bilimlarni taqdim etish va qayta ishlash funktsiyasi. Intellektual tizim atrofdagi dunyo haqida bilimlarni to'plashi, ularni pragmatik va izchillik nuqtai nazaridan tasniflashi va baholashi, yangi bilimlarni olishi, yangi bilimlarni bilimlar bazasida saqlanadigan bilimlar bilan bog'lashi kerak.

2) Fikrlash funktsiyasi. Intellektual tizim mantiqiy xulosa va to'plangan bilimlardagi bog'liqliklarni aniqlash mexanizmlari yordamida yangi bilimlarni shakllantirishga, shaxsiy bilimlar asosida umumlashtirilgan bilimlarni olishga va o'z faoliyatini mantiqiy rejalashtirishga qodir bo'lishi kerak.

3) Aloqa funktsiyasi. Intellektual tizim inson bilan tabiiy tilga yaqin tilda muloqot qila olishi va atrofdagi dunyoni, o'z faoliyatini tushuntirish, bilim orqali odamga yordam berish, uning xotirasida saqlanadigan va mantiqiy fikrlash vositalari yordamida muloqot qilishi, birinchi navbatda vizual va tovushni idrok etishda foydalanadigan kanallarga o'xshash kanallar orqali ma'lumot olishi, "o'zi uchun" yoki shaxsning iltimosiga binoan shakllantira olishi kerak.

SI tizimlarining rivojlanish tarixi

Inson bizning idrokimiz uchun mavjud bo'lgan eng murakkab ob'ektdir va fikrlash qobiliyati uning asosiy xususiyati — atributdir. Sun'iy intellect - bu inson atributini o'rganish va modellashtirishni maqsad qilgan fan. Fikrlashning tabiati qanday? Biz o'ylaganimizda, his qilganimizda, ko'rganimizda, tushunganimizda tanamizda qanday jarayonlar sodir bo'ladi? Asosan, miyamiz qanday ishlashini

tushunish va jonsiz tabiatni o'ylashga majbur qilish mumkinmi? Ming yillar davomida inson bu savollarni berib kelgan, ammo hozirgacha biz ularga to'liq aniqlik bilan javob bera olmaymiz.

Inson ongining sun'iy o'xshashini yaratishga urinishlar tarixi 700 yildan ortiq davom etadi. Tarixda inson ongini modellash tiradigan mashinani yaratishga qaratilgan birinchi urinish ispan ritsari, shoir, faylasuf, ilohiyotshunos, alkimyogar, ixtirochi Raymund Lulliyning nomi bilan bog'liq.

Bu odamning shaxsiyati katta qiziqish uyg'otadi. Qirolning sevimlisi, duelist va shoir, tarixchilar u haqida yozganidek, ritsar Raymund Lullius to'satdan dunyoviy o'yin-kulgidan voz kechib, donishmand bo'lish uchun monastirga boradi. Uning qarori ilm-fanni tushunish va ularning yordami bilan suyakli xonimini kasallikdan davolash ezgu maqsad tufayli yuzaga keladi. Afsuski, Lulliya o'z maqsadiga erisha olganmi yoki yo'qmi, ma'lum emas. Ma'lumki, 80 yoshida sun'iy intellekt bo'yicha ma'ruzalar o'qishga urinishlar vaqtida toshbo'ron qilingan.

Lulliy 1235 yilda tug'ilgan va 1315 yilda vafot etgan, uning davrida olimlar bir-biri bilan bog'liq bo'lgan, koinotning umumiy tasvirini beradigan va insoniyatni qiziqtirgan barcha savollarga javob beradigan ba'zi universal tushunchalar va haqiqatlarni izlash bilan band edilar.

O'z davri olimlarining an'analarini rivojlantirib, Lulliy aylanish imkoniyatiga ega bo'lgan aylanalar tizimidan iborat mashinani yaratdi. Har bir doira turli xil ranglarda bo'yalgan va lotin harflari bilan belgilangan sektorlarga bo'lingan. Doiralalar bir — biriga bog'langan va ularni aylantirish orqali ramzlar va ranglarning turli xil kombinatsiyalarini olish mumkin edi — va u haqiqat formulasi deb nomlangan.

Lulliyning mashinalari turli mavzularda ishlashlari va har xil savollarga javob berishi, munajjimlar bashoratini tuzish, kasalliklarga tashxis qo'yish va hosilni bashorat qilishi mumkin edi. Eng so'nggi versiyada Lulliya mashinasi harflar bilan belgilangan va turli xil tushunchalar, elementlar va bilim ob'ektlarini ramziy ma'noga ega bo'lgan turli xil ranglarga bo'yalgan 14 doiradan iborat edi. Doiralarni dastaklar tizimi boshqargan. Aylanish paytida ular taxminan 18 kvadrillion ($18 \cdot 10^{15}$)

harf va rang “haqiqatlari”ning turli xil birikmalarini hosil qilishi mumkin edi. Mashinaga so‘rovlar ichki doirani aylantirish orqali kiritildi, unda Nima? Nima uchun? Nimadan? Qancha? Qanday qilib? Qayerda? Qachon? Qaysi? Ikkalasidan qaysi biri? kabi to‘qqizta savol variantlari yozilgan.

Zamonaviy til bilan aytganda, Lulliya mashinasi mohiyatan bilimlar bazasi, kirish va chiqish moslamalari, tabiiy aloqa tili bilan ta’minlangan mexanik ekspert tizimi edi. Dunyo haqidagi barcha bilimlarni emas, balki ularning hech bo‘lmaganda bir qismini mantiqiy operatsiyalarga kamaytiring va keyin inson miyasiga emas, balki mexanik qurilmaga to‘plangan bilimlar bazasidan kelib chiqadigan “bilim formulalarini” chiqarish tartibini topsh-birinchi marta o‘rta asr ritsari Raymund Lullius tomonidan ilgari surilgan va amalga oshirilgan ushbu sun’iy intellekt g‘oyasi etti asr oshib bu kunlarda u o‘zining gullab-yashnashi va g‘alabasiga erishdi.

XX asrning 40-yillarida, elektron kompyuterlarning paydo bo‘lishi bilan sun’iy intellekt qayta tug‘ildi. Sun’iy intellekt mustaqil ilmiy yo‘nalishga ajratildi. "Sun’iy intellekt" (artificial intelligence) atamasining o‘zi 1956 yilda Stanford universitetida (AQSh) shu nomli ilmiy seminarda taklif etilgan.

O‘shandan beri sun’iy intellekt tarixi ikki ekstremal — optimizm va pessimizm o‘rtasida doimiy tortishuvlar va baxslarga sabab bo‘lib kelgan. Amerikalik iqtisodchi va sotsiolog, menejment nazariyasi, ijtimoiy jarayonlarni modellashtirish sohasidagi tadqiqotchi G. Simonning 1957 yilda amalga oshirgan mashhur bashoratlariga ko‘ra:

- yaqin o‘n yillikda kompyuter shaxmat bo‘yicha jahon chempioni unvonini qo‘lga kiritadi;
- o‘n yillar orasida kompyuter muhim yangi matematik teoremani kashf etadi va isbotlay oladi;
- o‘n yillar ichida psixologiya sohasidagi ko‘pgina nazariyalar kompyuter uchun dasturlar ko‘rinishini oladi.

Deyarli yarim asrdan so‘ng, biz Simonning bashoratlari asta-sekin amalga oshayotganini, u faqat bashoratlarni amalga oshish vaqtidangina xato qilganini ko‘ramiz. Shuni ham ta’kidlashimiz mumkinki, yosh kibernetika atrofidagi eyforiya

ham ijobiy, ham salbiy oqibatlarga olib keldi. Bir tomondan, u AQSh hukumati tomonidan katta grantlar ajratishda ifodalangan yangi ilmiy yo‘nalishga jamoatchilikning qiziqishini uyg‘otdi. Boshqa tomondan, kibernetika ko‘proq “hushyor fikrlaydigan” olimlar uchun juda qattiq tanqid ob‘ektiga aylandi.

Sun‘iy intellekt mustaqil fan sohasi sifatida tan olinganidan ko‘p o‘tmay, u ikkita asosiy yo‘nalishga bo‘lindi: neyrokibernetika va “qora quti” kibernetikasi. Ushbu yo‘nalishlarning birinchisi ba‘zan past darajadagi yoki yuqoriga, ikkinchisi esa yuqori darajadagi yoki pastga yo‘naltirilgan deb nomlanadi.

Neyrokibernetikaning asosiy g‘oyasini quyidagicha shakllantirish mumkin. Fikrlashga qodir bo‘lgan yagona narsa — bu inson miyasi. Shuning uchun har qanday fikrlash moslamasi inson miyasining qiyofasi va o‘xshashligida, uning tuzilishini, ishlash printsiptini takrorlashi kerak. Shunday qilib, neyrokibernetika miya tuzilishi va uning faoliyatini modellashtirish bilan shug‘ullanadi.

Ma‘lumki, inson miyasi ko‘plab o‘zaro bog‘liq nerv hujayralari-neyronlardan iborat. Shuning uchun neyrokibernetiklarning sa‘y — harakatlari neyronlarga o‘xshash elementlarni ishlab chiqishga va ushbu elementlarni tizimlarga-neyron tarmoqlari va neyrokompyuterlarga birlashtirishga qaratilgan. Birinchi neyron tarmoqlar va neyrokompyuterlar 1950-yillarning oxirida amerikalik olimlar V. Mak-Kulloq, V. Pitts va F. Rozenblatt tomonidan taklif qilingan va yaratilgan. Bu inson ko‘zini va uning miya bilan o‘zaro ta‘sirini modellashtiradigan qurilmalar edi.

Bugungi kunda neyrokompyuter va neyron tarmoq texnologiyalari sun‘iy intellektning eng istiqbolli va tez rivojlanayotgan sohalaridan biridir. Ushbu sohada yapon tadqiqotchilari katta yutuqlarga erishdilar. Ular VI avlod kompyuterini-miya tuzilishini modellashtiradigan va keng bilim bazasiga ega bo‘lgan neyrokompyuterni yaratdilar. Ushbu sohada rus olimlari katta yutuqlarga erishdilar. Mahalliy neyrokompyuterlar uzoq vaqtdan beri murakkab harbiy texnik ob‘ektlarni boshqarish uchun ishlatilgan.

Neyrokibernetikadan farqli o‘laroq, “qora quti” kibernetikasi fikrlash qurilmasining ishlash printsiptiga ahamiyat bermaydi. Asosiysi, uning funktsional faoliyatini etarli darajada modellashtirish. Sun‘iy intellektning ushbu yo‘nalishi

apparat bazasidan qat'i nazar, mavjud kompyuterlardan foydalangan holda aqli muammolarni hal qilish algoritmlarini izlashga qaratilgan.

Miya funktsiyalarini modellashtirish vazifasini o'z zimmasiga olgan olimlar jiddiy muammoga duch kelishdi. Ma'lum bo'lishicha, ko'p asrlik tadqiqot tarixiga qaramay, mavjud fanlarning hech biri (falsafa, psixologiya, tilshunoslik va boshqalar) inson tafakkurining o'ziga xos algoritmini taklif qila olmagan. Shuning uchun kibernetika o'z fikrlash modellarini yaratishi kerak edi.

XX asrning 50 yilari oxiri labirint qidirish modeli paydo bo'ldi. Ushbu yondashuvga ko'ra, intellektual muammoni hal qilish labirint bo'ylab harakatlanish shaklida taqdim etilgan juda ko'p sonli variantlarni ko'rib chiqish orqali amalga oshirildi. Bunday algoritmlarni yaratish, ularning tanqidchilarining fikriga ko'ra, Britaniya muzeyida saqlangan barcha kitoblarni qayta yozishga urinishdan ko'ra, maymunlarni yozuv mashinkalari orqasiga qo'yish va maymunlar ertami-kechmi tasodifan mazmunli so'z, ibora yoki sahifani chop eta olishiga umid qilishdan ko'ra aqlliyoq emas edi. Hozirgi vaqtda labirint qidirish modeli o'lik deb tan olingan va kompyuter o'yin dasturlarida cheklangan ravishda foydalanishlarga ega.

Kompyuter yordamida hal qilinadigan birinchi intellektual vazifalar mantiqiy o'yinlar (shashka, shaxmat), teoremlarning isboti edi. To'g'ri, bu erda Klod Shannonning "elektron sichqonchasi" kabi kibernetik o'yinchoqlarni ta'kidlash kerak, ular murakkab releli sxemasa bilan boshqarilgan. Bu sichqon labirintni "o'rganishi" va undan chiqish yo'lini topishi mumkin edi. Bundan tashqari, u allaqachon ma'lum bo'lgan labirintga joylashtirilgan, u chiqish yo'lini qidirmagan va darhol, to'siqlarga qaramasdan, labirintdan chiqib ketgan.

Amerikalik kibernetik A. Samuel tuzgan kompyuter dasturi unga shashka o'ynashga imkon beradi va o'yin davomida mashina o'qitiladi yoki hech bo'lmaganda to'plangan tajriba asosida o'z o'yinini takomillashtirish orqali o'rganilayotgandek taassurot qoldiradi. 1962 yilda ushbu dastur AQShdagi eng kuchli shaxmatchi R. Nili bilan jang qildi va g'alaba qozondi.

1960-yillarning boshida *evristik dasturlash davri boshlandi*. Ushbu atama muallifi amerikalik matematik Poya yozganidek, evristikaning maqsadi kashfiyot va

ixtirolarni amalga oshirish usullari va qoidalarini o'rganishdir. Gap shundaki, Arximed vannadan "Evrika" deb qichqirib turganida u qanday qilib suyuqlikka botgan jism o'z vaznini o'zi chiqarib yuborgan suv miqdori kabi yo'qotishini taxmin qilganini tushuntirmadi. Nyuton olma tushishini kuzatish orqali tortishish qonunini kashf etdi. Mendeleevga davriy jadval tuzish printsipi tushida namoyon bo'ldi. Shoirlar va musiqachilar ijodiy izlanishlarga yuksak tuyg'ularga ilhomlantiradigan tuyg'ularni tushunish mumkin emas.

Ijodiy fikrlash mexanizmlarini tushunish uchun evristik yondashuv mualliflari tajriba o'tkazdilar. Matematik mantiq bilan tanish bo'lmagan talabalar guruhi tanlandi. Har bir talaba darslikdan bir yoki bir nechta teoremlarni mustaqil ravishda isbotlashi kerak edi. Shu bilan birga, unga baland ovozda mulohaza yuritish, har qanday yozuvlarni yozish, noto'g'ri yo'l tanlanganligi aniq bo'lsa, ishni to'xtatish va yana boshqatdan boshlash vazifasi yuklatilgan.

Magnitofon yozuvlarini, hisob-kitoblarni, talabalarning qoralamalarini qayta ishlagandan so'ng, dasturchilar Evristikani topdilar — talabalar teoremlarni isbotlash uchun foydalanadigan usullar va keyin ushbu evristikalar yordamida evristik dasturlashning avlodi deb hisoblanadigan "nazariy mantiq" deb nomlangan dastur tuzildi. Ushbu dastur darslikdagi barcha teoremlarni isbotladi va kursning to'liq mantiqiy yakunlanishiga qadar etishmaganlarini qo'shimcha ravishda shakllantirdi.

Fikrlashni modellashtirish va sun'iy intellektni yaratish muammosiga yuqoridagi ikkita yondashuv bilan bir qatorda *evolyutsion dasturlash* (modellashtirish) deb nomlangan uchinchi tushuncha mavjud. Ushbu yondashuvning ma'nosi shundaki, insonni modellashtirish jarayoni uning evolyutsiyasi jarayonini modellashtirish bilan almashtiriladi.

1957 yilda amerikalik fiziolog F. Rozenblatt vizual idrok etish va tanib olish modelini — perseptronni taklif qildi. Tushunchalarni o'rganishga va taqdim etilgan ob'ektlarni tanib olishga qodir bo'lgan mashinaning paydo bo'lishi nafaqat fiziologlar uchun, balki boshqa bilim sohalari vakillari uchun ham juda qiziq bo'lib chiqdi va nazariy va eksperimental tadqiqotlarning katta oqimini keltirib chiqardi.

Perseptron yoki tanib olish jarayonini taqlid qiladigan har qanday dastur ikkita rejimda ishlaydi: o'rganish (o'qitish) rejimi va tanib olish rejimi. O'qitish rejimida o'qituvchi rolini o'ynaydigan kishi (odam, mashina, robot yoki tabiat) mashinaga ob'ektlarni taqdim etadi va ularning har biri haqida, qaysi tushunchaga (sinfga) tegishli ekanligini aytadi. Ushbu ma'lumotlarga ko'ra, kontseptsiyalarning rasmiy tavsifi bo'lgan hal qiluvchi qoida tuziladi. Tanib olish rejimida mashinaga yangi ob'ektlar taqdim etiladi (umuman olganda, ilgari taqdim etilganidan farq qiladi) va iloji bo'lsa, ularni to'g'ri tasniflashi kerak.

Tanib olishni o'rgatish muammosi boshqa intellektual vazifa bilan chambarchas bog'liq — bir tildan boshqasiga tarjima qilish, shuningdek, mashinani tilga o'rgatish kabi muammolari. Lug'atdan foydalanishning asosiy grammatik qoidalari va usullarini etarlicha rasmiy qayta ishlash va tasniflash bilan siz tarjima qilish uchun etarli bo'lgan algoritm yaratishingiz mumkin, aytaylik ilmiy yoki biznes matni. Ba'zi tillar uchun bunday tizimlar 60-yillarning oxirida yaratilgan. Biroq, etarlicha katta so'zlashuv matnini izchil tarjima qilish uchun uning ma'nosini tushunish kerak. Bunday dasturlar ustida ishlash uzoq vaqtdan beri davom etmoqda, ammo to'liq muvaffaqiyatga erishishga hali ham uzoq. Shuningdek, inson va mashina o'rtasidagi dialogni qisqartirilgan tabiiy tilda ta'minlaydigan dasturlar mavjud.

Mantiqiy fikrlashni modellashtirishga kelsak, teoremlarni isbotlashni avtomatlashtirish vazifasi bu erda yaxshi model vazifasi bo'lib xizmat qilishi mumkin. 1960 yildan boshlab birinchi darajali predikatlarni hisoblashda teoremlarning dalillarini topishga qodir bo'lgan bir qator dasturlar ishlab chiqildi. Amerikalik sun'iy intellekt bo'yicha mutaxassis Dj. Makkattiga ko'ra ya'ni "sog'lom fikr" bu deduktiv xulosalar chiqarish qobiliyatidir.

Savol-javob tizimini amalga oshiradigan K. Grin va boshqalarning dasturida bilimlar predikatlar mantig'i tilida aksiomalar to'plami shaklida yoziladi va mashinaga berilgan savollar teorema isbotlanishi kerak deb shakllantiriladi. Amerikalik matematik Xao Vangning "intellectual" dasturi katta qiziqish uyg'otmoqda. Ushbu dastur IBM-704 ishining 3 daqiqasida 220 ta nisbatan oddiy

lemma va teoremlarni fundamental matematik monografiyadan olib chiqdi, so'ngra 8,5 daqiqada yana 130 ta murakkab teoremlarni isbotladi, ularning ba'zilar hali matematiklar tomonidan topilmagan edi. To'g'ri, hozirgacha biron bir dastur matematiklarga kerak bo'lgan va tubdan yangi bo'lgan "posarez" deb nomlangan birorta teoremani chiqarmagan va isbotlamagan.

Sun'iy intellektning amaliy qo'llanmalarida katta yutuq 1970-yillarning o'rtalarida, universal fikrlash algoritmini izlashdan voz kechib, dasturchilar mutaxassislarining aniq bilimlarini modellashtirishni boshlaganlarida yuz berdi. Sun'iy intellektning yangi yo'nalishi-*ekspert tizimlari ochildi*. Ekspert tizimlarining paydo bo'lishi bilan intellektual axborot texnologiyalari sohasi birinchi marta iqtisodiy jihatdan samarali bo'ldi.

1980-yillarning o'rtalaridan boshlab sun'iy intellekt kompyuter sanoatining tijorat jihatidan eng jozibali yo'nalishlaridan biri hisoblanadi. Yillik investitsiyalar o'sib bormoqda, sanoat va harbiy ekspert tizimlari yaratilmoqda. Ekspert tizimlariga alternativa sifatida neyron tarmoq va neyrokompyuter texnologiyalari paydo bo'ladi va bozorni muvaffaqiyatli zabt etadi, bunda miyada sodir bo'ladigan jarayonlar singari bilimlar neyronlararo aloqalarda yo'qolib ketadi va tizimni dasturlash jarayoni uning o'qitish bilan almashtiriladi.

Hozirgi vaqtda mashinalarga katta amaliy ahamiyatga ega bo'lgan biznes yoki boshqa soxaga taluqli o'yinlarini o'ynashga imkon beradigan dasturlar mavjud va muvaffaqiyatli qo'llanilmoqda. Bu erda ham dasturlarga insonning o'rganish va moslashish qobiliyatini berish juda muhimdir. Amaliy ahamiyatga ega bo'lgan eng qiziqarli intellektual vazifalardan biri bu tasvirlarni va vaziyatlarni tanib olishni o'rganish vazifasidir. Bu masalalar ustida turli fanlar vakillari — fiziologlar, psixologlar, matematiklar, muhandislar shug'ullanishmoqda. Vazifaga bunday qiziqish nazariy tadqiqotlar natijalaridan keng amaliy foydalanishning istiqbollari bilan rag'batlantirildi: o'quvchi avtomatlar, sun'iy intellekt tizimlari, tibbiy tashxis qo'yish, sud ekspertizasini o'tkazish va boshqalar, shuningdek murakkab hissiy vaziyatlarni taniy oladigan va tahlil qila oladigan robotlardir.

SI tizimlarining juda katta yoʻnalishi **robototexnikadir**. Robotning aql-zakovati va universal kompyuterlarning aql-zakovati oʻrtasidagi asosiy farq nima?

Bu savolga javob berish uchun buyuk rus fiziologi I. M. Sechenovga tegishli boʻlgan quyidagi soʻlari oʻrinlidir: “... miya faoliyatining tashqi koʻrinishlarining barcha cheksiz xilma — xilligi nihoyat bitta hodisaga-mushaklarning harakatiga olib keladi”. Boshqacha qilib aytganda, insonning barcha intellektual faoliyati oxir-oqibat harakatlar orqali tashqi dunyo bilan faol aloqada boʻlishga qaratilgan. Xuddi shunday, robotning razvedka elementlari birinchi navbatda uning maqsadli harakatlarini tashkil etishga xizmat qiladi. Shu bilan birga, sof kompyuter SI tizimlarining asosiy maqsadi atrof-muhitni idrok etish yoki abstrakt harakatlarini tashkil qilish bilan bogʻliq emas balki mavhum yoki yordamchi xarakterga ega boʻlgan intellektual muammolarni hal qilishdan iboratdir.

Birinchi robotlarni aqlli deb atash qiyin. Faqat 60-yillarda universal kompyuterlar tomonidan boshqariladigan sezgir robotlar paydo boʻldi. Masalan, 1969 yilda elektrotexnika laboratoriyasida (Yaponiya) “sanoat aqlli robotlarini” loyihasini ishlab chiqish boshlandi. Ushbu rivojlanishning maqsadi vizual nazorat bilan oʻrnatish ishlarini bajarish uchun sunʼiy intellekt elementlari bilan boyitilgan sezgir boshqaruvchi robotini yaratishdir.

Robot manipulyatori olti daraja erkinlikka ega va NEAC-3100 mini-kompyuterlari tomonidan boshqariladi (operativ xotira hajmi 32000 soʻz, magnit disklardagi tashqi xotira hajmi 273000 soʻz), bu kuzatuv elektro-gidravlik tizimi tomonidan ishlab chiqilgan kerakli dasturiy harakatni hosil qiladi. Manipulyatorning tutqichi sezgir sensorlar bilan jihozlangan.

Vizual idrok etish tizimi sifatida obʼektlarning rangini aniqlash uchun qizil, yashil yoki koʻk filtrlar bilan jihozlangan ikkita kamera ishlatiladi. Kameraning koʻrish maydoni 64*64 katakka boʻlingan. Olingan maʼlumotlarni qayta ishlash natijasida robotni qiziqtirgan obekt egallagan maydon taxminan aniqlanadi. Bundan tashqari, ushbu obektni batafsil oʻrganish uchun aniqlangan maydon yana 4096 katakka boʻlinadi. Agar obʼekt tanlangan “oyina” ga mos kelmasa, u avtomatik ravishda harakat qiladi, xuddi odam obʼektga qaraganday. Elektrotexnika

laboratoriyasining roboti maxsus yorug'lik ostida tekisliklar va silindrsimon yuzalar bilan chegaralangan oddiy narsalarni taniy oldi. Ushbu eksperimental namunaning narxi taxminan 400 000 dollarni tashkil etgan.

Asta-sekin robotlarning xarakteristikalari monoton ravishda yaxshilandi, ammo hozirgi kunga qadar ular insonlar kabi aqilli emas, garchi ba'zi operatsiyalar allaqachon insonlardan yaxshiroq darajada amalga oshirayotgan bo'lsa xam.

Bu yerda N. M. Amosov va V. M. Glushkov rahbarligida robotlarning intellektual elementlarini ishlab chiqishga qaratilgan tadqiqotlar olib borilayotgan Kiyev kibernetika institutining ishlarini ajratib ko'rish mumkin. Ushbu tadqiqotlarda tasvirni aniqlash va nutqni aniqlash, mantiqiy xulosa (teoremlarni avtomatik isbotlash) va neyronga o'xshash tarmoqlar orqali boshqarish muammolariga alohida e'tibor qaratilgan.

Masalan, 70-yillarda yaratilgan transport avtonom integral robotining (Tair) modelini ko'rib keltirishimiz mumkin. Konstruktiv jihatdan Tair uch g'ildirakli shassi bo'lib, unda sensorli tizim va boshqaruv bloki o'rnatilgan. Sensor tizimi quyidagi sezgir vositalarni o'z ichiga oladi: optik masofani o'lchash moslamasi, ikkita radio mayoq va kompasli navigatsiya tizimi, aloqa sensorlari, aravaning egilish burchagi sensorlari, taymer va boshqalar. Tairni boshqa davlatlarda yaratilgan boshqa ko'plab tizimlardan ajratib turadigan xususiyat shundaki, uning tarkibida kompyuter yo'q. Boshqaruv tizimining asosini bordagi neyropaga o'xshash tarmoq tashkil etadi, unda sensorli ma'lumotlarni qayta ishlash, xatti-harakatlarni rejalashtirish va robot harakatini boshqarishning turli algoritmlari amalga oshiriladi. Shunday qilib, sun'iy intellektni modellashtirishning tarixiy jihatdan uchta asosiy yo'nalishi mavjud.

Birinch**i yondashuv** doirasida tadqiqot ob'ekti inson miyasining tuzilishi va mexanizmlari bo'lib, yakuniy maqsad fikrlash sirlarini ochishdir. Ushbu yo'nalishdagi tadqiqotlarning zarur bosqichlari psixofiziologik ma'lumotlarga asoslangan modellarni yaratish, ular bilan tajribalar o'tkazish, intellektual faoliyat mexanizmlari to'g'risida yangi farazlarni ilgari surish, modellarni takomillashtirish va boshqalar.

Ikkinchi yondashuv sun'iy intellektni o'rganish ob'ekti sifatida ko'rib chiqadi. Bu erda biz kompyuterlardan foydalangan holda intellektual faoliyatni modellashtirish haqida gapiramiz. Ushbu yo'nalishdagi ishlarning maqsadi kompyuterlarning algoritmik va dasturiy ta'minotini yaratishdir, bu sizga intellektual muammolarni insondan qolishmaydigan darajada hal qilishga imkon beradi.

Uchinchi yondashuv tabiiy va sun'iy intellekt imkoniyatlarining simbioziga aralash odam-mashina yoki interaktiv aqlli tizimlarni yaratishga qaratilgan. Ushbu tadqiqotlardagi eng muhim muammolar tabiiy va sun'iy intellekt o'rtasidagi funktsiyalarning optimal taqsimlanishi va inson va mashina o'rtasidagi muloqotni tashkil etishdir.

Ekspert tizimlari

Ekspert tizimlari deganda biz kompyuter xotirasida (bilimlar bazasida) saqlanadigan umumlashtirilgan jamoaviy tajriba asosida vaqt etishmasligi, qarama-qarshi va ishonchli bo'lmagan ma'lumotlar, shu jumladan oldindan aytib bo'lmaydigan vaziyatlarda rasmiylashtirish qiyin bo'lgan muammolarni hal qilish bo'yicha tavsiyalar berishni ta'minlaydigan dasturiy vositalar majmuini tushunamiz.

ETga katta qiziqish bir qator sabablarga ko'ra yuzaga keladi.

Birinchidan, ular kompyuter texnologiyalaridan foydalanish imkoniyatlarini sezilarli darajada kengaytiradigan norasmiy sohalarda keng ko'lamli muammolarni hal qilishga qaratilgan.

Ikkinchidan, ET yordamida dasturlashni bilmaydigan mutaxassislar o'zlarini qiziqtirgan dasturlarni mustaqil ravishda ishlab chiqishlari mumkin, bu esa kompyuter texnologiyalaridan foydalanish doirasini kengaytirishga imkon beradi.

Uchinchidan, amaliy muammolarni hal qilishda ET bilan jihozlanmagan inson mutaxassislarining imkoniyatlaridan kam bo'lmagan va ba'zan undan yuqori natijalarga erishadi.

ETlar an'anaviy dasturlash yondashuvini rad etmaydi yoki almashtirmaydi, ular an'anaviy dasturlardan farq qiladi, chunki ular norasmiy muammolarni hal qilishga qaratilgan va tahlil asosida ajratilgan quyidagi xususiyatlarga ega.

1. Yechimlar algoritmi oldindan ma'lum emas, balki ETning o'zi tomonidan evristik texnikaga asoslangan ramziy mulohazalar yordamida qurilgan.

2. Qabul qilingan qarorlarning aniqligi, ya'ni tizim foydalanuvchi nuqtai nazaridan echimni qanday olganligini "tushunadi".

3. Ularning harakatlari va bilimlarini tahlil qilish va tushuntirish qobiliyati.

4. Dasturlash bo'yicha professional bilimga ega bo'lmagan mutaxassis foydalanuvchidan yangi bilimlarni olish va ularning xatti-harakatlarini ularga mos ravishda o'zgartirish qobiliyati.

5. Foydalanuvchi bilan "do'stona", odatda tabiiy til interfeysini ta'minlash.

6. Muayyan ekspertiza sohasini cheklash.

7. Shubhali ma'lumotlar bilan mulohaza yuritish qobiliyati.

8. Iqtisodiy foydasi.

Kelajakda turli darajadagi **bilim tizimlaridan** (BT) foydalanish kutilmoqda, bu esa ishlarni rejalashtirish va tashkil etishda ma'sul shaxslar tomonidan qabul qilingan qarorlarni qo'llab-quvvatlash vazifalarining xilma-xilligini kengaytirishga imkon beradi. Bilim tizimlari va ET o'rtasidagi asosiy farqlar quyidagilardan iborat.

ETda bilimlar bazasi mutaxassislardan olingan ma'lumotlar, ya'ni ushbu ET mo'ljallangan ma'lum bir sohadagi etakchi mutaxassislardan asosida shakllantiriladi. Xulosa qilish mashinasi ma'lum bir bilim modeliga qat'iy yo'naltirilgan bo'lib, ushbu amaliy sohaning o'ziga xos xususiyatlarini aks ettiradi. Bilim tizimlarida bilimlar bazalari turli xil xususiyatlarning ko'plab manbalari bo'yicha shakllantiriladi, ular orasida adabiy manbalar, ekspert baholari, nizomlar va ko'rsatmalar va boshqalar bo'lishi mumkin. Shunday qilib, bilim tizimi ekspert tizimiga qaraganda kengroq tushunchadir. Shunga ko'ra, BT imkoniyatlari ET holatiga qaraganda kengroq. ETlar tor ixtisoslashuvi tufayli, tashkilotlar ierarxiyasining quyi darajalarida qo'llaniladi.

Neyropaketlar (neyron tarmoq va neyrokompyuter texnologiyalari).

Yaponlar tasvirlarni tanish uchun neyrokompyuterlardan (NK) foydalanishda katta yutuqlarga erishdilar. NKning birinchi faol elementi deb hisoblanishi mumkin bo'lgan qurilma yaratildi. Neyrokompyuterlar (ularning boshqa nomi neyron tarmoq

tizimlari) - bu yangi avlod kompyuterlari bo'lib, ular avvalgilaridan (shu jumladan beshinchi avlod kompyuterlaridan) sifat jihatidan farq qiladi, chunki ularda oldindan yaratilgan algoritmik dasturlar mavjud emas va ular inson miyasiga o'xshab alohida misollar yordamida o'rganishga qodir.

Neyron tarmoq tizimlarining ishlashi o'z-o'zini o'qitish printsipiga asoslanadi. Neyron tarmoq kompyuteriga (NTK) bir qator o'qitiladigan misollarning kirish parametrlarini taqdim etish va neyron tarmoq tuzilmasida chiqish parametrlarini mos ravishda sozlash natijasida real vaziyatlarni tahlil qilishda echim topishni ta'minlaydigan ma'lum aloqalar o'rnatiladi. Hozirgi vaqtda NTK tasvirni aniqlash muammolarini hal qilishda keng foydalanilmoqda.

Robototexnika tizimlari (RTT) - bu turli xil ishlab chiqarish funktsiyalarini mustaqil ravishda bajarishga qodir avtomatlashtirilgan qurilmalar.

Robotlar-bu inson mehnatini avtomatlashtirish uchun mo'ljallangan texnik qurilmalar. "Robot" so'zining o'zi XX asrning 20-yillarida paydo bo'lgan.

Hozirgi vaqtda sanoatda qattiq boshqaruv sxemasi bo'yicha ishlaydigan ko'plab robot-manipulyatorlardan foydalaniladi. Ulardan farqli o'laroq, aqlli robotlar o'z-o'zini o'qitish va o'zini tashkil qilish, o'zgaruvchan muhitga moslashish qobiliyatiga ega.

Harbiy ishlarda RTT razvedka vazifalarini (shu jumladan radiatsiya, kimyoviy va biologik), muhandislik vazifalarini, parvoz va harakatni boshqarish vazifalarini va boshqalarni bajarish uchun ishlatiladi. Yaqin kelajakda razvedka, mag'lubiyat va boshqaruv funktsiyalarini birlashtirgan va jangovar tartiblarda harakat qilayotgan qo'shnlarning yoki mustaqil ravishda jangovar robotlarni amaliy o'zlashtirish mumkin. Bunday robotlar atrof-muxitga moslashish xususiyatlarga ega bo'lishi kerak.

Noravshan tizimlar - noravshan tushunchalar va bilimlarni qayta ishlash, bu bilimlar bilan ishlash va noaniq xulosalar chiqarish uchun mo'ljallangan tizimlardir. Ushbu kompyuter tizimlarini qurish metodologiyasi L. Zade tomonidan taklif qilingan noravshan to'plamlar nazariyasiga asoslanadi. SI va noravshan to'plamlar nazariyasi o'rtasidagi bog'liqlik D. Zadening "odam raqamlar bilan emas, balki

noravshan tushunchalar bilan o‘ylaydi” degan gipotezasini qabul qilsak bu tabiiy jarayondir. Hozirgi vaqtda raqamli kompyuterlarga qaraganda noravshan bilimlar bilan ishlashga qodir bo‘lgan maxsus noravshan kompyuterlar yaratilgan.

Noravshan tizimlar yangi axborot texnologiyalarida alohida guruhga loyiq emas, balki SI bilan bog‘liq barcha noravshan intellektual tizimlarga kirib boradi: noravshan ekspert tizimlari, aqlli robotlar, tasvir va nutqni aniqlash, turli maqsadlar uchun avtomatlashtirilgan tizimlar, bilimlar bazasi, avtomatik loyihalash – bular noravshan tizimlar qo‘llanilgan sohalarning to‘liq ro‘yxati emas.

Kompyuter viruslari. Bugungi kunda ushbu turdagi dasturiy mahsulotlar bilan tanishishmagan kompyuter foydalanuvchisini topish qiyin.

Viruslarning so‘nggi avlodlari sun‘iy intellekt tizimlarining barcha xususiyatlariga ega. Ular kompyuterlar bo‘ylab erkin harakatlanadilar, mutatsiyaga uchraydilar va ko‘payadilar, o‘rganadilar, parametrlari va tuzilishini o‘zgartiraolad.

Internet paydo bo‘lishi bilan kompyuter viruslarining ta’siri sezilarli darajada oshdi. Mutaxassislarning fikriga ko‘ra, bugungi kunda biz boshdan kechirayotgan muammolar kompyuter viruslarining aqlli robotlar sohasiga kirib borishi bilan bizni kutayotgan istiqbollar bilan taqqoslaganda ahamiyatsiz bo‘lib tuyuladi.

Intellektual matematik modellashtirish. Bu sun‘iy intellekt usullaridan foydalangan holda kompyuterli matematik modellashtirish. Ushbu turdagi intellektual tizimlar matematik muammolarini hal qilish bilan shug‘ullanadigan professional matematikning ijodiy faoliyatiga taqlid qiladi. Ular kerakli teoremlar, matematik bog‘liqliklar va evristik qoidalarni o‘z ichiga olgan bilim bazalariga ega.

Sun‘iy intellektning rivojlantirish yo‘nalishlari

Sun‘iy intellektning ilmiy sohasini to‘liq tavsiflovchi yagona ta’rif bugungi kungacha mavjud emas. Bugungi kunda ko‘plab nuqtai nazarlar orasida uchta ustunlik mavjud. Birinchisiga ko‘ra - SI sohasidagi tadqiqotlar fundamental tadqiqotlar bo‘lib, unda an’anaviy ravishda intellektual deb hisoblangan va ilgari rasmiylashtirish va avtomatlashtirishga bo‘ysunmagan muammolarni hal qilish modellari va usullari ishlab chiqilgan. Ikkinchi nuqtai nazarga ko‘ra, yangi yo‘nalish kompyuterda muammolarni hal qilishning yangi g‘oyalari, tubdan farq qiladigan

dasturlash texnologiyasini ishlab chiqish, kompyuter arxitekturasiga o'tish, klassik arxitekturani rad etish bilan bog'liq, va nihoyat, uchinchi nuqtai nazar, ehtimol, eng pragmatikgi bo'lib, sun'iy intellekt sohasidagi ishlar natijasida ilgari yaratilgan tizimlar yaroqsiz bo'lgan muammolarni hal qiladigan ko'plab amaliy tizimlar tushiniladi.

Albatta, bu uchta nuqtai nazarning barchasi bir-biri bilan bog'liq, SI sohasida fundamental tadqiqotlar, yangi dasturlash texnologiyasi, yangi texnik vositalar arxitekturasini rivojlanmoqda va bularning barchasi turli sohalarda ishlashga mo'ljallangan amaliy tizimlarni yaratish uchun ishlatilmoqda. Hozirgi vaqtda SI tadqiqotlarida ettita asosiy yo'nalish mavjud.

1. Bilimlarni taqdimetish. Ushbu yo'nalish doirasida intellektual tizim (IT) xotirasida bilimlarni rasmiylashtirish va taqdim etish bilan bog'liq muammolar hal qilinadi. Buning uchun bilimlarni taqdim etishning maxsus modellari va bilimlarni tavsiflash uchun tillar ishlab chiqiladi, turli xil bilimlar ajratiladi. IT bilim olishi mumkin bo'lgan manbalar o'rganilmoqda va IT uchun bilim olish mumkin bo'lgan protseduralar va texnikalar yaratilmoqda. IT uchun bilimlarni taqdim etish muammosi juda dolzarbdir, chunki IT bu tizim bo'lib, uning faoliyati uning xotirasida saqlanadigan muammoli soha haqidagi bilimlarga asoslanadi.

2. Bilimlarni manipulyatsiya qilish. Muammolarni hal qilishda bilimlardan foydalanish uchun siz ITni ular bilan ishlashga o'rgatishingiz kerak. Ushbu yo'nalish doirasida bilimlarni to'ldirish usullari ularning to'liq tavsiflari asosida quriladi, tizimlar o'rganiladi, ITda saqlanadigan bilimlar tasnifi, bilimlarni umumlashtirish va ular asosida mavhum tushunchalarni shakllantirish protseduralari ishlab chiqiladi, mavjud bilimlar asosida ishonchli va aniq xulosa chiqarish usullari yaratiladi, fikrlash modellari taklif etiladi. Bilimlarni manipulyatsiya qilish bilimlarni taqdim etish bilan chambarchas bog'liq. Ushbu ikki yo'nalishni faqat shartli ravishda ajratish mumkin deb ishoniladi. Hozirgi kunda yaratilgan bilimlar bazasi nazariyasi birinchi va ikkinchi yo'nalishlarga oid tadqiqotlarni o'z ichiga oladi.

3. **Aloqa.** Ushbu yo‘nalishdagi vazifalar doirasiga quyidagilar kiradi: izchil matnlarni tushunish muammosi, nutqni tushunish va nutq sintezi, inson va IT o‘rtasidagi aloqa modellari nazariyasi. Xuddi shu muammolar doirasiga insonning talabiga binoan paydo bo‘lishi kerak bo‘lgan IT harakatlarining tushuntirishlarini shakllantirish vazifalari, shuningdek, aloqa jarayonida olingan turli xil modallikdagi (nutq, matn, vizual va boshqalar) xabarlarni yagona ichki tasvirga birlashtirish bilan bog‘liq vazifalar kiradi. Ushbu yo‘nalishdagi tadqiqotlar asosida lingvistik protsessor, savol-javob tizimlari, dialog tizimlari va boshqa ITlarni qurish usullari shakllantiriladi, ularning maqsadi inson va IT o‘rtasidagi aloqa uchun qulay sharoitlarni ta‘minlashdir.

4. **Idrok.** Ushbu yo‘nalish an‘anaviy ravishda muammolarni o‘z ichiga oladi uch o‘lchovli sahnalarni tahlil qilish, bilimlar bazasida vizual tasvirlar to‘g‘risidagi ma‘lumotlarni taqdim etish usullarini ishlab chiqish, vizual sahnalardan ularning matnli tavsifiga o‘tish usullarini va teskari o‘tish usullarini yaratish, kognitiv grafika protseduralarini ishlab chiqish, ITdagi ichki tasvirlar asosida sahnalarni yaratish vositalarini yaratish kiradi. SI mutaxassislari vizual (majoziy) ma‘lumotlarni qayta ishlash va uning ramziy (matnli) ma‘lumotlarni qayta ishlash bilan hosil qilingan IT intellektualligini oshirishda katta imkoniyatlarga egaligini ko‘rsatmoqda.

5. **O‘qitish.** Taxminlarga ko‘ra, IT, xuddi odam kabi, o‘rganishga qodir bo‘ladi - ular ilgari uchramagan muammolarni hal qilishi mumkin. Buni amalga oshirish uchun quyidagilar zarur: muammoli vaziyatni tavsiflash yoki ushbu vaziyatni kuzatish orqali muammoning shartlarini shakllantirish usullarini yaratish; ma‘lum muammolarni (misollarni) ma‘lum echimdan umumiy muammoni hal qilishga o‘tishni o‘rganish; IT uchun boshlang‘ich muammoni kichik qisimlarga ajratish usullarini yaratish, shunda ular IT uchun allaqachon ma‘lum echimlar bilan echi inkonini beradi; o‘quv jarayonining o‘zi uchun me‘yoriy va deklarativ modellarni ishlab chiqish, taqlid xatti-harakatlari nazariyasini yaratishdir.

6. **Xulq-atvor** (o‘zini tutishi). IT ba‘zi bir atrof-muhitda harakat qilishlari kerakligi sababli, ularga atrof-muhit, boshqa ITlar va odamlar bilan etarli darajada ta‘sir o‘tkazishga imkon beradigan maxsus xulq-atvor protseduralarini ishlab

chiqish kerak. Bunday o'zaro ta'sirga erishish uchun bir qator yo'nalishlarda tadqiqotlar o'tkazish va yaratish kerak: maqsadga muvofiq xatti-harakatlar, me'yoriy xatti-harakatlar, vaziyatli xatti-harakatlar modellari, dinamik vaziyatlarda ko'p darajali rejalashtirish va rejalarni tuzatishning maxsus usullari, ushbu vaziyatlarda IT qarorlarini qabul qilish modellari.

7. Aqlli dasturlash. Ushbu yo'nalish so'nggi yillarda rivojlana boshladi. U quyidagilarni o'z ichiga oladi: intellektual dasturlash uchun tillar (mantiqiy dasturlash, ob'ektga yo'naltirilgan tillar, bilimlarni taqdim etish va semantik belgilash tillari); dasturlarning avtomatik sintezi (deduktiv va induktiv usullar); asboblari; aqlli interfeyslar; ko'p agentli texnologiyalar.

SIning sanab o'tilgan yo'nalishlari turli darajadagi tadqiqotlar va shunga mos ravishda turli amaliy natijalar bilan tavsiflanadi.

Sun'iy intellektni o'rganish zarurati

Bilishimizcha, sun'iy intellekt odamlar kabi aqlli mashinalarni yaratishga intiladi. SI ni o'rganish uchun bizda ko'plab sabablar mavjud. Sabablari esa quyidagicha: -

Sun'iy intellekt o'zi ma'lumotlar yordamida o'rganishi mumkin.

Kundalik hayotimizda biz juda katta miqdordagi ma'lumotlar bilan ishlaymiz, va inson miyasi bunday miqdordagi ma'lumotlarni kuzatib borishga qodir emas. Shuning uchun hammasini avtomatlashtirishimiz kerak. Avtomatlashtirish uchun esa sun'iy intellektni o'rganishimiz zarur, chunki u ma'lumotlar asosida o'rganishi va takrorlanuvchi vazifalarni aniq va charchamasdan bajarishi mumkin..

Sun'iy intellekt o'zini o'zi o'rgatishi mumkin

Tizimning o'zini o'zi o'rgatishi juda zarur, chunki ma'lumotlarning o'zi doimo o'zgarib turadi va bunday ma'lumotlardan olingan bilimlar doimo yangilanib turishi kerak. Ushbu maqsadga erishish uchun biz SI dan foydalanishimiz mumkin, chunki sun'iy intellektga ega tizim o'zini o'zi o'rgatishi mumkin.

Sun'iy intellekt real vaqt rejimida javob berishi mumkin.

Sun'iy intellekt neyron tarmoqlari yordamida ma'lumotlarni yanada chuqurroq tahlil qilishi mumkin. Ushbu qobiliyat tufayli sun'iy intellekt sharoitlarga

asoslangan holatlarni real vaqt rejimida fikrlash va ularga javob berish imkoniyatiga ega.

Sun'iy intellekt yuqori aniqlikni ta'minlaydi.

Sun'iy intellekt chuqur neyron tarmoqlari yordamida juda yuqori aniqlikka erishishi mumkin. SI tibbiyot sohasida bemorlarning MRT natijalari asosida saraton kabi kasalliklarni aniqlashda yordam beradi.

Sun'iy intellekt ma'lumotlarni tartibga solib, ulardan maksimal foyda olish imkonini beradi.

Ma'lumotlar o'z-o'zini o'rganuvchi algoritmlardan foydalanadigan tizimlarning intellektual mulki hisoblanadi. Bizga ma'lumotlarni indekslash va tartibga solish uchun sun'iy intellekt kerak, shunda u har doim eng yaxshi natijalarni taqdim eta oladi.

Intellektni tushunish

Sun'iy intellekt yordamida aqlli tizimlar yaratish mumkin. Buning uchun biz intellekt konsepsiyasini tushunishimiz kerak, shunda miyamiz o'ziga o'xshash boshqa bir aqlli tizimni yaratishi mumkin bo'ladi.

Aql-idrok nima?

Tizimning hisoblash, mantiqiy fikrlash, o'zaro bog'liqliklar va analogiyalarni idrok etish, tajribadan o'rganish, xotiradan ma'lumotlarni saqlash va chiqarish, muammolarni hal qilish, murakkab g'oyalarni anglash, tabiiy tilni erkin ishlatish, tasniflash, umumlashtirish va yangi vaziyatlarga moslashish qobiliyati.

Aql-idrok turlari

Amerikalik rivojlanish psixologi G'ovard Gardnerning ta'rifiga ko'ra, intellekt bir nechta sohalarda namoyon bo'ladi.

№	Aql-idrok va tavsif	Misol
1	Lingvistik intellekt Fonologiya (nutq tovushlari), sintaksis (grammatika) va semantika (ma'no) mexanizmlarini gapirish, tan olish va ulardan foydalanish qobiliyati.	Hikoyachilar, ma'ruzachilar
2	Musiqiy intellekt Ovozli ma'nolarni yaratish, muloqot qilish va tushunish, balandlikni, ritmni tushunish qobiliyati.	Musiqachilar, qo'shiqchilar, bastakorlar

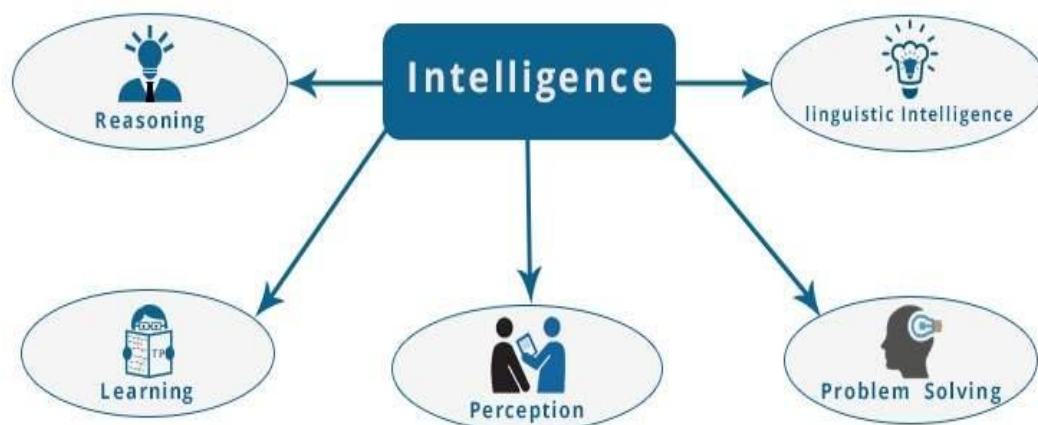
3	Mantiqiy-matematik intellekt Harakatlar yoki ob'ektlar yo'qligida munosabatlardan foydalanish va tushunish qobiliyati. Bu, shuningdek, murakkab va mavhum g'oyalarni tushunish qobiliyatidir.	Matematiklar, olimlar
4	Fazoviy intellekt Vizual yoki fazoviy ma'lumotni idrok etish, uni o'zgartirish va ob'ektlarga murojaat qilmasdan vizual tasvirlarni qayta yaratish, 3D tasvirlarni yaratish, shuningdek ularni ko'chirish va aylantirish qobiliyati.	Kartograflar, kosmonavtlar, fiziklar
5	Tana-kinestetik intellekt Muammolarni hal qilish yoki moda mahsulotlarini yaratish, nozik va qo'pol motorli ko'nikmalarni nazorat qilish va ob'ektlarni boshqarish uchun butun tanani yoki tananing qismlarini ishlatish qobiliyati.	O'yinchilar, raqqoslar
6	Intrapersonal intellekt O'z his-tuyg'ularini, niyatlarini va motivlarini farqlash qobiliyati.	Gautam Budda
7	Shaxslararo intellekt Boshqa odamlarning his-tuyg'ularini, e'tiqodlarini va niyatlarini tan olish va farqlash qobiliyati.	Ommaviy aloqa mutaxassislari, intervyu oluvchilar

Mashina yoki tizimda sun'iy intellekt bor deb aytish mumkin, agar u kamida bitta yoki barcha intellekt elementlari bilan ta'minlangan bo'lsa.

Aql-idrok nimadan iborat?

Intellekt nomoddiydir. U quyidagilardan iborat:

- Mulohaza yuritish (Reasoning)
- Ta'lim (Learning)
- Muammoni hal qilish (Problem Solving)
- Idrok (Perception)
- Lingvistik intellekt (Linguistic intelligence)



Mulohaza yuritish - Bu qarorlar qabul qilish, baholash va bashoratlash uchun asos yaratadigan jarayonlar to'plamidir. Umuman olganda, ikki turdagi jarayonlar mavjud:

Induktiv fikrlash	Deduktiv fikrlash
U keng umumiy xulosalar chiqarish uchun aniq kuzatuvlar olib boradi.	U umumiy bayonot bilan boshlanadi va aniq mantiqiy xulosaga kelish imkoniyatlarini tekshiradi.
Agar bayonotning barcha asoslari to'g'ri bo'lsa ham, induktiv fikrlash xulosaning noto'g'ri bo'lishiga imkon beradi.	Agar biror narsa umuman narsalar sinfi uchun to'g'ri bo'lsa, bu sinfning barcha a'zolari uchun ham to'g'ri keladi.
Misol - "Nita - o'qituvchi. Nita tirishqoq. Shuning uchun barcha o'qituvchilar mehnatsevar".	Misol - "60 yoshdan oshgan barcha ayollar buvilar. Shalini 65 yoshda. Shuning uchun Shalini buvisi."

Ta'lim olish (o'rganish)

O'qish qobiliyatiga odamlar, ayrim hayvonlar va sun'iy intellekt bilan ta'minlangan tizimlar ega. O'qish quyidagi toifalarga bo'linadi: -

Eshitish orqali o'qish - Bu eshitish bilan o'rganish. Masalan, talabalar ma'ruzalarning audio yozuvlarini tinglaydilar.

Epizodik ta'lim - Hodisalar ketma-ketligini kuzatish yoki boshdan kechirish orqali o'rganish bu bilimlarni chiziqli va tartiblangan shaklda eslab qolish jarayonidir.

Motorik o'rganish - Bu mushaklarning aniq harakatlari orqali o'rganishdir. Masalan, narsalarni yig'ish, yozish va hokazo.

Kuzatish orqali o'rganish - Kuzatish va taqlid qilish orqali o'rganish — bu boshqalarni kuzatib, ularning xatti-harakatlariga taqlid qilish orqali yangi bilim va ko'nikmalarni o'zlashtirish jarayonidir. Masalan, bola ota-onasining harakatlarini kuzatib, ularning gapirish, yurish yoki ma'lum bir faoliyatni bajarish usullarini o'rganadi.

Idrokni o'rgatish - Bu o'rganish turi avval ko'rilgan stimullarning tanib olishiga asoslangan bo'lib, inson yoki tizim ilgari kuzatilgan obyektlar yoki vaziyatlarni aniqlash va tasniflashga o'rganadi

Relations o'rganish -Bu o'rganish turi stimullarni nisbiy xususiyatlari asosida ajratishni o'z ichiga oladi, ya'ni stimullarning mutlaq emas, balki nisbiy

o'zgarishlarini tushunish va farqlashga asoslanadi. Masalan, agar biror taomni tayyorlashda oldingi marta juda tuzli bo'lsa, bu safar kartoshka tayyorlashda tuzni biroz kamroq qo'shish — bu nisbiy o'rganish jarayonidir.

- **Fazoviy ta'lim** - masalan, tasvirlar, ranglar, xaritalar va boshqalar yordamida o'rganish jarayonidir. Masalan, inson avval yo'lni xaritada ko'rib, uni aqliy tarzda rejalashtirib, keyin esa haqiqatda shu yo'l bo'ylab harakatlanishi mumkin
- **Rag'batlantiruvchi ta'lim** - bu ma'lum bir qo'zg'atuvchining mavjudligida muayyan xatti-harakatni o'rganishdir. Misol uchun, it eshik qo'ng'irog'ini eshitganida qulog'ini ko'taradi.

Muammoni hal qilish

Bu inson ma'lum yoki noma'lum to'siqlar bilan to'sib qo'yilgan yo'lni tanlash orqali mavjud vaziyatdan kerakli echimni idrok etadigan va unga erishishga harakat qiladigan jarayondir. Muammoni hal qilish, shuningdek, **qaror qabul qilishni** o'z ichiga oladi, bu istalgan maqsadga erishish uchun muqobillar to'plamidan eng mos variantni tanlash jarayonidir.

Idrok

Bu sensorli axborotni qabul qilish, izohlash, tanlash va tartibga solish jarayonidir. Idrok **anglashni** nazarda tutadi. Odamlarda idrokga sezgilar yordam beradi. SI sohasida idrok mexanizmi sensorlar orqali olingan ma'lumotlarni mazmunli tarzda birlashtiradi.

Lingvistik intellekt

Bu insonning og'zaki va yozma tillardan foydalanish, tushunish, gapirish va yozish qobiliyatidir. Bu shaxslararo muloqotda muhim ahamiyatga ega.

Bilimlarni taqdim etish modellari

Bilimlarni taqdim etish modellari - bu sun'iy intellekt sohasidagi eng muhim tadqiqot yo'nalishlaridan biridir. Nima uchun eng muhimlaridan biri? Chunki bilimsiz sun'iy intellekt printsiptial ravishda mavjud bo'lolmaydi. Darhaqiqat, mutlaqo hech narsani bilmaydigan odamni tasavvur qiling.

Bilimlarni taqdim etish modellari odatda quyidagilarga bo‘linadi **mantiqiy** (rasmiy) va **evristik** (rasmiylashtirilgan). Bilimlarni taqdim etishning mantiqiy modellari rasmiy tizim (nazariya) tushunchasiga asoslanadi. Rasmiy nazariyalarga predikatlarni hisoblash va har qanday aniq ishlab chiqarish tizimi misol bo‘la oladi. Mantiqiy modellar, qoida tariqasida, bir qator evristik strategiyalar bilan to‘ldirilgan birinchi darajali predikatlarni hisoblashdan foydalanadi. Ushbu usullar deduktiv tipdagi tizimlardir, ya’ni ular belgilangan chiqish qoidalari tizimidan foydalangan holda ma’lum bir yo‘naltiruvchi tizimidan chiqishni olish modelidan foydalanadilar. Predikat tizimlarining keyingi rivojlanishi induktiv tipdagi tizimlar bo‘lib, unda xulosa chiqarish qoidalari cheklangan miqdordagi o‘quv misollarini qayta ishlash asosida tizim tomonidan yaratiladi.

Bilimlarni taqdim etishning mantiqiy modellarida individual bilim birliklari o‘rtasida mavjud bo‘lgan munosabatlar faqat ishlatilgan rasmiy tizimning sintaktik qoidalari bilan ta’minlangan kambag‘al vositalar yordamida ifodalanadi.

Rasmiy modellardan farqli o‘laroq, evristik modellar ma’lum bir muammoli sohaning o‘ziga xos xususiyatlarini uzatuvchi turli xil vositalarga ega. Shuning uchun evristik modellar mantiqiy modellardan ustundir, agar iloji bo‘lsa, muammoli muhitni etarli darajada taqdim eting va ishlatilgan xulosa qoidalarining samaradorligi. Ekspert tizimlarida ishlatiladigan evristik modellarga tarmoq, ramka, ishlab chiqarish va ob’ektga yo‘naltirilgan modellar kiradi. Shuni ta’kidlash kerakki, ekspert tizimlarida bilimlarni namoyish etish uchun ishlatiladigan ishlab chiqarish modellari rasmiy ishlab chiqarish tizimlaridan farq qiladi, chunki ular yanada murakkab qoidalar dizaynidan foydalanadilar, shuningdek, ko‘pincha semantik tuzilmalar shaklida ifodalangan muammoli muhitning o‘ziga xos xususiyatlari to‘g‘risida evristik ma’lumotlarni o‘z ichiga oladi. Ekspert tizimlari uchun bilimlarni taqdim etishning asosiy modellariga quyidagilar kiradi:

- mantiqiy;
- funktsional;
- produksion;
- semantik tarmoqlar;

- freymilar.

Hisob-kitoblar predikatlar mantiq'i bilan ifodalangan bilimlarni ifodalash modellari mantiqiy, hisob-kitoblarni tasvirlash uchun ishlatiladigan dasturlash tili esa mantiqiy til deb ataladi. **Mantiqiy modellar** asosda *rasmiy tizim* kontseptsiya yotadi. Bilimlarni ifodalash uchun ishlatiladigan eng keng tarqalgan rasmiy tizim birinchi darajali predikatlarni hisoblashdir.

Predikatlarni hisoblash-bu rasmiy til bo'lib, unda bilimlar HAQIQAT yoki YOLG'ON ma'nolariga ega bo'lishi mumkin bo'lgan mantiqiy bayonotlar shaklida taqdim etiladi. Deduktiv tizimlarda predikatlarni hisoblash asosida qurilgan teoremlarni isbotlash, bilimlar universal xulosa qoidalarining qat'iy to'plami bilan to'ldirilgan ko'plab mantiqiy bayonotlar shaklida taqdim etiladi - predikatlarni hisoblash nuqtai nazaridan har doim adolatli bo'lgan yo'naltiruvchi va xulosalar o'rtasidagi munosabatlar to'g'risida eng umumiy xarakterdagi bayonotlar. Ushbu tizimlarda isbotlashni talab qiladigan bayonotlar berilgan haqiqiy bayonotlar to'plamiga asoslanib tasdiqlanishi yoki rad etilishi kerak bo'lgan teoremlar sifatida qoraladi. Mantiqiy tizimlarning muhim tarkibiy qismi xulosa qoidalarini qo'llash strategiyasidir: zamonaviy deduktiv tizimlarning ishlashi, qoida tariqasida, J.Robinson yoki uning modifikatsiyalaridan biri.

Predikatlarni hisoblashning asosiy afzalligi uning universalligi va deklarativligidir. Ushbu tilda ifodalangan bayonotlar ma'nosini o'zgartirmasdan kompyuterda ishlov berish uchun qulay shaklga aylantirilishi mumkin. Shu bilan birga, ushbu rasmiyatchilik muhim kamchiliklarga ham ega: protsessual va muammoli bilimlarni predikatlarni hisoblash asosida qurilgan tizimlarga kiritish qiyin, ular bilimlarning nomuvofiqligi bilan yaxshi kurashmaydilar va hokazo. Predikatlarni hisoblashning ko'rsatilgan kamchiliklari tufayli boshqa mantiqiy yondashuvlar ishlab chiqildi, masalan, induktiv tizimlar, bunda xulosa chiqarish qoidalari tizim tomonidan cheklangan miqdordagi o'quv misollarini qayta ishlash asosida yaratiladi, ko'p qiymatli yoki modal mantiqlardan foydalangan holda quriladi. Psevdofizik mantiqlar ham ishlab chiqilgan bo'lib, ularning xususiyati muammoli sohada amalga oshiriladigan munosabatlarning xususiyatlari to'g'risida

aniq bilimlarni olish qoidalarida foydalanish va lingvistik o'zgaruvchilar apparatidan foydalanishdir.

Mantiqiy dasturlashning "klassik" modeli atamalar, atom, dastur va so'rov tushunchalari bilan tavsiflanadi. Term - bu $f(t)$ shaklidagi o'zgaruvchi, doimiy yoki aralash $f(t_1, \dots, t_n)$, bu erda f - n -li funktsional belgisi va $t_1, \dots, t_n$ - termalar. Doimiylarni noular funktsional belgilar sifatida tushunish qulay. Atom formulasi(atom) $w(t_1, \dots, t_n)$, bu erda w - n -li predikat belgisi (munosabatlar nomi) va $t_1, \dots, t_n$ - termalar.

Komp'yuter dastur - bu ko'plab qoidalaridir. Qoida quyidagi shaklga ega:

$$A_0 \vdash A_1, \dots, A_m \ (m > 0),$$

bunda $A_0, A_1, \dots, A_m$ - atomlar. Atom A_0 sarlavha deb ataladi va $A_1, \dots, A_m$ - tana qoidalari. Tana bo'sh bo'lishi mumkin ($m=0$ da). Bunday qoidalar faktlar deb ataladi. Agar atom bo'lsa A_0 qoida sarlavhasida $w(t_1, \dots, t_n)$, keyin ular bu predikat belgisi w uchun qoida deb aytishadi. So'rov (maqsad) quyidagi shaklga ega:

$$\vdash C_1, \dots, C_r,$$

bu erda $r > 0$ va $C_1, \dots, C_r$ - atomlar.

Klassik modelning paydo bo'lishi Prolog tiliga qarzdor bo'lib, u tezda butun dunyoda mashhurlikka erishdi va hozirda sun'iy intellektning asosiy tillaridan biri bo'lib xizmat qilmoqda.

Soxani semantikasini modellashtirish nuqtai nazaridan mantiqiy model soxa qonunlarini aks ettiruvchi klassik bo'lmagan mantiqlarni modellashtirish, ishlab chiqarish qoidalari va evristikani modellashtirish uchun qulaydir. Mantiqiy tillarning operatsion semantikasi rasmiy hisob-kitoblarda dalillarni yaratishning mantiqiy protsedurasidir, bu esa bunday modellar uchun muammoni juft (shartlar, maqsad) deb o'ylashga imkon beradi, bu erda shartlar aksiomalar to'plami va maqsad teorema. Ushbu yo'nalishning amaliy tizimlarining rivojlanishi har bir foydalanuvchi muammosini hal qilish jarayonida amalga oshiriladigan mantiqiy xulosaning katta murakkabligi, protsessual va mantiqiy modellarni birlashtirishning murakkabligi, modellashtirish formalizmi sifatida klassik mantiqning kamchiliklari (bitta xato printsipi va boshqalar) bilan cheklanadi.

Bilimlar bazasi funktsiyalarning ko‘plab ta’riflari sifatida qaraladigan bilimlarni taqdim etish modeli **funktsional model** deyiladi. Ushbu model tenglik tiliga va tenglikni qisqartirish qoidalari yoki atamalarni qayta yozish qoidalari sifatida operatsion tushunishga asoslangan. Tenglik - bu $p=q$ maxsus shakldagi formulalar, bu erda p va q -atamalar. Termalar deganda funktsiyalar, o‘zgaruvchilar va doimiylarning (masalan, $f(x)=x^2-x+5$) belgilaridan (nomlaridan) sintaktik jihatdan to‘g‘ri tarzda tuzilgan iboralar tushuniladi. Dastur juftlik (E,T) sifatida paydo bo‘ladi, bu erda E - tengliklarning cheklangan to‘plami, T - asl atama va ushbu dastur tomonidan ishlab chiqarilgan hisoblash jarayoni iloji boricha asl atamani takroriy qayta yozishdan iborat. Ikkinchi holda, ular atamaning normal holatga qaytishi haqida gapirishadi normal shakl hisob-kitoblarning yakuniy natijasi sifatida qaraladi.

Turli xil matematik ma’nolar tenglikka kiritilishi mumkin. Tengliklarni, masalan, ob’ektlarning xususiyatlarini tavsiflovchi bayonotlar yoki tenglamalar sifatida ko‘rish mumkin. E’tibor bering, ikkinchi holda, funktsiyalar noma’lum deb hisoblanishi mumkin, keyin tenglik funktsiyalarning ta’riflari vazifasini bajaradi. Ma’nolarning bunday xilma-xilligi tenglik tilining tavsiflovchi semantikalarining xilma-xilligini keltirib chiqaradi. Agar biz formulaning berilgan aksiomalar to‘plamidan kelib chiqishi yoki uning tegishli matematik tuzilishdagi haqiqati (model-nazariy semantika) bilan qiziqsak, tengliklarga ba’zi mantiqiy (teng) hisob-kitoblarning formulalari sifatida qarash mumkin.

Funktsional modellar orasida eng mashhuri turli xil dasturlash tillarini qurish uchun ishlatiladigan bestip lambda hisobidir.

Soxa semantikasini modellashtirish nuqtai nazaridan funktsional model ham operatsion, ham tavsiflovchi semantikaga ega: funktsional dasturni hisoblash retsepti va kerakli ob’ektning konstruktiv ta’rifi sifatida ko‘rish mumkin. Protседurali dasturlashdan farqli o‘laroq, funktsional dasturlash nazariy va dalillarga asoslangan kelib chiqishga ega. Funktsional yondashuv bilan domen modeli, qoida tariqasida, rekursiv funktsiyalarning ta’riflari to‘plami bilan ifodalanadi. Protsessual modelda bo‘lgani kabi, dastur bitta muammoni hal qilish uchun so‘rovdur.

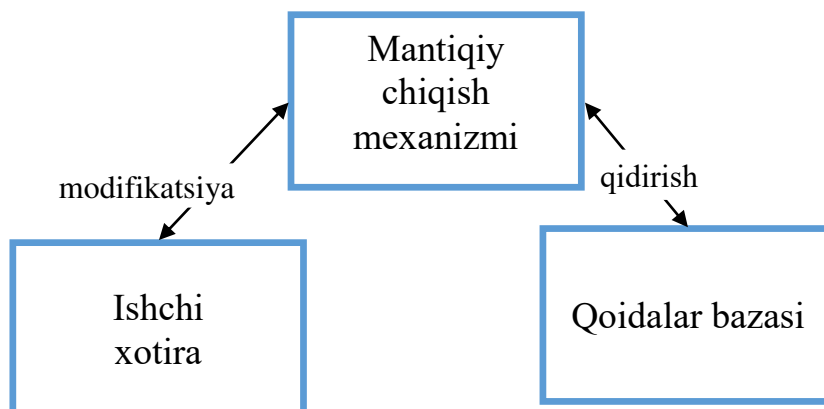
Funksional dasturlashning quyidagi xususiyatlari jozibali:

- funksional dasturlash tillari protsessual "qanday tillar" dan farqli o'laroq "nima tillar"ga tegishli;
- funksional dasturlash havolalar (funktсионallik) orqali shaffoflik xususiyatiga ega, ya'ni tilning har bir ifodasi hisoblash yoki turli funktsiyalar bilan bo'lishish orqali o'zgartirilishi mumkin bo'lmagan yagona miqdorni belgilaydi; funksional tillarda "nojo'ya ta'sirlar" yo'q - hisoblash jarayonida o'zgaruvchilar qiymatlarining dinamik o'zgarishi;
- funksional dasturlar, qoida tariqasida, protsessual hamkasblariga qaraganda qisqaroq bo'lib, yuqori darajadagi mavhumlik va tushunish imkoniyatiga ega;
- funksional dasturlar ularni optimallashtirish uchun rasmiy transformatsiyalar uchun javob beradi;
- funksional dasturlash tizimlari tabiiy parallellashtirishga imkon beradi va parallel kompyuterlarda amalga oshirilishi mumkin.

Produksino (maxsulot, natija) modeli, yoki qoidalarga asoslangan model sizga bilimlarni quyidagi jumlar shaklida taqdim etishga imkon beradi:

AGAR A UNDA B

Implikasiya oddiy mantiqiy ma'noda mantiqiy ergashish belgisi sifatida talqin qilinishi mumkin B haqiqiy A dan. ishlab chiqarishning boshqa talqinlari ham mumkin, masalan $A \rightarrow B$ harakatini amalga oshirish uchun zarur bo'lgan ba'zi shartlarni tavsiflaydi.



Agar ma'lum bir mahsulot to'plami tizim xotirasida saqlansa, ular *mahsulotlar tizimi* hosil qiladi. Mahsulotlar tizimida mahsulotlarni boshqarishning maxsus protseduralari belgilanishi kerak, ular yordamida mahsulotlar yangilanadi va ma'lum bir mahsulot aktuallashtirilganlar orasidan amalga oshiriladi.

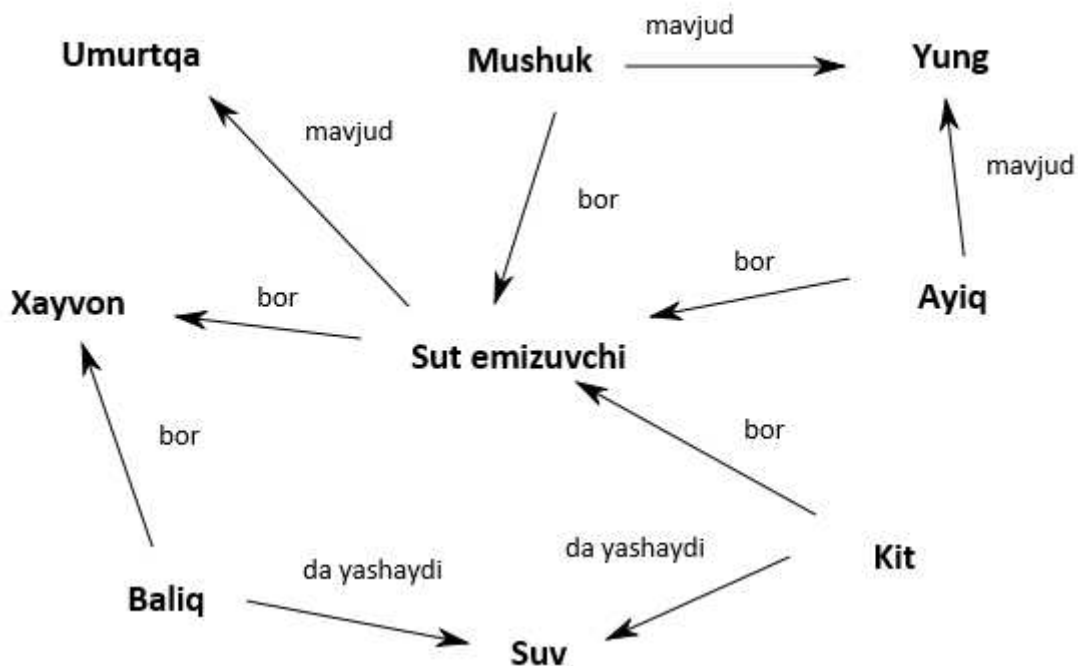
Mahsulotlar tizimiga qoidalar (mahsulotlar) bazasi, global ma'lumotlar bazasi va boshqaruv tizimi kiradi. Qoidalar bazasi – bu xotira sohasi bo'lib, unda if-then turidagi qoidalar ko'rinishidagi bilimlar to'plami mavjud. Global ma'lumotlar bazasi-haqiqiy ma'lumotlarni (faktlarni) o'z ichiga olgan xotira maydoni. Boshqaruv tizimi qoidalar bazasi va ma'lumotlar bazasidan foydalangan holda xulosalar chiqaradi. Xulosalarni shakllantirishning ikki yo'li mavjud – to'g'ridan-to'g'ri xulosalar va teskari xulosalar.

To'g'ridan-to'g'ri xulosalarda ma'lumotlar bazasidagi ma'lumotlar elementlaridan biri tanlanadi va agar ushbu element taqqoslanganda qoidaning chap tomoniga (posilkaga) mos keladigan bo'lsa, unda tegishli xulosa qoidadan chiqariladi va ma'lumotlar bazasiga joylashtiriladi yoki qoida bilan belgilangan amal bajariladi va ma'lumotlar bazasi tarkibi mos ravishda o'zgartiriladi.

Teskari xulosalarda jarayon belgilangan maqsaddan boshlanadi. Agar ushbu maqsad qoidaning o'ng tomoniga (xulosa) mos keladigan bo'lsa, u holda qoidaning asosi sub-maqsad yoki gipoteza sifatida qabul qilinadi. Ushbu jarayon sub-maqsadning ma'lumotlar bilan mos kelishiga qadar takrorlanadi.

Mahsulot modelidagi mahsulotlarning ko'pligi bilan mahsulot tizimining izchilligini tekshirish murakkablashadi, ya'ni, ko'plab qoidalar. Shu sababli, zamonaviy SI tizimlari ishlaydigan mahsulotlar soni, qoida tariqasida, mingdan oshmaydi.

Semantik tarmoq - bu yo'naltirilgan grafik, uning tepalari tushunchalar, yoylari esa ular orasidagi munosabatlardir. Atamasemantik semantik degan ma'noni anglatadi va semantikaning o'zi ramzlar va ular belgilaydigan ob'ektlar o'rtasidagi munosabatlarni o'rnatadigan fan, ya'ni, belgilarning ma'nosini belgilaydigan fan.



Tushunchalar odatda mavhum yoki aniq ob'ektlar paydo bo'ladi va aloqalar esa - bu quyidagilar: "bu" ("is"), "qism bor" ("has part"), "tegishli", "sevadi". Semantik tarmoqlarning o'ziga xos xususiyati uch turdagi munosabatlarning majburiy mavjudligi:

- sinf-sinf elementi;
- xossasi-qiyamat;
- sinf elementiga misol.

Semantik tarmoqlarning bir nechta tasniflarini kiritish mumkin. Masalan, munosabatlar turlari soni bo'yicha:

- bir hil (munosabatlarning yagona turi bilan);
- bir xil bo'lmagan (har xil turdagi munosabatlar bilan).

Munosabatlar turlari bo'yicha:

- binar (unda munosabatlar ikkita ob'ektni bog'laydi);
- juftlik (ikkitadan ortiq tushunchalarni bog'laydigan maxsus munosabatlar mavjud).

Ko'pincha semantik tarmoqlarda quyidagi munosabatlar qo'llaniladi:

- aloqalar turi "qism-butun" ("sinf-kichik sinf", "element-to'plam" va boshqalar);

- funksional aloqalar (odatda "ishlab chiqaradi", "ta'sir qiladi"fe'llari bilan belgilanadi...);
- miqdoriy (ko'proq, kamroq, teng...);
- fazoviy (uzoq, yaqin, orqada, pastda, yuqorida...);
- vaqtinchalik (oldin, keyinroq, davomida...);
- atribut aloqalari (mulkka ega bo'lish, borma'nosi...);
- mantiqiy aloqalar (va, yoki, emas) va boshqalar.

Tabiiy tillarni tavsiflash uchun ishlatiladigan semantik tarmoqlar "agent", "ob'ekt", "manba", "qabul qiluvchi", "vaqt", "joy", "maqsad" kabi yoylardan foydalanadi. Agent-bu harakatning tashabbuskori bo'lgan ob'ekt. Ob'ekt-harakatga duchor bo'lgan ob'ekt. Manba-harakatdan oldin ob'ektni joylashtirish. Qabul qilgich-harakatdan keyin ob'ektni joylashtirish. Vaqt-harakatni bajarish vaqti. Joy-bu harakat joyi. Maqsad-boshqa hodisaning harakati.

Semantik tarmoq kabi bilimlar bazasida echim topish muammosi berilgan savolga mos keladigan ba'zi bir pastki tarmoqqa mos keladigan tarmoq qismini topish vazifasiga kamayadi.

Semantik tarmoqlarni amalga oshirish uchun maxsus tarmoq tillari mavjud, masalan, NET va boshqalar. Semantik tarmoqlardan bilimlarni ifodalash tili sifatida foydalanadigan ekspert tizimlari keng ma'lum PROSPECTOR, CASNBT, TORUS.

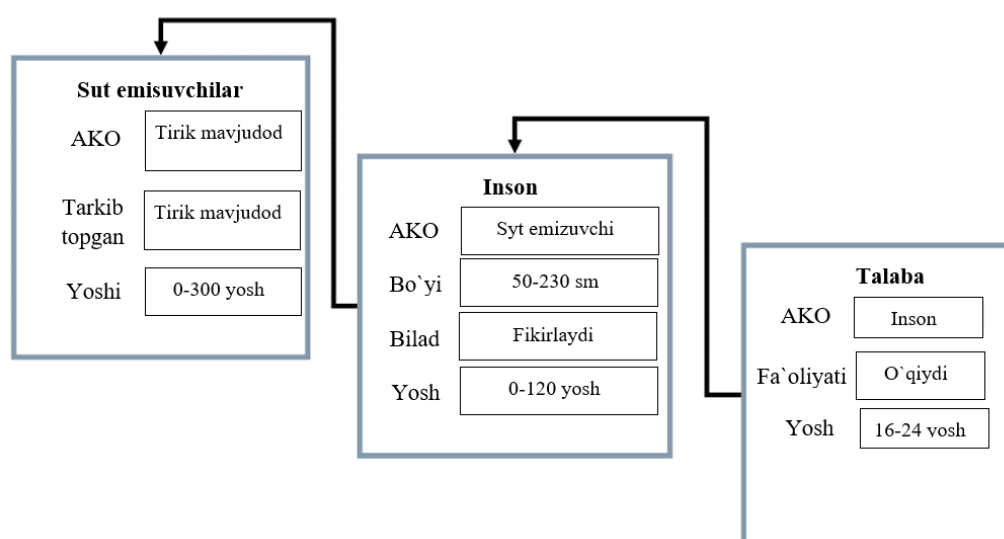
Semantik tarmoqning bir qator afzalliklarini ta'kidlash kerak:

- tushunchalar va hodisalarning tavsifi tabiiy tilga juda yaqin darajada amalga oshiriladi;
- tarmoqning turli qismlarini ulash imkoniyati ta'minlanadi;
- tushunchalar va hodisalar o'rtasidagi munosabatlar juda kichik va yaxshi rasmiylashtirilgan to'plamni hosil qiladi;
- ma'lumotlar va bilimlar bo'yicha har bir operatsiya uchun ushbu so'rovda zarur bo'lgan semantik xususiyatlarni qamrab oladigan barcha semantikani (yoki barcha bilimlarni), uning ba'zi bir qismini ifodalovchi to'liq tarmoqdan ajratish mumkin.

Modelning kamchiliklari semantik tarmoqda chiqishni izlashning murakkabligi.

Ko‘plab tushunchalarni o‘z ichiga olgan murakkab semantik tarmoqlarda tugunlarni yangilash jarayoni va ular orasidagi aloqalarni boshqarish qiyinlashadi. Shu bilan birga, tushunchalar orasidagi bilvosita umumiy aloqalar soni keskin oshadi.

Bilimlarni taqdim etishda **freymli yondashuvining** asosiy g‘oyasi shundaki, kontsepsiya yoki vaziyatga tegishli bo‘lgan hamma narsa "tarmoq orqali xiralashmaydi", balki ramkada taqdim etiladi.



Freym (Ramka) -bu vaziyatning xususiyatlari va ularning ma'nolaridan iborat bo'lgan tushuncha yoki vaziyatni tavsiflash uchun tuzilma.

Ramkani o'ziga xos xususiyatlarning butun to'plami bilan tushunchalarni tavsiflash uchun mo'ljallangan semantik tarmoqning bir qismi deb hisoblash mumkin. Grafik jihatdan ramka modeli semantik tarmoqqa o'xshash tarzda tasvirlangan, ammo asosiy farq shundaki, ramka tizimidagi tugun ramka shakliga ega, uning tuzilishini quyidagicha ifodalash mumkin:

FREYM NOMI :

(1-uyaning nomi: 1-uyaning ma'nosi),

(2-uyaning nomi: 2-uyaning ma'nosi),

(n-uyaning nomi: n-uyaning ma'nosi).

Xuddi shu yozuvni jadval shaklida taqdim etish mumkin, uni yana ikkita ustun bilan to'ldirish mumkin. Qo'shimcha ustunlar slotning turini va ramka nazariyasida ruxsat etilgan maxsus protseduralarning u yoki bu slotga biriktirilishini tavsiflash uchun mo'ljallangan. Boshqa freymning nomi slotning qiymati sifatida harakat qilishi mumkin; shunday qilib, freymlar tarmoqlari hosil bo'ladi.

Har bir uyaga bir yoki bir nechta protseduralar ulanishi mumkin, ular uyalar qiymatlari o'zgarganda amalga oshiriladi.

Ko'pincha protseduralar uyalar bilan bog'liq:

1. AGAR- QO'SHILSA. (yangi ma'lumotlar uyaga joylashtirilganda amalga oshiriladi);
2. AGAR - O'CHIRILSA. (ma'lumotlar uyadan o'chirilganda amalga oshiriladi);
3. AGAR - KERAK BO'LSA. (uyadan ma'lumot so'ralganda va u bo'sh bo'lganda amalga oshiriladi).

Ushbu protseduralar ushbu tugunga ma'lumotlarning biriktirilishini kuzatishi va qiymat o'zgarganda tegishli harakatlar amalga oshirilishini tekshirishi mumkin.

Ajratishnamunaviy ramkalar, yoki prototiplar, bilimlar bazasida saqlanadi vanamunaviy ramkalar, kiruvchi ma'lumotlar asosida Real vaziyatlarni ko'rsatish uchun yaratilgan. Frame modeli juda universaldir, chunki u dunyo haqidagi barcha bilimlarni namoyish etishga imkon beradi:

- freymlar-tuzilmalar, ob'ektlar va tushunchalarni belgilash uchun (kredit, garov, veksel);
- freym-rol (menejer, kassir, mijoz);
- freym-senariylari (bankrotlik, aktsiyadorlar yig'ilishi, ism kunini nishonlash);
- freym-vaziyatlari (signal, baxtsiz hodisa, qurilmaning ish rejimi) va boshqalar.

FREYMLar nazariyasining eng muhim xususiyati semantik tarmoqlar nazariyasidan olingan xususiyatlarni meros qilib olishdir. Ham freymlarda, ham semantik tarmoqlarda meros quyidagicha bo'ladi AKO-aloqalar (A-Kind-Of = bu). AKO uyasi ierarxiyaning yuqori darajadagi ramkasiga ishora qiladi, u erdan ular

bilvosita meros qilib olinadi, ya'ni shunga o'xshash uyalarning qiymatlari ko'chiriladi.

Bilimlarni namoyish etish modeli sifatida ramkalarning asosiy afzalligi-bu inson xotirasini tashkil etishning kontseptual asoslarini, shuningdek, moslashuvchanlik va ko'rinishni aks ettirish qobiliyatidir.

FRL (Frame Representation Language) ramka tarmoqlarida bilimlarni taqdim etishning maxsus tillari va boshqalar sanoat esni samarali qurishga imkon beradi. ANALYST, MODIS kabi freymga yo'naltirilgan ekspert tizimlari keng tarqalgan.

Sun'iy intellekt bilan nima bog'liq?

Sun'iy intellekt - bu keng tadqiqot sohasi. Ushbu tadqiqot sohasi haqiqiy dunyo muammolariga yechim topishga yordam beradi. Keling, sun'iy intellektning turli sohalarini ko'rib chiqaylik.

Mashinali o'rganish - Bu SIning eng mashhur sohalaridan biridir. SI asosiy kontseptsiyasi odamlar tajribadan o'rganishlari uchun ma'lumotlarga asoslangan mashinali o'rganishdir. Unda noma'lum ma'lumotlar bo'yicha bashorat qilish uchun ishlatilishi mumkin bo'lgan o'quv modellari mavjud.

Mantiqiy - Bu kompyuter dasturlarini bajarish uchun matematik mantiqdan foydalaniladigan yana bir muhim tadqiqot sohasidir. Unda naqsh solishtirish, semantik tahlil qilish va hokazolarni bajarish uchun qoidalar va faktlar mavjud.

Qidiruv - Bu bilimlar sohasi asosan shaxmat, krestiklar va nollar kabi o'yinlarda qo'llaniladi. Izlash algoritmlari izlash fazosida barcha imkoniyatlarni tekshirib chiqib, optimal yechimni taqdim etadi.

Sun'iy neyron tarmoqlari -Bu samarali hisoblash tizimlari tarmog'i bo'lib, uning markaziy mavzusi biologik neyron tarmoqlariga o'xshashlikdan olingan. **Sun'iy neyron tarmoqlari** (SNT) robototexnika, nutqni tanib olish, nutqni qayta ishlash va boshqa sohalarda qo'llanilishi mumkin.

Genetik algoritm - Genetik algoritmlar bir nechta dastur yordamida muammolarni hal qilishda yordam beradi. Natija eng mosini tanlash imkonini beradi.

Bilimlarni ifodalash - Bu soha, bizga faktlarni taqdim etish imkonini beradi, bu esa mashinaga ularni tushunishga yordam beradi. Qanday qilib bilimlar samarali taqdim etilsa, tizim shunchalik intellektual bo'ladi.

Sun'iy intellektni qo'llash

Ushbu bo'limda biz SI tomonidan qo'llab-quvvatlanadigan turli sohalarni ko'ramiz:

Qimor o'yinlari - Sun'iy intellekt shaxmat, poker, "xoch va nol" kabi strategik o'yinlarda muhim rol o'ynaydi, chunki u evristik bilimlar asosida ko'plab mumkin bo'lgan pozitsiyalarni oldindan tahlil qila oladi.

Tabiiy tilni qayta ishlash - Kompyuter bilan inson gapiradigan tabiiy tilda muloqot qilish imkoniyatini yaratadi.

Ekspert tizimlar - Ba'zi dasturlar kompyuter, dasturiy ta'minot va maxsus ma'lumotlarni birlashtirib, mantiqiy xulosa va tavsiyalar beradi. Bu tizimlar foydalanuvchilarga tushuntirish va tavsiyalarni taqdim etadi.

Ko'rish tizimlari - Bu tizimlar kompyuter orqali vizual ma'lumotlarni tushunadi, talqin qiladi va anglaydi. Masalan:

- Razvedka samolyoti suratga olgan tasvirlardan hududiy ma'lumot yoki xaritalar yaratish uchun foydalaniladi.
- Shifokorlar klinik ekspert tizimidan bemor tashxisini qo'yishda foydalanadilar.
- Politsiya jinoyatchining tasvirini kriminalist-rasmchining chizgan portreti asosida tanib olish uchun dasturlardan foydalanadi.

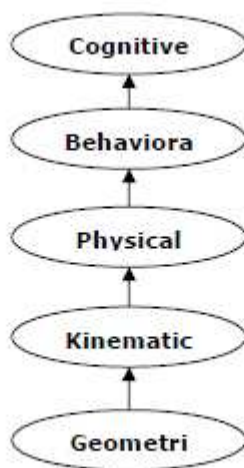
Nutqni tanib olish - Ba'zi intellektual tizimlar inson gapirayotganida nutqni eshitish va uning ma'nosini tushunish qobiliyatiga ega. Ular turli urg'ular, jargon so'zlar, fon shovqinlari yoki odamning sovuq tufayli o'zgaradigan ovozi qayta ishlay oladi.

Qo'lda yozilgan matnni tanib olish - Qo'lda yozilgan matnni tanib olish dasturlari qog'ozga ruchka bilan yoki ekran stylusi yordamida yozilgan matnni o'qiydi va uni tahrirlanadigan matnga aylantiradi.

Aqlli robotlar - Robotlar inson tomonidan belgilangan vazifalarni bajarishga qodir. Ular yorug‘lik, issiqlik, harorat oshishi, harakat, ovoz, zarba va bosim kabi fizik ma’lumotlarni aniqlash uchun datchiklarga ega. Ular kuchli protsessorlar, ko‘plab datchiklar va katta hajmdagi xotira bilan jihozlangan bo‘lib, intellektni namoyon etadi. Bundan tashqari, ular xatolaridan o‘rganib, yangi sharoitlarga moslasha oladi.

Kognitiv modellashtirish — bu kompyuter fanining inson tafakkur jarayonlarini o‘rganish va modellashtirish bilan shug‘ullanuvchi sohasi hisoblanadi. Sun’iy intellektning asosiy vazifasi — mashinani inson kabi fikrlashga undashdir. Inson tafakkur jarayonining eng muhim xususiyati muammolarni hal qilish qobiliyatidir. Shuning uchun, kognitiv modellashtirish ko‘proq insonlarning muammolarni qanday hal qilishini tushunishga harakat qiladi.

Ushbu modellar keyinchalik sun’iy intellektning turli ilovalari, masalan, mashinani o‘qitish, robototexnika, tabiiy tilni qayta ishlash va boshqalar uchun qo‘llanilishi mumkin. Quyida inson miyasi tafakkurining turli darajalarini ifodalovchi diagramma keltiriladi:



Agent va muhit

Ushbu bo‘limda agent va muhit tushunchalari hamda ularning sun’iy intellekt yaratishdagi ahamiyati haqida so‘z yuritamiz.

Agent — bu atrof-muhitni sezish uchun datchiklardan foydalanadigan va ushbu muhitga ta’sir ko‘rsatish uchun aktuatorlar orqali harakat qiladigan har qanday tizimdir.

- **Inson-agenti:** Insonning ko'z, quloq, burun, til va teri kabi sezgi organlari datchiklarga tenglashtiriladi, qo'l, oyoq va og'iz esa aktuator sifatida ishlaydi.
- **Robot-agenti:** Robotlarda datchiklar o'rnida kameralar va infraqizil masofa o'lchagichlar, aktuatorlar o'rnida esa turli motor va qo'zg'aluvchilar ishlatiladi.
- **Dasturiy ta'minot agenti** o'z dasturlari va harakatlari sifatida bit satrlarini kodlaydi .

Atrof-muhit - Ba'zi dasturlar klaviatura kiritish, ma'lumotlar bazasi, kompyuter fayl tizimlari va ekran belgilarini chiqarish bilan cheklangan to'liq sun'iy muhitda ishlaydi.

Aksincha, ba'zi dasturiy ta'minot agentlari (dasturiy ta'minot robotlari yoki dasturiy ta'minot botlari) dasturiy ta'minot botlarining keng, cheksiz sohalarida mavjud. Simulyator juda batafsil, murakkab muhitga ega. Dastur agenti real vaqt rejimida uzoq harakatlar to'plamidan tanlashi kerak. Dasturiy ta'minot boti mijozning onlayn afzalliklarini skanerlash uchun mo'ljallangan va mijozga haqiqiy va qurilgan muhitda ishlaydigan qiziqarli narsalarni ko'rsatadi.

Muhitning murakkabligi agentning qaror qabul qilish imkoniyatlarini va sun'iy intellekt tizimining moslashuvchanlik darajasini belgilaydi. Shuning uchun agent va muhitning o'zaro ta'siri sun'iy intellektning muvaffaqiyatida muhim rol o'ynaydi.

O'z-o'zini nazorat qilish uchun savollar

1. Aql, sun'iy intellekt, aqlli tizimlarga ta'rif bering.
2. Intellektual muammoning ta'riflarini va muammoni hal qilish algoritmlari.
3. Yigirmanchi asrning 50-60-yillarida sun'iy intellektning rivojlanishi bosqichlari.
4. Yigirmanchi asrning 70-80-yillarida SIning rivojlanishining asosiy yo'nalishlari.
5. Neyrokibernetika va "qora quti" kibernetikasi o'rtasidagi farq nima?
6. Robototexnika tarixi.

7. SI rivojlanishidagi tarixiy yondashuvlar.
8. Sun'iy intellektning rivojlanish yo'nalishlari.
9. Aqlli tizimlarning xususiyatlari.
10. Aqlli tizimlarning tasnifi.
11. Ekspert tizimlar.
12. Neyrokompyuterlar.
13. Noravshan tizimlar va robotlar.
14. Aqlli tizimlar.

1.2-§. PYTHON YORDAMIDA SUN'IY INTELLEKT – BOSHLANG'ICH KO'NIKMALAR

Ushbu bo'limda **Python** dasturlash tili bilan ishlashni boshlash asoslari va uning sun'iy intellektni yaratishdagi afzalliklari haqida bilib olamiz. Python sun'iy intellekt (SI) dasturlarini ishlab chiqishda ko'plab imkoniyatlar taqdim etuvchi kuchli vositadir.

Nega aynan Python sun'iy intellekt uchun?

Sun'iy intellekt kelajakning trend texnologiyalaridan biri hisoblanadi. Bu sohada bir qancha ilovalar allaqachon yaratilgan. Shu sababli ko'plab kompaniyalar va tadqiqotchilar bu sohaga qiziqish bildirmoqda. Ammo asosiy savol shundan iboratki, ushbu sun'iy intellekt ilovalari qaysi dasturlash tilida ishlab chiqilishi mumkin?

Bugungi kunda Lisp, Prolog, C++, Java va Python kabi turli dasturlash tillaridan foydalanish orqali sun'iy intellekt ilovalarini yaratish mumkin. Ular orasida Python dasturlash tili katta mashhurlikka ega bo'lib bormoqda. Bunga quyidagi sabablar ta'sir qilmoqda:

Oddiy sintaksis va kamroq kodlash

Python boshqa dasturlash tillariga qaraganda kamroq kod yozishni talab qiladi va juda sodda sintaksisga ega. Bu esa sun'iy intellekt ilovalarini ishlab chiqishda foydalidir. Ushbu xususiyat tufayli dasturlarni test qilish jarayoni osonlashadi va dasturchilar asosan kodlashdan ko'ra muammoni hal qilishga ko'proq e'tibor qaratishlari mumkin bo'ladi.

Sun'iy intellekt loyihalari uchun o'rnatilgan kutubxonalar

Python'ning sun'iy intellekt uchun ishlatilishidagi asosiy afzalliklardan biri uning o'rnatilgan kutubxonalari bilan ta'minlanishidir. Python'da sun'iy intellektning deyarli barcha turdagi loyihalari uchun mos keladigan kutubxonalar mavjud. Masalan, **NumPy**, **SciPy**, **matplotlib**, **nlTK**, va **SimpleAI** Python'ning muhim kutubxonalaridan hisoblanadi.

Ochiq kod manbai

Python — bu ochiq kodli dasturlash tili. Ushbu xususiyat uni dasturchilar hamjamiyatida keng tarqalgan va mashhur qiladi.

Python'ning xususiyatlari

Python — yuqori darajali, interpretatsiya qilinadigan, interaktiv va obyektga yo'naltirilgan skript tili. Bu til o'quvchiga oson tushunarli bo'lish uchun ishlab chiqilgan. Python'da inglizcha kalit so'zlardan foydalaniladi va boshqa tillardagi kabi murakkab punktuatsiya talab qilinmaydi. Sintaksis ham sodda va minimaldir. Python'ning asosiy xususiyatlari quyidagilarni o'z ichiga oladi:

- **O'rganishga oson** - Python kalit so'zlari kam, tuzilishi sodda va sintaksisi aniq belgilangan. Bu tilni yangi o'rganuvchilar uchun o'rganishni osonlashtiradi.
- **Oson o'qilishi** - Python kodi toza, tushunarli va vizual tarzda o'qilishi qulay.
- **Texnik xizmat ko'rsatishning qulayligi** - Python manba kodi boshqa tillarga qaraganda oddiyroq va uni texnik jihatdan boshqarish oson.
- **Keng standart kutubxona** - Python'ning standart kutubxonalari ko'plab platformalarda ishlaydi, jumladan UNIX, Windows va Macintosh.
- **Interaktiv rejim** - Python interaktiv rejimni qo'llab-quvvatlaydi, bu kod bo'laklarini sinab ko'rish va xatolarni tuzatish imkonini beradi.
- **Portativlik** - Python har xil apparat platformalarida ishlay oladi va barcha platformalarda bir xil interfeysni taklif qiladi.
- **Kengaytiruvchanlik** - Python interpretatoriga past darajali modullarni qo'shish imkoniyati mavjud. Bu modullar dasturchilarga samaradorlikni oshirish uchun o'zlariga mos vositalarni yaratish yoki sozlash imkoniyatini beradi.
- **Ma'lumotlar bazasi bilan ishlash** - Python barcha asosiy tijorat ma'lumotlar bazalari bilan interfeyslarni taqdim etadi.
- **Grafik interfeys bilan dasturlash** - Python grafik interfeysli ilovalarni yaratish va boshqa tizimlarga ko'chirish imkoniyatiga ega. Masalan, Windows MFC, Macintosh va Unix'dagi X Window System orqali grafik ilovalarni yaratish mumkin.

- **Masshtablilik** - Python katta hajmdagi dasturlarni boshqarish uchun yaxshi tuzilma va qo'llab-quvvatlashni taqdim etadi.

Python'ning muhim funksiyalari

Keling, endi Python'ning quyidagi muhim funksiyalarini ko'rib chiqamiz:

(izoh: Muammo yoki savolga oid funksiyalar ro'yxatini davom ettirish mumkin, agar kerak bo'lsa).

Python ni o'rnatish

Python tarqatilishi ko'p sonli platformalar uchun mavjud. Siz faqat platformangizga tegishli ikkilik kodni yuklab olishingiz va Python-ni o'rnatishingiz kerak.

Agar platformangiz uchun ikkilik kod mavjud bo'lmasa, manba kodini qo'lda kompilyatsiya qilish uchun sizga C kompilyatori kerak bo'ladi. Manba kodini kompilyatsiya qilish o'rnatish vaqtida talab qilinadigan xususiyatlar nuqtai nazaridan ko'proq moslashuvchanlikni ta'minlaydi.

Bu erda Python-ni turli platformalarda o'rnatishning qisqacha ko'rinishi -

Unix va Linuxga o'rnatish

Unix/Linux kompyuteringizga Python-ni o'rnatish uchun quyidagi amallarni bajaring.

Veb-brauzeringizni oching va <https://www.python.org/downloads> saytiga o'ting

Unix/Linux uchun mavjud siqilgan manba kodini yuklab olish uchun havolani bajaring.

Fayllarni yuklab oling va chiqarib oling.

Ba'zi opsiyalarni moslashtirmoqchi bo'lsangiz, *Modules/Setup* faylini tahrirlang .

yugurish. / skriptni sozlash

yaratish

o'rnatishni amalga oshiring

Bunday holda, Python standart `/usr/local/bin` papkasida o'rnatiladi va uning kutubxonalari `/usr/local/lib/pythonXX` da o'rnatiladi, bu erda XX Python versiyasidir.

Windowsga o'rnatish

Python-ni Windows kompyuteringizga o'rnatish uchun quyidagi amallarni bajaring.

Veb-brauzeringizni oching va <https://www.python.org/downloads> saytiga o'ting *python-XYZ.msi* o'rnatish fayli uchun havolaga rioya qiling , bu erda XYZ o'rnatishingiz kerak bo'lgan versiya.

python-XYZ.msi o'rnatuvchisidan foydalanish uchun Windows tizimingiz Microsoft Installer 2.0 ni qo'llab-quvvatlashi kerak. O'rnatish faylini mahalliy kompyuteringizga saqlang va kompyuteringiz MSI-ni qo'llab-quvvatlaydimi yoki yo'qligini bilish uchun uni ishga tushiring.

Yuklab olingan faylni ishga tushiring. Bu Python o'rnatish ustasini ochadi, undan foydalanish juda oson. Faqat standart sozlamalarni qabul qiling va o'rnatish tugashini kuting.

Macintoshga o'rnatish

Agar siz Mac OS X dan foydalanayotgan bo'lsangiz, Python 3 ni o'rnatish uchun Homebrew dan foydalanish tavsiya etiladi. Bu Mac OS X uchun ajoyib paket o'rnatuvchi va undan foydalanish juda oson. Agar sizda Homebrew bo'lmasa, uni quyidagi buyruq yordamida o'rnatishingiz mumkin -

```
$ ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Quyidagi buyruq yordamida paket menejerini yangilashimiz mumkin -

```
$ brew update
```

Endi tizimingizga Python3 ni o'rnatish uchun quyidagi buyruqni bajaring -

```
$ brew install python3
```

PATH (yo'lni) sozlansh

Dasturlar va boshqa bajariladigan fayllarni ko'plab kataloglarda topish mumkin, shuning uchun operatsion tizimlar OS bajariladigan fayllarni qidiradigan kataloglar ro'yxatini ko'rsatadigan qidiruv yo'lini taqdim etadi.

Yo‘l muhit o‘zgaruvchisida saqlanadi, bu operatsion tizim tomonidan qo‘llab-quvvatlanadigan nomlangan qatordir. Bu o‘zgaruvchi qobiq va boshqa dasturlar uchun mavjud bo‘lgan ma’lumotlarni o‘z ichiga oladi.

Yo‘l o‘zgaruvchisi Unix da PATH yoki Windows da Path deb ataladi (Unix katta harflarni sezgir, Windows emas).

Mac OS da o‘rnatuvchi yo‘l ma’lumotlarini boshqaradi. Har qanday maxsus katalogdan Python tarjimoniga qo‘ng‘iroq qilish uchun Python katalogini yo‘lingizga qo‘shishingiz kerak.

Unix/Linux-da yo‘lni sozlash

Unix-da ma’lum bir seans uchun yo‘lga Python katalogini qo‘shish uchun -

- In the csh shell

Type **setenv PATH "\$PATH:/usr/local/bin/python"** and press **Enter**.

- In the bash shell (Linux)

Type **export PATH = "\$PATH:/usr/local/bin/python"** and press **Enter**.

- In the sh or ksh shell

Type **PATH = "\$PATH:/usr/local/bin/python"** and press **Enter**.

Eslatma - */usr/local/bin/python* python katalogiga bo‘lgan yo‘l .

Windows-da yo‘lni sozlash

Windows-da ma’lum bir seans uchun *path* Python katalogini qanday qo‘shish mumkin -

- Buyruqlar satriga **%path%;C:\Python** yo‘lini kiriting va **Enter** ni bosing .

Eslatma - C:\Python - Python katalogiga bo‘lgan yo‘l.

Python ishga tushirish

Endi Python-ni ishga tushirishning turli usullarini ko‘rib chiqamiz.

Interaktiv tarjimon

Biz Python-ni Unix, DOS yoki buyruq qatori tarjimoni yoki buyruq qobig‘i oynasi bilan ta’minlaydigan har qanday boshqa tizimdan ishga tushirishimiz mumkin.

- Buyruqning satriga **python** yozing .
- Endi interaktiv tarjimonda kodlashni boshlang.

\$python # Unix / Linux

yoki

python% # Unix/Linux

yoki

C:>python#Windows/DOS

Bu erda barcha mavjud buyruq qatori variantlari ro'yxati.

№	Variantlar va tavsif
1	-d Bu disk raskadrovka chiqishini ta'minlaydi.
2	-o U optimallashtirilgan bayt kodini hosil qiladi (natijada .pyo fayllari).
3	-S Ishga tushganda Python yo'llarini qidirish uchun import saytini ishga tushirmang.
4	-v Verbose chiqish (import ko'rsatmalarini batafsil kuzatish).
5	-x o'rnatilgan sinfga asoslangan istisnolarni o'chiradi (faqat satrlardan foydalaning); 1.6 versiyasidan boshlab eskirgan.
6	-c cmd cmd qatori sifatida yuborilgan Python skriptini ishga tushiradi.
7	fayl Berilgan fayldan Python skriptini ishga tushiring.

Buyruqlar qatorida skript

Python skripti buyruq satrida quyida ko'rsatilganidek, ilovangizdagi tarjimonni chaqirish orqali bajarilishi mumkin -

\$ python script.py # Unix/Linux

yoki ,

python% script.py # Unix/Linux

yoki ,

C:>python script.py #Windows/DOS

Eslatma - Fayl ruxsati rejimi bajarishga ruxsat berilganligiga ishonch hosil qiling.

Integratsiyalashgan ishlab chiqish muhiti

Agar tizimingizda Python-ni qo'llab-quvvatlaydigan GUI ilovasi bo'lsa, siz Python-ni grafik foydalanuvchi interfeysi (GUI) muhitidan ham ishga tushirishingiz mumkin.

- **Unix** - IDLE Python uchun birinchi Unix IDE hisoblanadi.

- **Windows** – PythonWin Python uchun birinchi Windows interfeysi va GUI IDE.
- **Macintosh** - Pythonning Macintosh versiyasi IDLE IDE bilan birga asosiy veb-saytda mavjud va MacBinary yoki Binhex'd fayllari sifatida yuklab olinadi.

Agar siz muhitni to'g'ri sozlay olmasangiz, yordam uchun tizim administratoriga murojaat qilishingiz mumkin. Python muhitingiz to'g'ri tuzilganiga va muammosiz ishlayotganiga ishonch hosil qiling.

Anaconda deb nomlangan boshqa Python platformasidan ham foydalanishimiz mumkin. U yuzlab mashhur ma'lumotlar fanlari paketlarini, shuningdek Windows, Linux va macOS uchun conda paketi va virtual muhit menejerini o'z ichiga oladi. Siz uni operatsion tizimingizga muvofiq <https://www.anaconda.com/download/> havolasidan yuklab olishingiz mumkin . Ushbu qo'llanmada biz MS Windows uchun Python 3.6.3 versiyasidan foydalanamiz.

NumPy kutubxonasi, ko'p o'lchovli massiv

NumPy kutubxonasi C/C++ massivlariga o'xshash bir hil ma'lumotlarning (bir turdagi, odatda sonli) ko'p o'lchovli massivlariga va ularni qayta ishlash usullariga kirishni ta'minlaydi. NumPy kutubxonasi ostida xuddi shu massivlar dinamik C massivlari yordamida amalga oshiriladi, ular ro'yxatlardan farqli o'laroq, ma'lumotlarni unga bo'lgan ko'rsatgichla ko'rinishida emas, balki to'g'ridan-to'g'ri saqlanadi.

NumPy nima uchun kerak? Katta sonli massivlar bilan ishlash uchun maxsus kutubxonaga ehtiyoj pythonning tezligi tufayli yuzaga keladi. Pythonning moslashuvchanligi, ifodaliligi, dinamik tipliligi va boshqa ko'plab afzalliklari tezlikka katta tasir qiladi: pythonda yozilgan kod deyarli har doim C/C++ da yozilgan shunga o'xshash koddan sekinroq ishlaydi. Buni ko'rsatishning eng oson yo'li – quyidagi katta sikllar bilan ishlash misolida yanaada yaqol ko'rish mumkun bo'ladi.

Python

```
from time import perf_counter
from random import uniform

N = 10_000_000

# Ixtiyoriy sonlar ro'yxatini yaratish
a = [uniform(-1, 1) for i in range(N)] # yangi sintaksis, ixtiyoriy sonlar ro'yxati
b = [uniform(-1, 1) for i in range(N)]
c = [0.] * N

# Bu ro'yxatni shakilanish vaqtini o'lchash
t1 = perf_counter()

for i in range(N):
    c[i] = a[i] + b[i]

t2 = perf_counter()
print(f"{(t2 - t1) * 1000} ms") # perf_counter() - vaqt sekundlarda hisoblangan
```

C++

```
#include <iostream>
#include <chrono>
#include <vector>
#include <random>
#define N 10000000

int main(){
    // -1 dan 1 oralig'idagi ixtiyoriy sonlarni shakilantirish
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_real_distribution<> dis(-1.0, 1.0);

    // Ixtiyoriy sonlar massivini yaratish
    std::vector<double> a(N), b(N), c(N);
    for(int i=0; i < N; ++i){
        a[i] = dis(gen);
        b[i] = dis(gen);
    }

    // Vektorlarni qo'shish vaqtini o'lchash
    auto t1 = std::chrono::high_resolution_clock::now();

    for(int i = 0; i < N; ++i)
        c[i] = a[i] + b[i];

    auto t2 = std::chrono::high_resolution_clock::now();

    auto ms_int = std::chrono::duration_cast<std::chrono::milliseconds>(t2
- t1);
    std::cout << ms_int.count() << " ms";
}
```

Yuqorida ikkita manba kodli fayllarning tarkibi keltirilgan, ularning har biri 108 o‘lchovli raqamlarning ikkita vektorini qo‘shadi.

1. Python tilidagi brinchi dastur kodidagi vektorlarni ko‘rsatish uchun ro‘yxatlardan foydalanadi.
2. C++ tilidagi ikkinchi fayl *std::vector* dasturidagi vektorlarni ifodalash uchun ishlatiladi.

Ushbu fayllar havolada joylashgan va buyruq satrida quyidagi buyruqlar bilan ishga tushirilishi mumkin.

```
g++ loops.cpp -O2 -o loops
.\loops.exe
python loops.py
```

```
Python NumPy
from time import perf_counter
from numpy.random import uniform

N = 10_000_000

# Yxtiyoriy sonlar massivini yaratish
a = uniform(-1, 1, size=N)
b = uniform(-1, 1, size=N)

# Bu ro'yxatni shakilantirish vaqti
t1 = perf_counter()
c = a + b
t2 = perf_counter()
print(f"{{(t2 - t1) * 1000}} ms") # perf_counter() - vaqat sekundlarda
```

Umuman olganda, bu raqamlarga juda ko‘p omillar, shu jumladan tasodifiy omillar ta’sir qilishi mumkin. Shunga qaramay, farq shunchalik aniqki: C/C++ da sikkalar python-ga qaraganda ancha tez ishlaydi.

Python-dagi “o‘rtacha” dastur katta hajmdagi ma’lumotlarni qayta ishlashga duch kelmaydi va odatda ishlashdagi farq algoritmnini amalga oshirish uchun sarflangan vaqt bilan qoplanadi. Ammo ilmiy hisob-kitoblar ko‘pincha “ko‘p sonli raqamlarni bilan ishlash” ayniqsa sun’iy intellek masalalarida, bunday sekinlashuv sezilarli darajada bop‘ladi.

Kompilyatsiya qilingan tilning tezligi va python-ning qulayligini birlashtirish uchun ko‘plab kutubxonalar C/C++ (yoki boshqa kompilyatsiya qilingan til) da yozilgan kompilyatsiya qilingan modullar bilan birga keladi va python-da ular bilan ishlash uchun interfeysga “uzatiladi”. NumPy - bunday kutubxonalardan biridir.

NumPy ni o‘rnatish

Anaconda da NumPy o‘rnatilgan bo‘ladi yoki siz uni quyidagi buyruq bilan pip install yordamida o‘rnatishingiz mumkin.

```
python -m pip install numpy
```

NumPy-ni o‘rnatganingizdan so‘ng, uni dasturda ishlatish uchun uni import qilishingiz kerak.

```
import numpy as np
```

Numpy kutubxonasi shu qadar tez-tez va shu qadar ko‘p ishlatiladiki, NumPy-ni *np* taxallusi bilan import qilish an‘anasi paydo bo‘ldi. Kodning biron bir joyida *np* shaklidagi ifodani ko‘rgan har qanday dasturchi *np.some_method()* NumPy kutubxonasidan *some_method* usuli chaqirilishini taxmin qiladi va bu nomning orqasida nima borligini tushunish uchun *np* ob‘ektining ta‘rifini ko‘z bilan izlash shart emas.

NumPy massivlari haqida. Ma’lumotlar turi, o‘lchami, o‘qlari va shakli.

Quyidagi tasodifiy sonlardan NumPy massivi yaratiladi. Siz hali uning qanday yaratilganligini aniq bilmasligingiz mumkin. Faqat *array* nomi NumPy massiv bilan bog‘liqligini tushunish muhimdir.

```
array = np.random.random(size=(2, 3))
print(array)
[[0.5971054  0.44687181 0.63247176]
 [0.20003964 0.08926428 0.99762199]]
```

Kutubxonadagi massivlar NumPy turi bilan ifodalanadi *numpy.ndarray* (*n dimensional array*, *n* o‘lchovli massiv).

```
print(f"type(array)=)")
type(array)=<class 'numpy.ndarray'>
```

ndarray tipi tiplashgan ma'lumotlar konteyneridir, ya'ni u bir xil turdagi ma'lumotlarni saqlash uchun mo'ljallangan. Massiv ma'lumotlar turi *type* (*data type*) atributida saqlanadi. Ushbu turlar juda ko'p bo'lishi mumkin, ammo ularning aksariyati standart sonli tiplardir:

C/C++: `np.bool8`, `np.int8`, `np.int16`, `np.int32`, `np.int64`, `np.uint8`, `np.uint16`, `np.uint32`, `np.uint64`, `np.float16`, `np.float32`, `np.float64`, `np.complex64`, `np.complex128`.

NumPy massivlaridagi butun sonlar python-dandagi *int* tipiga o'xshamaydi: ular har doim ma'lum miqdordagi baytlarni egallaydi va to'ldirilishi mumkin, ya'ni C/C++ dangi butun son tipiga to'g'ri keladi.

Umuman olganda, NumPy massivlarida ixtiyoriy tipdagi obyektlar ham saqlanishi mumkin, ammo keyinchalik obyektlarning o'rniga massivda ularga havolalar saqlanadi. Ko'pincha NumPy massivlari sonlar saqlash uchun ishlatiladi.

```
print(f"{array.dtype=}")  
array.dtype=dtype('float64')
```

array massivining *dtype* atributining qiymatidan kelib chiqib, u 64 bitli suzuvchi nuqta raqamlarini saqlashini tushunish mumkin.

NumPy massivlari o'zlarining o'lchovlari sonini *ndim* (**n**umber of **d**imensions) atributida saqlaydi. NumPy massivlari har qanday o'lchovda bo'lishi mumkin: bir o'lchovli massivlar ko'pincha vektorlarni ifodalash uchun, ikki o'lchovli massivlar matritsalar va jadvallarni ifodalash uchun, yuqori o'lchovli massivlar tensorlarni ifodalash uchun, har qanday o'lchovli massivlar ko'pincha vektorli argumentning ba'zi funktsiyalarining qiymatlarini ko'p o'lchovli fazoda panjara ustiga ko'rsatish uchun ishlatiladi.

```
print(f"{array.ndim=}")  
array.ndim=2
```

array massivining *ndim* atributining qiymatiga asoslanib, biz uning ikki o'lchovli ekanligini ko'ramiz. NumPy massivlarining yana bir majburiy atributi-shakl (*shape*) — har bir o'lchov bo'ylab massivning o'lchamlari. Ushbu o'lchovlar o'qlar deb ataladi (birlikdagi *axis* va ko'plikdagi *axes* soni).

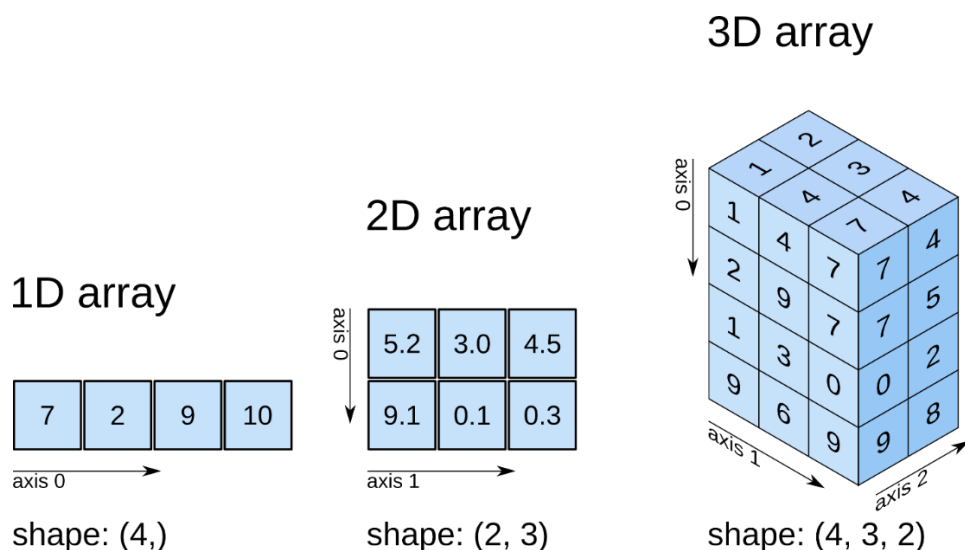
```
print(f"{array.shape}")  
array.shape=(2, 3)
```

array massivdagi *shape* atributi bizga birinchi o'lchov bo'ylab 2 ta element va ikkinchi o'lchov bo'ylab 3 ta element borligini aytadi. Boshqacha qilib aytganda, bu ikki qator va uchta ustundan iborat jadval deb aytishimiz mumkin.

Massivdagi elementlarning umumiy soni (har bir o'q bo'ylab elementlar sonining ko'paytmasi) *size* atributida saqlanadi.

```
print(f"{array.size}")  
array.size=6
```

Barcha massivlar tenglashtirilishi kerak, ya'ni, har xil uzunlikdagi qatorlardan iborat matritsalar bo'lishi mumkin emas. Quyidagi rasm yuqoridagi barcha tushunchalarning ma'nosini aniq ko'rsatib beradi.



Massiv yaratish

Siz NumPy — *ndarray* massivlarini juda ko'p miqdorda yaratishingiz mumkin, ammo ularni yaratish uchun kamdan-kam tipning o'z konstruktor ishlatiladi. Buning o'rniga, boshqa usullardan foydalanish ancha keng tarqalgan. Masalan, *numpy.array ndarray* ni boshqa ob'ektlardan quradi.

U ko'pincha raqamlar ro'yxatining python-dan NumPy massivlarini yaratish uchun ishlatiladi. Ma'lumotlar turi, agar u aniq nomlangan *dtype* parametri bilan ko'rsatilmagan bo'lsa, eng keng tarqalgan ma'lumotlar turi sifatida ko'rsatiladi.

Quyida to'rtta raqam ro'yxatidan bir o'lchovli massiv yaratiladi. Bunday holda, ma'lumotlar turi aniq ko'rsatilmagan va natijada suzuvchi nuqta (*float64*) bo'lgan raqamlar qatori olinadi, chunki dastlabki ro'yxatda *int* va *float* kabi raqamlar mavjud edi.

```
a1D = np.array([1., 2, 3, 4])
print(repr(a1D))
print(f"{a1D.dtype=}")
print(f"{a1D.ndim=}")
print(f"{a1D.shape=}")
array([1., 2., 3., 4.])
a1D.dtype=dtype('float64')
a1D.ndim=1
a1D.shape=(4,)
```

Quyidagi 2D massivlar qanday yaratilganligi ko'rsatilgan. Buning uchun ro'yxatlar ro'yxati unga kiritiladi, bu erda har bir ichki ro'yxat kelajakdagi jadvalning qatoridir. Chunki ushbu ro'yxatlarning barchasida faqat butun sonlar mavjud, keyin massivning yakuniy turi ham butun sonidir.

```
a2D = np.array([[1, 2], [3, 4]])
print(repr(a2D))
print(f"{a2D.dtype=}")
print(f"{a2D.ndim=}")
print(f"{a2D.shape=}")
array([[1, 2],
       [3, 4]])
a2D.dtype=dtype('int32')
a2D.ndim=2
a2D.shape=(2, 2)
```

Uch o'lchovli massivlar yuqoridagi kabi yaratilgan. Quyidagi keltirilgan misolga e'tibor bering, butun sonli ro'yxatlar ro'yxatining uzatiladigan ro'yxatidagi barcha raqamlar, ammo natijada paydo bo'lgan massiv *float32* tipiiga kiradi, chunki massivni yaratishda ushbu tip aniq ko'rsatilgan.

```
a3D = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]], dtype=np.float32)
print(repr(a3D))
print(f"{a3D.dtype=}")
print(f"{a3D.ndim=}")
print(f"{a3D.shape=}")
array([[[1., 2.],
       [3., 4.]],
      [[5., 6.],
       [7., 8.]]], dtype=float32)
a3D.dtype=dtype('float32')
```



```
a3D.ndim=3
a3D.shape=(2, 2, 2)
```

Massivlarni yaratadigan bir qator o'rnatilgan fuktssiyala ham mavjud. Ularning har biri ixtiyoriy argument *dtype* turi sifatida qabul qilinadi.

Usul	Argumentlar	Izoxi
np.arange	start, stop, end	analoglari <i>np.array(range)</i> , lekin <i>float</i> argument sifatida ishlatilishi mumkin
np.linspace	start, stop, num, endpoint=True	[start, stop] segmentini <i>num</i> segmentlariga bo'lish, agar <i>endpoint=False</i> sifatida uzatilsa, oxirgi nuqta kiritilmaydi
np.zeros	shape	Nollardan iborat bo'lgan berilgan formadagi massiv
np.ones	shape	Birlardan iborat bo'lgan berilgan formadagi massiv
np.random		Bu metod emas, balki tasodifiy qiymatlar massivlarini yaratish quyi modul
np.eye	n, m	Agar <i>m</i> ko'rsatilmagan bo'lsa, u holda $n \times n$ o'lchamdagi kvadrat birlik matritsasi, aks holda diagonaldagi birliklar bilan $n \times m$ o'lchamdagi nol matritsa
np.diag	v, k=0	Asosiy diagonalda ($k=0$) yoki <i>k</i> -diagonalda <i>v</i> vektorli matritsa
np.vand	x, N=None, increasing=False	<i>x</i> vektoridan Vandermond matritsasi

Biz ularnda asosiylarini keltirib o'tamiz. Nollar iborat massiv [np.zeros](#) usuli yordamida yaratiladi.

```
zeros = np.zeros(shape=(2, 2))
print(zeros)
[[0. 0.]
 [0. 0.]]
```

[np.ones](#) xam yuqoridagi kabi fazifani bajaradi, faqat bunda massiv birlardan iborat bo'ladi.

```
ones = np.ones(shape=(2, 2))
print(ones)
[[1. 1.]
 [1. 1.]]
```

[np.eye](#) funktsiyasi ko'pincha birlik massivlarni yaratish uchun ishlatiladi.

```
eye = np.eye(2)
print(eye)
[[1. 0.]
 [0. 1.]]
```

[np.diag](#) funksiyasi diagonalni to'ldirish uchun unga berilgan massiv elementlardan foydalangan holda diagonal massiv yaratadi.

```
diag = np.diag([3.14, 42])
print(diag)
[[ 3.14  0.   ]
 [ 0.   42.   ]]
```

Bundan tashqari, boshqa massivlar yaratishga imkon beradigan bir qator usullar mavjud. Ular quyidagi jadvalda keltirilgan.

Metod	Izoxi
<u>np.hstack</u>	Qatorlar bo'ylab massivlarni biriktirish
<u>np.vstack</u>	Ustunlar bo'ylab massivlarni biriktirish
<u>np.dstack</u>	"Chuqurlikda" massivlarni biriktirish
<u>np.block</u>	Bloklardan massiv tuzish

[np.block](#) usulini bu sizga boshqa massivlar ishlashi mumkin bo'lgan bloklardan massivlar yaratishga imkon beradi.

```
print(np.block([[zeros, ones], [eye, diag]]))
[[ 0.   0.   1.   1. ]
 [ 0.   0.   1.   1. ]
 [ 1.   0.   3.14  0. ]
 [ 0.   1.   0.   42. ]]
```

NumPy massivlarida arifmetik amallar

Все основные арифметические операторы (+, -, *, /, //, %, **) перегружены для работы с массивами. Определим скаляр и вектор, чтобы разобраться, какие действия они позволяют выполнять.

```
x = np.array([3, 13, 42])
alpha = 7
```

Massiv va skalyar

Agar arifmetik ifodada skalyar va massiv ishtirok etsa, natijada skalyar bilan ko'rsatilgan arifmetik amalni element bo'yicha qo'llash qiymatlari massiv bo'ladi. Ya'ni, masalan, massivni songa ko'paytirish mumkin.

```
print(f"{alpha * x = }")
print(f"{x * alpha = }")
alpha * x = array([ 21,  91, 294])
x * alpha = array([ 21,  91, 294])
```

Yoki vektorni xar bir element raqami bo'yicha qo'shish.

```
print(f"{alpha + x = }")
print(f"{x + alpha = }")
```

```

print(horizontal_line)

print(f"{alpha - x = }")
print(f"{x - alpha = }")
alpha + x = array([10, 20, 49])
x + alpha = array([10, 20, 49])

_____

alpha - x = array([ 4, -6, -35])
x - alpha = array([-4, 6, 35])

```

Vektorni raqamga va raqamni vektorga bo‘lish mumkin.

```

print(f"{alpha / x = }")
print(f"{x / alpha = }")
alpha / x = array([2.33333333, 0.53846154, 0.16666667])
x / alpha = array([0.42857143, 1.85714286, 6.          ])

```

Bundan tashqari, siz boshqa barcha arifmetik amallarni bajarishingiz mumkin.

```

print(f"{alpha // x = }")
print(f"{x // alpha = }")

print(horizontal_line)

print(f"{alpha % x = }")
print(f"{x % alpha = }")

print(horizontal_line)

print(f"{alpha ** x = }")
print(f"{x ** alpha = }")
alpha // x = array([2, 0, 0], dtype=int32)
x // alpha = array([0, 1, 6], dtype=int32)

_____

alpha % x = array([1, 7, 7], dtype=int32)
x % alpha = array([3, 6, 0], dtype=int32)

_____

alpha ** x = array([          343, -1895237401, 1626129905], dtype=int32)
x ** alpha = array([      2187,      62748517, -1388900736], dtype=int32)

```

Massiv ichida massiv

Agar ifodada ikkita massiv ishtirok etsa, unda ular bir xil shaklda bo‘lishi kerak va arifmetik operatsiya hali ham element bo‘yicha bajariladi. Shuni esda tutish kerakki, barcha arifmetik operatorlar elementlararo transformatsiyani amalga oshiradilar. Bu, qaysidir ma’noda, NumPy-ning standart xatti-harakati va NumPy-

ning boshqa ko‘plab harakatlarini element bo‘yicha bajaradi. Bundan tashqari, matritsani ko‘paytirish ko‘rib chiqiladi — bu asosan elementdan tashqari operatsiya.

```
x = np.array([3, 13, 42])
y = np.array([42, 13, 3])

print(f"x + y = ")
print(f"x + y = ")

print(horizontal_line)

print(f"x * y = ")
print(f"x / y = ")

print(horizontal_line)

print(f"x // y = ")
print(f"x % y = ")

print(horizontal_line)

print(f"x ** y = ")
x + y = array([45, 26, 45])
x + y = array([45, 26, 45])

-----
x * y = array([126, 169, 126])
x / y = array([ 0.07142857,  1.          , 14.          ])

-----
x // y = array([ 0,  1, 14])
x % y = array([3, 0, 0])

-----
x ** y = array([ 1914644777, -1692154371,  74088])
```

Matritsani ko‘paytirish

Operator `*` elementlarni ko‘paytirishni amalga oshiradi, shu jumladan ikki o‘lchovli massivlar uchun xam.

```
ones = np.ones(shape=(2, 2))
eye = np.eye(2)

print(f"{ones}\n*\n{eye}\n=\n{ones * eye} ")
[[1.  1.]
 [1.  1.]]
*
[[1.  0.]
 [0.  1.]]
=
[[1.  0.]
 [0.  1.]]
```

Matritsani ko‘paytirish uchun `@` operatori ishlatiladi.

```
print(f"{ones}\n@\n{eye}\n=\n{ones @ eye} ")
[[1. 1.]
 [1. 1.]]
@
[[1. 0.]
 [0. 1.]]
=
[[1. 1.]
 [1. 1.]]
```

Indekslash

Bir oʻlchovli *ndarray* massivlarini indeksatsiya qilish python ketma-ketligini indeksatsiya qilish bilan juda oʻxshash (masalan, roʻyxatlar).

```
print(a1D)
print(f"{a1D[0] = }, {a1D[-1] = }")
[1. 2. 3. 4.]
a1D[0] = 1.0, a1D[-1] = 4.0
```

Koʻp oʻlchovli indeksatsiya biroz yangi sintaksis bilan amalga oshiriladi. Masalan, ikki oʻlchovli massivning *i*-qatori va *j*-ustunining kesishmasida elementni olish uchun quyidagi sintaksis ishlatiladi.

```
array[i, j]
print(f"{a2D = }")

print(horizontal_line)

print(f"{a2D[1, 0] = }, {a2D[-1][-1] = }")
a2D = array([[1, 2],
            [3, 4]])

_____
a2D[1, 0] = 3, a2D[-1][-1] = 4
```

Agar indekslash ikki oʻlchovli massivga bitta raqam bilan qoʻllanilsa *i*, keyin siz *i*-qatorni olasiz.

```
print(f"{a2D[0] = }, {a2D[-1] = }")
a2D[0] = array([1, 2]), a2D[-1] = array([3, 4])
```

Bularning barchasi, umuman olganda, ixtiyoriy oʻlchovli massivlarga taalluqlidir. *n* oʻlchovli massiv elementini olish uchun *n* indekslar koʻrsatiladi, yaʼni har bir oʻlchov uchun indeks boʻyicha:

```
a[i_1, i_2, ..., i_n]
```

Massiv shaklini o'zgartirish

Asl massiv ma'lumotlarini saqlash orqali massivning o'lchami va shaklini o'zgartirish mumkin, ammo elementlarning umumiy soni o'zgarishsiz qolishi.

```
A = np.arange(12)
print(A)
[ 0  1  2  3  4  5  6  7  8  9 10 11]
```

[np.ndarray.reshape](#) usuli bir xil ma'lumotlar bilan yangi formadagi massiv yaratadi.

```
B = A.reshape(4, 3)
print(B)
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

[np.ndarray.resize](#) usuli massiv formasini joyida o'zgartiradi.

```
A.resize(2, 3, 2)
print(A)
[[[ 0  1]
   [ 2  3]
   [ 4  5]]

 [[ 6  7]
   [ 8  9]
   [10 11]]]
```

[np.ndarray.ravel](#) usuli ixtiyoriy o'lchovli massivni bir o'lchovli massivga aylantiradi.

```
A = A.ravel()
print(A)
[ 0  1  2  3  4  5  6  7  8  9 10 11]
```

View. Umumiy ma'lumotlarga egalik qilish va nusxalash

Aslida, *ndarray* massivlari biroz murakkabroq. *ndarray* massivlari bir xil ma'lumotlarni bir-biri bilan bo'lishi mumkin. Massivlarni ikki turga bo'lish mumkin: *base* va *view*, ikkinchisi o'z ma'lumotlariga ega emas, lekin *base* massivdagi ma'lumotlarga ishora qiladi.

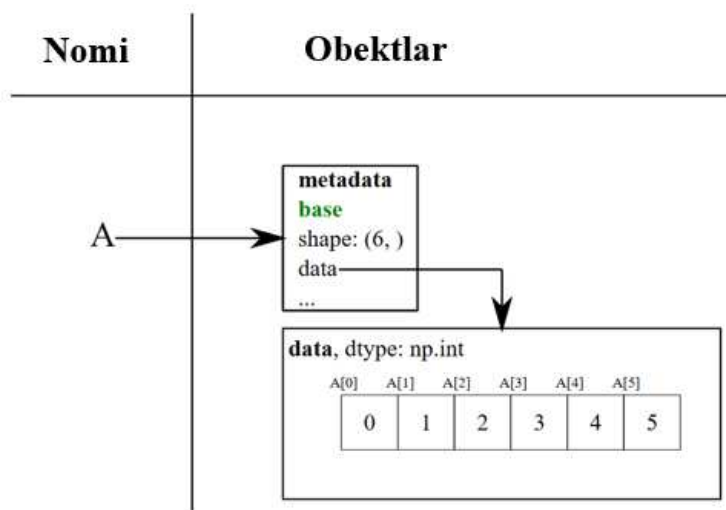
- *base* array ma'lumotlarga egalik qiladi;
- *view* faqat ularga boshqacha qarashga imkon beradi.

Har bir *ndarray*-bu nafaqat ma'lumotlarga ko'rsatgichni (*C* massivga juda o'xshash narsa) saqlaydigan ob'ekt, balki massivga ushbu ma'lumotlarni qanday talqin qilishni aytadigan metadata.

Misolni ko'rib chiqing. *A* qatorini yarating va uning ma'lumotlariga egalik qilishiga ishonch hosil qiling.

```
A = np.arange(6)
print(A)
print("A massiv belgilari: \n", A.flags)
[0 1 2 3 4 5]
A massiv belgilari:
  C_CONTIGUOUS : True
  F_CONTIGUOUS : True
  OWNDATA : True
  WRITEABLE : True
  ALIGNED : True
  WRITEBACKIFCOPY : False
  UPDATEIFCOPY : False
```

Biz [0, 1, 2, 3, 4, 5] ma'lumotlarga ega bo'lgan *np.int* tipli 6 o'lchamdagi massivni (*ndarray*) yaratdik. Biz yaratilgan massivga *A* nomini berdik



Endi *reshape* usuli yordamida *A* massivining shaklini o'zgartirib, yangi *B* massivini yarating. Keling, u bir xil ma'lumotlardan foydalanayotganiga ishonch hosil qilaylik.

```
B = A.reshape(2, 3)
print(B)
print("B massiv A massiv ma'lumotlaridan foydalanadi?", B.base is A)
print("B massiv belgilari: \n", B.flags)
[[0 1 2]
 [3 4 5]]
B massiv A massiv ma'lumotlaridan foydalanadi? True
```

```
B massiv belgilari:  
C_CONTIGUOUS : True  
F_CONTIGUOUS : False  
OWNDATA : False  
WRITEABLE : True  
ALIGNED : True  
WRITEBACKIFCOPY : False  
UPDATEIFCOPY : False
```

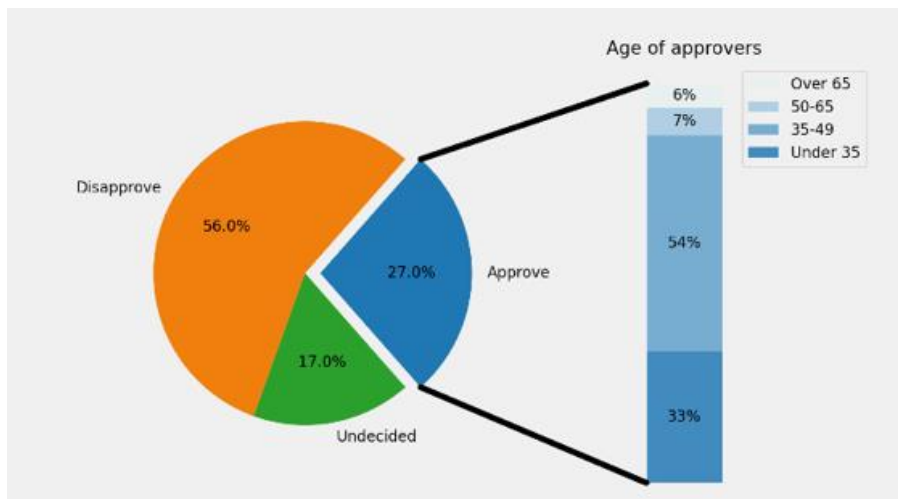
Grafiklar qurish uchun Matplotlib kutubxonasida foydalanish

Matplotlib - bu ma'lumotlarni vizualizatsiya qilish uchun ishlatiladigan mashhur Python kutubxonalaridan biri. U har qanday turdagi grafiklarni yaratish uchun ishlatiladi: vazifaga qarab chiziqli, doiraviy diagrammalar, chiziqli gistogrammalar va boshqalar. Ushbu bo'limda biz kutubxonani qanday import qilishni va ma'lumotlarni vizualizatsiya qilishning asosiy usullarini ko'rib chiqamiz.

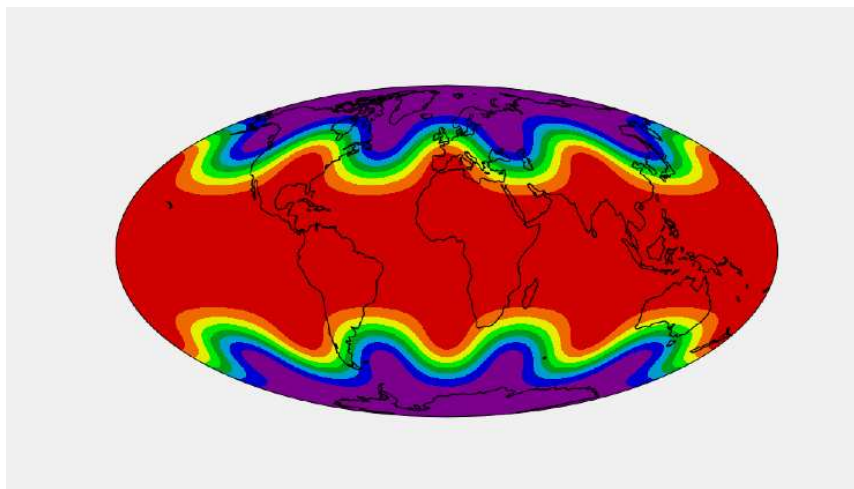
Matplotlib kutubxonasi Python-da ma'lumotlarni vizualizatsiya qilish paketi bo'lib, u sizga bir necha darajadagi ma'lumotlar bilan ishlash imkonini beradi:

- grafikni bir butun sifatida ko'rib chiqadigan *Pyplot* modulidan foydalanish;
- ob'ektga yo'naltirilgan interfeys orqali, har bir rasm yoki uning qismi alohida ob'ekt bo'lsa, ularning xususiyatlarini tanlab o'zgartirish va ko'rsatish imkonini beradi.

Ushbu bo'limda biz grafiklarni chizish uchun etarli bo'lgan *Pyplot* moduli bilan ishlaymiz. Matplotlib har qanday murakkablikdagi ma'lumotlarni vizuallashtirish uchun ishlatiladi. Kutubxona sizga har xil turdagi grafiklarni qurish imkonini beradi: chiziqli, uch o'lchovli, tarqoq chizmalar va boshqalar, shuningdek ularni birlashtirish. Qo'shimcha kutubxonalar ma'lumotlarni tahlil qilish imkoniyatlarini kengaytirish imkonini beradi. Masalan, **Cartopy** moduli kartografik ma'lumotlar bilan ishlash qobiliyatini qo'shadi.



Matplotlibda ikki turdagi grafiklarni birlashtirish



Cartopy moduli yordamida Yer xaritasida harorat taqsimotini vizualizatsiya qilish

Matplotlib-ning o'zi boshqa kutubxonalar uchun asosdir - masalan, **Seaborn** grafiklarni yaratishni osonlashtiradi va ularning tashqi ko'rinishini kosmetik jihatdan yaxshilash uchun ko'proq imkoniyatlarga ega. Ammo Matplotlib - bu ma'lumotlarni tahlil qilishda va ma'lumotlarni vizualizatsiya qilish uchun juda qulay kutubxona hisoblanadi.

Matplotlib — ni o'rganishda siz ikkita modul - **Pyplot** va **Pylab** haqida eslatib o'tishingiz kerak. Ishda qaysi birini ishlatishni va nima uchun paydo bo'lganligini tushunish muhimdir. Matplotlib kutubxonasi Python-da ma'lumotlarni vizualizatsiya qilish uchun to'plamdir. Pyplot-bu Matplotlib paketidagi modul. Siz uni ko'pincha kodda *matplotlib.pyplot* sifatida ko'rasiz. Modul avtomatik ravishda o'qlar, shakllar va boshqa komponentlarni yaratishga yordam beradi, bu qanday sodir bo'lishini o'ylamasdan. Bu ko'p hollarda ishlatiladigan Pyplot. Pylab-bu

Matplotlib to'plami bilan birga o'rnatiladigan yana bir modul. U bir vaqtning o'zida pyplot va numpy kutubxonasini interaktiv massivlar bilan ishlash yoki ma'lumotlar bilan ishlashda chizmachilik funksiyalariga kirish uchun import qiladi. Endi Pylab faqat tarixiy ahamiyatga ega-bu MATLAB-dan Matplotlib-ga o'tishni osonlashtirdi, chunki u import operatorlarisiz ishlashga imkon berdi (ya'ni MATLAB foydalanuvchilari ishlashga odatlangan). Siz Pylab bilan turli saytlardagi kod misollarida uchrashishingiz mumkin, ammo amalda moduldan foydalanishingiz shart emas.

Matplotlib kutubxonasini o'rnatish va foydalanish

Matplotlib-Windows, macOS va Linux-da Python-da ishlaydigan universal kutubxona. Google Colab yoki Jupyter Notebook bilan ishlashda Python va Matplotlib — ni o'rnatish shart emas. Ammo Agar siz Visual Studio Code kabi boshqa IDE-da kod yozishga qaror qilsangiz, avval Python-ni, so'ngra terminal orqali o'rganish kutubxonasini o'rnatishingiz kerak bo'ldi:

```
pip3 install matplotlib
```

Kutubxonani import qilish quyidagicha amalga oshiriladi:

```
import matplotlib.pyplot as plt
```

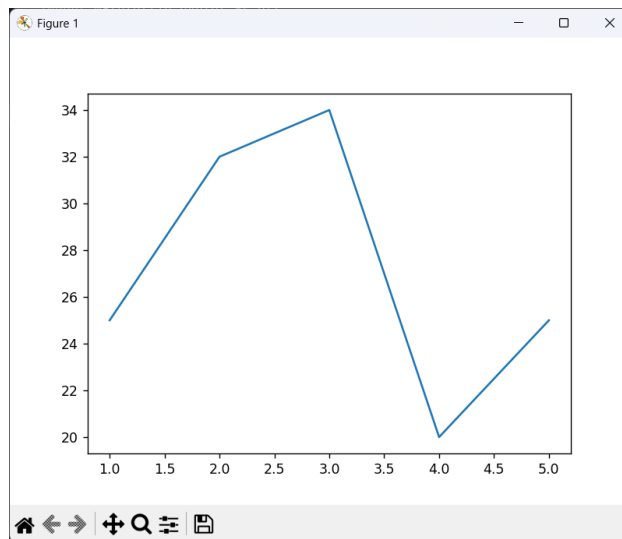
Kutubxona uchun **plt** qisqartmasi keng tarqalgan. Siz uni rasmiy hujjatlarda, kitoblarda va boshqa odamlarning noutbuklarida uchratasiz. Shunday qilib, undan foydalaning. Boshlash uchun biz ikkita o'zgaruvchini yaratamiz — x va y , ular x va y o'qlari bo'ylab nuqtalarning koordinatalarini o'z ichiga oladi:

```
x = [1, 2, 3, 4, 5]
y = [25, 32, 34, 20, 25]
```

Endi biz ushbu nuqtalarni bog'laydigan grafik tuzamiz:

```
plt.plot(x, y)
plt.show()
```

Natija:



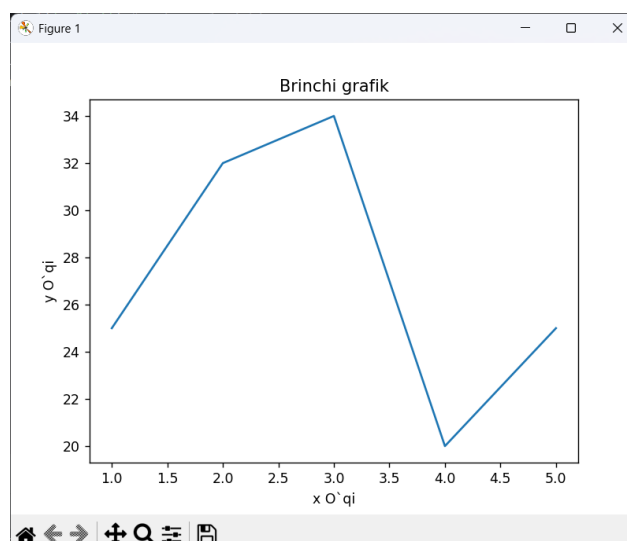
Biz oddiy chiziqli grafikni oldik. Keling, har bir kod qatorini tahlil qilaylik:

- `plt.plot()` — unga berilgan qiymatlarga muvofiq grafik tuzadigan standart funktsiya. Biz unga nuqta koordinatalarini berdik;
- `plt.show()` - ekranda ko'rsatilgan ma'lumotlarni ko'rsatish uchun mas'ul bo'lgan funktsiya. Siz buni ko'rsatmasligingiz mumkin, ammo keyin chiroyli rasmga qo'shimcha ravishda biz turli xil texnik ma'lumotlarni ko'ramiz.

Keling, birinchi grafigimizni sarlavha va eksa imzolari bilan to'ldiraylik:

```
plt.plot(x, y)
plt.xlabel('x o'qi') #x o'q nomlanishi
plt.ylabel('y o'qi') #y o'q nomlanishi
plt.title('Birinchi grafik') #Nomi
plt.show()
```

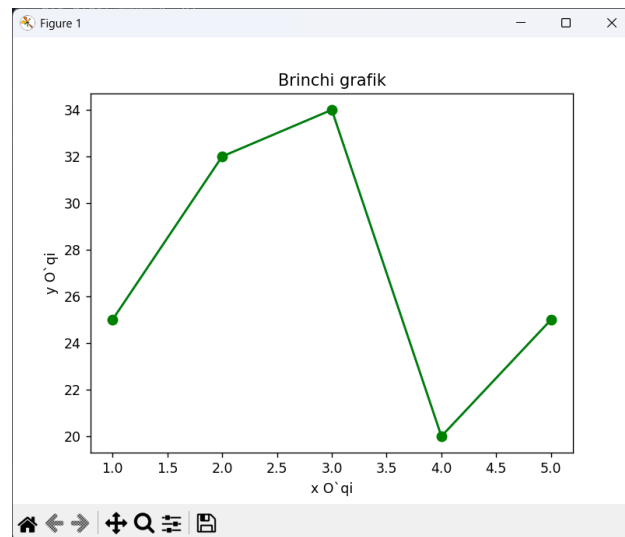
Natija:



Matplotlib yordamida har qanday grafikning ko‘rinishini sozlash mumkin. Masalan, biz chiziq rangini o‘zgartirishimiz mumkin, shuningdek koordinatalari o‘zgaruvchilarda berilgan nuqtalarni ajratib ko‘rsatishimiz mumkin:

```
plt.plot(x, y, color='green', marker='o', markersize=7)
```

Natija:



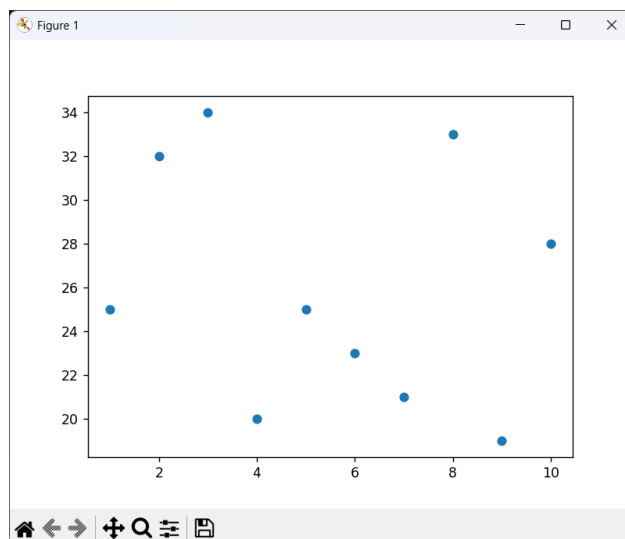
Tarqoq yoki scatterplot diagrammalar

Tarqoq diagramma qiymatlari turli o‘qlarda chizilgan ikkita o‘zgaruvchi o‘rtasidagi munosabatni baholash uchun ishlatiladi. Uni qurish uchun *plt.scatter()* funksiyasidan foydalaniladi, uning argumentlari diskret qiymatlarga ega o‘zgaruvchilar:

```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [25, 32, 34, 20, 25, 23, 21, 33, 19, 28]

plt.scatter(x, y)
plt.show()
```

Tarqoq diagramma ko‘plab individual nuqtalarga o‘xshaydi:



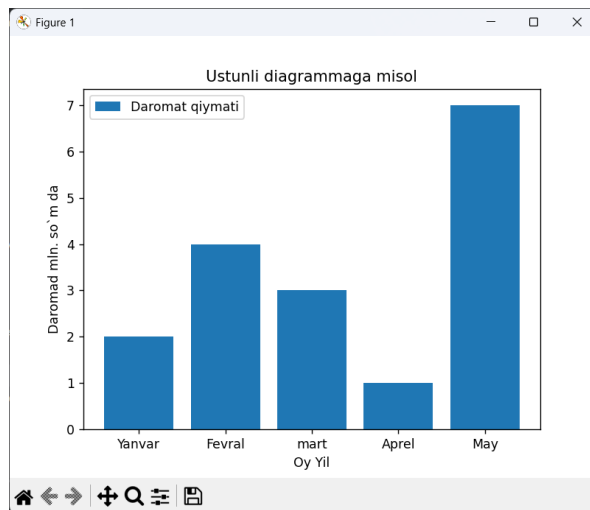
Ustunli diagramma

Ushbu turdagi diagramma sizga individual o'zgaruvchilar qiymatlarini qulay tarzda solishtirish imkonini beradi. Ustunli diagrammada ustunlar uzunligi ular ifodalagan o'lchovlarga mutanosibdir. Odatda, o'qlardan biri bitta toifaga, ikkinchisi esa uning diskret qiymatiga mos keladi. Misol uchun, ustunli diagramma sizga oylar bo'yicha foyda miqdorini aniq ko'rsatishga imkon beradi. Buni quyidagi misolda ko'raylik:

```
x = [Yanvar, 'Fevral', 'Mart', 'Aprel', 'may']
y = [2, 4, 3, 1, 7]

plt.bar(x, y, label='Daromat qiymati') #Yorliq parametri legenda
uchun qiymat nomini belgilash imkonini beradi.
plt.xlabel('Yil oyi')
plt.ylabel('Daromat mln. So'm da')
plt.title('Ustunli diagrammaga misol')
plt.legend()
plt.show()
```

Ustunli diagramma sizga foydaning oylar bo'yicha o'zgarishi dinamikasini ko'rish imkonini beradi:



Bir necha turdagi diagrammalarni kombinatsiyalash

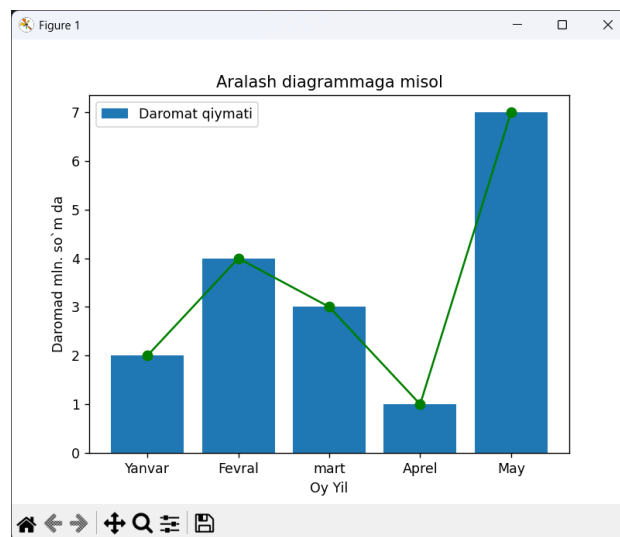
Ba'zi vazifalar uchun bir nechta grafik turlarini birlashtirish foydalidir, masalan, ustunli diagramma va chiziqli diagramma. Ustunlarga foyda qiymatlari bilan nuqta qo'shib, uni yanada ifodalibroq qilish mumkin bo'ladi:

```
x = [Yanvar, 'Fevral', 'Mart', 'Aprel', 'may']
y = [2, 4, 3, 1, 7]

plt.bar(x, y, label='Daromat qiymati') # Yorliq parametri legenda
uchun qiymat nomini belgilash imkonini beradi.
plt.plot(x, y, color='green', marker='o', markersize=7)

plt.xlabel('Yil oyi')
plt.ylabel('Daromat mln. So'm da')
plt.title('Aralash grafik')
plt.legend()
plt.show()
```

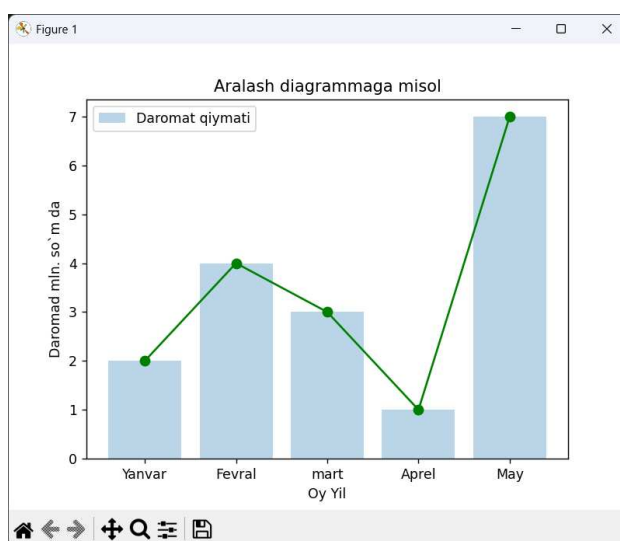
Natja:



Grafik paydo bo'ldi ammo endi chiziqli grafik yomon ko'rinadi - u shunchaki ustunlarning ko'k fonida yo'qolib qolmoqda. *alpha* parametridan foydalanib, ustunlarni shaffofligini boshqarish mumkin bo'ladi, misol uchun *alpha* ni 0.5 qiymatga belgilaylik:

```
plt.bar(x, y, label=Daromat qiymati, alpha=0.5)
```

Alpha parametri 0 dan 1 gacha bo'lgan qiymatlarni qabul qilishi mumkin, bu erda 0 to'liq shaffoflik va 1 shaffoflik yo'q. Natija:



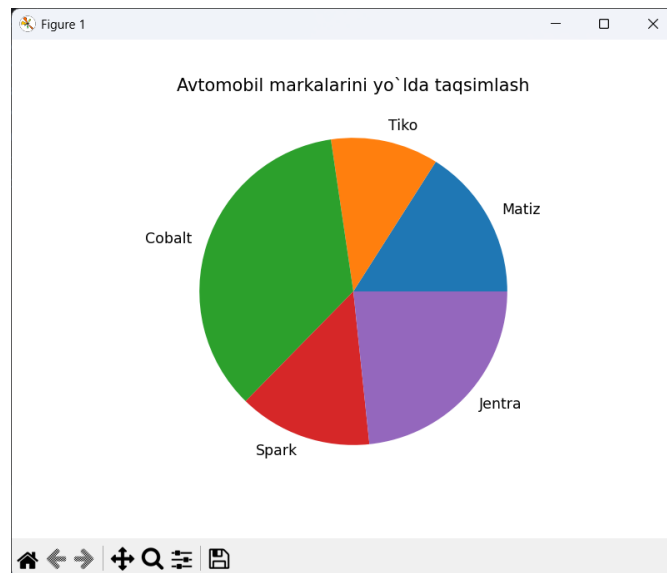
Doira diagrammasi

Doira diagrammasi guruhlar tarkibini ko'rsatish uchun ishlatiladi. Masalan, biz shahar yo'llarida qaysi avtomobil markalari ustunlik qilishini aniq ko'rsatishimiz mumkin:

```
vals = [24, 17, 53, 21, 35]
labels = ["Matiz", "Tiko", "Cobalt", "Spark", "Jentra"]

plt.pie(vals, labels=labels)
plt.title("Avtomobil markalarini yo'lda taqsimlash ")
plt.show()
```

Natija:

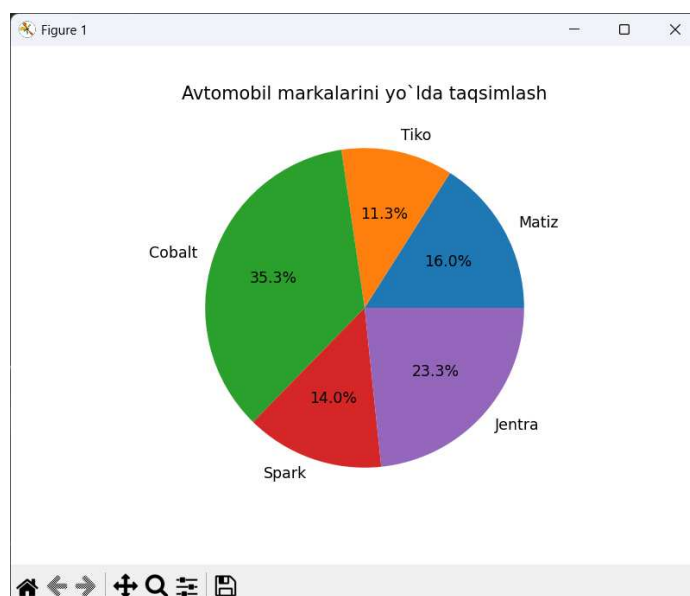


Shunday qilib, ma'lumotlar aniqroq, ammo har bir avtomobil markasiga qaysi ulush to'g'ri kelishi aniq emas. Shuning uchun doira diagrammalarini har doim foiz qiymatlari bilan to'ldirish yaxshiroqdir. Diagramani yaxshilash uchun *pie* funksiyasiga *autopct* parametrini qo'shish orqali kodimizni tahrirlaymiz:

```
vals = [24, 17, 53, 21, 35]
labels = ["Matiz", "Tiko", "Cobalt", "Spark", "Jentra"]

plt.pie(vals, labels=labels, autopct='%1.1f%%')
plt.title("Avtomobil markalarini yo`lda taqsimlash ")
plt.show()
```

Parametrga biz raqamni ko'rsatish formatini qo'shdik. Bizning holatda, bu verguldan keyin bitta belgi bo'lgan butun son. Kodni ishga tushirib quyidagi natijaga ega bo'lmiz:



Endi toifalarni taqqoslash osonroq, chunki biz raqamli qiymatlarni yaqol ko'rishimiz mumkin bo'ladi.

Murakkab vizualizatsiya

Keling ustunli grafik yarataylik. Bu bitta o'zgaruvchining qiymatlari nisbati dinamikasini baholashga imkon beradi. Keling, turli yillarda Android va iOS-dagi qurilmalar soni qanday bog'liqligini ko'rsatishga harakat qilaylik:

```
labels = ['2020', '2021', '2022', '2023', '2024']
android_users = [85, 85.1, 86, 86.2, 86]
ios_users = [14.5, 14.8, 13, 13.8, 14.0]

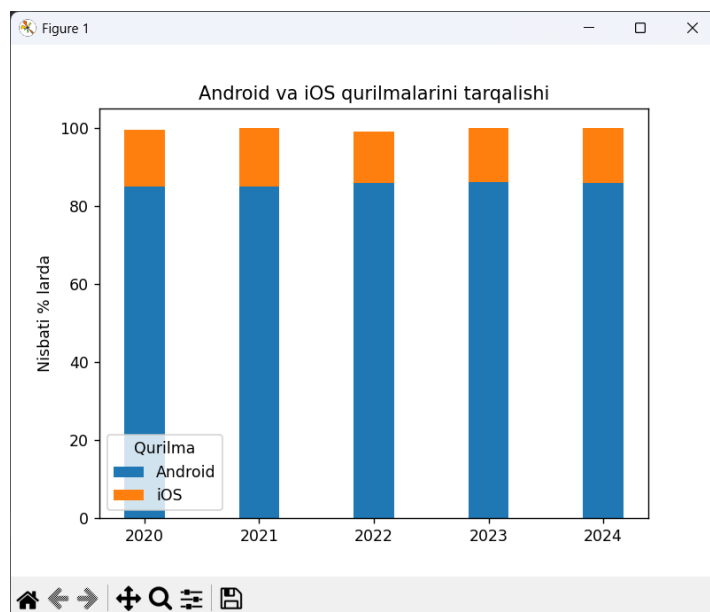
width = 0.35 #Ustun kengligi
fig, ax = plt.subplots()

ax.bar(labels, android_users, width, label='Android')
ax.bar(labels, ios_users, width, bottom=android_users,
        label='iOS') # Pastki parametrdan foydalanib, ustundagi
                    # qiymatlar android_users o'zgaruvchisi qiymatlaridan yuqori
                    # bo'lishi kerakligini aniqlaymiz.

ax.set_ylabel('Nisbati % larda')
ax.set_title('Android va iOS qurilmalarini tarqalishi')
ax.legend(loc='lower left', title='Qurilma') # Grafikning bir
        # qismini qoplamasligi uchun legendani pastki chap burchakka
        # o'tkazing

plt.show()
```

Natija:



Pandas kutubxonasi.

Pandas kutubxonasi Python-da ishlaydigan Data Science, Analytics va SI arsenalidagi eng muhim vositadir. Kuchli mashinani o'rganish vositalari va yorqin vizualizatsiya imkoniyatlari dasturchilar diqqatni jalb qiliali, ammo ma'lumotlar bilan ishlash loyihalarining aksariyati pandas-ga asoslangan. Pandas nomi panel ma'lumotlari (*ingl. panel data*) nomidan olingan. Ekonometriyada ular ko'p o'lchovli tuzilgan ma'lumotlar to'plamidir. Agar siz ma'lumot ilmi yoki SI sohasida ishlashni bo'lsangiz, unda sizning birinchi duch keladigan kutubxolalarinigzdan biri bu pads bo'lidi bo'lishi kerak. Ushbu bo'limda biz pandas haqida asosiy ma'lumotlarni bilan tanishi chiqamiz. Siz ushbu paketni qanday o'rnatishni va undan qanday foydalanishni bilib olasiz. Bundan tashqari, biz pandas matplotlib va scikit-learn kabi boshqa mashhur Python ma'lumotlarini tahlil qilish paketlari bilan qanday ishlashini ko'rib chiqamiz.

Pandas juda ko'p jarayonlarda foydalaniladi, shuning uchun uning imkoniyatlarini emas, balki nima qila olmasligini sanab o'tish yaxshiroqdir. Asosan, ushbu vosita sizning ma'lumotlaringiz uchun uydur. Pandas yordamida siz ma'lumotlaringizni tozalash, o'zgartirish va tahlil qilish imkoniyatlariga ega bo'lsiz.

Aytaylik, siz kompyuteringizda CSV ((*ingl. Comma-Separated Values qiymatlar vergul bilan ajratilgan*) jadval ma'lumotlarini ifodalash uchun mo'ljallangan matn formati. Jadval qatori vergul bilan ajratilgan bir yoki bir nechta maydonlarni o'z ichiga olgan matn qatoriga mos keladi)) formatida saqlangan ma'lumotlar to'plamini o'rganmoqchisiz. Pandas ushbu CSV — dan ma'lumotlarni *DataFrame*-ga chiqaradi-asosan jadval-va keyin sizga quyidagilarni bajarishga imkon beradi:

- Statistika hisoblashni amalga oshirish va ma'lumotlar haqidagi quyidagi savollarga javob bering mumkin:
 - Har bir ustun uchun o'rtacha, o'rtacha, maksimal yoki minimal nima?
 - A ustuni B ustuni bilan bog'liqligi?
 - C ustunidagi ma'lumotlar taqsimoti qanday ko'rinishga ega?

- Ma'lumotlarni tozalash, masalan, etishmayotgan qiymatlarni o'chirish yoki biron bir mezon bo'yicha qatorlar yoki ustunlarni filtrlash orqali.
- Matplotlib yordamida ma'lumotlarni vizualizatsiya qilish. Siz ustunli diagrammalar, chiziqli grafikalar, gistogrammalar tuzishingiz mumkin.
- Tozalangan, o'zgartirilgan ma'lumotlarni CSV va boshqa fayl formatlariga yoki ma'lumotlar bazasiga saqlash.

Modellashtirish yoki murakkab vizualizatsiyani boshlashdan oldin, ma'lumotlar to'plamining mohiyatini yaxshi tushunish kerak va pandas buning uchun eng yaxshi vositadir.

Pandas **Data Science** asboblari qutisida qanday o'rinni egallaydi? Pandas kutubxonasi nafaqat ma'lumotlar xaqidagi fanlari uchun asboblari to'plamining markaziy komponenti, balki ushbu to'plamdagi boshqa kutubxonalar bilan birgalikda ham qo'llaniladi. Pandas **NumPy** to'plami asosida qurilgan, ya'ni ko'plab NumPy tuzilmalari pandas-da ishlatiladi. Pandas-dagi ma'lumotlar ko'pincha **SciPy**-da statistik tahlil qilish uchun, **Matplotlib**-dan grafik funktsiyalarda va **scikit-learn**-da mashinani o'rganish algoritmlarida ishlatiladi.

Pandas-dan foydalanishni qachon boshlash kerak? Agar sizda Python dasturlash tajribasi bo'lmasa, unda siz hali pandasni o'rganishdan bosh tortganingiz ma'qul. Tajribali ishlab chiquvchi darajasi shart emas, lekin siz Python bilan ishlash asoslarini bilishingiz kerak (ro'yxatlar, kortejlar, lug'atlar, funktsiyalar, iteratsiyalar va boshqalar). Yuqorida aytib o'tilgan o'xshashlik tufayli avval NumPy bilan tanishish tavsiya etiladi. **Data science** yoki **SI** yo'nalishidagi masalalar bilan shug'ulanishdan oldin ham pandasni mustaqil ravishda o'rganishni boshlashli tavsiya etiladi.

Pandasni o'rnatish, import qilish va ishlash

Pandas-o'rnatish oson. Terminalni yoki buyruq satrini oching va quyidagi buyruqlardan biri yordamida paketni o'rnatang:

```
conda install pandas
```

Yoki:

```
pip install pandas
```

Pandasni import qilishda odatda qisqaroq paket nomi ishlatiladi, chunki siz uni tez-tez terishingiz kerak bo'ladi:

```
import pandas as pd
```

Pandas ning asosiy kompanetlari: Series n DataFrame

Pandasning ikkita asosiy komponenti **Series** va **DataFrame**. Seriya asosan ustun, DataFrame esa Seriya to'plamidan tashkil topgan ko'p o'lchovli jadvaldir.

Series			Series			DataFrame	
	apples			oranges			
0	3	+	0	0	=	0	3
1	2		1	3		1	2
2	0		2	7		2	0
3	1		3	2		3	1
							oranges
							0
							3
							7
							2

DataFrame va seriyasi juda o'xshash, chunki siz ulardan biri bilan bajarishingiz mumkin bo'lgan ko'plab operatsiyalarni boshqasi bilan ham bajarish mumkin, masalan, nol qiymatlarni to'ldirish va o'rtacha qiymatni hisoblash. Ma'lumotlar bilan ishlashni boshlaganimizda, siz ushbu komponentlarni amalda ko'rasiz.

Pandas da DataFrame -ni noldan yaratish.

DataFrame — ni to'g'ridan-to'g'ri Python-da qanday yaratishni bilish foydali u hujjatlarida topilgan yangi usullar va xususiyatlarni sinab ko'rishda sizga yordam beradi. DataFrame noldan yaratishning ko'plab usullari mavjud, ammo ajoyib variant oddiy lug'atdan foydalanishdir. Aytaylik, bizda olma va apelsin sotadigan mevali do'kon bor. Biz har bir meva uchun ustun va har bir mijoz sotib olish uchun qatorga ega bo'lishni xohlaymiz. Buni pandas uchun lug'at sifatida tashkil qilish uchun biz quyidagicha kodni shakilantiramiz:

```
data = { 'apples': [3, 2, 0, 1], 'oranges': [0, 3, 7, 2] }
```

So'ngra ma'lumotlarni DataFrame konstruktoriga o'tkazamiz:

```
purchases = pd.DataFrame(data)
```

```
purchases
```

Natija:

	Apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Bu qanday ishlaydi? Ma'lumotlardagi har bir kalit-qiymat elementi olingan ma'lumotlar doirasidagi ustunga mos keladi. Ushbu DataFrame indeksi dataframe-ni 0-3 raqamlari shaklida yaratishda berilgan. Ammo DataFrame ishga tushirilganda indeksni o'zingiz ham yaratishingiz mumkin. Mijozlar nomlari indeks sifatida bo'lsin:

```
purchases = pd.DataFrame(data, index=['June', 'Robert', 'Lily', 'David'])  
purchases
```

	Apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

Endi biz mijozning buyurtmasini uning nomi bilan topishimiz mumkin.

```
purchases.loc['June']
```

Eslatma: *loc locate* so'zining qisqartmasi bo'lib, ingliz tilida "o'rnini topish" degan ma'noni anglatadi. Natija:

```
apples 3  
oranges 0  
Name: June, dtype: int64
```

Dataframe pandas-da ma'lumotlarni qanday o'qish mumkin.

Ma'lumotlarni turli formatdagi fayllardan DataFrame-ga yuklash juda oson. Quyidagi misollarda biz olma va apelsin ma'lumotlaridan foydalanishda davom etamiz, ammo bu safar ular fayllardan keladi.

CSV fayllaridan ma'lumotlarni o'qish. CSV fayllaridan ma'lumotlarni yuklab olish uchun bitta satr kifoya qiladi:

```
df = pd.read_csv('purchases.csv')
df
```

Natija:

	Unnamed: 0	Apples	oranges
0	June	3	0
1	Robert	2	3
2	Lily	0	7
3	David	1	2

CSV - larda bizning DataFrame - miz kabi indekslar mavjud emas, shuning uchun biz faqat o'qish paytida *index_col*-ni aniqlashimiz kerak:

```
df = pd.read_csv('purchases.csv', index_col=0)
df
```

Natija:

	Apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

Bu erda biz indeks nol ustun bo'lishini belgilaymiz. Ko'pgina CSV-larda indeks ustuni yo'q, shuning uchun odatda bu qadam haqida tashvishlanishingiz shart emas.

JSON-dan ma'lumotlarni o'qish. Agar sizda Python-ning saqlanadigan lug'ati bo'lgan JSON (*ingl. JavaScript Object Notation - JavaScript-ga asoslangan ma'lumotlar almashinuvining matnli formati. Boshqa ko'plab matn formatlari singari, JSON ham o'qilishi oson. JSON formati Duglas Krokford tomonidan ishlab chiqilgan*) fayli bo'lsa, pandas uni osonlikcha o'qiy oladi:

```
df = pd.read_json('purchases.json')
df
```

Natija:

	Apples	oranges

David	1	2
June	3	0
Lily	0	7
Robert	2	3

E'tibor bering, bu safar biz to'g'ri indeksni oldik, chunki JSON-dan foydalanish indekslarni joylashuv orqali ishlashga imkon beradi. *data_file.json* faylini ochishga harakat qiling qanday ishlashini tushinish uchun uni oddit matn tedaktori orqali ochi ko'ring.

Pandas sizning JSON-ning tuzilishini tahlil qilish orqali DataFrame-ni qanday yaratishni tushunishga harakat qiladi va ba'zida u buni uddalay olmaydi. Siz tez-tez JSON satr formatiga qarab nomlangan *orient* argumentini so'rashingiz kerak bo'ladi. Bu haqida *read_json* hujjatlarida o'qishingiz mumkin.

SQL ma'lumotlar bazasidan ma'lumotlarni o'qish. Agar siz SQL ma'lumotlar bazasidagi ma'lumotlar bilan ishlayotgan bo'lsangiz, avval tegishli Python kutubxonasi yordamida ulanishni o'rnatishingiz va keyin so'rovni pandas-ga o'tkazishingiz kerak. Misol uchun biz **SQLite**-dan foydalanamiz. Avval siz *pysqlite3*-ni o'rnatishingiz kerak. Bunig uchun terminalda quyidagi buyruqni bajaring:

```
pip install pysqlite3
```

SQLite3 ma'lumotlar bazasiga ulanish uchun ishlatiladi, undan keyin **SELECT** so'rovi yordamida ma'lumotlar ramkasini yaratish uchun foydalanishimiz mumkin. Shunday qilib, avval biz SQLite ma'lumotlar bazasi fayliga ulanishni yaratamiz:

```
import sqlite3
con = sqlite3.connect("database.db")
```

Ushbu SQLite ma'lumotlar bazasida bizda xaridlar deb nomlangan jadval mavjud va bizning indeksimiz indeks ustunidir. SELECT so'rovini va *con*-ni topshirgandan so'ng, biz xarid qilish jadvalidan (*purchases*) o'qishimiz mumkin:

```
df = pd.read_sql_query("SELECT * FROM purchases", con)
```

```
df
```

Natija:

		Apples	oranges
0	June	3	0
1	Robert	2	3
2	Lily	0	7
3	David	1	2

CSV misolidagi kabi, biz `index_col= 'index'` ni uzatishimiz mumkin, lekin indeksni haqiqatdan keyin ham o'rnatishimiz mumkin:

```
df = df.set_index('index')  
df
```

Natija:

	Apples	oranges
Index		
June	3	0
Robert	2	3
Lily	0	7
David	1	2

Ma'lumotlarni CSV, JSON yoki SQL-ga qaytarish. Shunday qilib, ma'lumotlarni tozalash uchun uzoq vaqt ishlaganingizdan so'ng, siz ularni faylga saqlashga tayyormsiz. Pandas nafaqat o'qish uchun, balki ma'lumotlarni saqlash uchun ham intuitiv buyruqlarni taqdim etadi:

```
df.to_csv('new_purchases.csv')  
df.to_json('new_purchases.json')  
df.to_sql('new_purchases', con)
```

JSON va CSV fayllarini saqlaganimizda, ushbu funktsiyalarga tegishli kengaytmaga ega bo'lgan kerakli fayl nomini o'tkazishimiz kerak. SQL - dan foydalanganda biz yangi fayl yaratmaymiz, balki `con` o'zgaruvchimiz yordamida ma'lumotlar bazasiga yangi jadval kiritamiz.

O'z-o'zini nazorat qilish uchun savollar

1. Nega aynan Python sun'iy intellekt uchun?

2. Python'ning xususiyatlari?
3. Python ni o'rnatish?
4. Python integratsiyalashgan ishlab chiqish muhiti?
5. NumPy kutubxonasi nima uchun kerak?
6. NumPy da massivlari va ularni yaratish?
7. Numpy massivlarida arifmetik amallar bajarish?
8. Matplotlib yordamida diagrammalar yaratish?
9. Pandas nima uchun kerak?
10. Pandas ning asosiy kompanetlari: Series va DataFrame?

II-BOB. SUN'IY INTELLEKTADA O'RGATISH ALGORITIMLARI.

2.1-§. MASHINALI O'RGANISH - MACHINE LEARNING

O'rganish (o'qitish) bilim yoki ko'nikmalarni o'rganish yoki tajriba orqali bilim olishni anglatadi. Shundan kelib chiqqan holda, mashinaviy o'rganishni (Machine Learning - ML) quyidagicha ta'riflashimiz mumkin.

Ushbu sohani kompyuter fanlarining bir tarmog'i sifatida aniqlash mumkin, aniqrog'i, bu sun'iy intellektni qo'llash orqali kompyuter tizimlariga ma'lumotlar asosida o'rganish va tajriba asosida yaxshilanish imkonini beradigan soha. Bunda aniq dasturlash talab qilinmaydi.

Aslida, mashinaviy o'rganishning asosiy maqsadi kompyuterlarga avtomatik ravishda o'rganish imkonini berish, odamning aralashuvi holda. Endi savol tug'iladi: bunday o'rganishni qanday boshlash mumkin? O'rganish jarayonini ma'lumotlarni kuzatish orqali boshlash mumkin. Ma'lumotlar, shuningdek, ba'zi misollar, ko'rsatmalar yoki bevosita tajriba sifatida ham bo'lishi mumkin. So'ngra, ushbu kirish ma'lumotlaridan asoslanib, mashina eng yaxshi qarorni qabul qiladi va ma'lumotlar orasida ba'zi qonuniyatlarni qidiradi.

Mashinali o'rganish turlari (ML)

Mashinaviy o'rganish algoritmlari kompyuter tizimiga aniq dasturlashsiz o'rganishga yordam beradi. Bu algoritmlar nazorat ostidagi va nazoratsiz bo'lishi mumkin. Keling, endi ba'zi algoritmlarni ko'rib chiqaylik:

Boshqariladigan mashinani o'rganish algoritmlari

Bu mashinaviy o'rganish algoritmlarining eng ko'p ishlatiladigan turi hisoblanadi. U "nazorat ostida" deb ataladi, chunki algoritmni o'rganish jarayoni o'quv ma'lumotlar to'plami asosida o'qituvchining nazoratini amalga oshirishi mumkin. Ushbu turdagi algoritmlarda mumkin bo'lgan natijalar oldindan ma'lum bo'ladi va o'quv ma'lumotlari to'g'ri javoblar bilan belgilangan bo'ladi. Buni quyidagicha tushunish mumkin –

Faraz qilaylik, bizda kirish o'zgaruvchilari x va chiqish o'zgaruvchisi y mavjud va biz algoritmni kirish ma'lumotlarini chiqishlarga o'zgartirish funksiyasini o'rganish uchun qo'lladik, masalan.

$$Y=f(x)$$

Endi asosiy maqsad - tasvirlash funksiyasini shunchalik yaxshi yaqinlashtirishki, yangi kirish ma'lumotlari (x) mavjud bo'lganda, biz ushbu ma'lumotlar uchun chiqish o'zgaruvchisini (Y) bashorat qila olishimiz kerak. Nazorat ostida yuzaga keladigan muammolarni ikkita asosiy turga bo'lish mumkin.

Nazoratli mashinani o'rganishdagi muammolarni ikki turga bo'lish mumkin:

- **Klassifikatsiya:** Agar bizda kategoriyalangan chiqish ma'lumotlari mavjud bo'lsa, masalan, "qora", "o'quvchi", "o'quvchi emas" va hokazo, bunday holat klassifikatsiya muammosi deb ataladi.
- **Regressiya:** Agar bizda haqiqiy qiymatni chiqarish talab qilinadigan muammo mavjud bo'lsa, masalan, "masofa", "kilogramm" va hokazo, bunday holat regressiya muammosi deb ataladi.

O'qituvchisiz mashinani o'rganish algoritmlari

Nomidan ham ko'rinib turibdiki, ushbu turdagi mashinaviy o'rganish algoritmlarida hech qanday o'qituvchi yo'q, ya'ni ular hech qanday ko'rsatma bermaydi. Shu sababli, o'qituvchisiz mashinaviy o'rganish algoritmlari ba'zi odamlar tomonidan haqiqiy sun'iy intellekt deb ataladi. Buni quyidagicha tushunish mumkin:

- Faraz qilaylik, bizda kirish o'zgaruvchisi x bor, shunda nazorat ostida o'rganish algoritmlaridagi kabi tegishli chiqish o'zgaruvchilari bo'lmaydi.
- Oddiy so'zlar bilan aytganda, o'qituvchisiz o'rganishda to'g'ri javob bo'lmaydi va o'qituvchichi ham yo'q. Algoritmlar ma'lumotlarda qiziqarli qonuniyatlarni aniqlashga yordam beradi.

O'qituvchisiz o'rganish muammolari quyidagi ikki turga bo'linishi mumkin:

- **Klasterizatsiya:** Klasterizatsiya muammolarida ma'lumotlardagi ajralmas guruhlarini aniqlashimiz kerak. Masalan, mijozlarni ularning xarid qilish xulqiga qarab guruhlash.
- **Assotsiatsiya:** Bu muammo assotsiatsiya muammosi deb ataladi, chunki bunday turdagi muammolar bizning ma'lumotlarimizning katta qismlarini tasvirlaydigan qoidalarni aniqlashni talab qiladi. Masalan, x va y ni birgalikda xarid qiladigan mijozlarni izlash.

K-tizimli klasterlash, apriori assotsiatsiya algoritmi - bu o'qituvchisiz mashinaviy o'rganish algoritmlariga misollar.

Mustahkamlash bilan mashinani o'rganish algoritmlari

Bunday turdagi mashinaviy o'rganish algoritmlari ancha kamroq ishlatiladi. Ushbu algoritmlar tizimlarni aniq qarorlar qabul qilishni o'rgatadi. Aslida, mashina bir muhitda bo'lib, u doimiy ravishda sinov va xatolar usuli bilan o'rganadi. Bu algoritmlar o'tmish tajribasidan o'rganib, aniq qarorlar qabul qilish uchun mumkin bo'lgan barcha bilimlarni yig'ishga harakat qiladi. Markov qaror qabul qilish jarayoni - mustahkamlash asosida mashinaviy o'rganish algoritmlarining bir misolidir.

Mashinani o'rganishning eng keng tarqalgan algoritmlari

Ushbu bo'limda biz eng keng tarqalgan mashinani o'rganish algoritmlari bilan tanishamiz. Algoritmlar quyida tasvirlangan.

Chiziqli regressiya

Bu statistika va mashinani o'rganishda eng mashhur algoritmlardan biridir. Asosiy konsepsiya: Asosan, chiziqli regressiya — bu chiziqli model bo'lib, u kirish o'zgaruvchilari, masalan, x va bitta chiqish o'zgaruvchisi, masalan, y o'rtasida chiziqli bog'liqlikni taxmin qiladi. Boshqacha aytganda, y ni kirish o'zgaruvchilari x ning chiziqli kombinatsiyasidan hisoblash mumkin. O'zgaruvchilar orasidagi bog'liqlik eng yaxshi chiziqni tanlash orqali aniqlanishi mumkin.

Chiziqli regressiyaning turlari

Chiziqli regressiya quyidagi ikki turga bo'linadi:

- **Oddiy chiziqli regressiya:** Agar chiziqli regressiya algoritmda faqat bitta mustaqil o'zgaruvchi bo'lsa, u holda bu oddiy chiziqli regressiya deb ataladi.
- **Ko'p o'lchovli chiziqli regressiya:** Agar chiziqli regressiya algoritmda bir nechta mustaqil o'zgaruvchilar bo'lsa, bu ko'p o'lchovli chiziqli regressiya deb ataladi.

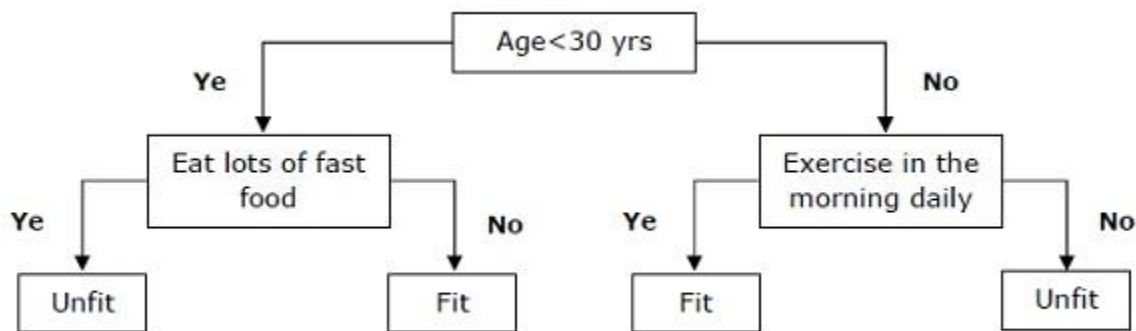
Chiziqli regressiya asosan uzluksiz o'zgaruvchi (o'zgaruvchilar) asosida haqiqiy qiymatlarni baholash uchun ishlatiladi. Masalan, bir kun ichida do'kondagi jami savdolarini haqiqiy qiymatlar asosida chiziqli regressiya yordamida baholash mumkin.

Logistik regressiya

Bu klassifikatsiya algoritmi bo'lib, logit regressiya deb ham tanilgan. Asosan, logistik regressiya — bu klassifikatsiya algoritmi bo'lib, u berilgan mustaqil o'zgaruvchilar to'plami asosida diskret qiymatlarni (masalan, 0 yoki 1, to'g'ri yoki noto'g'ri, ha yoki yo'q) baholash uchun ishlatiladi. Asosan, bu algoritm ehtimollikni taxmin qiladi, shuning uchun uning chiqish qiymatlari 0 dan 1 gacha bo'ladi.

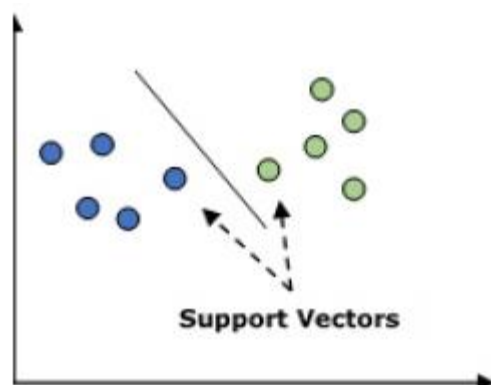
Qarorlar daraxti

Qaror daraxti — bu nazorat ostida o'rganish algoritmi bo'lib, asosan klassifikatsiya vazifalari uchun ishlatiladi. Aslida, bu mustaqil o'zgaruvchilar asosida rekursiv bo'linish shaklida ifodalangan klassifikator hisoblanadi. Qaror daraxti tugunlarga ega bo'lib, ular ildiz daraxtini tashkil qiladi. Ildiz daraxti — bu yo'naltirilgan daraxt bo'lib, uning "root" deb ataladigan tuguni mavjud. Root tugunida kiruvchi qirralar yo'q, barcha boshqa tugunlarda esa bitta kiruvchi qirra mavjud. Ushbu tugunlar "yaproqlar" yoki "qaror qabul qilish tugunlari" deb ataladi. Masalan, quyidagi qaror daraxtini ko'rib chiqaylik, bunda odamga mos keladimi yoki yo'qmi aniqlanadi.



Mashinaviy vektorlarni qo‘llash (Support Vector Machine - SVM)

U klassifikatsiya va regressiya vazifalarida ishlatiladi. Ammo asosan klassifikatsiya vazifalarida qo‘llaniladi. SVMning asosiy konsepsiyasi har bir ma’lumot elementini n -o‘lchovli fazoda nuqta sifatida tasvirlashda yotadi, bunda har bir obyektning qiymati ma’lum bir koordinataning qiymati bo‘ladi. Bu yerda n — bizda mavjud bo‘lgan obyektlar sonidir. Quyida SVM konsepsiyasini tushunish uchun oddiy grafikiy tasvir keltirilgan.



Yuqorida keltirilgan diagrammada bizda ikkita funksiya mavjud, shuning uchun avvalo, bu ikki o‘zgaruvchini ikki o‘lchovli fazoda tasvirlashimiz kerak, bunda har bir nuqta ikkita koordinatga ega bo‘ladi, ular "opornye vektory" deb ataladi. Chiziq ma’lumotlarni ikkita turli klassifikatsiyalangan guruhga bo‘lib tashlaydi. Ushbu chiziq esa klassifikator bo‘ladi.

Sodda Bayes klassifikatori.

Bayes klassifikatori (Naive Bayes classifier) — Bayes formulasiga asoslangan ehtimollik klassifikatori, ma’lum bir sinfdagi bir-biri bilan

xususiyatlarning mustaqilligi to'g'risida qat'iy (sodda) taxminlarga ega, bu bitta ko'p o'lchovli emas, balki bir o'lchovli ehtimollik zichligini baholash tufayli tasniflash vazifasini ancha soddalashtiradi.

Bunday holda, bir o'lchovli ehtimollik zichligi — bu har bir belgining mustaqilligini hisobga olgan holda alohida — alohida ehtimolini baholash va ko'p o'lchovli-bu ularning bog'liqligi holatidan kelib chiqadigan barcha xususiyatlarning kombinatsiyasi ehtimolini baholash. Shu sababli, ushbu klassifikator sodda deb nomlanadi, chunki u hisob-kitoblarni sezilarli darajada soddalashtirishga va algoritm samaradorligini oshirishga imkon beradi. Biroq, bunday taxmin amalda har doim ham to'g'ri emas va ba'zi hollarda prognozlar sifatining sezilarli darajada yomonlashishiga olib kelishi mumkin.

Bayes formulasining o'zi quyidagicha:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

bunda:

- $P(A/B)$ - B hodisasi bajarilgan taqdirda a hodisasining posterior ehtimoli;
- $P(B/A)$ - a hodisasi bajarilgan taqdirda b hodisasining shartli ehtimoli;
- $P(A)$ va $P(B)$ mos ravishda a va b hodisalarining apriori ehtimolliklari.

Mashinani o'rganish kontekstida Bayes formulasi quyidagi shaklga ega bo'ladi:

$$P(y_k|X) = \frac{P(y_k)P(X|y_k)}{P(X)}$$

bunda:

- $P(y_k/X)$ - namunaning x xususiyatlarini hisobga olgan holda y_k sinfiga tegishli bo'lishining posterior ehtimoli;
- $P(X/y_k)$ - ehtimollik, ya'ni berilgan y_k sinfiga x belgilarining ehtimoliyk;
- $P(y_k)$ - tasodifiy tanlangan kuzatuvning y_k sinfiga mansubligining aprior ehtimoliyk;
- $P(X)$ -x belgilarining aprior ehtimoli.

Agar ob'ekt bir emas, balki bir nechta X_1, X_2 belgilari bilan tavsiflansa ..., X_n , keyin formula quyidagi shaklni oladi:

$$P(y_k|X_1, X_2, \dots, X_n) = \frac{P(y_k) \prod_{i=1}^n P(X_i|y_k)}{P(X_1, X_2, \dots, X_n)}$$

Amalda, ushbu formulaning numeratori eng katta qiziqish uyg'otadi, chunki maxraj sinfga emas, balki faqat xususiyatlarga bog'liq va shuning uchun u ko'pincha turli sinflarning ehtimolliklarini taqqoslashda qoldiriladi. Oxir oqibat, tasniflash qoidasi maksimal posteriori ehtimollik bilan sinfni tanlashga mutanosib bo'ladi:

$$y_k \propto \arg \max_{y_k} P(y_k) \prod_{i=1}^n P(X_i|y_k)$$

Model parametrlarini, ya'ni ehtimolliklarni baholash uchun $P(y_k)$ va $P(X_i|y_k)$ odatda maksimal ehtimollik usuli qo'llaniladi, bu holda o'quv namunasidagi sinflar va xususiyatlarning paydo bo'lish chastotalariga asoslanadi.

Sodda Bayesning turlari

Scikit-learn kutubxonasida sodda Bayes klassifikatorining bir nechta amalga oshirilishi turlari mavjud bo'lib, ular ma'lum bir sinfdagi xususiyatlarning taqsimlanishi haqidagi taxminlar bilan ajralib turadi. Bularga quyidagilar kiradi:

- **Gauss sodda Bayes klassifikatori (GaussianNB)** normal (Gauss) taqsimotiga ega bo'lgan doimiy xususiyatlar bilan ishlash variantidir *вариант нормального*. Berilgan sinfdagi belgining ehtimoli quyidagi formula bo'yicha hisoblanadi:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

где μ_y и σ_y bu erda sinfdagi belgining o'rtacha va standart og'ishlari. Ushbu parametrlar o'quv ma'lumotlari bo'yicha maksimal ehtimollik usuli bilan baholanadi.

- **Multinomial sodda Bayes klassifikatori (MultinomialNB)** — bu multinomial taqsimotga ega bo'lgan diskret xususiyatlar bilan ishlash variantidir *мультиномиальное*. Bunday belgilar ko'pincha matnlarni tasniflash vazifalarida uchraydi, bu erda ular matndagi voqealar sonini

anglatadi. Berilgan sinfda belgining ehtimoli quyidagi formula bo'yicha hisoblanadi:

$$P(x_i|y) = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

bu erda i — bu belgi sinfda necha marta paydo bo'lishi; y , N_y — y sinfidagi barcha belgilarning umumiy soniy; n — turli xil belgilar soni; α – nol ehtimolliklarning paydo bo'lishiga to'sqinlik qiladigan tekislash parametri.

- **Qo'shimcha sodda Bayes klassifikatori (ComplementNB)**— *MultinomialNB* muvozanatsiz ma'lumotlar to'plamlari uchun mos bo'lgan MultinomialNB-ning takomillashtirilgan versiyasi. Belgilangan sinfda belgining ehtimolini baholash o'rniga, algoritm признака sinf uchun normallashtirilgan belgining og'irligini sinfni to'ldirishda, ya'ni boshqa barcha sinflarda belgining ehtimoli sifatida baholaydi. Shunday qilib, algoritm nafaqat sinfdagi xususiyatlarning chastotasini, balki ularning boshqa sinflarda yo'qligini ham hisobga oladi, bu esa uni namuna siljishiga nisbatan kamroq sezgir qiladi. Sinfni to'ldirishda belgining ehtimolini hisoblash formulasi quyidagicha:

$$\hat{\theta}_{ci} = \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}}$$

$$w_{ci} = \log \hat{\theta}_{ci}$$

$$w_{ci} = \frac{w_{ci}}{\sum_j |w_{cj}|}$$

где $\hat{\theta}_{ci}$ bu erda C sinfini to'ldirishda i belgining ehtimolini baholash при дополнении класса c , bu a_i ning tekislash parametriva C dan tashqari barcha sinflarda i xususiyatning chastotasi bilan hisoblanadic(bu holda d_{ij} -bui belgining j sinfida necha marta paydojbo'lishi); w_{ci} – bui C sinfi uchun i xususiyatning normallashtirilgan og'irligic. V Berilgan t xususiyat vektori uchun bashorat qilingan v sinftquyidagicha bo'ladi:

$$\hat{c} = \arg \min_c \sum_i t_i w_{ci}$$

- **Bernoulliev sodda Bayes klassifikatori (BernoulliNB)** diskret xususiyatlar bilan ishlashning yana bir variantidir, ammo бернуллиевскоеbernoulliev taqsimotiga ega. Bunday holda, belgilar ob'ektda ma'lum xususiyatlarning mavjudligi yoki yo'qligi uchun ikkilik ko'rsatkichlardir. Masalan, matnlarni tasniflash muammosida bu matnda ma'lum so'zlarning mavjudligi yoki yo'qligi bo'lishi mumkin. Berilgan sinfda belgining ehtimoli quyidagi formula bo'yicha hisoblanadi:

$$P(x_i|y) = P(x_i = 1|y)x_i + (1 - P(x_i = 1|y))(1 - x_i)$$

где $P(x_i = 1|y)$ bu erda ob'ekt y sinfiga tegishli bo'lsa, i belgining 1 (haqiqat) qiymatini olish ehtimoli; x_i — j (0 yoki 1) xususiyatining qiymati.

Kategorik sodda Bayes klassifikatori (CategoricalNB)— bu indeks bilan tavsiflangan har bir belgi o'z kategorik taqsimotiga ega degan taxminga asoslangan kategorik taqsimlangan ma'lumotlar uchun variant. Berilgan sinfda belgining ehtimoli quyidagi formula bo'yicha hisoblanadi:

$$P(x_i = t | y = c; \alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i}$$

bu erda $N_{tic} = |\{j \in J \mid x_{ij} = t, y_j = c\}|$ - bu belgi принимает значение C sinfidagi i qiymatini qabul qilish soni в классе c ; $N_c = |\{j \in J \mid y_j = c\}|$ - o'quv ma'lumotlaridagi C sinfidagi barcha xususiyatlarning umumiy soni; α -tekislash parametri; n_i - i xususiyatning mavjud qiymatlari soni .

Gauss taqsimoti bilan sodda Bayes klassifikatorining ishlash printsiplari.

Algoritm quyidagicha tuzilgan:

- 1) dastlab sinflarning apriori ehtimolliklari hisoblanadi;
- 2) keyin sinflar bo'yicha belgilarning o'rtacha va standart og'ishlari hisoblanadi;
- 3) olingan xususiyatlarning sinflar bo'yicha og'ishlariga asoslanib, Gauss taqsimoti bo'yicha test xususiyatlarining ehtimollik zichligi hisoblanadi;
- 4) bundan tashqari, posteriori ehtimolliklar sinflarning apriori ehtimolliklari va sinov xususiyatlarining ehtimollik zichligi mahsuloti sifatida hisoblanadi;

5) maksimal posteriori ehtimoli bo'lgan sinflar yakuniy prognoz bo'ladi.

Spamni filtrlash vazifalarida sodda Bayes

Spamni filtrlash kontekstida sodda Bayes klassifikatori spam va spam bo'lmagan xabarlarda so'zlarning paydo bo'lish chastotasiga va ularning ehtimolliklarini maksimal darajada oshirishga asoslangan. Bu holda soddalik xabardagi so'zlarning tartib va kontekstdan mustaqilligini taxmin qilishdan iborat bo'ladi. Keyin Bayes formulasi quyidagi shaklga ega bo'ladi:

$$P(C|M) \propto P(C) \prod_{i=1}^n P(w_i|C), \quad w_i \in M$$

buerda:

- C - sinf spam yoki spam emas;
- M - xabar;
- w_i m xabaridagi w_i — i so'zi;
- ϵ -mutanosiblik belgisi.

Yaxshiroq tushunish uchun quyidagi misolni ko'rib chiqing. Aytaylik, biz "**Hi, you won a discount and you can get The prize this evening**" xabarini **tasniflamoqchimiz.**" va bizda quyidagi xabarlardan iborat o'quv namunasi mavjud:

Message	Class
Hi, how are you?	Not spam
Congratulations, you won a prize!	Spam
Buy the product now and get a discount!	Spam
Let's walk this evening	Not spam

Birinchi qadam barcha noyob so'zlarning paydo bo'lish chastotasini va ularning spam va spam bo'lmagan xabarlardagi umumiy sonini hisoblashdir. Keyin har bir so'zni spam va spam bo'lmagan xabarlarda ushbu chastotalar asosida kutib olish ehtimoli hisoblab chiqiladi. Xabarda ilgari o'quv namunasida uchramagan so'zlar bo'lsa, anti-aliasing ishlatiladi. Anti-aliasing juda ko'p turli xil turlari

mavjud, ammo ularning eng sodda mohiyati xabarlardagi soʻzlarning chastotalarini hisoblashda 1 ni qoʻshishdir. Ushbu usul nol ehtimollik muammosidan qochadi. Quyida barcha soʻzlar uchun ehtimollik hisoblangan jadval mavjud.

Word	Frequency in Not Spam	Frequency in Spam	Probability in Not Spam	Probability in Spam
Hi	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
how	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
are	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
you	1 + 1 = 2	1 + 1 = 2	2 / 28 = 0.0714	2 / 33 = 0.06
Congratulations	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
won	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
a	0 + 1 = 1	2 + 1 = 3	1 / 28 = 0.0357	3 / 33 = 0.09
prize	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
Buy	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
the	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
product	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
now	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
and	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
get	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
discount	0 + 1 = 1	1 + 1 = 2	1 / 28 = 0.0357	2 / 33 = 0.06
Let's	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
walk	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
this	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
evening	1 + 1 = 2	0 + 1 = 1	2 / 28 = 0.0714	1 / 33 = 0.03
can	0 + 1 = 1	0 + 1 = 1	1 / 28 = 0.0357	1 / 33 = 0.03
Total amount of words	28	33		

Ushbu jadvalni noutbuklarda boshida havolalar orqali topish mumkin

Oxir-oqibat, xabarning spam yoki spam boʻlmasligi ehtimoli hisoblab chiqiladi va yakuniy prognoz maksimal ehtimollik sinfi boʻladi.

$$P(C|M) = P(C) \cdot P('Hi'|C) \cdot P('you'|C) \cdot P('won'|C) \cdot P('a'|C) \cdot P('discount'|C) \cdot P('and'|C) \cdot P('you'|C) \cdot P('can'|C) \cdot P('get'|C) \cdot P('the'|C) \cdot P('prize'|C) \cdot P('this'|C) \cdot P('evening'|C)$$

buerda:

$$C \in (Spam, Not Spam);$$

$$P(Spam) = P(Not Spam) = \frac{2}{4} = 0.5 ;$$

Xabarning spam bo'lish ehtimoli:

$$P(\text{Spam}|M) = 0.5 \cdot 0.03 \cdot 0.06 \cdot 0.06 \cdot 0.09 \cdot 0.06 \cdot 0.06 \cdot 0.06 \cdot 0.03 \cdot 0.06 \cdot 0.06 \cdot 0.06 \cdot 0.03 \cdot 0.03 \approx 6.12 \cdot 10^{-18}$$

Xabarning spam emasligi ehtimoli:

$$P(\text{Not Spam}|M) = 0.5 \cdot 0.0714 \cdot 0.0714 \cdot 0.0357 \cdot 0.0357 \cdot 0.0357 \cdot 0.0357 \cdot 0.0714 \cdot 0.0714 \approx 2.45 \cdot 10^{-18}$$

$$P(\text{Spam}|M) > P(\text{Not Spam}|M) \rightarrow \text{Xabar spam bo'lgani uchun.}$$

Shuni qo'shimcha qilish kerakki, amalda hisob-kitoblarning qulayligi uchun ko'pincha ehtimollik o'rniga ehtimollik logarifmi ishlatiladi.

Kerakli kutubxonalarni import qilish

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from mlxtend.plotting import plot_decision_regions
class GaussianNaiveBayes:
    def fit(self, X, y):
        classes, cls_counts = np.unique(y, return_counts=True)
        n_classes = len(classes)
        self.priors = cls_counts / len(y)

        # calculate the mean and standard deviations of features by classes
        self.X_cls_mean = np.array([np.mean(X[y == c], axis=0) for c in range(n_classes)])
        self.X_stds = np.array([np.std(X[y == c], axis=0) for c in range(n_classes)])

        # calculate the probability density of the feature according to the Gaussian distribution
        def pdf(self, x, mean, std):
            return (1 / (np.sqrt(2 * np.pi) * std)) * np.exp(-0.5 * ((x - mean) / std) ** 2)

        def predict(self, X):
            pdfs = np.array([self.pdf(x, self.X_cls_mean, self.X_stds) for x in X])
            posteriors = self.priors * np.prod(pdfs, axis=2) # shorten Bayes formula

            return np.argmax(posteriors, axis=1)
    def decision_boundary_plot(X, y, X_train, y_train, clf, feature_indexes, title=None):
        feature1_name, feature2_name = X.columns[feature_indexes]
        X_feature_columns = X.values[:, feature_indexes]
        X_train_feature_columns = X_train[:, feature_indexes]
        clf.fit(X_train_feature_columns, y_train)
```

```

plot_decision_regions(X=X_feature_columns, y=y.values, clf=clf)
plt.xlabel(feature1_name)
plt.ylabel(feature2_name)
plt.title(title)
//Ma'lumotlar to'plamini yuklab olish
X1, y1 = load_iris(return_X_y=True, as_frame=True)
X1_train, X1_test, y1_train, y1_test = train_test_split(X1.values, y1.values, random_state=0)
print(X1, y1, sep='\n')

```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
..	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

[150 rows x 4 columns]

```

0 0
1 0
2 0
3 0
4 0
..
145 2
146 2
147 2
148 2
149 2

```

Name: target, Length: 150, dtype: int64

Modellarni o'qitish va natijalarni baholash

Uning soddaligiga qaramay, bu holda algoritm mukammal natijani ko'rsatdi, mutlaqo barcha namunalarni to'g'ri tasnifladi, bu yuqori umumlashtirish qobiliyatiga ega moslashuvchan hal qiluvchi chegarani qurish tufayli mumkin. Bundan qiziqarli xulosa qilishimiz mumkinki, ba'zi holatlarda sodda modellar murakkab modellarga qaraganda ancha yaxshi ishlashi mumkin, buni keyinchalik boshqa algoritmlar misolida ko'rish mumkin.

Sodda Bayes

```

nb_clf = GaussianNaiveBayes()
nb_clf.fit(X1_train, y1_train)
nb_clf_pred_res = nb_clf.predict(X1_test)
nb_clf_accuracy = accuracy_score(y1_test, nb_clf_pred_res)

print(f'Naive Bayes classifier accucacy: {nb_clf_accuracy}')
print(nb_clf_pred_res)

Naive Bayes classifier accuracy: 1.0
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0 1]

```

Sodda Bayes (scikit-learn)

```

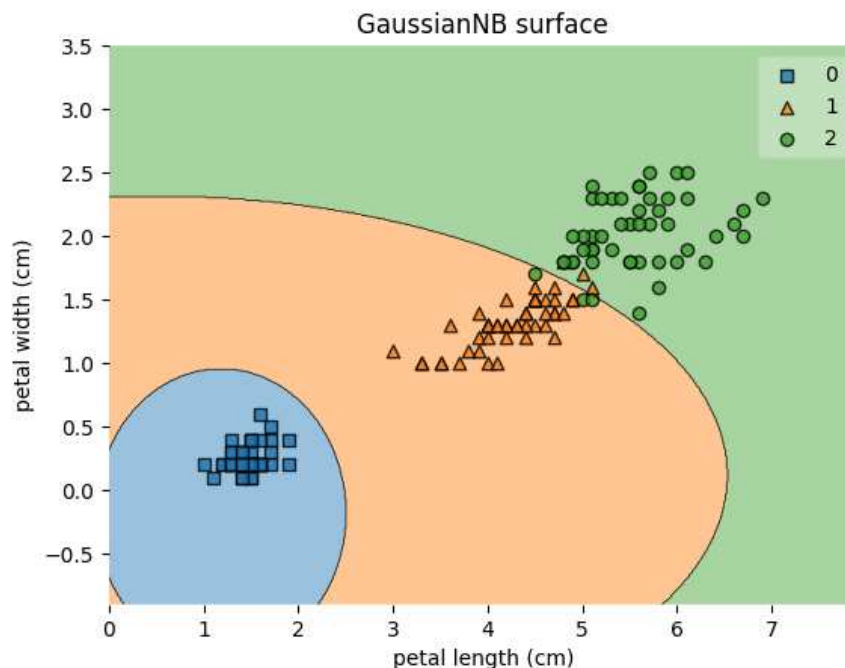
sk_nb_clf = GaussianNB()
sk_nb_clf.fit(X1_train, y1_train)
sk_nb_clf_pred_res = sk_nb_clf.predict(X1_test)
sk_nb_clf_accuracy = accuracy_score(y1_test, sk_nb_clf_pred_res)

print(f'sk Naive Bayes classifier accucacy: {sk_nb_clf_accuracy}')
print(sk_nb_clf_pred_res)

feature_indexes = [2, 3]
title1 = 'GaussianNB surface'
decision_boundary_plot(X1, y1, X1_train, y1_train, sk_nb_clf, feature_indexes, title1)

sk Naive Bayes classifier accuracy: 1.0
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0 1]

```



Sodda Bayes klassifikatorining afzalliklari va kamchiliklari

Afzalliklari:

- amalga oshirish va talqin qilish qulayligi;

- parametrlarni sozlash deyarli talab qilinmaydi;
- ko‘p holatlarda yuqori ish tezligi va prognozlarning aniqligi;
- shovqin va chiqindilarga nisbatan yaxshi qarshilik ko‘rsatadi, chunki u ehtimollik taqsimotiga va belgilarning mustaqilligi haqidagi sodda taxminlarga asoslanadi.

Kamchiliklari:

- belgilarning mustaqilligi haqidagi taxmin buzilgan taqdirda, prognozlarning aniqligi sezilarli darajada pasayishi mumkin;
- может отдавать предпочтение к классам с большим balanssiz ma’lumotlar mavjud bo‘lganda ko‘proq namunalarga ega sinflarga ustunlik berishi mumkin.

K - eng yaqin qo‘shnilar (K-Nearest Neighbors - KNN)

U klassifikatsiya va regressiya vazifalari uchun ishlatiladi. KNN asosan klassifikatsiya vazifalarini hal qilish uchun keng qo‘llaniladi. Ushbu algoritmnining asosiy konsepsiyasi shundan iboratki, u mavjud bo‘lgan barcha murojaatlarni saqlaydi va yangi murojaatlarni o‘zining K yaqin qo‘shnilarining ovozlari ko‘paytirilgan holda tasniflaydi. So‘ngra, yangi yozuv eng ko‘p uchragan K yaqin qo‘shnisining klassiga tayinlanadi va masofa funktsiyasi yordamida o‘lchanadi. Masofa funktsiyalari sifatida Yevklid, Minkovski va Xemming masofalari ishlatilishi mumkin. KNNni ishlatishda quyidagilarni hisobga olish zarur –

- KNNlar tasniflash muammolari uchun ishlatiladigan boshqa algoritmlarga qaraganda hisoblash jihatidan qimmatroq.
- O‘zgaruvchilarni normalizatsiya qilish kerak, aks holda yuqori diapazonga ega bo‘lgan o‘zgaruvchilar ularning noaniq bo‘lishiga olib kelishi mumkin.
- KNNda biz shovqinni olib tashlash kabi dastlabki ishlov berish bosqichida ishlashimiz kerak.

K - Klasterlash degan ma’noni anglatadi

Nomidan ko‘rinib turibdiki, u klasterlash muammolarini hal qilish uchun ishlatiladi. Aslida, bu nazoratsiz o‘rganishning bir turi. K-Means klasterlash algoritmining asosiy mantig‘i ma’lumotlar to‘plamini bir nechta klasterlarga

tasniflashdir. K-vositalardan foydalangan holda klasterlarni shakllantirish uchun quyidagi amallarni bajaring:

- K-means har bir klaster uchun centroidlar deb nomlanuvchi **k** sonli nuqtani tanlaydi.
- Endi har bir ma'lumot nuqtasi o'zining eng yaqin markazlari bilan klaster hosil qiladi, ya'ni. k klasterlar.
- Endi u mavjud klaster a'zolari asosida har bir klasterning markazlarini topadi.
- Konvergensiya paydo bo'lgunga qadar ushbu amallarni takrorlashimiz kerak.

Tasodifiy o'rmon

Bu nazorat ostidagi klassifikatsiya algoritmi hisoblanadi. Tasodifiy o'rmon algoritmining afzalligi shundaki, uni klassifikatsiya vazifalari uchun ham, regressiya vazifalari uchun ham ishlatish mumkin. Aslida, bu qaror daraxtlari to'plami (ya'ni, o'rmon) yoki boshqa so'z bilan aytganda, qaror daraxtlari ansambli hisoblanadi. Tasodifiy o'rmonning asosiy konsepsiyasi shundan iboratki, har bir daraxt o'z klassifikatsiyasini beradi va o'rmon bu klassifikatsiyalardan eng yaxshilarini tanlaydi. Quyida tasodifiy o'rmon algoritmining afzalliklari keltirilgan:

- Tasodifiy o'rmon klassifikatori klassifikatsiya va regressiya vazifalari uchun ishlatilishi mumkin.
- Ular yo'qolgan qiymatlarni ishlay oladi.
- U o'rmondagi daraxtlar soni ko'paytirilganida ham, modelga to'liq mos kelmaydi.

O'z-o'zini nazorat qilish uchun savollar

1. Mashinali o'rganish turlari?
2. O'qituvchisiz mashinani o'rganish algoritmlari?
3. Mashinani o'rganishning eng keng tarqalgan algoritmlari?
4. Chiziqli regressiyaning turlari?
5. Qarorlar daraxti?
6. Sodda Bayes klassifikatori?
7. Sodda Bayesning turlari?
8. Sodda Bayes klassifikatorining afzalliklari va kamchiliklari

2.2-§. MA'LUMOTLAR TAYYORLASH

Biz allaqachon nazoratli va nazoratsiz mashinani o'qitish algoritmlarini o'rganganmiz. Ushbu algoritmlar o'quv jarayonini boshlashi uchun ma'lumotlarning belgilangan shaklda bo'lishi talab qilinadi. Ma'lumotlar mashinani o'qitish algoritmlariga kirish sifatida foydalanilishi uchun ularni ma'lum bir shaklda tayyorlash yoki formatlashimiz kerak.

Ushbu bo'lim mashinani o'qitish algoritmlari uchun ma'lumotlarni tayyorlash jarayoniga bag'ishlangan.

Ma'lumotlarni oldindan qayta ishlash

Kundalik hayotimizda biz ko'p miqdordagi ma'lumotlar bilan ishlaymiz, ammo bu ma'lumotlar odatda xom shaklda bo'ladi. Mashinani o'qitish algoritmlariga kirish sifatida ma'lumotlarni taqdim qilish uchun ularni mazmunli ma'lumotlarga aylantirishimiz kerak. Mana shu jarayonda **ma'lumotlarni oldindan qayta ishlash** muhim o'rin tutadi.

Oddiyroq qilib aytganda, mashinani o'qitish algoritmlariga ma'lumotlarni yetkazishdan oldin ularni oldindan qayta ishlash zarur bo'ladi. Bu jarayon xom ma'lumotlarni sifatli va foydalanishga yaroqli ko'rinishga keltirishga xizmat qiladi.

Ma'lumotlarni oldindan qayta ishlash bosqichlari.

Python-da ma'lumotlarni oldindan qayta ishlash uchun quyidagi amallarni bajaring:

1-qadam - Foydali paketlarni import qilish - Agar biz Python-dan foydalanayotgan bo'lsak, bu ma'lumotlarni ma'lum bir formatga aylantirish uchun birinchi qadam bo'ladi, ya'ni. oldindan ishlov berish. Buni quyidagicha amalga oshirish mumkin.

```
import numpy as np
import sklearn.preprocessing
```

Bu erda biz quyidagi ikkita paketdan foydalandik:

- **NumPy** - NumPy o'z mohiyatiga ko'ra, kichik ko'p o'lchovli massivlar uchun juda katta tezlikni yo'qotmasdan, ixtiyoriy yozuvlarning katta ko'p o'lchovli

massivlarini samarali boshqarish uchun mo'ljallangan umumiy maqsadli massivlarni qayta ishlash paketidir .

- **Sklearn** - Ushbu paket ko'plab umumiy yordamchi funktsiyalarni va xom xususiyat vektorlarini mashinani o'rganish algoritmlari uchun mosroq tasvirga aylantirish uchun o'zgartirish sinflarini taqdim etadi .

2-qadam – Namuna ma'lumotlarini aniqlash – Paketlarni import qilgandan so'ng, biz ushbu ma'lumotlarga oldindan ishlov berish usullarini qo'llashimiz uchun ba'zi namunaviy ma'lumotlarni aniqlashimiz kerak. Endi biz quyidagi ma'lumotlar misollarini aniqlaymiz.

```
input_data = np.array([2.1, -1.9, 5.5],  
                      [-1.5, 2.4, 3.5],  
                      [0.5, -7.9, 5.6],  
                      [5.9, 2.3, -5.8])
```

3-qadam - Oldindan ishlov berish usulini qo'llash - Ushbu bosqichda biz oldindan ishlov berish usullaridan birini qo'llashimiz kerak. Keyingi bo'limda ma'lumotlarni qayta ishlash usullari tasvirlangan.

Ma'lumotlarni oldindan qayta ishlash usullari

Ma'lumotlarni qayta ishlash usullari quyida tavsiflangan -

Binarizatsiya

Bu raqamli qiymatlarni mantiqiy qiymatlarga aylantirishimiz kerak bo'lganda foydalaniladigan oldindan ishlov berish usuli. Biz kirish ma'lumotlarini binarlash uchun o'rnatilgan usuldan foydalanishimiz mumkin, masalan, chegara qiymati sifatida 0,5 dan quyidagi tarzda foydalanish.

```
data_binarized = preprocessing.Binarizer(threshold = 0.5).transform(input_data)  
print("\nBinarized data:\n", data_binarized)
```

Endi, yuqoridagi kodni ishga tushirgandan so'ng, biz quyidagi natijani olamiz, 0,5 dan yuqori bo'lgan barcha qiymatlar 1 ga aylantiriladi va 0,5 dan past bo'lgan barcha qiymatlar 0 ga aylantiriladi.

Ikkilik ma'lumotlar

```
[[ 1.  0.  1.]
```

[0. 1. 1.]
[0. 0. 1.]
[1. 1. 0.]]

O‘rtacha olib tashlash

Bu mashinani o‘rganishda qo‘llaniladigan yana bir keng tarqalgan oldindan ishlov berish usuli. U asosan xususiyat vektoridan o‘rtachani olib tashlash uchun ishlatiladi, shunda har bir xususiyat nolga teng bo‘ladi. Xususiyat vektoridagi ob’ektlarning siljishini ham bartaraf etishimiz mumkin. Namuna ma’lumotlariga o‘rtacha olib tashlashni oldindan qayta ishlash texnikasini qo‘llash uchun biz quyida ko‘rsatilgan Python kodini yozishimiz mumkin. Kod kirish ma’lumotlarining o‘rtacha va standart og‘ishini ko‘rsatadi.

```
print("Mean = ", input_data.mean(axis = 0))  
print("Std deviation = ", input_data.std(axis = 0))
```

Yuqoridagi kod satrlarini bajarganimizdan so‘ng, biz quyidagi natijaga erishamiz.

```
Mean = [ 1.75    -1.275    2.2]  
Std deviation = [ 2.71431391  4.20022321  4.69414529]
```

Endi quyidagi kod kiritilgan ma’lumotlarning o‘rtacha va standart og‘ishini olib tashlaydi.

```
data_scaled = preprocessing.scale(input_data)  
print("Mean =", data_scaled.mean(axis=0))  
print("Std deviation =", data_scaled.std(axis = 0))  
= preprocessing.scale(input_data)  
print("Mean =", data_scaled.mean(axis=0))  
print("Std deviation =", data_scaled.std(axis = 0))
```

Yuqoridagi kod satrlarini bajarganimizdan so‘ng, biz quyidagi natijaga erishamiz.

```
Mean = [ 1.11022302e-16 0.00000000e+00 0.00000000e+00]  
Std deviation = [ 1.          1.          1.]
```

Masshtablash

Bu xususiyat vektorlarini o'lchash uchun foydalaniladigan ma'lumotlarni qayta ishlashning yana bir usuli. Xususiyat vektorlarini masshtablash zarur, chunki har bir xususiyatning qiymatlari tasodifiy qiymatlar oralig'ida o'zgarishi mumkin. Boshqacha qilib aytganda, biz masshtabni muhim deb aytishimiz mumkin, chunki biz biron bir xususiyat sun'iy ravishda katta yoki kichik bo'lishini xohlamaymiz. Quyidagi Python kodi yordamida biz kiritilgan ma'lumotlarni o'lchashimiz mumkin, ya'ni. ob'ekt vektorlari.

```
data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0,1))
data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
print("\nMin max scaled data:\n", data_scaled_minmax)
```

Yuqoridagi kod satrlarini bajarganimizdan so'ng, biz quyidagi natijaga erishamiz. Minimal va maksimal ma'lumotlar shkalasi.

```
[ [ 0.48648649 0.58252427 0.99122807 ]
[ 0.         1.         0.81578947 ]
[ 0.27027027 0.         1.         ]
[ 1.         0.99029126 0.         ]]
```

Normallashtirish

Bu xususiyat vektorlarini o'zgartirish uchun ishlatiladigan ma'lumotlarni qayta ishlashning yana bir usuli. Ushbu turdagi modifikatsiya ob'ekt vektorlarini an'anaviy shkalada o'lchash uchun zarurdir. Quyida mashinani o'rganishda ishlatilishi mumkin bo'lgan ikki turdagi normalizatsiya mavjud

L1 Normallashtirish

Eng kam mutlaq og'ish deyiladi. Ushbu turdagi normalizatsiya qiymatlarni o'zgartiradi, shunda mutlaq qiymatlar har doim har bir satrda 1 ga teng bo'ladi. Buni kiritishda quyidagi Python kodidan foydalanib amalga oshirish mumkin.

```
# Normalize data
data_normalized_l1 = preprocessing.normalize(input_data, norm = 'l1')
print("\nL1 normalized data:\n", data_normalized_l1)
```

Yuqoridagi kod qatori quyidagi chiqishni hosil qiladi;

```
L1 normalized data:
[[ 0.22105263 -0.2      0.57894737]
```

```
[ -0.2027027  0.32432432  0.47297297]
[ 0.03571429 -0.56428571  0.4      ]
[ 0.42142857  0.16428571 -0.41428571]]
```

L2 Normallash

Eng kichik kvadratlar usuli deb ham ataladi . Ushbu turdagi normalizatsiya qiymatlarni har bir satrda kvadratchalar yig'indisi har doim 1 bo'lishi uchun o'zgartiradi. Buni kiritishda quyidagi Python kodidan foydalanib amalga oshirish mumkin.

```
# Normalize data
data_normalized_l2 = preprocessing.normalize(input_data, norm = 'l2')
print("\nL2 normalized data:\n", data_normalized_l2)
```

Yuqoridagi kod qatori quyidagi chiqishni hosil qiladi.

```
L2 normalized data:
[[ 0.33946114 -0.30713151  0.88906489]
 [ -0.33325106  0.53320169  0.7775858 ]
 [ 0.05156558 -0.81473612  0.57753446]
 [ 0.68706914  0.26784051 -0.6754239 ]]
```

Ma'lumotlarni belgilash

Mashinalarni o'qitish algoritmlari ma'lum formatdagi ma'lumotlar bilan ishlaydi. Ushbu formatni tayyorlashda muhim bosqichlardan biri — ma'lumotlarni belgilashdir. Bu degani, har bir ma'lumot birligi algoritm uchun tegishli toifadagi qiymat (yoki "belgi") bilan ta'minlangan bo'lishi kerak.

Masalan, agar bizda **klassifikatsiya** vazifasi bo'lsa, ma'lumotlar to'plamida bir nechta **belgilar** bo'ladi. Bu belgilar so'zlar, raqamlar va boshqa formatlarda bo'lishi mumkin.

Sklearn kutubxonasida mashinaviy o'qitish algoritmlari asosan **sonli belgilar** bilan ishlaydi. Shuning uchun agar ma'lumot belgilar boshqa formatda bo'lsa, ularni sonli ko'rinishga o'tkazish talab qilinadi. Ushbu jarayon **belgilarni kodlash** deb ataladi.

Yorliqlarni kodlash bosqichlari

Python-da ma'lumotlar yorliqlarini kodlash uchun quyidagi amallarni bajaring:

1-qadam - Foydali paketlarni import qilish

Agar biz Python-dan foydalanayotgan bo'lsak, bu ma'lumotlarni ma'lum bir formatga aylantirish uchun birinchi qadam bo'ladi, ya'ni oldindan ishlov berish.

```
import numpy as np
from sklearn import preprocessing
```

2-bosqich - Yorliq misollarini aniqlash

Paketlarni import qilgandan so'ng, biz yorliq kodlovchisini yaratishimiz va o'rgatishimiz uchun ba'zi namuna yorliqlarini aniqlashimiz kerak. Endi biz quyidagi namuna belgilarini aniqlaymiz.

```
# Sample input labels
input_labels = ['red', 'black', 'red', 'green', 'black', 'yellow', 'white']
```

3-qadam - Yorliq kodlovchi ob'ektini yaratish va o'rgatish

Ushbu bosqichda biz yorliq kodlovchisini yaratishimiz va uni o'rgatishimiz kerak. Quyidagi Python kodi bunga yordam beradi.

```
# Creating the label encoder
encoder = preprocessing.LabelEncoder()
encoder.fit(input_labels)
```

Quyida yuqoridagi Python kodini ishga tushirgandan so'ng natija bo'ladi.

```
LabelEncoder()
```

4-qadam - Tasodifiy tartiblangan ro'yxatni kodlash orqali ishlashni sinab ko'ring

Ushbu qadam tasodifiy tartiblangan ro'yxatni kodlash orqali ishlashni sinash uchun ishlatilishi mumkin. Xuddi shu narsani amalga oshirish uchun siz Python-da quyidagi kodni yozishingiz mumkin.

```
# encoding a set of labels
test_labels = ['green', 'red', 'black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
```

Yorliqlar quyidagicha chop etiladi.

```
Labels = ['green', 'red', 'black']
```

Endi biz kodlangan qiymatlar ro'yxatini, ya'ni raqamlarga aylantirilgan so'z belgilarini quyidagicha olishimiz mumkin.

```
print("Encoded values =", list(encoded_values))
```

Kodlangan qiymatlar quyidagicha chop etiladi.

```
Encoded values = [1, 2, 0]
```

5-qadam - Tasodifiy sonlarni dekodlash bosqichlari.

Ushbu qadam tasodifiy raqamlar to'plamini dekodlash orqali ishlashni tekshirish uchun ishlatilishi mumkin. Xuddi shu narsani amalga oshirish uchun siz Python-da quyidagi kodni yozishingiz mumkin.

```
# decoding a set of values
encoded_values = [3,0,4,1]
decoded_list = encoder.inverse_transform(encoded_values)
print("\nEncoded values =", encoded_values)
```

Endi kodlangan qiymatlar quyidagicha chiqariladi.

```
Encoded values = [3, 0, 4, 1]
print("\nDecoded labels =", list(decoded_list))
```

Endi shifrlangan qiymatlar quyidagicha chiqariladi.

```
Decoded labels = ['white', 'black', 'yellow', 'green']
```


Belgilangan va belgilanmagan ma'lumotlar

Belgilanmagan ma'lumotlar asosan atrof-muhitdan osongina olinadigan tabiiy yoki sun'iy ob'ektlarning namunalaridan iborat. Bularga audio, video, fotosuratlar, yangiliklar maqolalari va boshqalar kiradi.

Boshqa tomondan, etiketli ma'lumotlar yorliqsiz ma'lumotlar to'plamini oladi va yorliqsiz ma'lumotlarning har bir qismini mazmunli teg, teg yoki sinf bilan oshiradi. Misol uchun, agar bizda fotosurat bo'lsa, unda fotosuratning mazmuniga qarab yorliq qo'yilishi mumkin, ya'ni o'g'il yoki qiz, hayvon yoki boshqa biror narsaning fotosurati. Yorliqlash ma'lumotlari ma'lum bir yorliqsiz ma'lumotlar bo'lagi haqida inson tajribasi yoki mulohazalarini talab qiladi.

Belgilanmagan ma'lumotlar juda ko'p va ularni olish oson bo'lgan ko'plab stsenariylar mavjud, ammo etiketli ma'lumotlarni izohlash uchun ko'pincha inson mutaxassisi kerak bo'ladi. Yarim nazorat ostida o'rganish yaxshi modellarni yaratish uchun etiketli va yorliqsiz ma'lumotlarni birlashtirishga urinadi.

O'z-o'zini nazorat qilish uchun savollar

1. Ma'lumotlarni oldindan qayta ishlash?
2. Ma'lumotlarni oldindan qayta ishlash bosqichlari?
3. Ma'lumotlarni oldindan qayta ishlash usullari?
4. O'rtacha olib tashlash?
5. Masshtablash?
6. Normallashtirish?
7. Ma'lumotlarni belgilash?

2.3-§. KO‘RSATMA BO‘YISH O‘RGATISH: KLASIFIKATSIYA.

Ushbu bobda biz boshqariladigan o‘rgatish – klassifikatsiyani amalga oshirishga diqqatni qaratamiz. Klassifikatsiya usuli yoki modeli kuzatilgan qiymatlardan ma’lum bir xulosa chiqarishga harakat qiladi. Klassifikatsiya vazifasida bizda kategoriyalangan chiqish ma’lumotlari mavjud, masalan, “Qora” yoki “Oq”, yoki “O‘rgatish” va “O‘rgatilmagan”. Klassifikatsion modelni qurishda bizga o‘quv ma’lumotlari to‘plami kerak, unda ma’lumotlar nuqtalari va ularga tegishli belgilangan yorliqlar mavjud. Masalan, agar biz tasvirning avtomobil ekanligini tekshirmoqchi bo‘lsak, biz “car” (avtomobil) va “no car” (avtomobil emas) bilan bog‘liq ikki klassdan iborat o‘quv to‘plamini yaratamiz. Keyin, biz modelni o‘quv namunalarini ishlatib o‘rgatamiz. Klassifikatsiya modellari asosan yuzni tanish, spamni aniqlash va boshqalar kabi vazifalarda qo‘llaniladi.

Python-da klassifikator yaratish bosqichlar

Python-da klassifikator yaratish uchun biz Python 3 va Scikit-learn kutubxonasidan foydalanamiz, bu mashinani o‘rgatish uchun vosita hisoblanadi. Quyidagi amallarni bajarish orqali Python-da klassifikator yaratishingiz mumkin:

1-qadam - Import Scikit-learn

Bu Python-da klassifikator yaratish yo‘lidagi birinchi qadam bo‘ladi. Ushbu bosqichda biz Python-dagi eng yaxshi mashina o‘rganish modullaridan biri bo‘lgan Scikit-learn deb nomlangan Python paketini o‘rnatamiz. Quyidagi buyruq paketni import qilishga yordam beradi.

Import Sklearn

2-qadam - Scikit-o‘rganish ma’lumotlar to‘plamini import qilish

Shu nuqtada, biz mashinani o‘rganish modelimiz uchun ma’lumotlar to‘plami bilan ishlashni boshlashimiz mumkin. Bu erda biz Viskonsindagi ko‘krak bezi saratoni diagnostikasi ma’lumotlar bazasidan foydalanamiz. Ma’lumotlar to‘plami ko‘krak bezi saratoni o‘smalari haqida turli xil ma’lumotlarni, shuningdek, **xavfli** yoki **xavfli tasniflash belgilarini o‘z ichiga oladi** . Ma’lumotlar to‘plami 569 ta

o'simtadan 569 ta misol yoki ma'lumotni o'z ichiga oladi va o'simta radiusi, tuzilishi, silliqligi va maydoni kabi 30 ta atribut yoki xususiyatlar haqida ma'lumotni o'z ichiga oladi. Quyidagi buyruq yordamida biz Scikit-learn ko'krak saratoni ma'lumotlar to'plamini import qilishimiz mumkin.

```
from sklearn.datasets import load_breast_cancer
```

Endi quyidagi buyruq ma'lumotlar to'plamini yuklaydi.

```
data = load_breast_cancer()
```

Quyida muhim lug'at kalitlari ro'yxati keltirilgan:

- Tasniflash teglari nomlari (target_names)
- Haqiqiy teglar (target)
- Atribut/xususiyat nomlari (feature_names)
- Atribut (data)

Endi quyidagi buyruq yordamida biz har bir muhim ma'lumot to'plami uchun yangi o'zgaruvchilar yaratishimiz va ma'lumotlarni belgilashimiz mumkin. Boshqacha qilib aytganda, biz quyidagi buyruqlar yordamida ma'lumotlarni tartibga solishimiz mumkin.

```
label_names = data['target_names']  
labels = data['target']  
feature_names = data['feature_names']  
features = data['data']
```

Endi buni aniqroq qilish uchun biz sinf yorliqlarini, birinchi ma'lumotlar namunasi yorlig'ini, funksiya nomlarini va funksiya qiymatini quyidagi buyruqlar yordamida chop etishimiz mumkin.

```
print(label_names)
```

Yuqoridagi buyruq mos ravishda yomon va yaxshi bo'lgan sinflarning nomlarini chiqaradi. Bu quyidagi chiqishda ko'rsatilgan -

```
['malignant' 'benign']
```

Endi quyidagi buyruq ularning ikkilik qiymatlari 0 va 1 bilan taqqoslanganligini ko'rsatadi. Bu erda 0 xavfli saratonni va 1 yaxshi saratonni anglatadi. Siz quyidagi natijani olasiz.

```
print(labels[0])  
0
```

Quyidagi ikkita buyruq funksiya nomlarini va ularning qiymatlarini yaratadi.

```
print(feature_names[0])  
mean radius  
print(features[0])  
[ 1.79900000e+01 1.03800000e+01 1.22800000e+02 1.00100000e+03  
 1.18400000e-01 2.77600000e-01 3.00100000e-01 1.47100000e-01  
 2.41900000e-01 7.87100000e-02 1.09500000e+00 9.05300000e-01  
 8.58900000e+00 1.53400000e+02 6.39900000e-03 4.90400000e-02  
 5.37300000e-02 1.58700000e-02 3.00300000e-02 6.19300000e-03  
 2.53800000e+01 1.73300000e+01 1.84600000e+02 2.01900000e+03  
 1.62200000e-01 6.65600000e-01 7.11900000e-01 2.65400000e-01  
 4.60100000e-01 1.18900000e-01]
```

Yuqoridagi ma'lumotlardan ma'lumotlarning birinchi namunasi radiusi 1,7990000e + 01 bo'lgan xavfli o'sma ekanligini ko'rishimiz mumkin.

3-qadam - Ma'lumotlarni to'plamlarga ajratish

Ushbu bosqichda biz ma'lumotlarimizni ikki qismga ajratamiz, ya'ni o'quv to'plami va test to'plami. Ma'lumotlarni ushbu to'plamlarga ajratish juda muhim, chunki biz modelimizni ko'rinmas ma'lumotlarda sinab ko'rishimiz kerak. Ma'lumotlarni to'plamlarga bo'lish uchun sklearn-da **train_test_split()** funksiyasi mavjud . Quyidagi buyruqlar yordamida biz ushbu to'plamlardagi ma'lumotlarni ajratishimiz mumkin.

```
from sklearn.model_selection import train_test_split
```

Yuqoridagi buyruq train_test_split funksiyasini sklearn'dan import qiladi va quyidagi buyruq ma'lumotlarni trening va testga ajratadi. Quyidagi misolda biz sinov uchun ma'lumotlarning 40% dan foydalanamiz, qolgan ma'lumotlar esa modelni o'rgatish uchun ishlatiladi.

```
train, test, train_labels, test_labels = train_test_split(features, labels, test_size = 0.40,  
random_state = 42)
```

4-qadam – Model yaratish

Ushbu bosqichda biz modelimizni quramiz. Modelni yaratish uchun biz Naive Bayesdan foydalanamiz. Modelni yaratish uchun siz quyidagi buyruqlardan foydalanishingiz mumkin.

```
from sklearn.naive_bayes import GaussianNB
```

Yuqoridagi buyruq GaussianNB modulini import qiladi. Endi quyidagi buyruq modelni ishga tushirishga yordam beradi.

```
gnb = GaussianNB()
```

Modelni gnb.fit() yordamida ma'lumotlarga moslashtirib o'rgatamiz.

```
model = gnb.fit(train, train_labels)
```

5-qadam - Model va uning aniqligini baholang

Ushbu bosqichda biz test ma'lumotlarimiz asosida bashorat qilish orqali modelni baholaymiz. Shunda uning aniqligini ham bilib olamiz. Bashorat qilish uchun biz *predict()* funksiyasidan foydalanamiz. Quyidagi buyruq buni amalga oshirishga yordam beradi.

```
preds = gnb.predict(test)  
print(preds)  
  
[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1  
 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0  
 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0  
 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 0 0  
 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0  
 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0  
 1 1 0 1 1 1 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 0  
 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 1 0 1]
```

Yuqoridagi 0 va 1 qatorlar o'sma sinflari uchun taxmin qilingan qiymatlarni ifodalaydi - yomon va yaxshi. Endi ikkita massivni, ya'ni **test_labels** va **preds** ni solishtirish orqali biz modelimizning to'g'riligini bilishimiz mumkin. Biz aniqlikni aniqlash uchun **accuracy_score()** funksiyasidan foydalanamiz . Buning uchun quyidagi buyruqni ko'rib chiqing.

```
from sklearn.metrics import accuracy_score  
print(accuracy_score(test_labels,preds))  
0.951754385965
```

Natija Bayes tasniflagichining aniqligi 95,17% ekanligini ko'rsatadi. Shunday qilib, yuqoridagi qadamlar bilan biz Python-da klassifikatorni yaratishimiz mumkin.

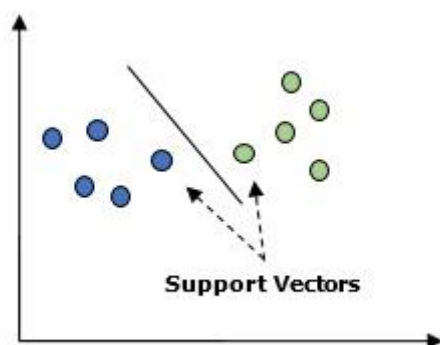
Tayanch vektorlar mashinasi (SVM)

Tayanch vektorlar mashinasi (Support Vector Machine - SVM) nazorat ostidagi mashinani o'rganish algoritmi bo'lib, u regressiya va klassifikatsiya vazifalari uchun ishlatilishi mumkin.

Asosiy kontseptsiya

SVM algoritmi har bir ma'lumot elementini n -o'lchovli fazodagi nuqta sifatida ifodalaydi. Bu yerda har bir obyektning qiymati uning koordinatalarining muayyan qiymatiga mos keladi. Ushbu fazodagi nuqtalar yordamida ma'lumotlarni ajratuvchi chiziq (yoki o'lchamga qarab tekislik yoki gipertekislik) hosil qilinadi. **n** bu fazodagi xususiyatlar (obyektlar) soniga teng bo'lib, SVM maqsadi ma'lumotlarni ikki yoki undan ortiq sinfga ajratuvchi optimal tekislikni topishdir.

Quyidagi diagramma SVM kontseptsiyasini tushunishga yordam beruvchi oddiy grafik tasvirni ifodalaydi:



Yuqoridagi diagrammada biz ikkita funktsiyaga egamiz. Shuning uchun, biz birinchi navbatda bu ikki o‘zgaruvchini ikki o‘lchovli fazoga solishimiz kerak, bu erda har bir nuqta qo‘llab-quvvatlash vektorlari deb ataladigan ikkita koordinataga ega. Chiziqli ma’lumotlarni ikki xil tasniflangan guruhga ajratadi. Ushbu qator tasniflagich bo‘ladi.

Bu erda biz *scikit-learn* va *iris* ma’lumotlar to‘plamidan foydalangan holda SVM klassifikatorini yaratmoqchimiz. Scikitlearn kutubxonasi **sklearn.svm** moduliga ega va tasniflash uchun `sklearn.svm.svc` ni taqdim etadi. Quyida 4 ta xususiyatga asoslangan iris sinfini aniqlash uchun SVM tasniflagichi keltirilgan.

Ma’lumotlar to‘plami

Biz iris ma’lumotlar to‘plamidan foydalanamiz, unda har biri 50 ta misoldan iborat 3 ta sinf mavjud, bunda har bir sinf iris o‘simligining ma’lum bir turiga tegishli. Har bir namunada to‘rtta belgi mavjud, ya’ni sepal uzunligi, sepal kengligi, gulbarg uzunligi va gulbarg kengligi. 4 ta xususiyatga asoslangan iris sinfini aniqlash uchun SVM tasniflagichi quyida ko‘rsatilgan.

Yadro

Bu SVM tomonidan qo‘llaniladigan usul. Bular asosan past o‘lchamli kirish maydonini oladigan va uni yuqori o‘lchamli fazoga aylantiradigan funktsiyalardir. U uzluksiz vazifani ajraladigan vazifaga aylantiradi. Yadro funktsiyasi har qanday bo‘lishi mumkin: chiziqli, polinom, rbf va sigmasimon. Ushbu misolda biz chiziqli yadrodan foydalanamiz.

Endi quyidagi paketlarni import qilaylik.

```
import pandas as pd
import numpy as np
```

```
from sklearn import svm, datasets
import matplotlib.pyplot as plt
```

Endi kiritilgan ma'lumotlarni yuklang.

```
iris = datasets.load_iris()
```

Biz birinchi ikkita funktsiyadan foydalanamiz.

```
X = iris.data[:, :2]
y = iris.target
```

Biz asl ma'lumotlardan foydalanib, qo'llab-quvvatlovchi vektor mashinasining chegaralarini quramiz. Grafikni chizish uchun panjara yaratamiz.

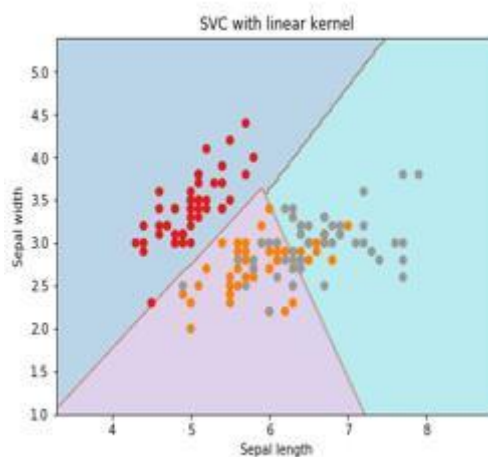
```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
np.arange(y_min, y_max, h))
X_plot = np.c_[xx.ravel(), yy.ravel()]
```

Biz tartibga solish parametrining qiymatini belgilashimiz kerak.

```
C = 1.0
```

Biz SVM klassifikatori ob'ektini yaratishimiz kerak.

```
Svc_klassifikatori = svm_classifier . SVC ( yadro = "chiziqli" ,
C = C , qaror_funksiyasi_shakli = 'ovr' ). mos ( X , y )
Z = svc_classifier . bashorat qilish ( X_plot )
Z = Z. qayta shakllantiring ( xx.shape ).
plt . raqam ( figsiz = ( 15 , 5 ))
plt . subplot ( 121 )
plt . kontur ( xx , yy , Z , cmap = plt . sm . tab10 , alfa = 0,3 )
plt . tarqalish ( X[:, 0 ], X[:, 1 ], c = y , cmap = plt . sm . Set 1 )
plt . xlabel ( 'Sepal uzunligi' )
plt . ylabel ( 'Sepal kengligi' )
plt . xlim ( xx . min () , xx . maks () )
plt . sarlavha ( 'chiziqli yadroli SVC' )
```

Logistik regressiya

Asosan, logistik regressiya modeli nazorat ostidagi tasniflash algoritmlari oilasining bir a'zosidir. Logistik regressiya logistik funktsiyadan foydalanib, ehtimolliklarni baholash orqali bog'liq va mustaqil o'zgaruvchilar o'rtasidagi munosabatni o'lchaydi.

Bu erda, agar qaram va mustaqil o'zgaruvchilar haqida gapiradigan bo'lsak, u holda qaram o'zgaruvchi biz bashorat qiladigan maqsadli sinf o'zgaruvchisidir va boshqa tomondan, mustaqil o'zgaruvchilar maqsadli sinfni bashorat qilish uchun foydalanmoqchi bo'lgan xususiyatlardir.

Logistik regressiyada ehtimolliklarni baholash voqea sodir bo'lish ehtimolini bashorat qilishni anglatadi. Misol uchun, do'kon egasi do'konga kirgan mijoz o'yin stantsiyasini (masalan,) sotib oladimi yoki yo'qmi, bashorat qilishni xohlaydi. Xaridor ko'plab xususiyatlarga ega bo'ladi - jinsi, yoshi va boshqalar. voqea sodir bo'lish ehtimolini bashorat qilish uchun do'kon sotuvchisi tomonidan hisobga olinadi, ya'ni o'yin stantsiyasini sotib olish yoki olmaslik. Logistik funktsiya sigmasimon egri chiziq bo'lib, u turli parametrlarga ega funktsiyani chizish uchun ishlatiladi.

Kerakli talablar

Tizimimizga **Tkinter** paketini o'rnatishimiz kerak. Uni <https://docs.python.org/2/library/tkinter.html> dan o'rnatish mumkin. Endi quyidagi kod yordamida logistik regressiya yordamida klassifikator yaratishimiz mumkin -

Avval biz ba'zi paketlarni import qilamiz:

```
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
```

Endi biz namuna ma'lumotlarini aniqlashimiz kerak, uni quyidagicha bajarish mumkin:

```
X = np.array([[2, 4.8], [2.9, 4.7], [2.5, 5], [3.2, 5.5], [6, 5], [7.6, 4],
              [3.2, 0.9], [2.9, 1.9], [2.4, 3.5], [0.5, 3.4], [1, 4], [0.9, 5.9]])
y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])
```

Keyinchalik, logistik regressiya klassifikatorini yaratishimiz kerak, uni quyidagicha bajarish mumkin:

```
Classifier_LR = linear_model.LogisticRegression(solver = 'liblinear', C = 75)
```

Va nihoyat, biz ushbu tasniflagichni o'rgatishimiz kerak:

```
Classifier_LR.fit(X, y)
```

Endi natijani qanday tasavvur qilishimiz mumkin? Buni Logistic_visualize() – nomli funksiya yaratish orqali amalga oshirish mumkin:

```
Def Logistic_visualize(Classifier_LR, X, y):
    min_x, max_x = X[:, 0].min() - 1.0, X[:, 0].max() + 1.0
    min_y, max_y = X[:, 1].min() - 1.0, X[:, 1].max() + 1.0
```

Yuqoridagi satrda biz to'rda ishlatiladigan minimal va maksimal X va Y qiymatlarini aniqladik. Bundan tashqari, biz to'rni qurish uchun qadam hajmini aniqlaymiz.

```
mesh_step_size = 0.02
```

X va Y qiymatlarining to'rini quyidagicha aniqlaymiz:

```
x_vals, y_vals = np.meshgrid(np.arange(min_x, max_x, mesh_step_size),
```

```
np.arange(min_y, max_y, mesh_step_size))
```

Quyidagi kod yordamida biz klassifikatorni tarmoqli panjarada ishga tushirishimiz mumkin:

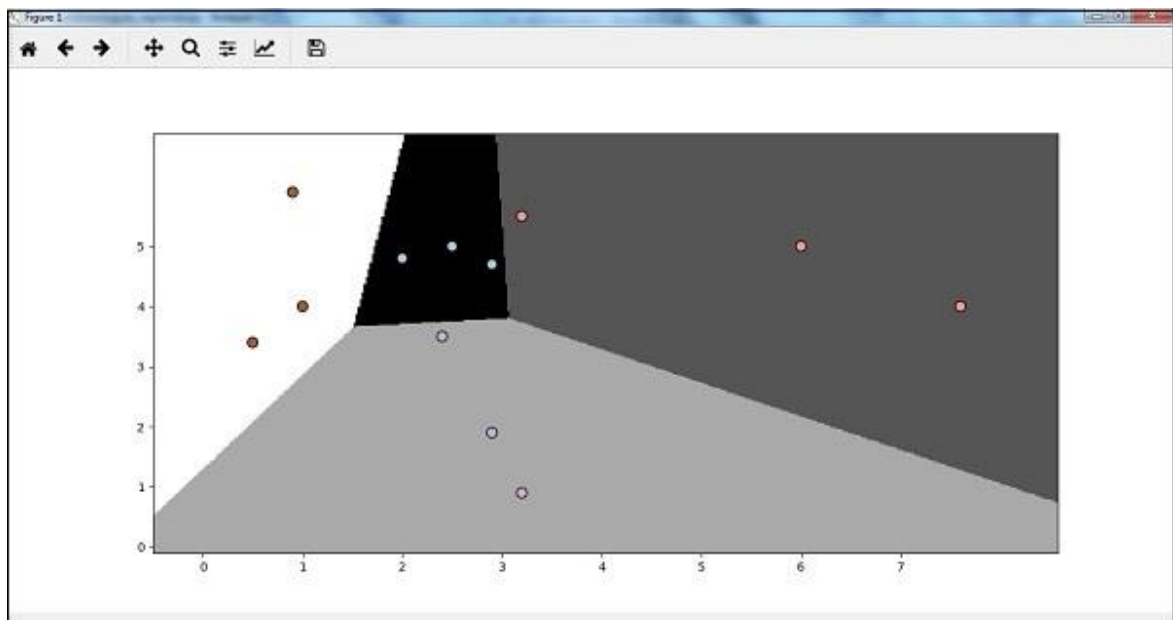
```
output = classifier.predict(np.c_[x_vals.ravel(), y_vals.ravel()])
output = output.reshape(x_vals.shape)
plt.figure()
plt.pcolormesh(x_vals, y_vals, output, cmap = plt.cm.gray)

plt.scatter(X[:, 0], X[:, 1], c = y, s = 75, edgecolors = 'black',
linewidth=1, cmap = plt.cm.Paired)
```

Quyidagi kod qatori grafik chegaralarini ko'rsatadi:

```
plt.xlim(x_vals.min(), x_vals.max())
plt.ylim(y_vals.min(), y_vals.max())
plt.xticks((np.arange(int(X[:, 0].min() - 1), int(X[:, 0].max() + 1), 1.0)))
plt.yticks((np.arange(int(X[:, 1].min() - 1), int(X[:, 1].max() + 1), 1.0)))
plt.show()
```

Endi, kodni ishga tushirgandan so'ng, biz quyidagi natijani olamiz, logistik regressiya tasniflagichi:



Qaror daraxti klassifikatori

Qaror daraxti asosan ikkilik daraxtga asoslangan blok-sxema bo'lib, unda har bir tugun kuzatuvlar guruhini ma'lum bir xususiyat o'zgaruvchisi bo'yicha ajratadi.

Bu yerda biz qaror daraxti klassifikatorini erkak yoki ayol jinsini bashorat qilish uchun yaratamiz. Biz juda kichik ma'lumotlar to'plamini ishlatamiz, unda 19 ta namuna mavjud bo'ladi. Ushbu namunalar ikkita xususiyat – "bo'y" va "soch uzunligi" dan iborat bo'ladi.

Majburiy shart

Quyidagi klassifikatorni yaratish uchun bizga **pydotplus** va **graphviz** o'rnatilgan bo'lishi kerak. Asosan, **graphviz** - bu Dot fayllar yordamida grafik chizish uchun mo'ljallangan vosita, **pydotplus** esa Graphviz-ning Dot tilini qo'llab-quvvatlovchi moduldir. Uni paket menejeri yoki `pip` yordamida o'rnatish mumkin.

Endi qaror daraxti klassifikatorini quyidagi Python kodi yordamida qurishimiz mumkin:

Avval ba'zi muhim kutubxonalarni quyidagicha import qilamiz:

```
import pydotplus
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.metrics import classification_report
from sklearn import cross_validation
import collection
```

Endi biz ma'lumotlar to'plamini quyidagicha ta'minlashimiz kerak:

```
X = [[165,19],[175,32],[136,35],[174,65],[141,28],[176,15],[131,32],
[166,6],[128,32],[179,10],[136,34],[186,2],[126,25],[176,28],[112,38],
[169,9],[171,36],[116,25],[196,25]]

Y = ['Man','Woman','Woman','Man','Woman','Man','Woman','Man','Woman',
'Man','Woman','Man','Woman','Woman','Woman','Man','Woman','Woman','Man']
data_feature_names = ['height','length of hair']

X_train, X_test, Y_train, Y_test = cross_validation.train_test_split
(X, Y, test_size=0.40, random_state=5)
```

Ma'lumotlar to'plamini taqdim etgandan so'ng, biz modelni moslashtirishimiz kerak, buni quyidagicha amalga oshirish mumkin:

```
clf = дерево.DecisionTreeClassifier()
clf = clf.подходит(X,Y)
```

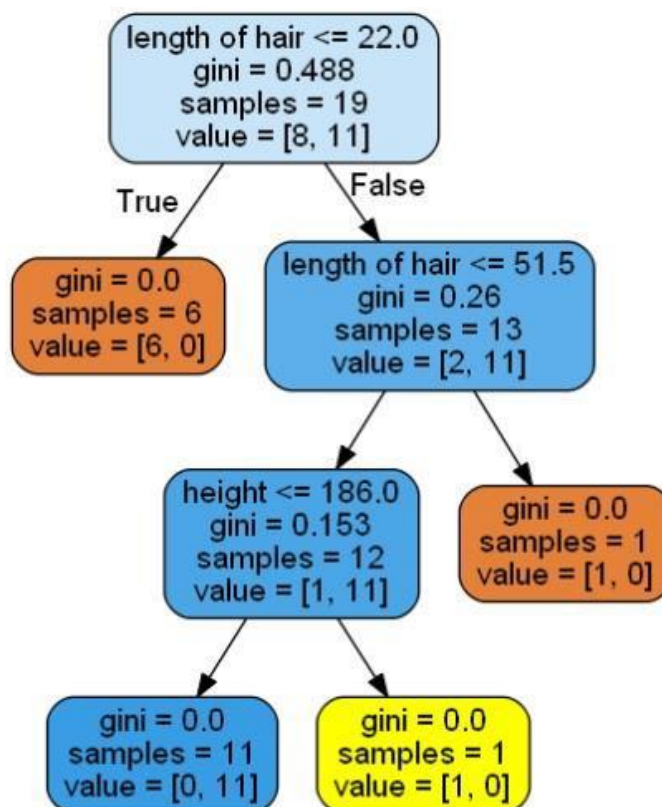
Quyidagi Python kodi yordamida bashorat qilish mumkin:

```
prediction = clf.predict([[133,37]])  
print(prediction)
```

Biz qarorlar daraxtini quyidagi Python kodidan foydalanib tasavvur qilishimiz mumkin.

```
dot_data = tree.export_graphviz(clf,feature_names = data_feature_names,  
                                out_file = None, filled = True, rounded = True)  
graph = pydotplus.graph_from_dot_data(dot_data)  
colors = ('orange', 'yellow')  
edges = collections.defaultdict(list)  
  
for edge in graph.get_edge_list():  
    edges[edge.get_source()].append(int(edge.get_destination()))  
  
for edge in edges: edges[edge].sort()  
  
for i in range(2):dest = graph.get_node(str(edges[edge][i]))[0]  
    dest.set_fillcolor(colors[i])  
graph.write_png('Decisiontree16.png')
```

U yuqoridagi kod uchun bashoratni ['**Woman**'] sifatida chiqaradi va quyidagi qarorlar daraxtini yaratadi.



Uni sinab ko‘rish uchun biz bashoratdagi funktsiya qiymatlarini o‘zgartirishimiz mumkin.

Tasodifiy o‘rmon klassifikatori

Biz bilamizki, ansambl usullari — bu mashinani o‘rganish modellarini kuchliroq mashinani o‘rganish modeliga birlashtiradigan usullardir. Tasodifiy o‘rmon — qaror daraxtlari to‘plami bo‘lib, ansambl usullaridan biridir. Bu alohida qaror daraxtidan yaxshiroq, chunki bashorat qilish imkoniyatlarini saqlab qolgan holda natijalarni o‘rtacha olish orqali haddan tashqari moslashuvni kamaytiradi.

Quyida biz **scikit-learn** kutubxonasidan foydalangan holda saraton ma’lumotlar to‘plamida tasodifiy o‘rmon modelini amalga oshiramiz.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
import matplotlib.pyplot as plt
import numpy as np
```

Endi biz shunday va minus kabi bajarilishi mumkin bo‘lgan ma’lumotlar to‘plamini taqdim etishimiz kerak:

```
cancer = load_breast_cancer()
X_train, X_test, y_train,
y_test = train_test_split(cancer.data, cancer.target, random_state = 0)
```

Ma’lumotlar to‘plamini taqdim etgandan so‘ng, biz modelni moslashtirishimiz kerak, buni quyidagicha amalga oshirish mumkin.

```
forest = RandomForestClassifier(n_estimators = 50, random_state = 0)
forest.fit(X_train, y_train)
```

Endi o‘qitish va test to‘plamining aniqligini oling: Agar biz baholovchilar sonini ko‘paytirsak, test to‘plamining aniqligi ham ortadi.

```
print('Accuracy on the training subset:(.3f)',format(forest.score(X_train,y_train)))
print('Accuracy on the training subset:(.3f)',format(forest.score(X_test,y_test)))
```

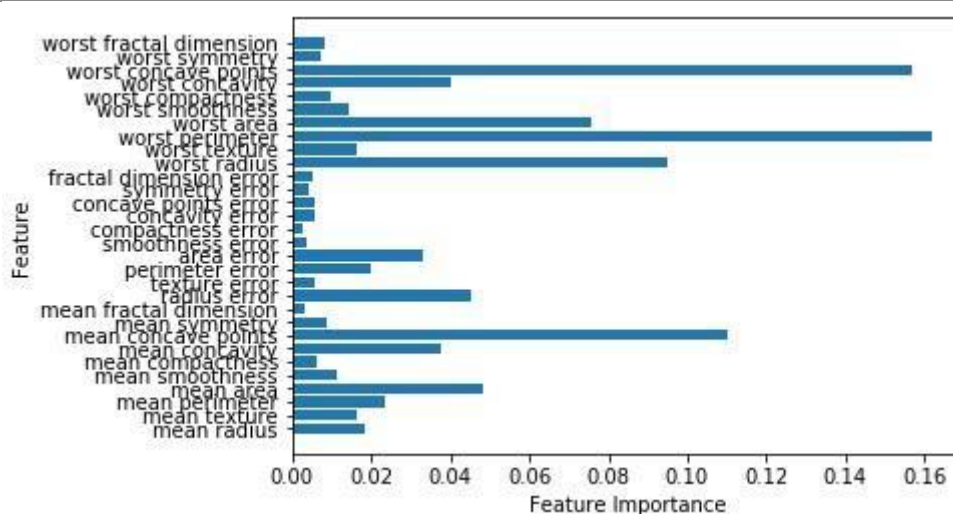
Xulosa

```
Accuracy on the training subset:(.3f) 1.0
```

Accuracy on the training subset:(.3f) 0.965034965034965

Endi, qaror daraxti kabi, tasodifiy o'rmonda qaror daraxtiga qaraganda xususiyat og'irliklari haqida yaxshiroq tushuncha beradigan **feature_importance** moduli mavjud. U quyidagicha tuzilishi va ko'rsatilishi mumkin.

```
n_features = cancer.data.shape[1]
plt.barh(range(n_features),forest.feature_importances_, align='center')
plt.yticks(np.arange(n_features),cancer.feature_names)
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.show()
```



Klassifikatorning ishlashi

Mashinani o'rganish algoritmini amalga oshirgandan so'ng, biz model qanchalik samarali ekanligini aniqlashimiz kerak. Samaradorlikni o'lchash mezonlari ma'lumotlar to'plami va ko'rsatkichlarga asoslangan bo'lishi mumkin. Turli xil mashinani o'rganish algoritmlarini baholash uchun turli xil ishlash ko'rsatkichlaridan foydalanishimiz mumkin. Misol uchun, agar klassifikator turli ob'ektlar tasvirlarini ajratish uchun ishlatilsa, biz o'rtacha aniqlik, AUC va boshqalar kabi tasniflash ko'rsatkichlaridan foydalanishimiz mumkin. Qaysidir ma'noda, mashinani o'rganish modelimizni baholash uchun biz tanlagan ko'rsatkich juda muhim, chunki ko'rsatkichlarni tanlash mashinani o'rganish algoritmining ishlashi qanday o'lchanishi va taqqoslanishiga ta'sir qiladi. Quyida ba'zi ko'rsatkichlar -

Chalkashlik matritsasi

U asosan tasniflash muammosini hal qilish uchun ishlatiladi, bunda chiqish ikki yoki undan ortiq turdagi sinflar bo'lishi mumkin. Bu tasniflagichning ishlashini o'lchashning eng oddiy usuli. Chalkashlik matritsasi asosan ikki o'lchovli jadval, ya'ni "Haqiqiy" va "Prognoz qilingan". Ikkala o'lchovda ham "haqiqiy ijobiy (TP)", "haqiqiy salbiy (TN)", "noto'g'ri ijobiy (FP)", "noto'g'ri salbiy (FN)" mavjud.

		Actual	
		1	0
Predicted	1	True Positives (TP)	False Positives (FP)
	0	False Negatives (FN)	True Negatives (TN)

Confusion Matrix

Yuqoridagi chalkashlik matritsasida 1 ijobiy sinfni va 0 salbiy sinfni ifodalaydi.

Quyida chalkashlik matritsasi - bilan bog'liq atamalar keltirilgan

- **Haqiqiy ijobiy**lar - TP - bu ma'lumotlar nuqtasining haqiqiy klassi 1 bo'lgan va bashorat qilingan sinf ham 1 bo'lgan holatlar.
- **Haqiqiy salbiy** - TN - bu ma'lumotlar nuqtasining haqiqiy klassi 0 bo'lgan va bashorat qilingan sinf ham 0 bo'lgan holatlar.
- **Noto'g'ri ijobiy** - FP - bu ma'lumotlar nuqtasining haqiqiy klassi 0, bashorat qilingani esa 1 bo'lgan holatlar.
- **Noto'g'ri salbiy** - FN ma'lumotlar nuqtasining haqiqiy klassi 1 bo'lgan va bashorat qilingan sinf ham 0 bo'lgan holatlardir.

Aniqlik

Chalkashlik matritsasining o'zi ishlash ko'rsatkichi emas, lekin deyarli barcha ishlash matritsalarini chalkashlik matritsasiga asoslanadi. Ulardan biri aniqlikdir. Tasniflash muammolarida buni barcha turdagi bashoratlarga nisbatan model tomonidan tuzilgan noto'g'ri bashoratlar soni sifatida aniqlash mumkin. Aniqlikni hisoblash formulasi quyidagicha -

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Aniqlik

U asosan hujjatlarni qidirish uchun ishlatiladi. Bu qaytarilgan to'g'ri hujjatlar soni sifatida belgilanishi mumkin. Quyida - aniqligini hisoblash formulasi keltirilgan

$$Precision = \frac{TP}{TP + FP}$$

Fikr-mulohaza yoki sezgirlik

Bu model tomonidan qaytarilgan ijobiy natijalar soni sifatida aniqlanishi mumkin. Quyida modelning eslab qolish/sezuvchanligini hisoblash formulasi keltirilgan -

$$Recall = \frac{TP}{TP + FN}$$

Xususiyatlar

Bu model tomonidan qaytarilgan negativlar soni sifatida aniqlanishi mumkin. Bu ko'rib chiqishning mutlaqo aksi. Quyida modelning o'ziga xosligini hisoblash formulasi -

$$Specificity = \frac{TN}{TN + FP}$$

Sinf nomutanosibligi muammosi

Sinf nomutanosibligi - bu bir sinfga tegishli kuzatuvlar soni boshqa sinflarga tegishli kuzatuvlar sonidan sezilarli darajada kam bo'lgan stsenariy. Masalan, bu muammo, ayniqsa, noyob kasalliklarni, firibgar bank operatsiyalarini va hokazolarni aniqlashimiz kerak bo'lgan stsenariyda sezilarli.

Nomutanosib sinflar misoli

Nomutanosib sinflar mashina o'rganish muammolarida keng uchraydi. Bu holatda, bir sinf namunalarining soni boshqasiga nisbatan sezilarli darajada kamroq bo'ladi.

Total observations = 5000
Fraudulent Observations = 50
Non-Fraudulent Observations = 4950
Event Rate = 1%

Yechim

Sinflarni muvozanatlash muvozanatsiz sinflar muammosini hal qilish vazifasini bajaradi. Sinf muvozanatining asosiy maqsadi ozchilik sinfidan foydalanish chastotasini oshirish yoki ko'pchilik sinfidan foydalanish chastotasini kamaytirishdir. Quyida sinflar nomutanosibligi muammosini hal qilishning yondashuvlari keltirilgan

Qayta namuna olish

Qayta namuna olish - bu o'qitish va sinovdan o'tkazish uchun namuna ma'lumotlar to'plamini qayta tiklash uchun foydalaniladigan bir qator usullar. Modelning aniqligini oshirish uchun qayta namuna olish amalga oshiriladi. Quyida ba'zi qayta namuna olish usullari keltirilgan -

- **Tasodifiy kam namuna olish** - Bu usul ko'pchilik sinflarning misollarini tasodifiy yo'q qilish orqali sinflarning taqsimlanishini muvozanatlashtirishga qaratilgan. Bu ko'pchilik va ozchilik sinfi misollari muvozanatlashtirilgunga qadar amalga oshiriladi.

Total observations = 5000
Fraudulent Observations = 50
Non-Fraudulent Observations = 4950
Event Rate = 1%

Bunday holda, biz firibgarlik bo'lmagan holatlardan almashtirilmasdan namunalarning 10 foizini olamiz va keyin ularni firibgarlik bilan bog'liq holatlar bilan birlashtiramiz -

Tasodifiy kam tanlanishdan keyin firibgarliksiz kuzatuvlar = 4950 dan 10% = 495.

Soxta kuzatuvlar bilan birlashtirilgandan keyin kuzatuvlarning umumiy soni = 50 + 495 = 545

Shuning uchun, kam namuna olishdan keyin yangi ma'lumotlar to'plami uchun hodisa tezligi endi = 9%

Ushbu usulning asosiy afzalligi shundaki, u bajarish vaqtini qisqartirishi va saqlashni yaxshilashi mumkin. Ammo boshqa tomondan, u o'quv ma'lumotlari namunalari sonini kamaytirish orqali foydali ma'lumotlarni bekor qilishi mumkin.

- **Tasodifiy kamroq namuna olish (Random Undersampling)**

Bu usul klasslar taqsimotini muvozanatlash uchun ko'pchilik klassdan tasodifiy ravishda namunalarni chiqarib tashlashni o'z ichiga oladi. Bu jarayon shu bilan davom etadi, bir vaqtning o'zida ko'pchilik va ozchilik klasslaridagi namunalar soni tenglashtiriladi.

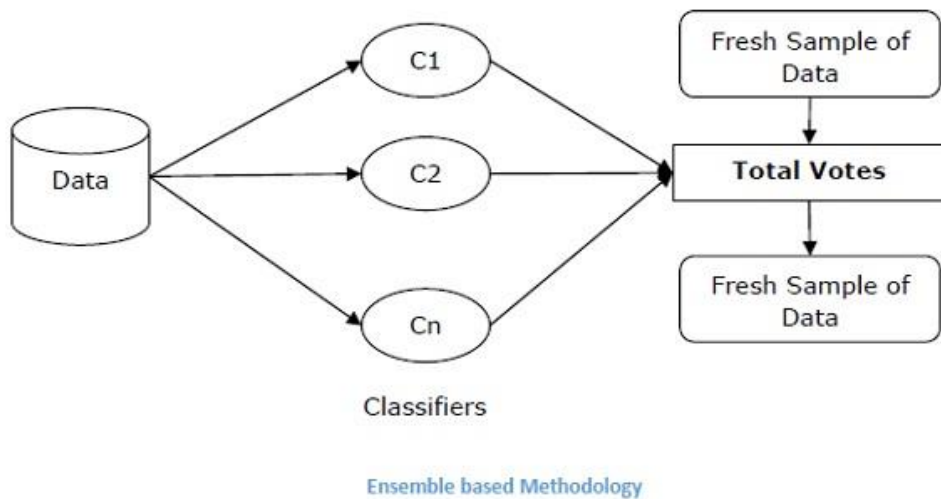
Total observations = 5000
Fraudulent Observations = 50
Non-Fraudulent Observations = 4950
Event Rate = 1%

Agar biz 50 ta soxta kuzatuvni 30 marta takrorlasak, u holda ozchilik kuzatuvlarini takrorlagandan keyin soxta kuzatuvlar 1500 ta bo'ladi. Va keyin qayta namuna olishdan keyin yangi ma'lumotlardagi kuzatuvlarning umumiy soni $4950 + 1500 = 6450$ bo'ladi. Shuning uchun hodisa tezligi yangi ma'lumotlar to'plami $1500 / 6450 = 23\%$ bo'ladi.

Ushbu usulning asosiy afzalligi shundaki, foydali ma'lumotlar yo'qolmaydi. Ammo boshqa tomondan, u kichikroq sinf voqealarini takrorlaganligi sababli, haddan tashqari moslashish ehtimoli ortadi.

Ansambl usullari

Ushbu metodologiya asosan mavjud tasniflash algoritmlarini muvozanatsiz ma'lumotlar to'plamlari uchun mos qilish uchun o'zgartirish uchun ishlatiladi. Ushbu yondashuvda biz dastlabki ma'lumotlar asosida bir nechta ikki bosqichli tasniflagichlarni yaratamiz va keyin ularning bashoratlarini jamlaymiz. Tasodifiy o'rmon klassifikatori ansambl asosidagi klassifikatorga misol bo'la oladi.



O‘z-o‘zini nazorat qilish uchun savollar

1. Klassifikatsiya?
2. Python-da klassifikator yaratish bosqichlar?
3. Ma’lumotlarni to‘plamlarga ajratish?
4. Model yaratish?
5. Model va uning aniqligini baholash?
6. Tayanch vektorlar mashinasi (SVM)?
7. Logistik regressiya?
8. Qaror daraxti klassifikatori?
9. Tasodifiy o‘rmon klassifikatori?
10. Chalkashlik matritsasi?
11. Ansambl usullari?

2.4-§. KO‘RSATMA BO‘YISH O‘RGANISH: REGRESSIYA

Regressiya — bu statistikada va mashinani o‘rganishda eng muhim vositalardan biridir. Agar biz aytadigan bo‘lsak, mashinani o‘rganish yo‘li regressiyadan boshlanadi, noto‘g‘ri bo‘lmaydi. Uni parametrik usul deb ta’riflash mumkin, chunki u bizga ma’lumotlar asosida qarorlar qabul qilish imkonini beradi yoki boshqa so‘zlar bilan aytganda, kirish va chiqish o‘zgaruvchilarining o‘zaro bog‘liqligini o‘rganish orqali ma’lumotlarga asoslangan prognozlarni qilish imkonini beradi. Bu yerda chiqish o‘zgaruvchilari, kirish o‘zgaruvchilariga bog‘liq bo‘lib, uzluksiz qiymatga ega haqiqiy sonlar sifatida ko‘rib chiqiladi. Regressiyada kirish va chiqish o‘zgaruvchilari o‘rtasidagi o‘zaro bog‘liqlik muhimdir, chunki bu bizga chiqish o‘zgaruvchisining qiymati kirish o‘zgaruvchisi o‘zgarganida qanday o‘zgarishini tushunishga yordam beradi. Regressiya ko‘pincha narxlarni, iqtisodiyotni, o‘zgarishlarni va boshqa ko‘plab narsalarni prognoz qilishda qo‘llaniladi.

Python da regressorlarni yaratish

Ushbu bo‘limda biz bitta va ko‘p o‘zgaruvchan regressorlarni qanday yaratishni o‘rganamiz.

Chiziqli regressor / Yagona o‘zgaruvchan regressor

Keling, bir nechta kerakli paketlarni ko‘rib chiqaylik.

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt
```

Endi biz kiritilgan ma’lumotlarni taqdim etishimiz kerak va biz ma’lumotlarimizni linear.txt deb nomlangan faylda saqladik.

```
input = 'D:/ProgramData/linear.txt'
```

np.loadtxt funksiyasidan foydalanib yuklashimiz kerak .

```
input_data = np.loadtxt(input, delimiter=',')
X, y = input_data[:, :-1], input_data[:, -1]
```

Keyingi qadam modelni o'rgatishdir. O'qitish va sinovdan o'tkazish uchun misollar keltiraylik.

```
training_samples = int(0.6 * len(X))
testing_samples = len(X) - num_training

X_train, y_train = X[:training_samples], y[:training_samples]

X_test, y_test = X[training_samples:], y[training_samples:]
```

Endi chiziqli regressor ob'ektini yaratishimiz kerak.

```
reg_linear = linear_model.LinearRegression()
```

O'quv namunalari yordamida ob'ektni o'rgating.

```
reg_linear.fit(X_train, y_train)
```

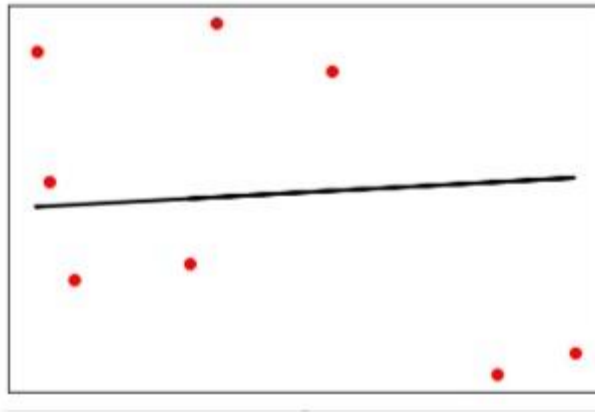
Sinov ma'lumotlari asosida bashorat qilishimiz kerak.

```
y_test_pred = reg_linear.predict(X_test)
```

Endi grafik tuzing va ma'lumotlarni tasavvur qiling.

```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_test, y_test_pred, color = 'black', linewidth = 2)
plt.xticks(())
plt.yticks(())
plt.show()
```

Natija



Endi chiziqli regressiyamizning ishlashini quyidagicha hisoblashimiz mumkin:

```
print("Performance of Linear regressor:")
print("Mean absolute error =", round(sm.mean_absolute_error(y_test, y_test_pred), 2))
print("Mean squared error =", round(sm.mean_squared_error(y_test, y_test_pred), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, y_test_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test, y_test_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))
```

Natija

Chiziqli regressor ishlashi:

```
Mean absolute error = 1.78
Mean squared error = 3.89
Median absolute error = 2.01
Explain variance score = -0.09
R2 score = -0.09
```

Yuqoridagi kodda biz ushbu kichik ma'lumotdan foydalanganmiz. Agar sizga katta ma'lumotlar to'plami kerak bo'lsa, kattaroq ma'lumotlar to'plamini import qilish uchun `sklearn.dataset` dan foydalanishingiz mumkin.

```
2,4.82,9,4.72,5,53.2,5.56,57.6,43.2,0.92,9,1.92,4,
3.50,5,3.41,40,9,5.91,2,2.58,3.2,5.65,1,1.54,5,
1.22,3,6.32,1,2.8
```

Ko'p o'zgaruvchan regressor

Avval bir nechta kerakli paketlarni import qilaylik:

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
```

Endi biz kiritilgan ma'lumotlarni taqdim etishimiz kerak va biz ma'lumotlarimizni linear.txt deb nomlangan faylda saqladik.

```
input = 'D:/ProgramData/Mul_linear.txt'
```

np.loadtxt funksiyasidan foydalanib yuklaymiz.

```
input_data = np.loadtxt(input, delimiter=',')
X, y = input_data[:, :-1], input_data[:, -1]
```

Keyingi qadam modelni o'rgatishdir; Biz ta'lim va sinov namunalarini taqdim etamiz.

```
training_samples = int(0.6 * len(X))
testing_samples = len(X) - num_training
X_train, y_train = X[:training_samples], y[:training_samples]
X_test, y_test = X[training_samples:], y[training_samples:]
```

Endi chiziqli regressor ob'ektini yaratishimiz kerak.

```
reg_linear_mul = linear_model.LinearRegression()
```

O'quv namunalari yordamida ob'ektni o'rgating.

```
reg_linear_mul.fit(X_train, y_train)
```


Endi, nihoyat, sinov ma'lumotlaridan foydalanib, bashorat qilishimiz kerak.

```
y_test_pred = reg_linear_mul.predict(X_test)

print("Performance of Linear regressor:")
print("Mean absolute error =", round(sm.mean_absolute_error(y_test, y_test_pred), 2))
print("Mean squared error =", round(sm.mean_squared_error(y_test, y_test_pred), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, y_test_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test, y_test_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))
```

Natija

Chiziqli regressor ishlashi:

```
Mean absolute error = 0.6
Mean squared error = 0.65
Median absolute error = 0.41
Explain variance score = 0.34
R2 score = 0.33
```

Endi biz 10 darajali ko'phadni yaratamiz va regressorni o'rgatamiz. Biz ma'lumotlar nuqtasiga misol keltiramiz.

```
polynomial = PolynomialFeatures(degree = 10)
X_train_transformed = polynomial.fit_transform(X_train)
datapoint = [[2.23, 1.35, 1.12]]
poly_datapoint = polynomial.fit_transform(datapoint)
poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)
print("\nLinear regression:\n", reg_linear_mul.predict(datapoint))
print("\nPolynomial regression:\n", poly_linear_model.predict(poly_datapoint))
```

Natija

Chiziqli regressiya:

```
[2.40170462]
```

Polinomli regressiya:

[1.8697225]

Yuqoridagi kodda biz ushbu kichik ma'lumotdan foydalanganmiz. Agar sizga katta ma'lumotlar to'plami kerak bo'lsa, kattaroq ma'lumotlar to'plamini import qilish uchun *sklearn.dataset* dan foydalanishingiz mumkin.

```
2,4.8,1.2,3.22.9,4.7,1.5,3.62.5,5,2.8,23.2,5.5,3.5,2.16,5,  
2,3.27.6,4,1.2,3.23.2,0.9,2.3,1.42.9,1.9,2.3,1.22.4,3.5,  
2.8,3.60.5,3.4,1.8,2.91,4,3,2.50.9,5.9,5.6,0.81.2,2.58,  
3.45,1.233.2,5.6,2,3.25.1,1.5,1.2,1.34.5,1.2,4.1,2.32.3,  
6.3,2.5,3.22.1,2.8,1.2,3.6
```

O'z-o'zini nazorat qilish uchun savollar

1. Python da regressorlarni yaratish?
2. Chiziqli regressor / Yagona o'zgaruvchan regressor?
3. Ko'p o'zgaruvchan regressor?

2.5-§. MANTIQUIY DASTURLASH

Bu bo‘limda biz mantiqiy dasturlashni ko‘rib chiqamiz. Mantiqiy dasturlash – bu ma’lumotlarni tashkil etish va ular bilan mantiqiy operatsiyalarni bajarish uchun matematik mantiqdan foydalanishdir. Sun’iy intellekt va mashinaviy o‘qitish sohalarida mantiqiy dasturlash juda muhim ahamiyatga ega, chunki u to‘g‘ri qarorlar qabul qilishda yordam beradi. Bu usul yordamida biz tizimlarni qarorlar qabul qilishga o‘rgatamiz, shu bilan birga dasturiy ta’minot tizimlarida mantiqiy va matematik ifodalarni avtomatik ravishda ishlashini ta’minlaymiz.

Konsepsiya

Mantiqiy dasturlash — bu ikki so‘zning, ya’ni mantiq va dasturlashning kombinatsiyasi. Mantiqiy dasturlash — bu dasturlash paradigmlaridan biri bo‘lib, unda muammolar dastur ko‘rsatmalari yordamida faktlar va qoidalariga asoslangan tarzda ifodalanadi, lekin bu tizim rasmiy mantiqiy tizim asosida amalga oshiriladi. Boshqa dasturlash paradigmlariga, masalan, obyektga yo‘naltirilgan, funksional, deklarativ, protsedural va boshqalarga o‘xshab, bu ham dasturlashga xos yondashuvdir.

Mantiqiy dasturlash yordamida qanday muammolarni hal qilish mumkin

Mantiqiy dasturlash faktlar va qoidalar yordamida muammolarni hal qiladi. Shu sababli, ular mantiqiy dasturlashning qurilish bloklari deb ataladi. Mantiqiy dasturlashdagi har bir dastur uchun maqsadni aniqlash kerak. Mantiqiy dasturlashda muammolarni qanday hal qilish mumkinligini tushunish uchun biz qurilish bloklari — **faktlar va qoidalar** haqida bilishimiz kerak.

Faktlar

Haqiqatan ham, har bir mantiqiy dastur maqsadga erishish uchun faktlarga muhtoj. Faktlar dastur va ma’lumotlar haqidagi haqiqatni ifodalovchi bayonotlardir. Masalan, **Delhi** Hindistonning poytaxti degan bayonot fakt sifatida ko‘rsatilishi mumkin.

Qoidalar

Haqiqatan ham, qoidalar - bu bizga predmet sohasiga oid xulosalar chiqarish imkonini beradigan cheklovlardir. Qoidalar asosan turli faktlarni ifodalash uchun mantiqiy takliflar (predlogatsiyalar) shaklida yoziladi. Masalan, agar biz qandaydir o‘yin yaratmoqchi bo‘lsak, unda barcha qoidalar belgilanishi kerak.

Qoidalar mantiqiy dasturlashda ma’lum bir holatga oid xulosalar va qarorlar chiqarishga imkon beradi. Ular ma’lum shartlarga asoslangan xulosalarni chiqarish uchun ishlatiladi, va bu shartlar faktlar bilan bog‘lanadi

Mantiqiy dasturlashda har qanday muammoni hal qilish uchun qoidalar juda muhimdir. Qoidalar aslida faktlarni ifoda eta oladigan mantiqiy xulosalardir. Quyida – qoida sintaksisi keltirilgan

$A : - B_1, B_2, \dots, B_n.$

Bu erda A - bosh va $B_1, B_2, \dots, B_n$ - tana.

Masalan – ajdod (X, Y) :- ota (X, Y).

ajdod (X, Z): - ota (X, Y), ajdod (Y, Z).

Buni har bir X va Y uchun o‘qish mumkin, agar X Y ning otasi bo‘lsa va Y Z ning ajdodi bo‘lsa, X Z ning ajdodidir. Y va Y Z ning ajdodidir.

Foydali paketlarni o‘rnatish

Python-da mantiqiy dasturlashni boshlash uchun biz quyidagi ikkita paketni o‘rnatishimiz kerak -

Kanren — bu bizga biznes mantiqini yaratish uchun kodni soddalashtirish usulini beradi. Bu bizga mantiqni qoidalar va faktlar nuqtai nazaridan ifodalash imkonini beradi. Kanren — bu mantiqiy dasturlashni Python dasturlash tilida amalga oshirish uchun kutubxona bo‘lib, ular yordamida faktlar va qoidalar asosida yechimlar topish mumkin.

`pip install kanren`

SymPy — bu Python kutubxonasi bo‘lib, simvolli matematika uchun mo‘ljallangan. Uning maqsadi to‘liq funksional kompyuter algebra tizimi (CAS) bo‘lish, shu bilan birga oddiy va o‘qilishi oson kodni saqlab qolish va kengaytirish imkoniyatlarini taqdim etishdir.

SymPy kutubxonasini o'rnatish uchun quyidagi buyruqni terminalga kiriting:

```
pip install sympy
```

Mantiqiy dasturlashga misollari

Quyida mantiqiy dasturlash yordamida hal qilinishi mumkin bo'lgan ba'zi misollar keltirilgan

Matematik ifodalarni moslashtirish

Aslida biz mantiqiy dasturlash yordamida noma'lum qiymatlarni juda samarali tarzda topishimiz mumkin. Quyidagi Python kodi sizga topishga yordam beradi matematik ifoda -

Avval quyidagi paketlarni import qilishni o'ylab ko'ring:

```
from kanren import run, var, fact
from kanren.assoccomm import eq_assoccomm as eq
from kanren.assoccomm import commutative, associative
```

Biz foydalanadigan matematik amallarni aniqlashimiz kerak:

```
add = 'add'
mul = 'mul'
```

Qo'shish ham, ko'paytirish ham kommunikativ jarayonlardir. Demak, biz buni aniqlashtirishimiz kerak va buni quyidagicha qilish mumkin -

```
fact(commutative, mul)
fact(commutative, add)
fact(associative, mul)
fact(associative, add)
```

O'zgaruvchilarni aniqlash majburiydir; buni quyidagicha bajarish mumkin -

```
a,b= var('a'), var('b')
```

Biz ifodani asl naqshga moslashtirishimiz kerak. Bizda quyidagi boshlang'ich naqsh mavjud, bu asosan $(5 + a) * b$:

```
Original_pattern = (mul, (add, 5,a),b)
```

Bizda asl naqshga mos keladigan quyidagi ikkita ibora mavjud:

```
exp1 = (mul, 2, (add, 3, 1))  
exp2 = (add,5,(mul,8,1))
```

Chiqish quyidagi buyruq yordamida chop etilishi mumkin:

```
print(run(0, (a,b), eq(original_pattern, exp1)))  
print(run(0, (a,b), eq(original_pattern, exp2)))
```

Ushbu kodni ishga tushirgandan so'ng biz quyidagi natijani olamiz:

```
((3,2))  
()
```

a va **b** uchun qiymatlarni ifodalaydi . Birinchi ifoda asl naqshga mos keldi va **a** va **b** uchun qiymatlarni qaytardi , lekin ikkinchi ifoda asl naqshga mos kelmadi, shuning uchun hech narsa qaytarilmadi.

Odiy sonlarni tekshirish

Mantiqiy dasturlash yordamida biz sonlar ro'yxatidan tub sonlarni topishimiz va tub sonlarni ham hosil qilishimiz mumkin. Quyidagi Python kodi raqamlar ro'yxatidan tub sonni topadi va birinchi 10 ta tub sonni yaratadi.

Keling, avval quyidagi paketlarni import qilishni ko'rib chiqaylik.

```
from kanren import isvar, run, membero  
from kanren.core import success, fail, goaleval, condeseq, eq, var  
from sympy.ntheory.generate import prime, isprime  
import itertools as it
```

Endi biz berilgan raqamlar asosida tub sonlarni ma'lumotlar sifatida tekshiradigan prime_check funksiyasini aniqlaymiz.

```
def prime_check(x):
if isvar(x):
    return condeseq([(eq,x,p)] for p in map(prime, it.count(1)))
else:
    return success if isprime(x) else fail
```

Endi biz ishlatiladigan o‘zgaruvchini e’lon qilishimiz kerak:

```
x = var()
print((set(run(0,x,(membero,x,(12,14,15,19,20,21,22,23,29,30,41,44,52,62,65,85)),
(prime_check,x))))))
print((run(10,x,prime_check(x))))
```

Yuqoridagi kodning chiqishi quyidagicha bo‘ladi:

```
{ 19, 23, 29, 41 }
(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)
```

Jumboq yechish

Mantiqiy dasturlash 8 ta jumboq, zebra boshqotirmasi, sudoku, N-queen va boshqalar kabi ko‘plab muammolarni hal qilish uchun ishlatilishi mumkin. Bu erda biz zebra jumboqning shunday ko‘rinishdagi versiyasiga misol keltiramiz:

There are five houses.
The English man lives in the red house.
The Swede has a dog.
The Dane drinks tea.
The green house is immediately to the left of the white house.
They drink coffee in the green house.
The man who smokes Pall Mall has birds.
In the yellow house they smoke Dunhill.
In the middle house they drink milk.
The Norwegian lives in the first house.
The man who smokes Blend lives in the house next to the house with cats.
In a house next to the house where they have a horse, they smoke Dunhill.
The man who smokes Blue Master drinks beer.

The German smokes Prince.

The Norwegian lives next to the blue house.

They drink water in a house next to the house where they smoke Blend.

Python yordamida **zebra kimga tegishli (who owns zebra)** degan savolni hal qilamiz. Kerakli paketlarni import qilaylik:

```
from kanren import *  
from kanren.core import lall  
import time
```

Endi biz ikkita funktsiyani aniqlashimiz kerak - **left()** va **next()** kimning uyi chap tomonda yoki kimning uyi yonida ekanligini tekshirish:

```
def left(q, p, list):  
    return membro((q,p), zip(list, list[1:]))  
def next(q, p, list):  
    return conde([left(q, p, list)], [left(p, q, list)])
```

Endi biz uy o'zgaruvchisini quyidagicha e'lon qilamiz:

```
houses = var()
```

Biz lall paketi yordamida qoidalarni quyidagicha belgilashimiz kerak. 5 ta uy bor:

```
rules_zebraproblem = lall(  
    (eq, (var(), var(), var(), var(), var()), houses),  
    (membro,('Englishman', var(), var(), var(), 'red'), houses),  
    (membro,('Swede', var(), var(), 'dog', var()), houses),  
    (membro,('Dane', var(), 'tea', var(), var()), houses),  
    (left,(var(), var(), var(), var(), 'green'),  
    (var(), var(), var(), var(), 'white'), houses),  
    (membro,(var(), var(), 'coffee', var(), 'green'), houses),  
    (membro,(var(), 'Pall Mall', var(), 'birds', var()), houses),  
    (membro,(var(), 'Dunhill', var(), var(), 'yellow'), houses),
```



```
(eq,(var(), var(), (var(), var(), 'milk', var(), var()), var(), var()), houses),
(eq,((('Norwegian', var(), var(), var(), var()), var(), var(), var(), var()), houses),
(next,(var(), 'Blend', var(), var(), var()),
(var(), var(), var(), 'cats', var()), houses),
(next,(var(), 'Dunhill', var(), var(), var()),
(var(), var(), var(), 'horse', var()), houses),
(membero,(var(), 'Blue Master', 'beer', var(), var()), houses),
(membero,('German', 'Prince', var(), var(), var()), houses),
(next,('Norwegian', var(), var(), var(), var()),
(var(), var(), var(), var(), 'blue'), houses),
(next,(var(), 'Blend', var(), var(), var()),
(var(), var(), 'water', var(), var()), houses),
(membero,(var(), var(), var(), 'zebra', var()), houses)
)
```

Endi oldingi cheklovlar bilan hal qiluvchini ishga tushiring:

```
solutions = run(0, houses, rules_zebraproblem)
```

Quyidagi kod yordamida biz yechimni – dan chiqarishimiz mumkin:

```
output_zebra = [house for house in solutions[0] if 'zebra' in house][0][0]
```

Quyidagi kod yechimni chop etishga yordam beradi:

```
print ('\\n'+ output_zebra + 'owns zebra.')
```

Yuqoridagi kodning chiqishi quyidagicha bo'ladi:

```
German owns zebra.
```

O'z-o'zini nazorat qilish uchun savollar

1. Mantiqiy dasturlash
2. Matematik ifodalarni moslashtirish
3. Odiy sonlarni tekshirish

2.6-§. NAZORATSIZ O'RGANISH: KLASTERLASH

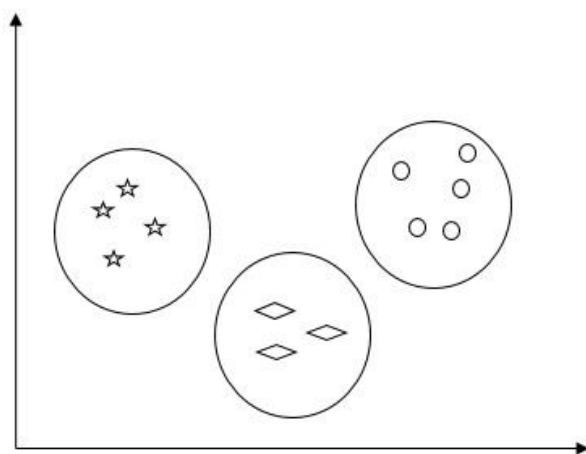
Nazoratsiz mashinani o'rgatish algoritmlarida ma'lumotlarni boshqarib turuvchi yoki yo'l-yo'riq ko'rsatuvchi nazoratchi bo'lmaydi. Shu sababli, bu usul ba'zilar tomonidan haqiqiy sun'iy intellektga yaqin deb hisoblanadi.

Nazoratsiz o'rganishda to'g'ri javob yoki yo'naltiruvchi o'qituvchi mavjud emas. Algoritmlar o'qitilish uchun ma'lumotlarda qiziqarli qonuniyatlarni topishni o'z zimmlariga oladi.

Klasterlash nima?

Klasterlash — bu nazoratsiz o'rgatish usullaridan biri bo'lib, ko'plab sohalarda keng qo'llaniladigan statistik ma'lumotlarni tahlil qilish usulidir. Klasterlash asosan kuzatuvlar to'plamini "klasterlar" deb ataladigan kichik to'plamlarga ajratish vazifasidir. Bunda bir klaster ichidagi kuzatuvlar ma'lum bir ma'noda bir-biriga o'xshash, boshqa klasterlardagi kuzatuvlardan esa farq qilishi kerak. Oddiy qilib aytganda, klasterlashning asosiy maqsadi — ma'lumotlarni o'xshashlik va o'xshamaslik asosida guruhlariga ajratishdir.

Masalan: Quyidagi diagrammada bir xil turdagi ma'lumotlar har xil klasterlarda ko'rsatilgan:



Ma'lumotlarni Klasterlash Algoritmlari

Quyida ma'lumotlarni klasterlash uchun ishlatiladigan mashhur algoritmlar haqida ma'lumot keltirilgan

K-o'rtacha Klasterlash Algoritmi

K-o‘rtacha klasterlash — mashhur algoritm bo‘lib, unda klasterlar sonini oldindan belgilash talab qilinadi. Ushbu algoritm yassi klasterlash usullariga kiradi va iterativ usulda ishlaydi. Bu interaktiv klasterlash algoritmidir. Ushbu algoritm uchun siz quyida keltirilgan amallarni bajarishingiz kerak -

1-qadam - Biz kichik guruhlarining kerakli sonini **K** ko‘rsatishimiz kerak.

2-qadam - Klasterlar sonini yozib oling va har bir ma’lumot nuqtasini tasodifiy klasterga tayinlang. Boshqacha qilib aytganda, biz ma’lumotlarimizni klasterlar soniga qarab tasniflashimiz kerak.

Ushbu bosqichda klaster markazlarini hisoblash kerak.

Bu iterativ algoritm bo‘lgani uchun biz global optima topmagunimizcha yoki boshqacha qilib aytganda, markazlar o‘zlarining optimal joylariga yetguncha har bir iteratsiya bilan **K**-o‘rtachalarining joylashuvini yangilashimiz kerak.

Quyidagi kod Pythonda K-o‘rtacha klasterlash algoritmini amalga oshirishda yordam beradi. Biz Scikit-learn modulidan foydalanamiz. Kerakli paketlarni import qilaylik:

```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
from sklearn.cluster import KMeans
```

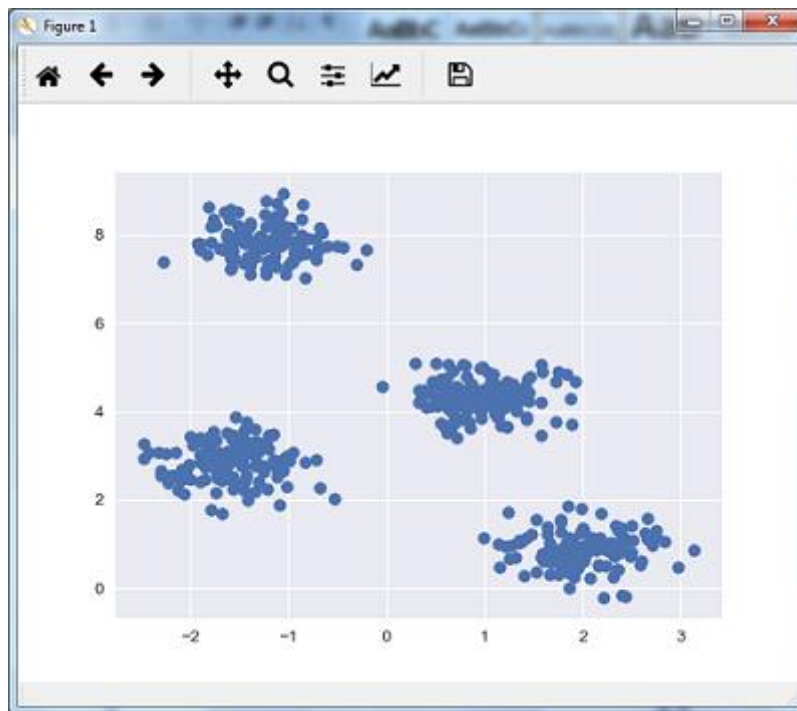
Quyidagi kod qatori **sklearn.dataset** paketidagi **make_blob** yordamida to‘rtta blobdan iborat 2D ma’lumotlar to‘plamini yaratishga yordam beradi.

```
from sklearn.datasets.samples_generator import make_blobs

X, y_true = make_blobs(n_samples = 500, centers = 4,
                        cluster_std = 0.40, random_state = 0)
```

Biz quyidagi kod yordamida ma’lumotlar to‘plamini tasavvur qilishimiz mumkin:

```
plt.scatter(X[:, 0], X[:, 1], s = 50);
plt.show()
```



Bu erda biz kmeanlarni KMeans algoritmi sifatida kerakli miqdordagi klasterlar parametri (n_clusters) bilan ishga tushiramiz.

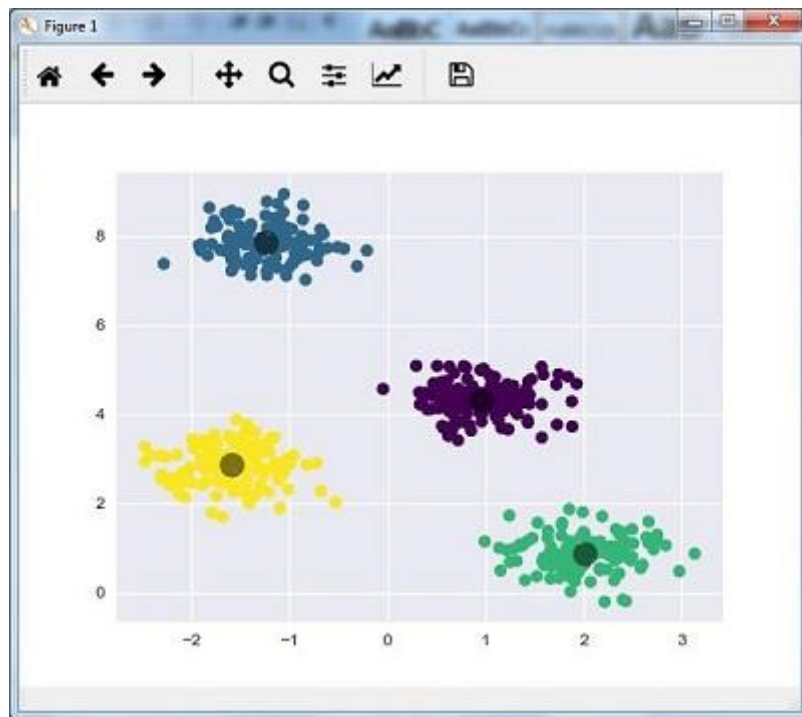
```
kmeans = KMeans(n_clusters = 4)
```

Biz kirish ma'lumotlaridan foydalanib, K-o'rtacha modelini o'rgatishimiz kerak.

```
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
plt.scatter(X[:, 0], X[:, 1], c = y_kmeans, s = 50, cmap = 'viridis')
centers = kmeans.cluster_centers_
```

Quyidagi kod bizga ma'lumotlarimiz asosida mashina natijalarini grafik va vizualizatsiya qilishda yordam beradi, shuningdek, topilishi kerak bo'lgan klasterlar soniga qarab konfiguratsiyani tanlashga yordam beradi.

```
plt.scatter(centers[:, 0], centers[:, 1], c = 'black', s = 200, alpha = 0.5);
plt.show()
```



O‘rtacha siljish algoritmi

Bu nazoratsiz o‘rganishda qo‘llaniladigan yana bir mashhur va kuchli klasterlash algoritmidir. U hech qanday taxmin qilmaydi, shuning uchun u parametrik bo‘lmagan algoritmdir. U ierarxik klasterlash yoki o‘rtacha siljishli klaster tahlili deb ham ataladi. Quyida ushbu algoritmning asosiy bosqichlari keltirilgan:

- Avvalo, biz o‘z klasteriga tayinlangan ma’lumotlar nuqtalaridan boshlashimiz kerak.
- Endi u centroidlarni hisoblab chiqadi va yangi markazlarning joylashuvini yangilaydi.
- Ushbu jarayonni takrorlash orqali biz klaster cho‘qqisini yaqinlashtiramiz, ya’ni. yuqori zichlikdagi hududga.
- Ushbu algoritm markazlar endi harakatlanmaydigan nuqtada to‘xtaydi.

Quyidagi koddan foydalanib, biz Python-da o‘rtacha siljish klasterlash algoritmini amalga oshiramiz. Biz Scikit-learn modulidan foydalanamiz. Kerakli paketlarni import qilaylik:

```
import numpy as np
from sklearn.cluster import MeanShift
```

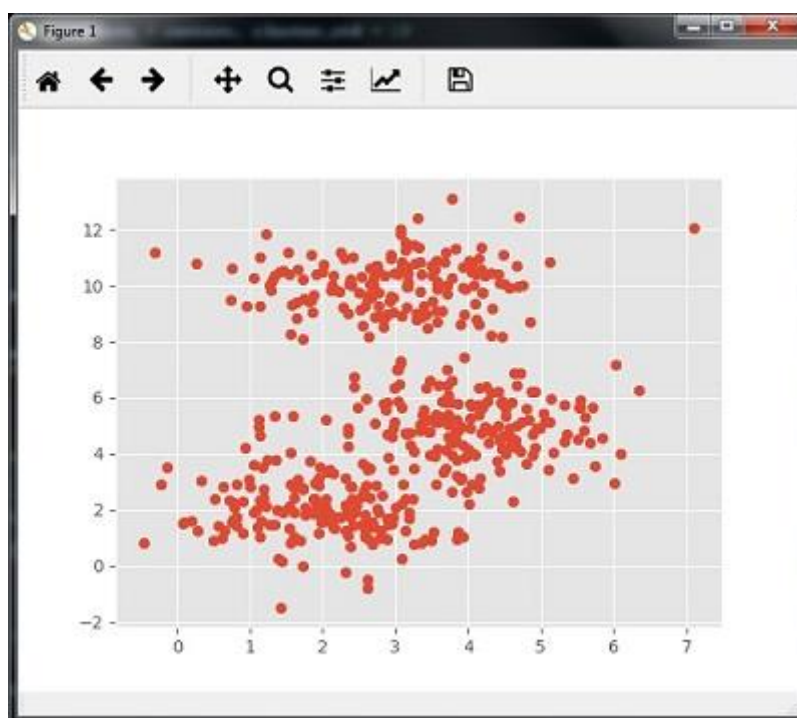
```
import matplotlib.pyplot as plt
from matplotlib import style
style.use("ggplot")
```

Quyidagi kod **sklearn.dataset** paketidagi **make_blob** yordamida to‘rtta blobdan iborat 2D ma’lumotlar to‘plamini yaratishga yordam beradi .

```
from sklearn.datasets.samples_generator import make_blobs
```

Quyidagi kod yordamida ma’lumotlar to‘plamini tasavvur qilishimiz mumkin

```
centers = [[2,2],[4,5],[3,10]]
X, _ = make_blobs(n_samples = 500, centers = centers, cluster_std = 1)
plt.scatter(X[:,0],X[:,1])
plt.show()
```



Endi biz kiritilgan ma’lumotlarga asoslangan o‘rtacha siljish klaster modelini o‘rgatishimiz kerak.

```
ms = MeanShift()
ms.fit(X)
labels = ms.labels_
```

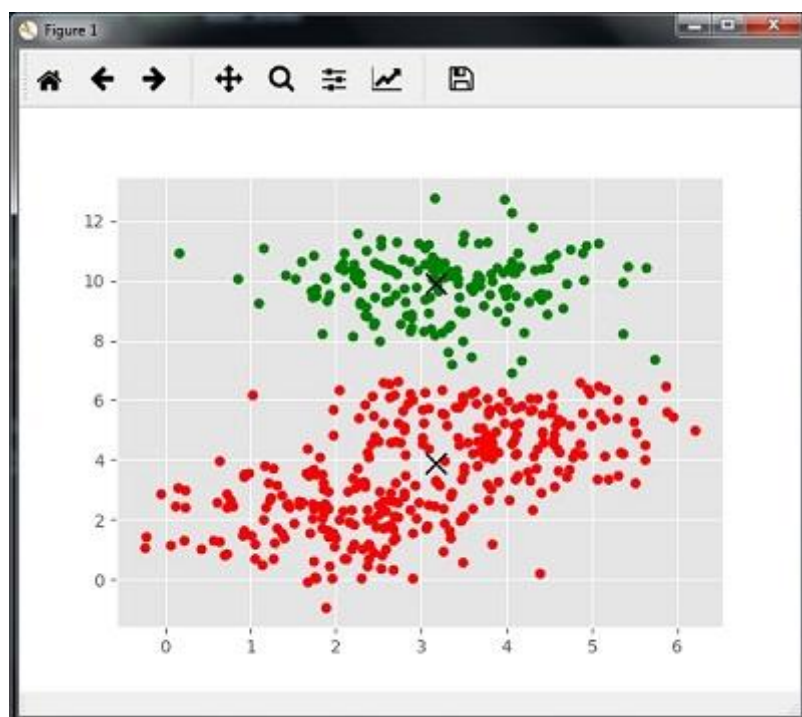
```
cluster_centers = ms.cluster_centers_
```

Quyidagi kod klaster markazlarini va kirishga muvofiq kutilgan klasterlar sonini chiqaradi

```
print(cluster_centers)
n_clusters_ = len(np.unique(labels))
print("Estimated clusters:", n_clusters_)
[[ 3.23005036  3.84771893]
 [ 3.02057451  9.88928991]]
Estimated clusters: 2
```

Quyidagi kod sizga bizning ma'lumotlarimiz asosida grafik yaratish va mashina natijalarini tasavvur qilishda yordam beradi, shuningdek, topilishi kerak bo'lgan klasterlar soniga qarab konfiguratsiyani tanlang.

```
colors = 10*['r.', 'g.', 'b.', 'c.', 'k.', 'y.', 'm.']
for i in range(len(X)):
    plt.plot(X[i][0], X[i][1], colors[labels[i]], markersize = 10)
plt.scatter(cluster_centers[:,0], cluster_centers[:,1],
            marker = "x", color = 'k', s = 150, linewidths = 5, zorder = 10)
plt.show()
```



Klasterlash samaradorligini o'lchash

Haqiqiy dunyo ma'lumotlari tabiiy ravishda bir nechta alohida klasterlarga ajratilmagan. Shu sababli, ularni tasavvur qilish va xulosa chiqarish oson emas. Shuning uchun biz klasterlash samaradorligini va uning sifatini o'lchashimiz kerak. Buni siluet tahlili yordamida amalga oshirish mumkin.

Siluet tahlili

Ushbu usul yordamida klasterlar orasidagi masofani o'lchash orqali klasterlash sifatini tekshirish mumkin. Asosan, u siluet ballini taqdim etish orqali klasterlar soni kabi parametrlarni baholash usulini taqdim etadi. Bu ko'rsatkich bitta klasterdagi har bir nuqta qo'shni klasterlardagi nuqtalarga qanchalik yaqinligini o'lchaydigan ko'rsatkichdir.

Siluetni baholash tahlili

Ballar qator qiymatlarga ega $[-1, 1]$. Quyida ushbu baholashning tahlili -

- **+1 ball** - +1 ga yaqin ball namunaning qo'shni klasterdan uzoqda ekanligini ko'rsatadi.
- **0 ball** - 0 ball namunaning ikkita qo'shni klaster o'rtasidagi qaror chegarasida yoki juda yaqin ekanligini ko'rsatadi.
- **Bal -1** - Salbiy ball namunalar noto'g'ri klasterlarga tayinlanganligini ko'rsatadi.

Siluet ballini hisoblash

Ushbu bo'limda biz *silhouette* ballini qanday hisoblashni o'rganamiz. Siluet reytingini quyidagi formula yordamida hisoblash mumkin:

$$silhouettescore = \frac{(p - q)}{\max(p, q)}$$

Bu yerda p - eng yaqin klasterdagi ma'lumotlar nuqtasi qismi bo'lmagan nuqtalargacha bo'lgan o'rtacha masofa. Va q - o'z klasteridagi barcha nuqtalargacha bo'lgan o'rtacha klaster ichidagi masofa.

Klasterlarning optimal sonini topish uchun **sklearn** paketidan **metrics** modulini import qilish orqali klasterlash algoritmini qayta ishga tushirishimiz kerak

. Quydagi misolda biz klasterlarning optimal sonini topish uchun K-means klasterlash algoritmini ishga tushiramiz. Rasmda ko‘rsatilganidek, kerakli paketlarni import qiling:

```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
from sklearn.cluster import KMeans
```

Quydagi kod yordamida biz **sklearn.dataset** paketidagi **make_blob** yordamida to‘rtta blobdan iborat ikki o‘lchovli ma’lumotlar to‘plamini yaratamiz.

```
from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples = 500, centers = 4, cluster_std = 0.40, random_state = 0)
```

O‘zgaruvchilarni rasmda ko‘rsatilganidek ishga tushiring -

```
scores = []
values = np.arange(2, 10)
```

Biz K-means modelini barcha qiymatlar bo‘yicha takrorlashimiz va uni kiritilgan ma’lumotlarga o‘rgatishimiz kerak.

```
for num_clusters in values:
    kmeans = KMeans(init = 'k-means++', n_clusters = num_clusters, n_init = 10)
    kmeans.fit(X)
```

Endi Evklid masofa metrikasi yordamida joriy klasterlash modeli uchun siluet ballini baholang -

```
score = metrics.silhouette_score(X, kmeans.labels_,
metric = 'euclidean', sample_size = len(X))
```

Quydagi kod qatori klasterlar sonini hamda siluet reytingini ko‘rsatishga yordam beradi.

```
print("\nNumber of clusters =", num_clusters)
```

```
print("Silhouette score =", score)
scores.append(score))
```

Siz quyidagi natijaga erishasiz:

```
Number of clusters = 9
Silhouette score = 0.340391138371
num_clusters = np.argmax(scores) + values[0]
print('\nOptimal number of clusters =', num_clusters)
```

Endi klasterlarning optimal soni uchun natija quyidagicha bo'ladi:

```
Optimal number of clusters = 2
```

Eng yaqin qo'shnilar qidirish

Agar biz filmni tavsiya qilish tizimi kabi tizimlarini yaratmoqchi bo'lsak, u holda biz eng yaqin qo'shni qidirish tushunchasini organishimiz kerak. Buning sababi, tavsiyanoma tizimi eng yaqin qo'shnilar tushunchasidan foydalanadi.

Eng yaqin qo'shni qidirish tushunchasini ma'lum ma'lumotlar to'plamidan kirish nuqtasiga eng yaqin nuqtani topish jarayoni sifatida aniqlash mumkin. Ushbu KNN (K-Nearest Neighbors) algoritmining asosiy qo'llanilishi kirish ma'lumotlar nuqtasining turli sinflarga yaqinligi asosida ma'lumotlar nuqtasini tasniflaydigan tasniflash tizimlarini qurishdir.

Quyidagi Python kodi berilgan ma'lumotlar to'plamining **K** eng yaqin qo'shnilarini topishga yordam beradi. Quyida ko'rsatilgandek kerakli paketlarni import qiling. Bu erda biz **sklearn** paketidagi **NearestNeighbors** modulidan foydalanamiz.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors
```

Hozir Keling aniqlaymiz kiritish ma'lumotlar.

```
A = np.array([[3.1, 2.3], [2.3, 4.2], [3.9, 3.5], [3.7, 6.4], [4.8, 1.9],
```

```
[8.3, 3.1], [5.2, 7.5], [4.8, 4.7], [3.5, 5.1], [4.4, 2.9],)]
```

Endi biz eng yaqin qo'shnilarni aniqlashimiz kerak:

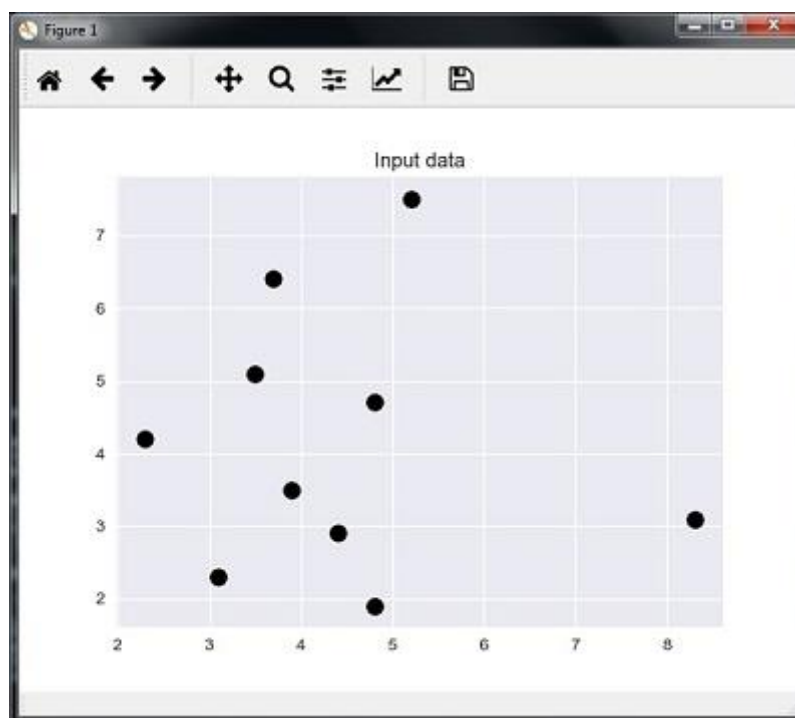
```
k = 3
```

Shuningdek, biz eng yaqin qo'shnilarni topish uchun ishlatilishi mumkin bo'lgan test ma'lumotlarini taqdim etishimiz kerak:

```
test_data = [3.3, 2.9]
```

Quyidagi kod biz aniqlagan kirish ma'lumotlarini tasavvur qilish va chizish imkonini beradi -

```
plt.figure()
plt.title('Input data')
plt.scatter(A[:,0], A[:,1], marker = 'o', s = 100, color = 'black')
```



K eng yaqin qo'shnilar (**KNN**) algoritmini yaratish va o'rgatish uchun biz quyidagi bosqichlarni bajarishimiz kerak.

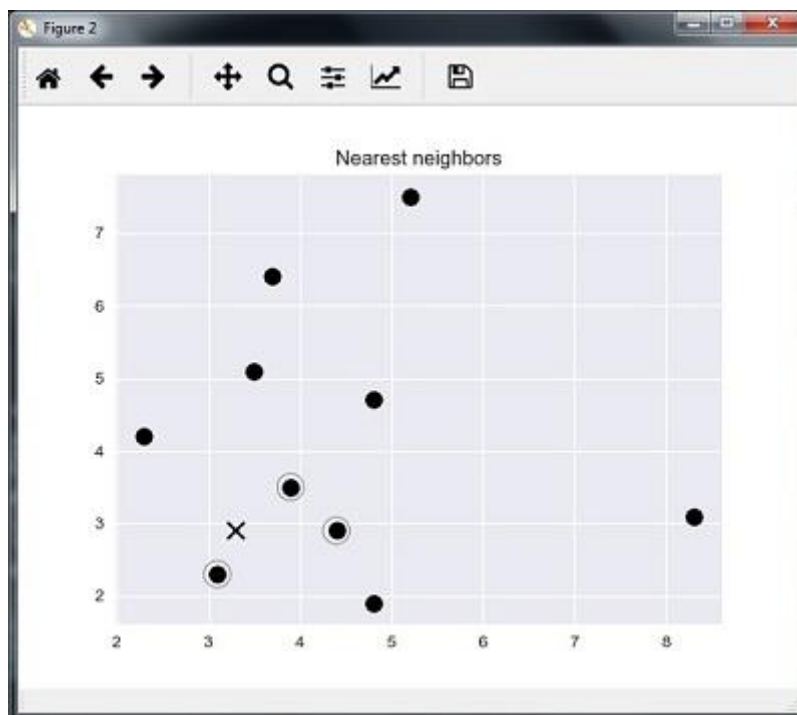
```
knn_model = NearestNeighbors(n_neighbors = k, algorithm = 'auto').fit(X)
distances, indices = knn_model.kneighbors([test_data])
```

K eng yaqin qo'shnilarni chiqarish uchun Python'da quyidagi kodni yozishingiz mumkin.

```
print("\nK Nearest Neighbors:")
for rank, index in enumerate(indices[0][:k], start = 1):
    print(str(rank) + " is", A[index])
```

Sinov ma'lumotlari nuqtasi bilan birga eng yaqin qo'shnilarni vizualizatsiya qilishimiz mumkin

```
plt.figure()
plt.title('Nearest neighbors')
plt.scatter(A[:, 0], X[:, 1], marker = 'o', s = 100, color = 'k')
plt.scatter(A[indices][0][:, 0], A[indices][0][:, 1],
            marker = 'o', s = 250, color = 'k', facecolors = 'none')
plt.scatter(test_data[0], test_data[1],
            marker = 'x', s = 100, color = 'k')
plt.show()
```



Xulosa

K eng yaqin qo'shnilar

1 is [3.1 2.3]

2 is [3.9 3.5]

3 is [4.4 2.9]

K-eng yaqin qo'shni tasniflagichi

Yaqin qo'shnilar (K-Nearest Neighbors - KNN) klassifikatori — bu ma'lumotlar nuqtasini klassifikatsiya qilishda eng yaqin qo'shnilarni aniqlash asosida ishlaydigan modeldir. Biz avvalgi bo'limda KNN algoritmini amalga oshirdik, endi esa shu algoritmdan foydalangan holda KNN klassifikatorini yaratmoqchimiz.

KNN tasniflagichi tushunchasi

K-eng yaqin qo'shni klassifikatsiyasining asosiy konsepsiyasi yangi tanlangan namunani klassifikatsiya qilish uchun eng yaqin bo'lgan oldindan belgilangan sonli, ya'ni "k" — o'rganish to'plamlarining masofalarini topishdan iborat. Yangi namunalar o'zlarining teglarini aynan shu qo'shnilardan oladi. KNN klassifikatorlari foydalanuvchi tomonidan belgilangan qo'shnilar sonini aniqlash uchun doimiy bir konstanta bilan ishlaydi. Masofa bo'yicha eng keng tarqalgan tanlov standart evklid masofasidir. KNN klassifikatori o'rganilgan namunalar bilan bevosita ishlaydi, ya'ni o'rganish uchun qoidalar yaratmaydi. KNN algoritmi barcha mashinani o'rganish algoritmlaridan eng oddiylaridan biri hisoblanadi va ko'plab klassifikatsiya va regressiya vazifalarida, masalan, belgilarni tanib olish yoki tasvirlarni tahlil qilishda ancha muvaffaqiyatli bo'lgan.

Misol

Raqamlarni aniqlash uchun KNN klassifikatorini yaratamiz. Buning uchun biz MNIST ma'lumotlar to'plamidan foydalanamiz. Biz ushbu kodni Jupyter daftariga yozamiz.

Quyida ko'rsatilgandek kerakli paketlarni import qiling.

Bu erda biz **sklearn.neighbors** paketidagi **KNeighborsClassifier** modulidan foydalanamiz:

```
from sklearn.datasets import *  
import pandas as pd
```

```
%matplotlib inline
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import numpy as np
```

Quyidagi kod qaysi rasmni sinab ko‘rishimiz kerakligini tekshirish uchun raqamli tasvirni ko‘rsatadi.

```
def Image_display(i):
    plt.imshow(digit['images'][i], cmap = 'Greys_r')
    plt.show()
```

Endi biz MNIST ma’lumotlar to‘plamini yuklashimiz kerak. Hammasi bo‘lib 1797 ta rasm mavjud, ammo biz dastlabki 1600 ta rasmdan o‘quv to‘plami sifatida foydalanamiz va qolgan 197 tasi sinov uchun saqlanadi.

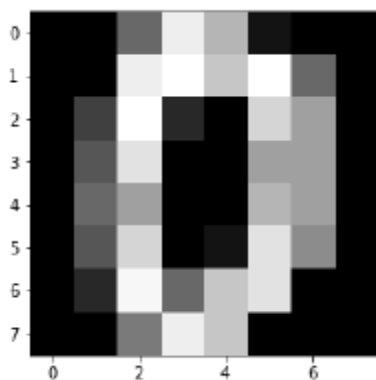
```
digit = load_digits()
digit_d = pd.DataFrame(digit['data'][0:1600])
```

Endi tasvirlarni ko‘rsatayotganda biz natijani quyidagicha ko‘rishimiz mumkin:

```
Image_display(0)
```

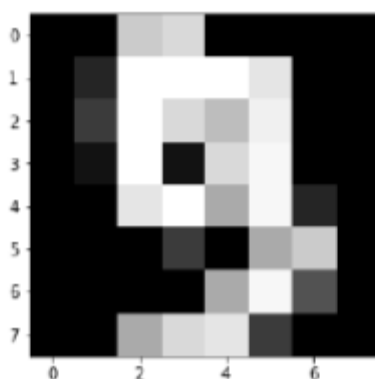
Display rasmlar Image_display(0)

Tasvir 0 quyidagicha ko‘rsatiladi -



Rasmlarni ko‘rsatish (9)

9-rasm quyidagicha ko'rsatiladi.



digit.keys()

Endi biz o'quv va sinov ma'lumotlar to'plamini yaratishimiz va test ma'lumotlar to'plamini KNN tasniflagichlariga taqdim etishimiz kerak.

```
train_x = digit['data'][:1600]
train_y = digit['target'][:1600]
KNN = KNeighborsClassifier(20)
KNN.fit(train_x, train_y)
```

Quyidagi chiqish K eng yaqin qo'shni tasniflagich konstruktorini hosil qiladi.

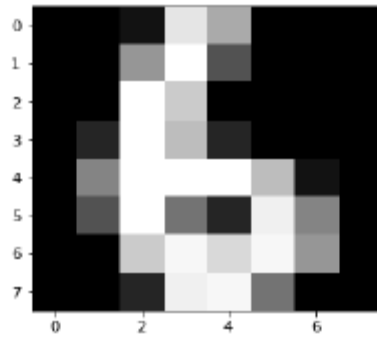
```
KNeighborsClassifier(algorithm = 'auto', leaf_size = 30, metric = 'minkowski',
metric_params = None, n_jobs = 1, n_neighbors = 20, p = 2,
weights = 'uniform')
```

Biz o'quv to'plami bo'lgan 1600 dan katta ixtiyoriy raqamni taqdim etish orqali test to'plamini yaratishimiz kerak.

```
test = np.array(digit['data'][1725])
test1 = test.reshape(1,-1)
Image_display(1725)
```

Rasmni ko'rsatish (6)

6-rasm quyidagicha ko'rsatiladi.



Endi biz test ma'lumotlarini quyidagicha bashorat qilamiz:

```
KNN.predict(test1)
```

Yuqoridagi kod quyidagi chiqishni hosil qiladi:

```
array([6])
```

Endi quyidagilarni ko'rib chiqing:

```
digit['target_names']
```

Yuqoridagi kod quyidagi chiqishni hosil qiladi:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

O'z-o'zini nazorat qilish uchun savollar

1. Klasterlash nima?
2. Ma'lumotlarni Klasterlash Algoritmлари?
3. O'rtacha siljish algoritmi?
4. Klasterlash samaradorligini o'lchash Eng yaqin qo'shnilarini qidirish?
5. KNN tasniflagichi tushunchasi?

2.7-§. TABIIY TILNI QAYTA ISHLASH

Tabiiy tilni qayta ishlash (Natural Language Processing - NLP) sun'iy intellekt usuliga kiradi va u aqlli tizimlar bilan tabiiy til, masalan, ingliz tilida muloqot qilish uchun mo'ljallangan.

Tabiiy tilni qayta ishlash, masalan, robotdan o'z buyruqlaringizni bajarishini yoki dialogga asoslangan klinik ekspert tizimidan yechim eshitishni xohlaganingizda zarur bo'ladi. NLP sohasiga, odamlar ishlatadigan tabiiy tillar yordamida foydali vazifalarni bajarish uchun kompyuterlarni yaratish kiradi. NLP tizimining kirish va chiqish ma'lumotlari quyidagilar bo'lishi mumkin:

- Nutq
- Yozma matn

NLP komponentlari

Ushbu bo'limda biz NLP ning turli komponentlari haqida ma'lumot olishamiz. NLP ikki asosiy komponentdan iborat. Komponentlar quyidagi tarzda tavsiflanadi:

Tabiiy tilni tushunish (Natural Language Understanding - NLU)

Bu quyidagi vazifalarni o'z ichiga oladi:

- Tabiiy tilda berilgan kirish ma'lumotlarini foydali taqdimotlarga aylantirish.
- Tilning turli jihatlarini tahlil qilish.

Tabiiy tilni yaratish (NLG)

Bu jarayon ichki taqdimotdan tabiiy til shaklida mazmunli frazalar va jummalarni yaratishdan iborat. U quyidagi vazifalarni o'z ichiga oladi:

- **Matnni rejalashtirish** — bu bilimlar bazasidan tegishli kontentni ajratib olishni o'z ichiga oladi.
- **Jumla rejalashtirish** — bu kerakli so'zlarni tanlash, ma'noli frazalarni yaratish va jumlaning ohangini sozlashni o'z ichiga oladi.
- **Matnni amalga oshirish** — bu jumla rejasini jumla tuzilishiga aylantirishni anglatadi.

NLU ni qiyinchiliklar

NLU shakl va tuzilma jihatidan juda boy, ammo u noaniqliklarga ega. Noaniqliklarning turli darajalari bo'lishi mumkin:

- **Leksik noaniqlik**

Bu juda oddiy darajada, so'zlar darajasida paydo bo'ladi. Masalan, "doska" so'zini ot yoki fe'l sifatida qabul qilish?

- **Sintaksisdagi noaniqlik**

Jumla turlicha tahlil qilinishi mumkin. Masalan, "U qizil shapkalik qurpani ko'tarib oldi" — u shapka yordamida qurpani ko'tarib oldimi yoki u qurpani qizil shapka bilan ko'tarib oldimi?

- **Ko'rsatkich noaniqligi**

Biror narsaga joylovchi zamonlar orqali murojaat qilish. Masalan: "Rima Gauriga bordi. U: 'Men charchadim', dedi." — Kim charchadi?

NLP terminologiyasi

Endi NLP terminologiyasida ba'zi muhim atamalarni ko'rib chiqamiz:

- **Fonologiya** — bu tovushlarning tizimli tashkilotini o'rganish.
- **Morfologiya** — bu so'zlarni oddiy ma'noli birliklardan qurishni o'rganish.
- **Morfema** — bu tilda ma'no ifodalovchi eng oddiy birlik.
- **Sintaksis** — bu so'zlarni jumla tuzish uchun tartibga solish. Shuningdek, bu jumladagi va frazalardagi so'zlarning tuzuvchilar rolini aniqlashni o'z ichiga oladi.
- **Semantika** — bu so'zlarning ma'nosiga va so'zlarni ma'noli frazalar va jumlar sifatida birlashtirishga taalluqlidir.
- **Pragmatika** — bu jumlarni turli holatlarda ishlatish va tushunishga, shuningdek, bu qanday qilib jumlaning talqiniga ta'sir qilishiga oid bo'ladi.

- **Diskurs** — bu bir jumlaning keyingi jumlani qanday ta'sir qilishi mumkinligini ko'rib chiqadi.
- **Dunyo haqidagi bilimlar** — bu dunyo haqidagi umumiy bilimlarni o'z ichiga oladi.

NLP bosqichlari

Ushbu bo'limda NLP ning turli bosqichlari ko'rsatilgan.

- **Leksik tahlil**

Bu so'zlarning tuzilishini aniqlash va tahlil qilishni o'z ichiga oladi. Tildagi leksikon so'zlar va iboralarning umumiy to'plamidir. Leksik tahlil matnning barcha bo'laklarini — abzaslar, jumlar va so'zlarga ajratishni o'z ichiga oladi.

- **Sintaksis tahlili**

Bu jumladagi so'zlarni grammatik jihatdan tahlil qilish va so'zlarni o'zaro bog'lashni ko'rsatish uchun ularni tartibga solishni o'z ichiga oladi. "Maktab o'g'il bola tomon o'tmoqda" kabi jumla ingliz tilidagi sintaksis analizatori tomonidan noto'g'ri deb baholanadi.

- **Semantik tahlil**

Bu matndan aniq ma'no yoki lug'at ma'nosini chiqarib olishni anglatadi. Matn mazmunlilik uchun tekshiriladi. Bu sintaksis strukturalari va vazifa sohasi ob'ektlarini solishtirish orqali amalga oshiriladi. Semantik analizator "issiqlikdagi muzqaymoq" kabi jummalarni inkor etadi.

- **Diskurs bilan integratsiya**

Har qanday jumlaning ma'nosi undan oldingi jumlaning ma'nosiga bog'liq. Bundan tashqari, bu jumladan keyingi jumlaning ma'nosini ham belgilaydi.

- **Pragmatik tahlil**

Bu bosqichda aytilgan narsa haqiqatan nima deganini qayta ko'rib chiqish amalga oshiriladi. Bu, shuningdek, haqiqiy dunyo haqidagi bilimlarni talab qiladigan til jihatlarini aniqlashni o'z ichiga oladi.

NLTK paketi

NLTK (Natural Language Toolkit) — bu tabiiy tilni qayta ishlash (NLP) uchun Python dasturlash tilida yaratilgan keng qoʻllaniladigan kutubxonadir. U matnni tahlil qilish, leksik tahlil, sintaksis va semantik tahlil, shuningdek, koʻplab NLP vazifalarini bajarish uchun turli vositalar va algoritmlar taqdim etadi.

NLTK yordamida quyidagi amallarni bajarish mumkin:

- Matnni leksik tahlil qilish, jummalarni va soʻzlarni ajratish.
- Sintaksis tahlili, grammatik strukturalarni aniqlash.
- Matndan maʼno chiqarish va semantik tahlil qilish.
- Tabiiy tilni qayta ishlashda ishlatiladigan boshqa usullar, masalan, soʻzlarni aniqlash va maʼno chiqarish.

NLTK paketi oʻz ichiga quyidagi funksiyalarni oladi:

- **Tokenization:** Matnni soʻzlar yoki iboralarga ajratish.
- **POS Tagging:** Har bir soʻzning grammatik funksiyasini aniqlash.
- **Parsing:** Sintaksis tahlili, yaʼni jumlarlar tuzilishini tekshirish.
- **Stemming:** Soʻz ildizlarini topish.
- **Lemmatization:** Soʻzning leksik asosini aniqlash.
- **WordNet:** Leksik maʼlumotlar bazasi, soʻzlarning sinonimlari, antonimlari va boshqa lingvistik xususiyatlarini oʻrganish.

NLTK ning asosiy afzalliklari:

- Kengaytirilgan vositalar toʻplami.
- Oʻrganish uchun qulay interfeys.
- Turli xil NLP vazifalarini bajarishda yordam beradi.

NLTK-ni import qilish

Uni ishlatishdan oldin NLTK-ni oʻrnatishimiz kerak. Uni quyidagi buyruq yordamida oʻrnatish mumkin:

```
pip install nltk
```

NLTK uchun konda paketini yaratish uchun quyidagi buyruqdan foydalaning -

```
conda install -c anaconda nltk
```

Endi NLTK paketini o‘rnatganimizdan so‘ng uni python buyruq qatori orqali import qilishimiz kerak. Biz uni Python buyruq satriga quyidagi buyruqni yozish orqali import qilishimiz mumkin:

```
>>> import nltk
```

NLTK ma’lumotlari yuklash

Endi, NLTK-ni import qilgandan so‘ng, biz kerakli ma’lumotlarni yuklashimiz kerak. Buni Python buyruq satrida quyidagi buyruq yordamida amalga oshirish mumkin:

```
>>> nltk.download()
```

Boshqa kerakli paketlarni o‘rnatish

NLTK yordamida tabiiy tillarni qayta ishlash dasturlarini yaratish uchun biz kerakli paketlarni o‘rnatishimiz kerak. Paketlar quyidagicha.

```
gensim
```

Bu ko‘plab ilovalar uchun foydali bo‘lgan mustahkam semantik modellashtirish kutubxonasi. Biz uni quyidagi buyruqni bajarish orqali o‘rnatishimiz mumkin.

```
pip install gensim
```

Namuna

Gensim paketini to‘g‘ri ishlashi uchun ishlatiladi . Biz uni quyidagi buyruqni ishga tushirish orqali o‘rnatishimiz mumkin.

```
pip install pattern
```

Tokenizatsiya, stemming va lemmatizatsiya konsepsiyalari

Ushbu bo‘limda biz tokenizatsiya, stemming va lemmatizatsiyaning nima ekanligini ko‘rib chiqamiz.

Tokenizatsiya

Uni berilgan matnni, ya’ni belgilar ketma-ketligini tokenlar deb ataladigan kichikroq birliklarga ajratish jarayoni sifatida aniqlash mumkin. Tokenlar so‘zlar, raqamlar yoki tinish belgilari bo‘lishi mumkin. Bu so‘z segmentatsiyasi deb ham ataladi. Quyida tokenizatsiyaning oddiy misoli:

Kirish - Mango, banan, ananas va olma hammasi mevalardir.

Xulosa

Mango	Banana	Pineapple	and	Apple	all	are	Fruits

Berilgan matnni parchalash jarayoni so‘z chegaralarini aniqlash orqali amalga oshirilishi mumkin. So‘zning oxiri va yangi so‘zning boshlanishi so‘z chegaralari deyiladi. So‘zlarning imlo tizimi va tipografik tuzilishi chegaralarga ta’sir qiladi.

Python NLTK modulida bizda turli xil tokenizatsiya bilan bog‘liq paketlar mavjud bo‘lib, ulardan matnni talablarimizga muvofiq tokenlarga bo‘lish uchun foydalanishimiz mumkin. Ba’zi paketlar quyidagicha.

sent_tokenize paketi

Nomidan ko‘rinib turibdiki, ushbu paket kiritilgan matnni jummalarga ajratadi. Biz ushbu paketni quyidagi Python kodidan foydalanib import qilishimiz mumkin:

```
from nltk.tokenize import sent_tokenize
```

word_tokenize paketi

Ushbu paket kiritilgan matnni so‘zlarga ajratadi. Biz ushbu paketni quyidagi Python kodidan foydalanib import qilishimiz mumkin:

```
from nltk.tokenize import word_tokenize
```

WordPunctTokenizer pakate

Ushbu paket kiritilgan matnni soʻzlarga, shuningdek, tinish belgilariga ajratadi. Biz ushbu paketni quyidagi Python kodidan foydalanib import qilishimiz mumkin -

```
from nltk.tokenize import WordPunctTokenizer
```

Stemming

Soʻzlar bilan ishlashda biz grammatik sabablarga koʻra koʻplab oʻzgarishlarga duch kelamiz. Bu erda oʻzgaruvchanlik tushunchasi biz bir xil soʻzlarning turli shakllari bilan shugʻullanishimiz kerakligini anglatadi, masalan, **demokratiya**, **demokratik** va **demokratiklashtirish**. Mashinalar uchun bu turli soʻzlar bir xil asosiy shaklga ega ekanligini tushunish juda muhimdir. Shunday qilib, matnni tahlil qilishda soʻzlarning asosiy shakllarini ajratib olish foydali boʻladi.

Bunga stemming orqali erishishimiz mumkin. Demak, biz ayta olamizki, oʻzak soʻzlarning uchlarini kesish orqali soʻzlarning asosiy shakllarini ajratib olishning evristik jarayonidir.

Python NLTK modulida bizda poydevor bilan bogʻliq turli xil paketlar mavjud. Ushbu paketlardan asosiy soʻz shakllarini olish uchun foydalanish mumkin. Ushbu paketlar algoritmlardan foydalanadi. Baʼzi paketlar quyidagicha -

PorterStemmer paketi

Ushbu Python paketi asosiy shaklni chiqarish uchun Porter algoritmidan foydalanadi. Biz ushbu paketni quyidagi Python kodidan foydalanib import qilishimiz mumkin:

```
from nltk.stem.porter import PorterStemmer
```

“**writing**” soʻzini kiritsak, oʻzakdan keyin “**write**” soʻzini olamiz.

Lancaster Stemmer paketi

Ushbu Python toʻplami asosiy shaklni chiqarish uchun Lancaster algoritmidan foydalanadi. Biz ushbu paketni quyidagi Python kodidan foydalanib import qilishimiz mumkin:

```
from nltk.stem.lancaster import LancasterStemmer
```

“writing” soʻzini kiritsak , oʻzakdan keyin **“write”** soʻzini olamiz .

SnowballStemmer paketi

Ushbu Python toʻplami asosiy shaklni olish uchun qor toʻpi algoritmidan foydalanadi. Biz ushbu paketni quyidagi Python kodidan foydalanib import qilishimiz mumkin:

```
from nltk.stem.snowball import SnowballStemmer
```

“writing” soʻzini kiritsak , oʻzakdan keyin **“write”** soʻzini olamiz.

Bu algoritmlarning barchasi turli darajadagi qatʼiylikka ega. Agar biz ushbu uchta algoritmni solishtirsak, Porter stemmers eng qatʼiy emas va Lancaster eng qatʼiydir. Snowball stemmer ham tezlik, ham qatʼiylilik nuqtai nazaridan foydalanish uchun qulay.

Lemmatizatsiya

Lemmatizatsiya yordamida soʻzlarning asosiy shaklini ham ajratib olishimiz mumkin. Asosan, bu vazifa soʻzlarning lugʻat va morfologik tahlili yordamida, odatda, faqat flektiv sonlarni olib tashlash maqsadida amalga oshiriladi. Har qanday soʻzning bu asosiy shakli lemma deb ataladi.

Stemming va lemmatizatsiya oʻrtasidagi asosiy farq soʻzlarning lugʻat va morfologik tahlilidan foydalanishdir. Yana bir farqi shundaki, oʻzak yasashda soʻz yasovchi bogʻlangan soʻzlar koʻpincha yoʻq qilinadi, lemmatizatsiya paytida esa, odatda, lemmaning har xil flektiv shakllari yoʻq qilinadi. Misol uchun, agar kirish soʻzi sifatida saw soʻzini kiritsak, u holda "s" soʻzini qaytarishi mumkin, lekin lemmatizatsiya belgisi feʼl yoki ot ekanligiga qarab koʻrish yoki koʻrish soʻzini qaytarishga harakat qiladi.

Python NLTK modulida biz soʻzning asosiy shakllarini olish uchun foydalanishimiz mumkin boʻlgan lemmatizatsiya jarayoni bilan bogʻliq quyidagi paketga egamiz -

WordNetLemmatizer paketi

Ot yoki fe'l sifatida ishlatilishiga qarab uning asosiy shaklini chiqaradi . Biz ushbu paketni quyidagi Python kodidan foydalanib import qilishimiz mumkin.

```
from nltk.stem import WordNetLemmatizer
```

Chunking: ma'lumotlarni bo'laklarga bo'lish

Bu tabiiy tilni qayta ishlashning muhim jarayonlaridan biridir. Bog'lanishning asosiy maqsadi nutq qismlarini va ot so'z birikmalari kabi qisqa iboralarni aniqlashdir. Biz allaqachon tokenizatsiya jarayonini, tokenlarni yaratishni o'rganib chiqdik. Chiqarish asosan ushbu tokenlarni belgilashni o'z ichiga oladi. Boshqacha qilib aytganda, bo'linish bizga jumlaning tuzilishini ko'rsatadi.

Keyingi bo'limda biz turli xil bo'linish turlari haqida bilib olamiz.

Parchalanish turlari

Bog'lanishning ikki turi mavjud.

Bo'laklarga bo'lish

Bu parchalanish jarayonida ob'ekt, narsalar va boshqalar. Ular umumiy bo'lib, til mavhum bo'ladi. Tasdiqlash uchun ko'proq imkoniyatlar. Bu jarayonda biz kichraytiramiz. Misol uchun, agar biz savolni qismlarga ajratsak: "avtomobillarning maqsadlari nimadan iborat"? Biz "transport" javobini olishimiz mumkin. Ushbu bloklarga bo'linish jarayonida ob'ekt, narsalar va boshqalar. Ular aniqroq bo'lib, til aniqroq bo'ladi. Bloklarga bo'linishda chuqurroq struktura hisobga olinadi. Bu jarayonda biz miqyosni kengaytiramiz. Misol uchun, agar biz "Mashina haqida menga aniq aytib bering" degan savolni qisqartirsak? Biz mashina haqida kichikroq ma'lumotlarni olamiz.

Misol

Ushbu misolda biz Python-da NLTK moduli yordamida gapdagi ot fragmentlarini topishga imkon beruvchi bo'laklash turkumi bo'lgan ot iborasini bajaramiz.

Ot so'z birikmasini amalga oshirish uchun pythonda ushbu bosqichlarni bajaring – chunking.

1-qadam - Ushbu bosqichda biz qismlarga ajratish grammatikasini aniqlashimiz kerak. Bu biz rioya qilishimiz kerak bo'lgan qoidalardan iborat bo'ladi.

2-qadam - Ushbu bosqichda biz parcha tahlilchisini yaratishimiz kerak. U grammatikani tahlil qiladi va natija beradi.

3-qadam - Ushbu oxirgi bosqichda chiqish daraxt shaklida yaratiladi.

Kerakli NLTK paketini quyidagi tarzda import qilaylik.

```
Import nltk
```

Endi biz jumlani aniqlashimiz kerak. Bu yerda DT aniqlovchi, VBP fe'l, JJ sifat, IN bosh gap, NN esa ot ma'nosini bildiradi.

```
sentence=[("a","DT"),("clever","JJ"),("fox","NN"),("was","VBP"),  
          ("jumping","VBP"),("over","IN"),("the","DT"),("wall","NN")]
```

Endi grammatikani berishimiz kerak. Bu yerda grammatikani muntazam ifoda shaklida beramiz.

```
grammar = "NP:{ <DT>?<JJ>*<NN> }"
```

Grammatikani tahlil qiladigan tahlilchini aniqlashimiz kerak.

```
parser_chunking = nltk.RegexpParser(grammar)
```

Tahlilchi gapni quyidagicha tahlil qiladi.

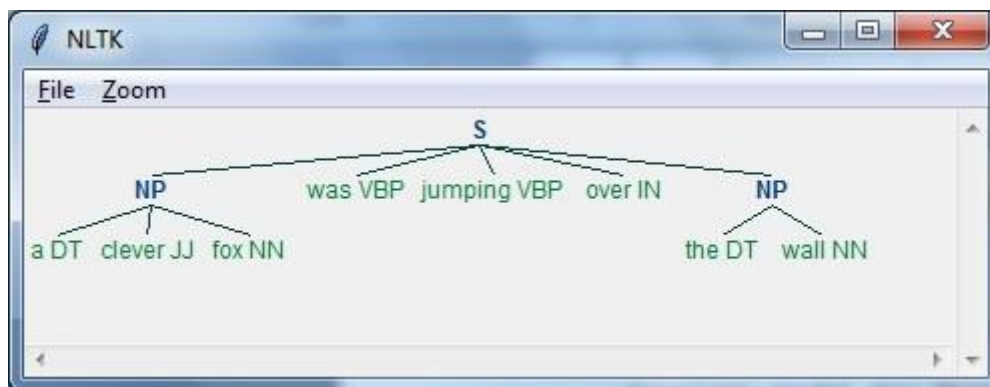
```
parser_chunking.parse(sentence)
```

Keyinchalik biz chiqishni olishimiz kerak. Chiqish **output_chunk** deb nomlangan oddiy o'zgaruvchida hosil bo'ladi.

```
Output_chunk = parser_chunking.parse(sentence)
```

Quyidagi kodni bajarganimizdan so'ng, biz mahsulotimizni daraxt sifatida ko'rsatishimiz mumkin.

```
output.draw()
```



“So‘zlar sumkasi” (BoW) modeli

Tabiiy tilni qayta ishlash modeli bo‘lgan Word sumkasi (BoW) asosan matndan xususiyatlarni ajratib olish uchun ishlatiladi, shunda matn modellashtirishda, masalan, mashinani o‘rganish algoritmlarida foydalanish mumkin.

Endi nima uchun biz matndan xususiyatlarni ajratib olishimiz kerak degan savol tug‘iladi. Buning sababi shundaki, mashinani o‘rganish algoritmlari xom ma’lumotlar bilan ishlay olmaydi va ular undan mazmunli ma’lumot olishlari uchun raqamli ma’lumotlarga muhtoj. Matn ma’lumotlarini raqamli ma’lumotlarga aylantirish xususiyatni ajratib olish yoki xususiyatni kodlash deb ataladi.

Qanday ishlaydi

Bu matndan xususiyatlarni ajratib olishning juda oddiy usuli. Aytaylik, bizda matnli hujjat bor va biz uni raqamli ma’lumotlarga aylantirmoqchimiz yoki undan xususiyatlarni chiqarmoqchimiz deylik, keyin birinchi navbatda ushbu model hujjatdagi barcha so‘zlardan lug‘at chiqaradi. Keyin, hujjatning matritsa atamasidan foydalanib, model quriladi. Shunday qilib, BoW hujjatni faqat so‘zlar to‘plami sifatida ifodalaydi. Hujjatdagi so‘zlarning tartibi yoki tuzilishi haqidagi har qanday ma’lumot o‘chiriladi.

Hujjat atamasi matritsasi tushunchasi

BoW algoritmi hujjat atamasi matritsasi yordamida model tuzadi. Nomidan ko‘rinib turibdiki, hujjat atamasi matritsasi - bu hujjatda uchraydigan turli xil so‘zlarning matritsasi. Ushbu matritsadan foydalanib, matnli hujjat turli so‘zlarning vaznli birikmasi sifatida ifodalanishi mumkin. Eshik chegarasini o‘rnatish va muhimroq so‘zlarni tanlash orqali biz hujjatlardagi barcha so‘zlarning

gistogrammasini yaratishimiz mumkin, undan xususiyat vektori sifatida foydalanish mumkin. Quyida matritsa - hujjat atamasi tushunchasini tushunish uchun misol keltirilgan

Misol

Aytaylik, bizda quyidagi ikkita jumla bor -

- **Taklif 1** - Biz soʻzlar sumkasidan foydalanamiz.
- **Taklif 2** - Xususiyatlarni ajratib olish uchun soʻzlar sumkasi modeli qoʻllaniladi.

Endi, ushbu ikki jumalani koʻrib chiqsak, bizda quyidagi 13 xil soʻz bor -

- Biz - we
- bor - are
- foydalanish - using
- da - the
- sumka - bag
- lar - of
- Soʻzlar - words
- Model - model
- Bu – is
- Ishlatilgan- used
- uchun - for
- Chiqazib olish - extracting
- Xususiyatlari - features

Endi biz har bir jumladagi soʻzlar sonidan foydalangan holda har bir jumla uchun gistogramma tuzishimiz kerak -

- | |
|---|
| <ul style="list-style-type: none">• 1-taklif - [1,1,1,1,1,1,1,0,0,0,0,0]• 2-taklif - [0,0,0,1,1,1,1,1,1,1,1,1] |
|---|

Shunday qilib, biz xususiyat vektorlarini ajratib oldik. Har bir xususiyat vektori 13 oʻlchovli, chunki bizda 13 ta alohida soʻz bor.

Statistika tushunchasi

Statistika tushunchasi atamaFrequency-teskari hujjat chastotasi (tf-idf) deb ataladi. Hujjatdagi har bir soʻz muhim ahamiyatga ega. Statistik maʼlumotlar har bir soʻzning ahamiyatini tushunishga yordam beradi.

Soʻslar chastotasi (tf)

Bu hujjatdagi har bir soʻzning paydo boʻlish chastotasining oʻlchovidir. Uni har bir soʻz sonini maʼlum hujjatdagi soʻzlarning umumiy soniga boʻlish yoʻli bilan olish mumkin.

Hujjatlarning teskari chastotasi (idf)

Bu maʼlum hujjatlar toʻplamidagi maʼlum bir hujjat uchun soʻz qanchalik noyob ekanligini koʻrsatadigan oʻlchovdir. Idf ni hisoblash va xususiyat vektorini shakllantirish uchun biz tez-tez uchraydigan soʻzlarning ogʻirligini kamaytirishimiz kerak, masalan, kamdan-kam uchraydigan soʻzlarni ogʻirligi.

NLTKda soʻzlar modelini yaratish

Ushbu boʻlimda biz ushbu jumladan vektorlarni yaratish uchun CountVectorizer yordamida qatorlar toʻplamini aniqlaymiz. Keling import zarur paket:

```
from sklearn.feature_extraction.text import CountVectorizer
```

Hozir aniqlash toʻplam takliflar

```
Sentences = ['We are using the Bag of Word model', 'Bag of Word model is  
used for extracting the features.']  
  
vectorizer_count = CountVectorizer()  
  
features_text = vectorizer.fit_transform(Sentences).todense()  
  
print(vectorizer.vocabulary_)
```

Yuqoridagi dastur quyida koʻrsatilgandek chiqishni ishlab chiqaradi. Bu yuqoridagi ikkita jumlada 13 xil soʻz borligini koʻrsatadi -

```
{'we': 11, 'are': 0, 'using': 10, 'the': 8, 'bag': 1, 'of': 7,  
'word': 12, 'model': 6, 'is': 5, 'used': 9, 'for': 4, 'extracting': 2, 'features': 3}
```

Bular mashinani o'rganish uchun ishlatilishi mumkin bo'lgan xususiyat vektorlari (matndan raqamli shaklgacha).

Muammoni hal qilish

Ushbu bo'limda biz bir nechta tegishli muammolarni hal qilamiz.

Kategoriyalarni bashorat qilish

Hujjatlar to'plamida nafaqat so'zlar, balki so'zlar toifasi ham muhimdir; u yoki bu so'z matnning qaysi turkumiga kiradi? Misol uchun, biz berilgan jumla elektron pochta, yangiliklar, sport, kompyuter va boshqalar sifatida tasniflanadimi yoki yo'qligini taxmin qilmoqchimiz. Quyidagi misolda biz tf-idf dan hujjat toifasini qidirish uchun xususiyat vektorini shakllantirish uchun foydalanamiz. Biz 20 ta sklearn newsgroup ma'lumotlar to'plamidagi ma'lumotlardan foydalanamiz.

Biz kerakli paketlarni import qilishimiz kerak.

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
```

Kategoriya xaritasini aniqlang. Biz besh xil toifadan foydalanamiz: din, avtomobillar, sport, elektronika va kosmik.

```
category_map = {'talk.religion.misc': 'Religion', 'rec.autos': 'Autos',
                'rec.sport.hockey': 'Hockey', 'sci.electronics': 'Electronics', 'sci.space': 'Space'}
```

Yaratilish tarbiyaviy o'rnatish:

```
training_data = fetch_20newsgroups(subset = 'train',
                                   categories = category_map.keys(), shuffle = True, random_state = 5)
```

Hisoblash vektorizatorini yarating va *counts* atamasini ajratib oling:

```
vectorizer_count = CountVectorizer()
train_tc = vectorizer_count.fit_transform(training_data.data)
print("\nDimensions of training data:", train_tc.shape)
```

tf-idf transformatori quyidagicha yaratilgan:

```
tfidf = TfidfTransformer()
train_tfidf = tfidf.fit_transform(train_tc)
```

Endi test ma'lumotlarini aniqlaymiz:

```
input_data = [
    'Discovery was a space shuttle',
    'Hindu, Christian, Sikh all are religions',
    'We must have to drive safely',
    'Puck is a disk made of rubber',
    'Television, Microwave, Refrigerated all uses electricity' ]
```

Yuqoridagi ma'lumotlar bizga multinomial Naive Bayes klassifikatorini o'rgatishda yordam beradi:

```
classifier = MultinomialNB().fit(train_tfidf, training_data.target)
```

Hisoblash - vektorizator yordamida kiritilgan ma'lumotlarni o'zgartiring

```
input_tc = vectorizer_count.transform(input_data)
```

Endi biz vektorlashtirilgan ma'lumotlarni tfidf transformator - yordamida o'zgartiramiz:

```
input_tfidf = tfidf.transform(input_tc)
```

Biz bashorat qilaylik toifalar chiqish:

```
predictions = classifier.predict(input_tfidf)
```

Chiqish quyidagicha hosil bo'ladi.

```
for sent, category in zip(input_data, predictions):
    print('\nInput Data:', sent, '\n Category:', \
        category_map[training_data.target_names[category]])
```

Turkum bashoratchisi quyidagi natijani hosil qiladi.

Dimensions of training data: (2755, 39297)

Input Data: Discovery was a space shuttle

Category: Space

Input Data: Hindu, Christian, Sikh all are religions

Category: Religion

Input Data: We must have to drive safely

Category: Autos

Input Data: Puck is a disk made of rubber

Category: Hockey

Input Data: Television, Microwave, Refrigerated all uses electricity

Category: Electronics

Jinsni qidirish

Ushbu muammoni o'rnatishda ismlarni berish orqali jinsni (erkak yoki ayol) aniqlash uchun tasniflagich o'qitiladi. Xususiyat vektorini qurish va tasniflagichni o'rgatish uchun biz evristikadan foydalanishimiz kerak. Biz scikit-learn paketidagi etiketli ma'lumotlardan foydalanamiz. Quyida jinsni aniqlash tizimini yaratish uchun Python kodi. Kerakli paketlarni import qilaylik:

```
import random

from nltk import NaiveBayesClassifier
from nltk.classify import accuracy as nltk_accuracy
from nltk.corpus import names
```

Endi biz kiritilgan so'zdan oxirgi N harfni ajratib olishimiz kerak. Bu harflar funksiya vazifasini bajaradi:

```
def extract_features(word, N = 2):
    last_n_letters = word[-N:]
    return {'feature': last_n_letters.lower()}

if __name__ == '__main__':
```


NLTK-da mavjud bo'lgan yorliqli ismlar (erkak va ayol) yordamida trening ma'lumotlarini yarating:

```
male_list = [(name, 'male') for name in names.words('male.txt')]
female_list = [(name, 'female') for name in names.words('female.txt')]
data = (male_list + female_list)

random.seed(5)
random.shuffle(data)
```

Endi test ma'lumotlari quyidagicha yaratiladi -

```
namesInput = ['Rajesh', 'Gaurav', 'Swati', 'Shubha']
```

Quyidagi kod yordamida o'qitish va sinov uchun ishlatiladigan namunalar sonini aniqlang.

```
train_sample = int(0.8 * len(data))
```

Endi biz aniqlikni solishtirishimiz uchun turli uzunliklarda takrorlashimiz kerak:

```
for i in range(1, 6):
    print('\nNumber of end letters:', i)
    features = [(extract_features(n, i), gender) for (n, gender) in data]
    train_data, test_data = features[:train_sample],
    features[train_sample:]
    classifier = NaiveBayesClassifier.train(train_data)
```

Klassifikatorning aniqligini quyidagicha hisoblash mumkin:

```
accuracy_classifier = round(100 * nltk_accuracy(classifier, test_data), 2)
print('Accuracy = ' + str(accuracy_classifier) + '%')
```

Endi biz natijani taxmin qilishimiz mumkin:

```
for name in namesInput:
    print(name, '==>', classifier.classify(extract_features(name, i)))
```

Yuqoridagi dastur quyidagi natijani yaratadi:

Number of end letters: 1 Accuracy = 74.7% Rajesh -> female Gaurav -> male Swati -> female Shubha -> female
Number of end letters: 2 Accuracy = 78.79% Rajesh -> male Gaurav -> male Swati -> female Shubha -> female
Number of end letters: 3 Accuracy = 77.22% Rajesh -> male Gaurav -> female Swati -> female Shubha -> female
Number of end letters: 4 Accuracy = 69.98% Rajesh -> female Gaurav -> female Swati -> female Shubha -> female
Number of end letters: 5 Accuracy = 64.63% Rajesh -> female Gaurav -> female Swati -> female Shubha -> female

Yuqoridagi chiqishda biz yakuniy harflarning maksimal sonidagi aniqlik ikkita ekanligini va yakuniy harflar soni ortishi bilan kamayib borishini ko‘rishimiz mumkin.

Mavzuni modellashtirish: Matn ma’lumotlaridagi qonuniylikni aniqlash

Biz bilamizki, hujjatlar odatda mavzular bo‘yicha guruhlanadi. Ba’zan biz matndagi ma’lum bir mavzuga mos keladigan naqshlarni aniqlashimiz kerak. Buni amalga oshirish texnikasi mavzuni modellashtirish deb ataladi. Boshqacha qilib

aytganda, mavzuni modellashtirish mavhum mavzularni yoki ma'lum hujjatlar to'plamidagi yashirin tuzilmani aniqlash usulidir.

Mavzuni modellashtirish texnikasidan quyidagi stsenariylarda foydalanishimiz mumkin.

Matnni tasnifi

Mavzuni modellashtirish tasniflashni yaxshilashi mumkin, chunki u har bir so'zni alohida xususiyat sifatida ishlatmasdan, o'xshash so'zlarni birlashtiradi.

Tavsiya tizimlari

Mavzuni modellashtirish yordamida biz o'xshashlik ko'rsatkichlari yordamida tavsiya tizimlarini yaratishimiz mumkin.

Mavzularni modellashtirish algoritmlari

Mavzuni modellashtirish algoritmlar yordamida amalga oshirilishi mumkin. Algoritmlar quyidagicha -

Yashirin Dirixlet taqsimoti (Latent Dirichlet Allocation - LDA)

Ushbu algoritmi mavzuni modellashtirish uchun eng mashhur hisoblanadi. Mavzuni modellashtirishni amalga oshirish uchun ehtimollik grafik modellaridan foydalanadi. LDA algoritmini ishlatish uchun Python-da gensim paketini import qilishimiz kerak.

Yashirin semantik tahlil (LDA) yoki yashirin semantik indeks (Latent Semantic Indexin - LSI)

Bu algoritmi chiziqli algebraga asoslangan. U asosan hujjat atamasi matritsasi uchun SVD (Singular Value Decomposition) tushunchasidan foydalanadi.

Manfiy bo'lmagan matritsa faktorizatsiyasi (Non-Negative Matrix Factorization - NMF)

Shuningdek, u chiziqli algebraga asoslanadi. Mavzuni modellashtirish uchun yuqoridagi barcha algoritmlarda parametr sifatida **mavzular soni**, kirish sifatida **Hujjat-Word matritsasi** va chiqish sifatida **WTM (Word Topic Matrix)** va **TDM (Topic Document Matrix)** bo'ladi.

O‘z-o‘zini nazorat qilish uchun savollar

1. Tabiiy tilni tushunish (Natural Language Understanding - NLU)?
2. NLP terminologiyasi?
3. NLP bosqichlari?
4. NLTK-ni import qilish?
5. Tokenizatsiya, stemming va lemmatizatsiya konsepsiyalari?
6. Lemmatizatsiya?
7. “So‘zlar sumkasi” (BoW) modeli?
8. Hujjat atamasi matritsasi tushunchasi?
9. Statistika tushunchasi?
10. Kategoriyalarni bashorat qilish?

2.8-§. VAQTLAR QATORI MA'LUMOTLARI TAHLIL QILISH

Berilgan kirish ketma-ketligidagi keyingi elementni bashorat qilish mashinani o'rganishdagi yana bir muhim tushunchadir. Ushbu bobda vaqt qatorlari ma'lumotlarini tahlil qilish haqida batafsil tushuntirish berilgan.

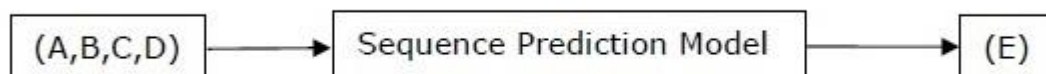
Vaqt qatorining ma'lumotlari ma'lum vaqt oralig'ida topilgan ma'lumotlarni anglatadi. Agar biz mashinani o'rganishda ketma-ketlikni bashorat qilishni istasak, unda biz ketma-ket ma'lumotlar va vaqt bilan shug'ullanishimiz kerak. Qatro ma'lumotlari ketma-ket ma'lumotlar uchun izohdir. Ma'lumotlarni tartiblash ketma-ket ma'lumotlarning muhim xususiyatidir.

Ketma-ketlik yoki vaqt qatori tahlilining asosiy tushunchasi

Ketma-ketlik yoki vaqtli ketma-ketlik tahlili ilgari kuzatilgan ma'lumotlarga asoslangan holda berilgan kirish ketma-ketligidagi keyingisini bashorat qilish uchun mo'ljallangan. Bashorat keyin sodir bo'lishi mumkin bo'lgan har qanday narsaga taalluqli bo'lishi mumkin: belgi, raqam, keyingi kun uchun ob-havo, nutqdagi keyingi atama va boshqalar. Qimmatli qog'ozlar bozori tahlili, ob-havo prognozi va mahsulot tavsiyalari kabi ilovalarda ketma-ketlikni tahlil qilish juda foydali bo'lishi mumkin.

Misol

Ketma-ketlikni bashorat qilishni tushunish uchun quyidagi misolni ko'rib chiqaylik. Bu erda **A, B, C, D qiymatlari berilgan va siz** ketma-ketlikni bashorat qilish modeli yordamida **E** qiymatini taxmin qilishingiz kerak .



Foydali paketlarni o'rnatish

Python yordamida vaqt qatorlari ma'lumotlarini tahlil qilish uchun biz quyidagi paketlarni o'rnatishimiz kerak.

Pandas

Pandas ochiq manbali, BSD litsenziyasiga ega kutubxona bo'lib, Python uchun yuqori unumdorlik, foydalanish uchun qulay ma'lumotlar tuzilmalari va

ma'lumotlarni tahlil qilish vositalarini taqdim etadi. Pandas-ni quyidagi buyruq yordamida o'rnatishingiz mumkin.

```
pip install pandas
```

Conda paket menejeri yordamida o'rnatmoqchi bo'lsangiz , quyidagi buyruqdan foydalanishingiz mumkin.

```
conda install -c anaconda pandas
```

hmmlearn

Bu Python-da yashirin Markov Modellarini (HMM) o'rganish uchun oddiy algoritmlar va modellardan iborat ochiq manba, BSD litsenziyalangan kutubxona. Siz uni quyidagi buyruq yordamida o'rnatishingiz mumkin.

```
pip install hmmlearn
```

Conda paket menejeri yordamida o'rnatmoqchi bo'lsangiz , quyidagi buyruqdan foydalanishingiz mumkin.

```
conda install -c omnia hmmlearn
```

PyStruct

Bu tuzilgan o'rganish va bashorat qilish kutubxonasi. PyStruct-da qo'llaniladigan o'rganish algoritmlari shartli tasodifiy maydonlar (CRF), maksimal chegara Markov tasodifiy tarmoqlari (M3N) yoki Strukturaviy qo'llab-quvvatlash vektor mashinalari kabi nomlar bilan ishlaydi. Siz uni quyidagi buyruq yordamida o'rnatishingiz mumkin.

```
pip install pystruct
```

CVXOPT

Python dasturlash tiliga asoslangan konveks optimallashtirish uchun ishlatiladi. Bu, shuningdek, bepul dasturiy ta'minot to'plami. Siz uni quyidagi buyruq yordamida o'rnatishingiz mumkin.

```
pip install cvxopt
```

Conda paket menejeri yordamida o'rnatmoqchi bo'lsangiz , quyidagi buyruqdan foydalanishingiz mumkin.

```
conda install -c anaconda cvdoxt
```

Pandas: Vaqt qatori ma'lumotlaridan statistik ma'lumotlarni qayta ishlash, kesish va olish

Vaqt qatorlari ma'lumotlari bilan ishlashingiz kerak bo'lsa, Pandas juda foydali vositadir. Pandas yordamida siz quyidagilarni qilishingiz mumkin:

- **pd.date_range** paketidan foydalanib sana oralig'ini yarating
- **pd.series** paketidan foydalanib, pandas sanalar bo'yicha indekslang
- **ts.resample** paketi yordamida qayta namuna olish
- chastotani o'zgartirish

Misol

Quyidagi misolda Pandas yordamida vaqt qatorlari ma'lumotlarini qayta ishlash va kesish ko'rsatilgan. Esda tutingki, biz bu yerda har oylik Arktika tebranishi ma'lumotlaridan foydalanmoqdamiz, ularni *monthly.ao.index.b50.current.ascii* saytidan yuklab olish va foydalanishimiz uchun matn formatiga aylantirish mumkin.

Vaqt qatori ma'lumotlarini qayta ishlash

Vaqt qatorlari ma'lumotlarini qayta ishlash uchun siz quyidagi amallarni bajarishingiz kerak bo'ladi. Birinchi qadam quyidagi paketlarni import qilishni o'z ichiga oladi.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Keyin quyidagi kodda ko'rsatilganidek, kirish faylidan ma'lumotlarni o'qiy oladigan funktsiyani belgilang.

```
def read_data(input_file):
    input_data = np.loadtxt(input_file, delimiter = None)
```

Endi ushbu ma'lumotlarni vaqt qatoriga aylantiring. Buning uchun vaqt seriyamiz uchun sana oralig'ini yarating. Ushbu misolda biz ma'lumotlar chastotasi sifatida bir oy saqlaymiz. Bizning faylimiz 1950 yil yanvariga oid ma'lumotlarni o'z ichiga oladi.

```
dates = pd.date_range('1950-01', periods = input_data.shape[0], freq = 'M')
```

Ushbu bosqichda biz quyida ko'rsatilgandek Pandas Series yordamida vaqt seriyasi ma'lumotlarini yaratamiz.

```
output = pd.Series(input_data[:, index], index = dates)
return output

if __name__ == '__main__':
```

Kirish fayli yo'lini bu yerda ko'rsatilganidek kiriting.

```
input_file = "/Users/admin/AO.txt"
```

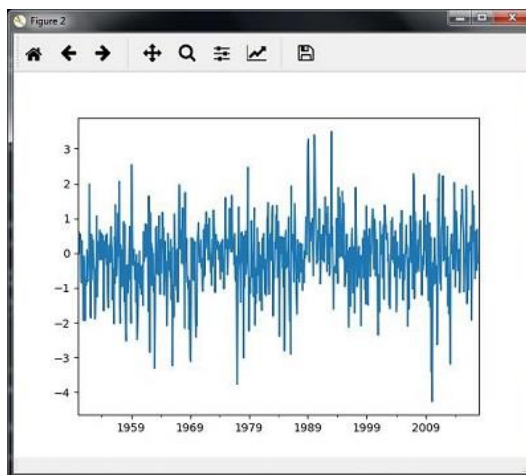
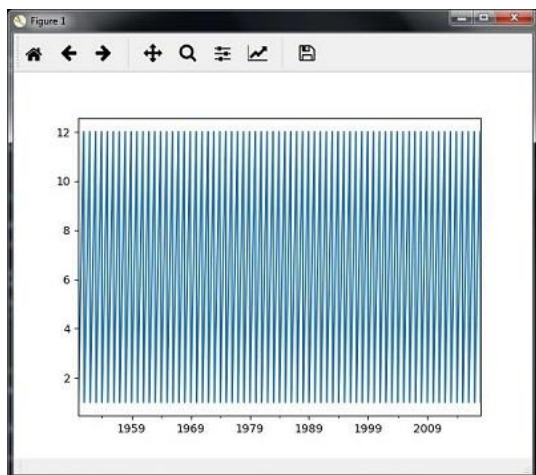
Endi bu yerda ko'rsatilganidek, ustunni vaqt seriyasi formatiga aylantiring.

```
timeseries = read_data(input_file)
```

Nihoyat, ko'rsatilgan buyruqlar yordamida ma'lumotlarni chizing va vizualizatsiya qiling.

```
plt.figure()
timeseries.plot()
plt.show()
```

Quyidagi rasmlarda ko'rsatilgandek grafiklarni ko'rasiz.

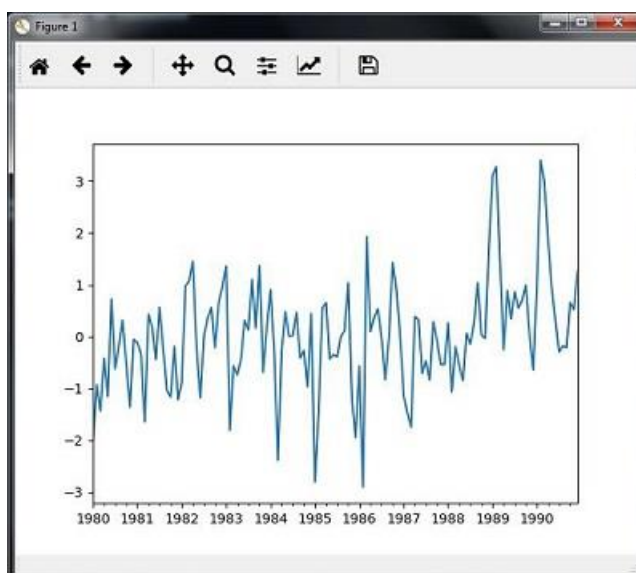


Vaqt qatori ma'lumotlarini kesish

Kesish vaqt qatori ma'lumotlarining faqat bir qismini ajratib olishni o'z ichiga oladi. Misolning bir qismi sifatida biz faqat 1980 yildan 1990 yilgacha bo'lgan ma'lumotlarni qisqartirdik. Iltimos, ushbu vazifani bajaradigan quyidagi kodga e'tibor bering.

```
timeseries['1980':'1990'].plot()  
<matplotlib.axes._subplots.AxesSubplot at 0xa0e4b00>  
  
plt.show()
```

Vaqt qatori ma'lumotlarini kesish uchun kodni ishga tushirganingizda, bu erda rasmda ko'rsatilganidek, quyidagi grafikni kuzatishingiz mumkin.



Vaqt qatorlari ma'lumotlaridan statistik ma'lumotlarni olish

Ba'zi muhim xulosalar chiqarishingiz kerak bo'lgan hollarda, berilgan ma'lumotlardan ba'zi statistik ma'lumotlarni chiqarib olishingiz kerak bo'ladi. O'rtacha, dispersiya, korrelyatsiya, maksimal va minimal bu statistik ma'lumotlardan ba'zilari. Berilgan vaqt qatori ma'lumotlaridan bunday statistik ma'lumotlarni chiqarmoqchi bo'lsangiz, quyidagi koddan foydalanishingiz mumkin.

Ma'nosi

Bu yerda ko'rsatilgandek o'rtachani topish uchun **mean()** funksiyasidan foydalanishingiz mumkin.

```
timeseries.mean()
```

Keyin muhokama qilingan misolda ko‘radigan natija quyidagicha bo‘ladi -

```
-0,11143128165238671
```

Maksimum

Bu yerda ko‘rsatilgandek maksimalni topish uchun **max()** funksiyasidan foydalanishingiz mumkin.

```
timeseries.max()
```

Keyin muhokama qilingan misolda ko‘radigan natija quyidagicha bo‘ladi.

```
3.4952999999999999
```

Minium

Bu erda ko‘rsatilganidek, **min()** funktsiyasidan minimalni topish uchun foydalanishingiz mumkin.

```
timeseries.min()
```

Keyin muhokama qilingan misolda ko‘radigan natija quyidagicha bo‘ladi.

```
-4,2656999999999998
```

Bir vaqtning o‘zida barchaini olish

Agar siz bir vaqtning o‘zida barcha statistik ma’lumotlarni hisoblamogchi bo‘lsangiz, bu erda ko‘rsatilgandek **describe()** funksiyasidan foydalanishingiz mumkin.

```
timeseries.describe()
```

Keyin muhokama qilingan misolda ko‘radigan natija quyidagicha bo‘ladi.

```
count 817.000000  
mean -0.111431  
std 1.003151  
min -4.265700  
25% -0.649430
```

```
50%    -0.042744
75%     0.475720
max      3.495300
dtype: float64
```

Qayta namuna olish

Siz boshqa chastotada ma'lumotlarni qayta namuna olishingiz mumkin. Qayta namuna olishning ikkita varianti:

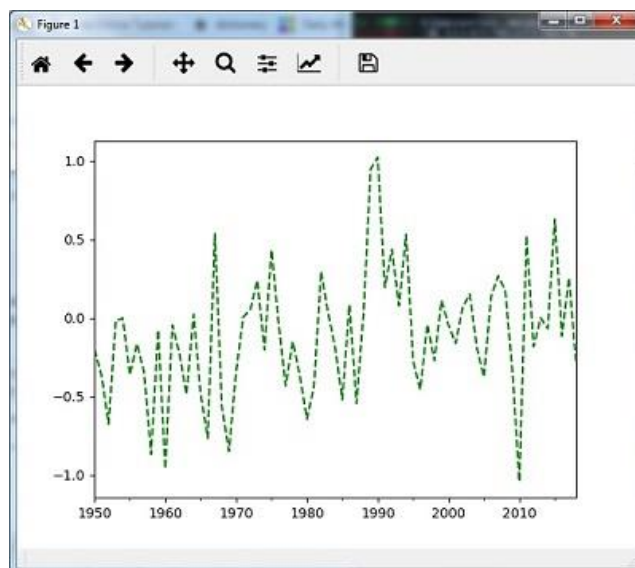
- Vaqt davri
- Usul

mean() bilan qayta namuna olish

mean() usuli yordamida ma'lumotlardan qayta namuna olish uchun quyidagi koddan foydalanishingiz mumkin.

```
timeseries_mm = timeseries.resample("A").mean()
timeseries_mm.plot(style = 'g--')
plt.show()
```

Keyin **mean()** – dan foydalanib, qayta namuna olish natijasi sifatida quyidagi sxemani kuzatishingiz mumkin.

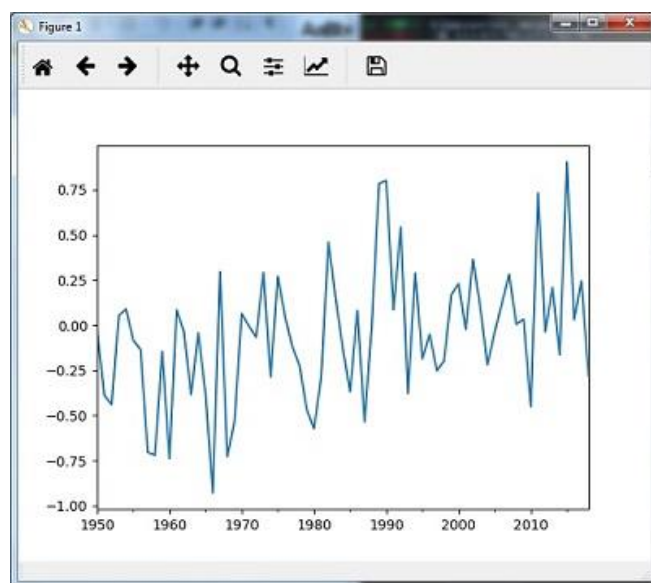


median() bilan qayta namuna olish

median() usuli yordamida ma'lumotlarni qayta namunalash uchun quyidagi koddan foydalanishingiz mumkin.

```
timeseries_mm = timeseries.resample("A").median()
timeseries_mm.plot()
plt.show()
```

Keyin **median()** – yordamida qayta namuna olish natijasida quyidagi syujetni kuzatishingiz mumkin.

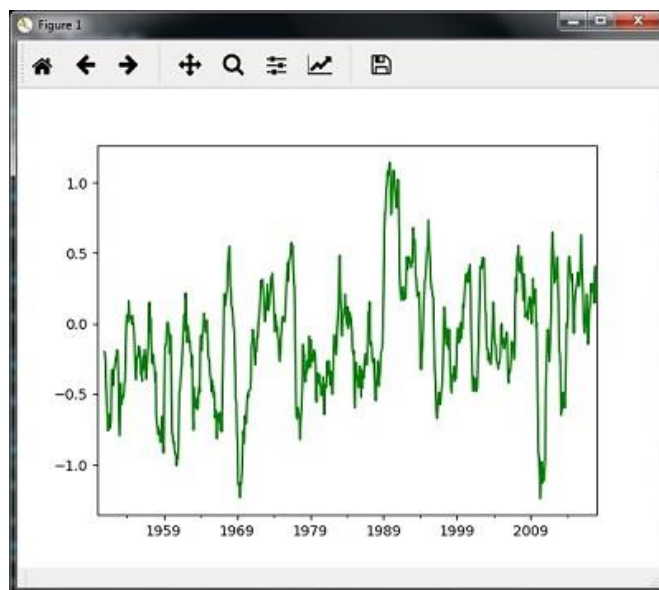


O'tkinchi qiymat

Harakatlanuvchi o'rtachani hisoblash uchun quyidagi koddan foydalanishingiz mumkin.

```
timeseries.rolling(window = 12, center = False).mean().plot(style = '-g')  
plt.show()
```

Keyin harakatlanuvchi o'rtacha - natijasida quyidagi jadvalni kuzatishingiz mumkin.



Yashirin Markov modeli (HMM) yordamida ketma-ket ma'lumotlarni tahlil qilish

HMM doimiy va kengaytiriladigan ma'lumotlarni qayta ishlash uchun keng qo'llaniladigan statistik model bo'lib, vaqt seriyali fond bozori tahlili, ishlash testi va nutqni aniqlash kabi. Ushbu bo'limda Yashirin Markov Modeli (HMM) yordamida ketma-ket ma'lumotlar tahlili batafsil tavsiflanadi.

Yashirin Markov modeli (HM)

HMM - bu Markov zanjiri kontsepsiyasiga asoslangan stoxastik model bo'lib, kelajakdagi statistik ma'lumotlarning ehtimoli undan oldingi har qanday holatga emas, balki faqat jarayonning hozirgi holatiga bog'liq degan taxminga asoslanadi. Misol uchun, tanga otishda biz beshinchi otish natijasi boshlar bo'ladi, deb ayta olmaymiz. Buning sababi shundaki, tanga xotirasi yo'q va keyingi natija oldingi natijaga bog'liq emas.

Matematik jihatdan HMM quyidagi o'zgaruvchilardan iborat:

(S) Xolati

HMMda mavjud bo'lgan yashirin holatlar to'plami . S harfi bilan belgilanadi.

Chiqish belgilar (O)

Bu HMMda mavjud bo'lishi mumkin bo'lgan chiqish belgilari to'plami. U O harfi bilan belgilanadi.

Holatga o'tish ehtimoli matritsasi (A)

Bu bir holatdan boshqa davlatlarning har biriga o'tish ehtimoli. A harfi bilan belgilanadi.

Kuzatishdan tashqari ehtimollik matritsasi (B)

Bu ma'lum bir holatda ramzni chiqarish/kuzatish ehtimoli. Bu B harfi bilan belgilanadi.

Oldingi ehtimollik matritsasi (π)

Bu tizimning turli holatlaridan ma'lum bir holatda boshlash ehtimoli. π bilan belgilanadi.

$\lambda = (S, O, A, B, \pi)$ sifatida aniqlash mumkin .

Qayerda,

• $S = \{s_1, s_2, \dots, s_N\}$ - N ta mumkin bo'lgan holatlar to'plami,

• $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_M\}$ M mumkin bo'lgan kuzatish belgilari to'plamini ifodalaydi,

- $\mathbf{A} - \mathbf{N} \times \mathbf{N}$ holatga o'tish ehtimoli matritsasi (TPM),
- $\mathbf{B} - \mathbf{N} \times \mathbf{M}$ kuzatuv yoki o'zgaruvchan ehtimollik matritsasi (EPM),
- π - boshlang'ich holatning ehtimollik taqsimotining N-o'lchovli vektori.

Misol: Qimmatli qog'ozlar bozori ma'lumotlarini tahlil qilish

Ushbu misolda biz HMM ning ketma-ket yoki vaqt seriyali ma'lumotlari bilan qanday ishlashi haqida tasavvurga ega bo'lish uchun birja ma'lumotlarini bosqichma-bosqich tahlil qilamiz. E'tibor bering, biz ushbu misolni Python-da amalga oshirmoqdamiz.

Quyida ko'rsatilgandek kerakli paketlarni import qiling.

```
import datetime
import warnings
```

Bu yerda ko'rsatilganidek, **matplotlib.finance** paketidagi fond bozori ma'lumotlaridan foydalaning.

```
import numpy as np
from matplotlib import cm, pyplot as plt
from matplotlib.dates import YearLocator, MonthLocator
try:
    from matplotlib.finance import quotes_historical_yahoo_och1
except ImportError:
    from matplotlib.finance import (
        quotes_historical_yahoo as quotes_historical_yahoo_och1)
from hmmlearn.hmm import GaussianHMM
```

Ma'lumotlarni boshlanish va tugash sanasidan, ya'ni bu erda ko'rsatilgandek ikkita aniq sana o'rtasida yuklang.

```
start_date = datetime.date(1995, 10, 10)
end_date = datetime.date(2015, 4, 25)
quotes = quotes_historical_yahoo_och1('INTC', start_date, end_date)
```

Ushbu bosqichda biz har kuni yakuniy tirnoqlarni chiqaramiz. Buning uchun quyidagi buyruqdan foydalaning.

```
closing_quotes = np.array([quote[2] for quote in quotes])
```

Endi biz har kuni sotilgan aksiyalar hajmini chiqaramiz. Buning uchun quyidagi buyruqdan foydalaning.

```
volumes = np.array([quote[5] for quote in quotes])[1:]
```

Bu erda quyida berilgan koddan foydalanib, aksiyalarning yopilish narxidan foiz farqini oling.

```
diff_percentages = 100.0 * np.diff(closing_quotes) / closing_quotes[:-1]
dates = np.array([quote[0] for quote in quotes], dtype = np.int)[1:]
training_data = np.column_stack([diff_percentages, volumes])
```

Ushbu bosqichda Gauss HMM ni yarating va o'rgating. Buning uchun quyidagi koddan foydalaning.

```
hmm = GaussianHMM(n_components = 7, covariance_type = 'diag', n_iter = 1000)
with warnings.catch_warnings():
    warnings.simplefilter('ignore')
    hmm.fit(training_data)
```

Endi ko'rsatilgan buyruqlar yordamida HMM modeli yordamida ma'lumotlarni yarating.

```
num_samples = 300
samples, _ = hmm.sample(num_samples)
```

Nihoyat, ushbu bosqichda biz grafikni tuzamiz va grafik chiqishi sifatida sotilgan aksiyalarning farqi va hajmini ingl. Farq foizlarini chizish va tasavvur qilish uchun quyidagi koddan foydalaning.

```
plt.figure()
plt.title('Difference percentages')
plt.plot(np.arange(num_samples), samples[:, 0], c = 'black')
```

Savdo qilingan aksiyalar hajmini rejalashtirish va tasavvur qilish uchun quyidagi koddan foydalaning.

```
plt.figure()
plt.title('Volume of shares')
plt.plot(np.arange(num_samples), samples[:, 1], c = 'black')
plt.ylim(ymin = 0)
plt.show()
```

O‘z-o‘zini nazorat qilish uchun savollar

1. Ketma-ketlik yoki vaqt qatori tahlilining asosiy tushunchasi?
2. Vaqt qatori ma'lumotlarini qayta ishlash?
3. Vaqt qatori ma'lumotlarini kesish?
4. Qayta namuna olish?
5. Yashirin Markov modeli (HMM)?

2.9-§. NUTQNI TANISH (SPEECH RECOGNITION)

Ushbu bobda biz Python yordamida SI nutqni aniqlash haqida bilib olamiz. Nutq kattalar uchun eng asosiy muloqot vositasidir. Nutqni qayta ishlashning asosiy maqsadi inson va mashinaning o‘zaro ta’sirini ta’minlashdir.

Nutqni qayta ishlash tizimi asosan uchta vazifani bajaradi:

- **Birinchidan** , nutqni aniqlash, bu mashinaga biz aytgan so‘zlar, iboralar va jummalarni olish imkonini beradi.
- **Ikkinchidan** , tabiiy tilni qayta ishlash, mashinaga nima haqida gapirayotganimizni tushunishga imkon beradi va
- **Uchinchidan** , nutq sintezi, mashinaga gapirishga imkon beradi.

Nutqni aniqlashga, odamlar tomonidan aytilgan so‘zlarni tushunish jarayoniga qaratilgan . Esda tutingki, nutq signallari mikrofon tomonidan qabul qilinadi va keyin tizim tomonidan tushunilishi kerak.

Nutqni tan tizimini yaratish

Nutqni aniqlash yoki avtomatik nutqni aniqlash (Automatic Speech Recognition - ASR) robototexnika kabi sun’iy intellekt loyihalarining diqqat markazidir. ASRsiz kognitiv robotni odam bilan o‘zaro aloqada bo‘lishini tasavvur qilib bo‘lmaydi. Biroq, nutqni tan oluvchini yaratish unchalik oson emas.

Nutqni aniqlash tizimini ishlab chiqishdagi qiyinchiliklar

Yuqori sifatli nutqni aniqlash tizimini ishlab chiqish juda qiyin vazifadir. Nutqni aniqlash texnologiyasining murakkabligi quyida tavsiflanganidek, bir qator parametrlar bilan keng tavsiflanishi mumkin:

- **Lug‘at hajmi** - Lug‘at hajmi ASR rivojlanishining qulayligiga ta’sir qiladi. Yaxshiroq tushunish uchun quyidagi lug‘at o‘lchamlarini ko‘rib chiqing.
 - Kichik lug‘at 2-100 so‘zdan iborat, masalan, ovoqli menyu tizimidagi kabi
 - O‘rtacha kattalikdagi lug‘at bir necha 100-1000 so‘zdan iborat, masalan, ma’lumotlar bazasini qidirish muammosi

- Katta lugʻat odatdagi diktant vazifasi kabi bir necha 10 000 soʻzdan iborat.

Lugʻat qanchalik katta boʻlsa, tanib olish shunchalik qiyin boʻladi.

- **Kanal xususiyatlari** - Kanal sifati ham muhim parametrdir. Masalan, inson nutqi toʻliq chastota diapazoniga ega boʻlgan yuqori tarmoqli kengligini oʻz ichiga oladi, telefon nutqi esa cheklangan chastota diapazoniga ega past tarmoqli kengligidan iborat. Eʼtibor bering, ikkinchisi ancha murakkab.
- **Gapirish rejimi** - ASRni rivojlantirishning qulayligi, shuningdek, nutq rejimiga, yaʼni nutqning ajratilgan soʻz rejimida yoki bogʻlangan soʻz rejimida yoki doimiy nutq rejimida boʻlishiga bogʻliq. Eʼtibor bering, davomiy nutqni tanib olish qiyinroq.
- **Nutq uslubi** - Oʻqilgan nutq rasmiy yoki oʻz-oʻzidan va boʻshashgan boʻlishi mumkin. Ikkinchisini tanib olish qiyinroq.
- **Spikerga bogʻliqlik** - Nutq maʼruzachiga bogʻliq, maʼruzachiga moslashtirilgan yoki spikerdan mustaqil boʻlishi mumkin. Mustaqil maʼruzachi yaratish - eng qiyini.
- **Shovqin turi** - Shovqin ASRni loyihalashda eʼtiborga olinadigan yana bir omil. Signal va shovqin nisbati fon shovqini kamroq yoki koʻproq boʻlgan akustik muhitga qarab turli diapazonlarda boʻlishi mumkin:
 - Agar signalning shovqin nisbati 30 dB dan katta boʻlsa, u yuqori diapazon deb hisoblanadi
 - Agar signalning shovqin nisbati 30 dB dan 10 dB gacha boʻlsa, u oʻrtacha SNR hisoblanadi.
 - Signalning shovqin nisbati 10 dB dan kam boʻlsa, u past diapazon deb hisoblanadi

Misol uchun, statsionar, insoniy boʻlmagan shovqin, fon nutqi va boshqa karnaylardan oʻzaro bogʻlanish kabi fon shovqinining turi ham vazifaning qiyinligini oshiradi.

- **Mikrofon xususiyati** - Mikrofon sifati yaxshi, oʻrtacha yoki oʻrtachadan past boʻlishi mumkin. Bundan tashqari, ogʻiz va mikrofon orasidagi masofa farq

qilishi mumkin. Nutqni tanib olish tizimlarini ishlab chiqishda ushbu omillarni ham hisobga olish kerak.

Ushbu qiyinchiliklarga qaramay, tadqiqotchilar nutqning turli jihatlari, masalan, nutq signalini tushunish, ma'ruzachi va urg'uni aniqlash bo'yicha keng ko'lamli ishladilar. Nutqni tanuvchi tizimlar yaratish uchun quyida keltirilgan amallarni bajarishingiz kerak bo'ladi:

Ovozli signallarni vizualizatsiya qilish - fayldan o'qish va u bilan ishlash

Bu nutqni aniqlash tizimini yaratishdagi birinchi qadamdir, chunki u audio signalning tuzilishi haqida tushuncha beradi. Quyida audio signallar bilan ishlash uchun bajarishingiz mumkin bo'lgan ba'zi umumiy qadamlar:

Yozib olish - Fayldan audio o'qish kerak bo'lganda, avval uni mikrofon yordamida yozib oling.

Namuna - Mikrofondan yozib olishda signallar raqamlashtirilgan shaklda saqlanadi. Ammo u bilan ishlash uchun mashinaga ularni diskret raqamli shaklda kerak. Shuning uchun biz ma'lum bir chastotada namuna olishimiz va signalni diskret raqamli shaklga aylantirishimiz kerak. Namuna olish uchun yuqori chastotani tanlash odamlar signalni tinglaganlarida uni uzluksiz audio signal sifatida qabul qilishlarini anglatadi.

Misol

Quyidagi misol faylda saqlanadigan Python yordamida audio signalni tahlil qilishning bosqichma-bosqich yondashuvini ko'rsatadi. Ushbu audio signalning chastotasi 44,100 Gts ni tashkil qiladi.

Kerakli pakatlarni bu yerda ko'rsatilganidek import qiling:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
```

Hozir saqlangan audio fayl o'qing. U ikkita qiymatni qaytaradi: namuna tezligi va audio signal. Bu yerda ko'rsatilganidek, saqlangan audio faylga yo'lni ko'rsating:

```
frequency_sampling, audio_signal = wavfile.read("/Users/admin/audio_file.wav")
```

Ko'rsatilgan buyruqlar yordamida audio namuna tezligi, signal ma'lumotlari turi va signal davomiyligi kabi parametrlarni ko'rsatish:

```
print('\nSignal shape:', audio_signal.shape)
print('Signal Datatype:', audio_signal.dtype)
print('Signal duration:', round(audio_signal.shape[0] / float(frequency_sampling), 2), 'seconds')
```

Bu qadam quyida ko'rsatilgandek signalni normallashtirishni o'z ichiga oladi:

```
audio_signal = audio_signal / np.power(2, 15)
```

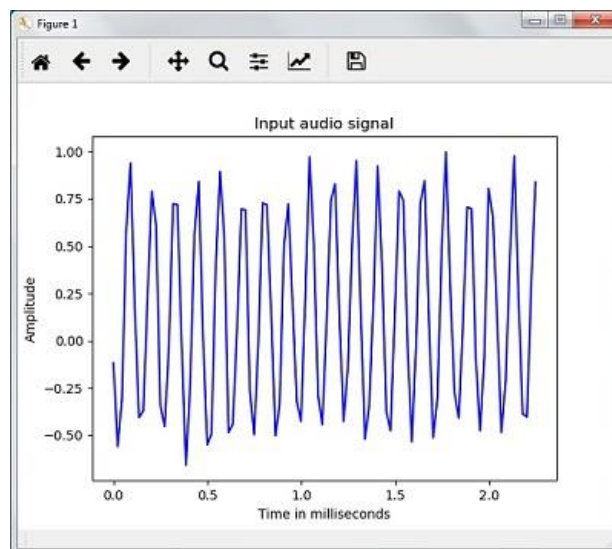
Ushbu bosqichda biz vizualizatsiya uchun ushbu signaldan dastlabki 100 ta qiymatni chiqaramiz. Buning uchun quyidagi buyruqlardan foydalaning:

```
audio_signal = audio_signal[:100]
time_axis = 1000 * np.arange(0, len(signal), 1) / float(frequency_sampling)
```

Endi quyida berilgan buyruqlar yordamida signalni tasavvur qiling:

```
plt.plot(time_axis, signal, color='blue')
plt.xlabel('Time (milliseconds)')
plt.ylabel('Amplitude')
plt.title('Input audio signal')
plt.show()
```

Rasmda ko'rsatilganidek, yuqoridagi audio signal uchun olingan chiqish grafigi va ma'lumotlarni ko'rishingiz mumkin



```
Signal shape: (132300,)
Signal Datatype: int16
Signal duration: 3.0 seconds
```

Ovoz signalining xarakteristikasi: chastota domenini o'zgartirish

Ovozli signalni tavsiflash vaqt domeni signalini chastota domeniga aylantirishni va uning chastota komponentlarini tushunishni o'z ichiga oladi. Bu muhim qadam, chunki u signal haqida juda ko'p ma'lumot beradi. Ushbu transformatsiyani amalga oshirish uchun siz Furiye transformatsiyasi kabi matematik vositadan foydalanishingiz mumkin.

Misol

Quyidagi misolda faylda saqlanadigan Python yordamida signalni qanday xarakterlash bosqichma-bosqich ko'rsatilgan. Shuni esda tutingki, bu erda biz Fourier transformatsiyasining matematik vositasidan uni chastota sohasiga aylantirish uchun foydalanamiz.

Kerakli pakatlarni bu yerda ko'rsatilganidek import qiling:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
```

Hozir o'qing saqlangan audio fayl . U ikkita qiymatni qaytaradi: namuna tezligi va audio signal. Bu erda berilgan buyruqda ko'rsatilganidek, saqlangan audio faylga yo'lni ko'rsating:

```
frequency_sampling, audio_signal = wavfile.read("/Users/admin/sample.wav")
```

Ushbu bosqichda biz quyida berilgan buyruqlar yordamida audio signalni namuna olish tezligi, signal ma'lumotlari turi va uning davomiyligi kabi parametrlarni ko'rsatamiz:

```
print('\nSignal shape:', audio_signal.shape)
print('Signal Datatype:', audio_signal.dtype)
print('Signal duration:', round(audio_signal.shape[0] /
float(frequency_sampling), 2), 'seconds')
```

Ushbu bosqichda biz signalni quyidagi buyruqda ko'rsatilganidek normallashtirishimiz kerak:

```
audio_signal = audio_signal / np.power(2, 15)
```

Ushbu bosqich signalning uzunligi va yarmini ajratib olishni o'z ichiga oladi.

Buning uchun quyidagi buyruqlardan foydalaning:

```
length_signal = len(audio_signal)
half_length = np.ceil((length_signal + 1) / 2.0).astype(np.int)
```

Endi biz chastota domeniga aylantirish uchun matematik vositalarni qo'llashimiz kerak. Bu erda biz Furiye transformatsiyasidan foydalanamiz.

```
signal_frequency = np.fft.fft(audio_signal)
```

Endi chastota domenidagi signalni normallashtiring va uni kvadratga aylantiring:

```
signal_frequency = abs(signal_frequency[0:half_length]) / length_signal
signal_frequency **= 2
```

Keyin chastotaga aylantirilgan signalning uzunligi va yarmini ajratib oling:

```
len_fts = len(signal_frequency)
```

E'tibor bering, Furiye o'zgartirilgan signal juft va toq registrlar uchun ham sozlanishi kerak.

```
if length_signal % 2:
    signal_frequency[1:len_fts] *= 2
else:
    signal_frequency[1:len_fts-1] *= 2
```

Endi quvvatni desiballarda (dB) ajratib oling:

```
signal_power = 10 * np.log10(signal_frequency)
```

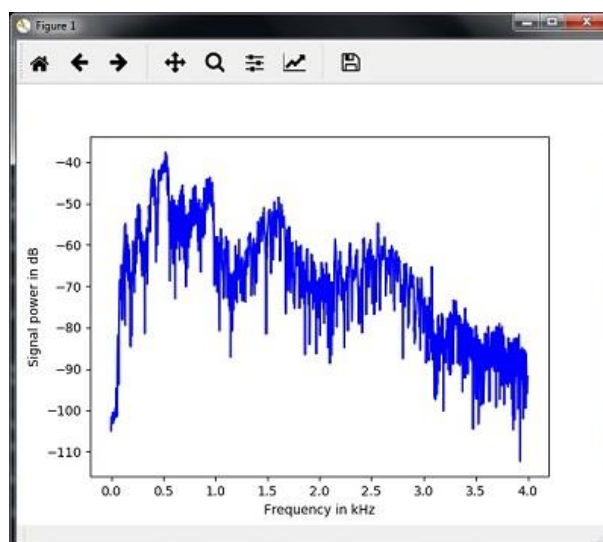
X o'qi uchun chastotani kHz ga sozlang:

```
x_axis = np.arange(0, len_half, 1) * (frequency_sampling / length_signal) / 1000.0
```

Endi signal xarakteristikasini quyidagicha tasavvur qiling:

```
plt.figure()
plt.plot(x_axis, signal_power, color='black')
plt.xlabel('Frequency (kHz)')
plt.ylabel('Signal power (dB)')
plt.show()
```

Quyidagi rasmda ko'rsatilganidek, yuqoridagi kodning chiqish grafigini kuzatishingiz mumkin:



Monoton ovoz signalini yarating

Siz hozirgacha ko'rgan ikkita qadam signallarni o'rganish uchun muhimdir. Endi bu qadam ba'zi bir oldindan belgilangan parametrlar bilan audio signalni yaratmoqchi bo'lsangiz foydali bo'ladi. E'tibor bering, ushbu bosqichda audio signal chiqish faylida saqlanadi.

Misol

Quyidagi misolda biz faylda saqlanadigan Python yordamida monoton signal yaratmoqchimiz. Buning uchun siz quyidagi amallarni bajarishingiz kerak bo'ladi. Rasmda ko'rsatilganidek, kerakli paketlarni import qiling:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import write
```

Chiqish fayli saqlanishi kerak bo'lgan faylni belgilang:

```
output_file = 'audio_signal_generated.wav'
```

Endi rasmda ko'rsatilganidek, siz tanlagan parametrlarni belgilang:

```
duration = 4 # in seconds
frequency_sampling = 44100 # in Hz
frequency_tone = 784
min_val = -4 * np.pi
max_val = 4 * np.pi
```

Ushbu bosqichda biz - rasmda ko'rsatilganidek, audio signalni yaratishimiz mumkin:

```
t = np.linspace(min_val, max_val, duration * frequency_sampling)
audio_signal = np.sin(2 * np.pi * tone_freq * t)
```

Endi audio faylni chiqish faylida saqlang:

```
write(output_file, frequency_sampling, signal_scaled)
```

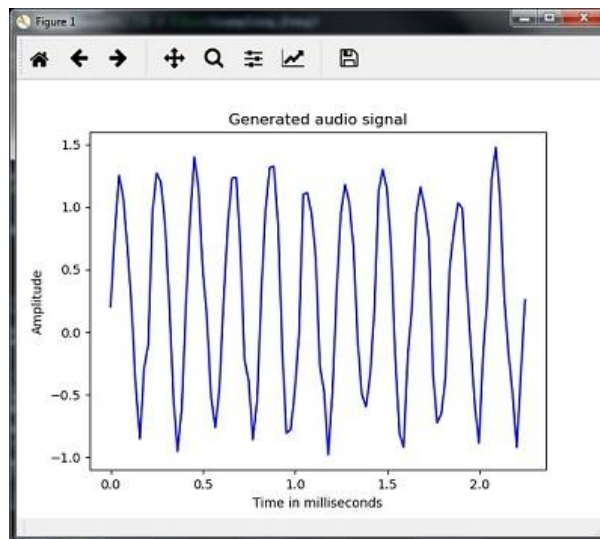
Rasmda ko'rsatilganidek, bizning grafigimiz uchun dastlabki 100 ta qiymatni chiqarib oling:

```
audio_signal = audio_signal[:100]
time_axis = 1000 * np.arange(0, len(signal), 1) / float(sampling_freq)
```

Endi yaratilgan audio signalni quyidagicha tasavvur qiling:

```
plt.plot(time_axis, signal, color='blue')
plt.xlabel('Time in milliseconds')
plt.ylabel('Amplitude')
plt.title('Generated audio signal')
plt.show()
```

Bu erda berilgan rasmda ko'rsatilgandek grafikni kuzatishingiz mumkin:



Nutqdan funktsiyalarni ajratib olish

Bu nutqni aniqlash qurilmasini yaratishda eng muhim qadamdir, chunki nutq signalini chastota domeniga aylantirgandan so'ng, biz uni foydalanish mumkin bo'lgan xususiyat vektor shakliga aylantirishimiz kerak. MFCC, PLP, PLP-RASTA va boshqalar kabi xususiyatlarni ajratib olishning turli usullaridan foydalanishimiz mumkin. shu maqsadda.

Misol

Quyidagi misolda biz MFCC texnikasidan foydalangan holda Python-dan foydalanib, bosqichma-bosqich Signal-dan xususiyatlarni chiqaramiz. Kerakli paketlarni bu yerda ko'rsatilganidek import qiling:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from python_speech_features import mfcc, logfbank
```

Endi saqlangan audio faylni o'qing. U ikkita qiymatni qaytaradi - namuna tezligi va audio signal. Saqlangan audio faylga yo'lni belgilang.

```
frequency_sampling, audio_signal = wavfile.read("/Users/admin/audio_file.wav")
```

E'tibor bering, biz bu erda tahlil qilish uchun dastlabki 15 000 ta namunani olamiz.

```
audio_signal = audio_signal[:15000]
```

MFCC usullaridan foydalaning va MFCC xususiyatlarini ajratib olish uchun quyidagi buyruqni bajaring -

```
features_mfcc = mfcc(audio_signal, frequency_sampling)
```

Endi MFCC parametrlarini rasmda ko'rsatilganidek chiqaring.

```
print('\nMFCC:\nNumber of windows =', features_mfcc.shape[0])  
print('Length of each feature =', features_mfcc.shape[1])
```

Endi quyida berilgan buyruqlar yordamida MFCC funksiyalarini chizing va tasavvur qiling.

```
features_mfcc = features_mfcc.T  
plt.matshow(features_mfcc)  
plt.title('MFCC')
```

Ushbu bosqichda biz - rasmda ko'rsatilganidek, filtrlar banki funktsiyalari bilan ishlaymiz. O'chirish funktsiyalari banka filtrlar.

```
filterbank_features = logfbank(audio_signal, frequency_sampling)
```

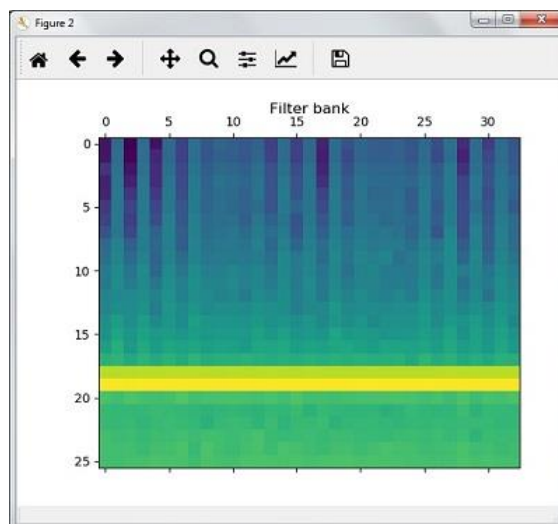
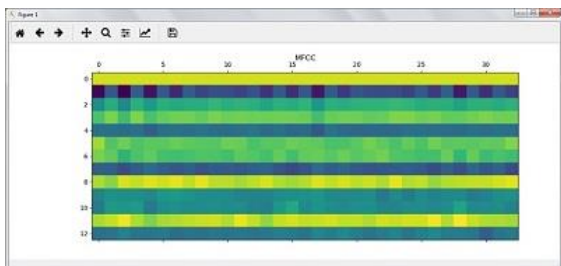
Endi filtr banki parametrlarini chop eting.

```
print('\nFilter bank:\nNumber of windows =', filterbank_features.shape[0])  
print('Length of each feature =', filterbank_features.shape[1])
```

Endi filtrlar banki funktsiyalarini chizing va tasavvur qiling.

```
filterbank_features = filterbank_features.T  
plt.matshow(filterbank_features)  
plt.title('Filter bank')  
plt.show()
```

Yuqorida tavsiflangan amallarni bajarish natijasida siz quyidagi natijalarni ko'rishingiz mumkin: rasm. MFCC uchun 1 va rasm. 2 filtr banki uchun



Og‘zaki so‘zni aniqlash

Nutqni aniqlash degani, odamlar gapirganda, mashina buni tushunadi. Buni amalga oshirish uchun biz Python-da Google Speech API-dan foydalanamiz. Buning uchun biz quyidagi paketlarni o‘rnatishimiz kerak:

- **Pyaudio** - uni *pip install Pyaudio* buyrug‘i yordamida o‘rnatish mumkin.
- **Speech Recognition** - Ushbu paketni *pip install SpeechRecognition* yordamida o‘rnatish mumkin.
- **Google-Speech-API** - U *pip install google-api-python-client* buyrug‘i yordamida o‘rnatilishi mumkin.

Misol

Og‘zaki so‘zlarni qanday aniqlashni tushunish uchun quyidagi misolni ko‘rib chiqing..Rasmda ko‘rsatilganidek, kerakli paketlarni import qiling.

```
import speech_recognition as sr
```

Pastda ko‘rsatilgan kabi ob‘ekt yarating.

```
recording = sr.Recognizer()
```

Endi **Microphone()** moduli ovozni kirish sifatida qabul qiladi.

```
with sr.Microphone() as source: recording.adjust_for_ambient_noise(source)
print("Please Say something:")
audio = recording.listen(source)
```

Endi Google API ovozni taniydi va natijani beradi.

```
try:  
    print("You said: \n" + recording.recognize_google(audio))  
except Exception as e:  
    print(e)
```

Quyidagi natijani ko‘rishingiz mumkin.

Please Say Something:

You said:

Misol uchun, agar siz **camuf.uz** deb aytgan bo‘lsangiz, tizim uni quyidagicha to‘g‘ri taniydi.

camuf.uz

O‘z-o‘zini nazorat qilish uchun savollar

1. Nutqni tan tizimini yaratish?
2. Nutqni aniqlash tizimini ishlab chiqishdagi qiyinchiliklar?
3. Ovozli signallarni vizualizatsiya qilish?
4. Ovoz signalining xarakteristikasi?
5. Monoton ovoz signalini yarating?
6. Nutqdan funktsiyalarni ajratib olish?
7. Og‘zaki so‘zni aniqlash?

2.10-§. EVRISTIK QIDIRISH

Evristik qidiruv sun'iy intellektda muhim rol o'ynaydi. Ushbu bobda siz bu haqda batafsil ma'lumotga ega bo'lasiz.

Sun'iy intellektda evristik qidiruv tushunchasi

Evristika — bu bizni ehtimoliy yechimga olib boruvchi empirik qoidadir. Sun'iy intellekt sohasidagi aksariyat muammolar eksponentsial xarakterga ega bo'lib, ko'plab mumkin bo'lgan yechimlarga ega. Qaysi yechimlarning to'g'ri ekanligini aniq bilib bo'lmaydi, va barcha yechimlarni tekshirish juda qimmatga tushadi.

Shunday qilib, evristikadan foydalanish qidiruvni toraytiradi va noto'g'ri variantlarni istisno qiladi. Qidiruv fazosida qidiruvni boshqarish uchun evristikadan foydalanish usuli evristik qidiruv deb ataladi. Evristik usullar juda foydali, chunki ular yordamida qidiruv jarayonini tezlashtirish mumkin.

Malumotsiz va axborotli qidiruv o'rtasidagi farq

Boshqaruv strategiyalari yoki qidiruv usullarining ikki turi mavjud: ma'lumotsiz va axborotli. Bu yerda aytilganidek, ular batafsil tushuntirilgan:

Ma'lumotsiz qidiruv

U ko'r-ko'rona qidirish yoki ko'r-ko'rona nazorat qilish strategiyasi deb ham ataladi. U faqat muammoning ta'rifi haqidagi ma'lumotlarni o'z ichiga oladi va shartlar haqida boshqa hech qanday qo'shimcha ma'lumot mavjud emasligi sababli shunday nomlanadi. Ushbu turdagi qidiruv usullari yechimni olish uchun butun davlat maydoni bo'ylab qidirish imkonini beradi. Kenglik-birinchi qidiruv (BFS) va chuqurlik-birinchi qidiruv (DFS) ma'lumotsiz qidiruvga misoldir.

Axborotli qidiruv

U evristik qidiruv yoki evristik boshqaruv strategiyasi deb ham ataladi. U shtatlar haqida ba'zi qo'shimcha ma'lumotlarni o'z ichiga olgani uchun shunday nomlangan. Ushbu qo'shimcha ma'lumot tadqiqot va kengaytirish uchun bola tugunlarining afzalliklarini hisoblash uchun foydalidir. Har bir tugun u bilan

bog‘langan evristik funktsiyaga ega bo‘ladi. Eng yaxshi birinchi qidiruv (BFS), A*, Average va Analysis ma’lumotli qidiruvga misollardir.

Cheklovlarni qanoatlantirish muammolari (Constraint Satisfaction Problems - CSP)

Cheklov tushunchasi cheklash yoki chegaralashni anglatadi. Sun’iy intellektda cheklovlarni qanoatlantirish muammolari — bu belgilangan cheklovlar asosida hal qilinishi lozim bo‘lgan masalalardir. Asosiy e’tibor bunday muammolarni hal qilish jarayonida cheklovlarni buzmaslikka qaratilishi kerak. Yakuniy yechimga kelganda, CSP barcha belgilangan cheklovlarni qanoatlantirishi lozim.

Haqiqiy dunyo muammolari cheklovlarni qanoatlantirish orqali hal qilinadi

Oldingi bo‘limlarda cheklovlarni qanoatlantirishga oid muammolarni yaratish muhokama qilindi. Endi esa, ushbu yondashuvni haqiqiy dunyo muammolariga qo‘llaymiz. Quyida cheklovlarni qanoatlantirish orqali hal qilinadigan ba’zi haqiqiy muammolar misollari keltirilgan:

Algebraik munosabatni yechish

Cheklovlarni qanoatlantirish vazifasi yordamida algebraik munosabatlarni hal qilishimiz mumkin. Ushbu misolda biz oddiy algebraik munosabatni, ya’ni $a \times 2 = b$, yechishga harakat qilamiz. U biz belgilagan diapazon ichida **a** va **b** qiymatlarini qaytaradi. Ushbu Python dasturini yakunlaganingizdan so‘ng, cheklovlarni qanoatlantirishga oid masalalarni hal qilish asoslarini tushunishingiz mumkin bo‘ladi.

E’tibor bering, dastur yozishdan oldin Python-ning **python-constraint** nomli paketini o‘rnatishimiz kerak. Uni quyidagi buyruq yordamida o‘rnatishingiz mumkin:

```
pip install python-constraint
```

Quyidagi qadamlar sizga cheklovni qondirish yordamida algebraik munosabatlarni yechish uchun Python dasturini ko'rsatadi. Quyidagi buyruq yordamida **constraint** paketini import qiling:

```
from constraint import *
```

Quyida ko'rsatilganidek, **problem()** nomli modul ob'ektini yarating:

```
problem = Problem()
```

Endi o'zgaruvchilarni aniqlaymiz. E'tibor bering, bu erda bizda ikkita o'zgaruvchi bor a va b va biz 10 ni ularning diapazoni sifatida belgilaymiz, ya'ni biz birinchi 10 raqam ichida yechimga erishdik.

```
problem.addVariable('a', range(10))  
problem.addVariable('b', range(10))
```

Keyinchalik, biz ushbu muammoga qo'llamoqchi bo'lgan maxsus cheklovni aniqlaymiz. E'tibor bering, bu erda biz $a * 2 = b$ cheklovidan foydalanamiz .

```
problem.addConstraint(lambda a, b: a * 2 == b)
```

quyidagi buyruq yordamida **getSolution()** modul obyektini yarating:

```
solutions = problem.getSolutions()
```

Nihoyat, quyidagi buyruq yordamida chiqishni chop qiling.

```
print (solutions)
```

Yuqoridagi dasturning chiqishini quyidagicha kuzatishingiz mumkin:

```
 [{'a': 4, 'b': 8}, {'a': 3, 'b': 6}, {'a': 2, 'b': 4}, {'a': 1, 'b': 2}, {'a': 0, 'b': 0}]
```

Sehrli kvadrat

Sehrli kvadrat — bu turli xil sonlarni, odatda butun sonlarni, kvadrat shaklidagi panjara ichida joylashtirish bo‘lib, bunda har bir qator, ustun va diagonal bo‘ylab sonlarning yig‘indisi bir xil qiymatni beradi. Ushbu qiymat "sehrli konstant" deb ataladi.

Quyida sehrli kvadratlarni generatsiya qilish uchun oddiy Python kodini bosqichma-bosqich bajarish keltirilgan:

Quyida ko‘rsatilgandek **magic_square** nomli funktsiyani belgilang:

```
def magic_square(matrix_ms):  
    iSize = len(matrix_ms[0])  
    sum_list = []
```

Quyidagi kod kvadratlarni vertikal joylashtirish kodini ko‘rsatadi:

```
for col in range(iSize):  
    sum_list.append(sum(row[col] for row in matrix_ms))
```

Quyidagi kod kvadratlarni gorizontal ravishda chizish uchun kodni ko‘rsatadi:

```
sum_list.extend([sum (lines) for lines in matrix_ms])
```

Quyidagi kod kvadratlarni gorizontal joylashtirish kodini ko‘rsatadi:

```
dlResult = 0  
for i in range(0,iSize):  
    dlResult +=matrix_ms[i][i]  
sum_list.append(dlResult)  
drResult = 0  
for i in range(iSize-1,-1,-1):  
    drResult +=matrix_ms[i][i]  
sum_list.append(drResult)  
  
if len(set(sum_list))>1:  
    return False  
return True
```

Endi matritsa qiymatini kiriting va natijani tekshiring:

```
print(magic_square([[1,2,3], [4,5,6], [7,8,9]]))
```


Chiqarish **noto‘g‘ri** bo‘lishini sezishingiz mumkin, chunki yig‘indi bir xil raqamga mos kelmaydi.

```
print(magic_square([[3,9,2], [3,5,7], [9,1,6]]))
```

To‘g‘ri bo‘lishini sezishingiz mumkin, chunki yig‘indi bir xil raqamga teng, ya’ni bu erda **15**.

O‘z-o‘zini nazorat qilish uchun savollar

1. Sun’iy intellektda evristik qidiruv tushunchasi?
2. Malumotsiz va axborotli qidiruv o‘rtasidagi farq?
3. Cheklolarni qanoatlantirish muammolari (Constraint Satisfaction Problems - CSP)?
4. Sehrli kvadrat?

2.11-§. SUN'IY INTELLEKT BILAN PYTHON YORDAMIDA O'YINLAR

O'yinlarda strategiya muhim rol o'ynaydi. Har bir o'yinchi yoki jamoa o'yin boshlanishidan oldin o'z strategiyasini ishlab chiqishi kerak, va o'yin davomida mavjud vaziyatga qarab bu strategiyani o'zgartirish yoki yangisini ishlab chiqish zarur bo'ladi. Bu jarayon o'yinchining harakatlarini va qarorlarini shakllantiradi, hamda raqiblarning harakatlariga javob berish uchun moslashuvchanlikni talab qiladi.

Qidiruv algoritmlari

Sizga shuningdek yuqoridagi kabi strategiyaga ega bo'lgan kompyuter o'yinlarini ko'rib chiqish kerak bo'ladi. Diqqat qiling, aynan qidiruv algoritmlari kompyuter o'yinlaridagi strategiyani belgilaydi.

Qanday ishlaydi

Qidiruv algoritmlarining maqsadi – ishtirokchilar maqsadga erishishlari va g'alaba qozonishlari uchun optimal yurishlar to'plamini topishdir. Ushbu algoritmlar har bir o'yin uchun turli xil g'alaba shartlaridan foydalanib, eng yaxshi yurishlarni qidiradilar.

Kompyuter o'yinini daraxt shaklida tasavvur qiling. Biz bilamizki, daraxtda tugunlar mavjud. Ildizdan boshlagan holda, biz g'alaba keltiradigan yakuniy tugunga optimal yurishlar bilan yetib borishimiz mumkin. Bu qidiruv algoritmlarining vazifasi. Bunday daraxtdagi har bir tugun kelajakdagi holatni ifodalaydi. Qidiruv algoritmlari har bir bosqich yoki o'yin tugunida qaror qabul qilish uchun bu daraxtni ko'rib chiqadilar.

Kombinatsiyalangan qidiruv

Qidiruv algoritmlarini qo'llashning asosiy kamchiligi shundaki, ular tabiatan to'liqdir, shuning uchun ular yechim topish uchun butun qidiruv maydonini o'rganadilar, bu esa resurslarni isrof qiladi. Agar ushbu algoritmlar yakuniy yechimni topish uchun butun qidiruv maydonini izlashi kerak bo'lsa, bu qiyinroq bo'lar edi.

Ushbu turdagi muammolarni bartaraf etish uchun biz qidiruv maydonini o'rganish uchun evristikadan foydalanadigan va mumkin bo'lgan noto'g'ri harakatlarni bartaraf etish orqali uning hajmini kamaytiradigan kombinatsiyalangan qidiruvdan foydalanishimiz mumkin. Shuning uchun bunday algoritmlar resurslarni tejashga qodir. Bo'sh joy topish va resurslarni tejash uchun evristikadan foydalanadigan ba'zi algoritmlar bu erda muhokama qilinadi.

Minimax algoritmi

Bu kombinatsion qidiruv tomonidan qo'llaniladigan strategiya bo'lib, qidiruv strategiyasini tezlashtirish uchun evristikani ishlatadi. Minimax strategiyasining konsepsiyasini ikki o'yinchi o'yinlari misolida tushunish mumkin, unda har bir o'yinchi raqibining keyingi yurishini oldindan taxmin qilishga harakat qiladi va bu funksiyani minimallashtirishga intiladi. Bundan tashqari, g'alaba qozonish uchun, o'yinchi har doim o'zining shaxsiy funksiyalaridan maksimal darajada foydalanishga intiladi, bu esa joriy holatga asoslanadi. Evristika bunday turdagi strategiyalar, masalan, Minimaxda muhim rol o'ynaydi. Har bir daraxt tuguni bilan evristik funksiyaning bog'lanishi bo'ladi. Ushbu evristikaga asoslanib, o'yinchi eng katta foyda keltiradigan tugunga qarab bir qadamni amalga oshirishga qaror qiladi.

Alfa-beta kesish

Minimax algoritmining asosiy muammosi shundaki, u daraxtning ahamiyatsiz qismlarini o'rganishi mumkin, bu esa resurslarning isrof bo'lishiga olib keladi. Shuning uchun, daraxtning qaysi qismi muhim va qaysi qismi ahamiyatsiz ekanligini hal qilishga yordam beradigan strategiya bo'lishi kerak va ahamiyatsiz qismlar o'rganilmasdan qoldirilishi lozim. Alfa-beta kesish bu kabi strategiyalardan biridir.

Alfa-beta kesish algoritmining asosiy maqsadi – daraxtning hech qanday echimi bo'lmagan qismlarini qidirishni oldini olishdir. Alfa-beta kesishning asosiy konsepsiyasi – ikki chegarani, Alfa (maksimal minimal chegara) va Beta (minimal maksimal chegara) deb nomlangan, ishlatishdir. Bu ikki parametr, mumkin bo'lgan echimlar to'plamini cheklovchi qiymatlardir. U hozirgi tugun qiymatini alfa va beta parametrlarining qiymati bilan taqqoslaydi, shunda echim mavjud bo'lgan daraxtning o'sha qismiga o'tish va qolganini chiqarib tashlash mumkin bo'ladi.

Negamax algoritmi

Ushbu algoritm minimaks algoritmiga o'xshash, lekin u yanada nafisroq amalga oshirilgan. Minimax algoritmini qo'llashning asosiy kamchiligi shundaki, bizga ikkita turli evristik funksiyalarni aniqlash kerak. Bu evristik metodlar o'rtasidagi bog'lanish shundaki, o'yin holati bir o'yinchi uchun qanchalik yaxshi bo'lsa, boshqa o'yinchi uchun shunchalik yomon bo'ladi. Negamax algoritmida ikkita evristik funksiyaning bitta evristik funksiya yordamida bajariladi.

O'yinlar uchun botlarni yaratish

Ishni **easySI** kutubxonasini o'rnatishidan boshlashimiz kerak. Bu ikki o'yinchi o'yinlarini yaratish uchun barcha funktsiyalarni ta'minlaydigan sun'iy intellekt asosidir. Siz uni quyidagi buyruq yordamida yuklab olishingiz mumkin.

```
pip install easySI
```

Oxirgi tanga o'ynash uchun bot

Ushbu o'yinda tonna tangalar bo'ladi. Har bir o'yinchi ushbu qoziqdan ma'lum miqdordagi tangalarni olishi kerak. O'yinning maqsadi qoziqdagi oxirgi tangani olish emas. Biz **EasySI** kutubxonasi **TwoPlayersGame** sinfidan meros qolgan **LastCoinStanding** sinfidan foydalanamiz. Quyidagi kod ushbu o'yin uchun Python kodini ko'rsatadi. Rasmda ko'rsatilganidek, kerakli paketlarni import qiling:

```
from easyAI import TwoPlayersGame, id_solve, Human_Player, AI_Player
from easyAI.AI import TT
```

TwoPlayerGame sinfidan sinfni meros qilib oling:

```
class LastCoin_game(TwoPlayersGame):
    def __init__(self, players):
```

Endi o'yinchilarni va o'yinni kim boshlashini aniqlang.

```
self.players = players
self.nplayer = 1
```

Endi o'yindagi tangalar sonini aniqlang, bu erda biz o'yin uchun 15 tanga foydalanamiz.

```
self.num_coins = 15
```

O'yinchi har bir turda olishi mumkin bo'lgan maksimal tangalar sonini aniqlang.

```
self.max_coins = 4
```

Endi biz quyidagi kodda ko'rsatilganidek, ba'zi aniq narsalarni aniqlashimiz kerak. Aniqlash mumkin harakat qiladi .

```
def possible_moves(self):  
    return [str(a) for a in range(1, self.max_coins + 1)]
```

Aniqlash o'chirish tangalar.

```
def make_move(self, move):  
    self.num_coins -= int(move)
```

Oxirgi tangani kim olganini aniqlang.

```
def win_game(self):  
    return self.num_coins <= 0
```

O'yinni qachon to'xtatish kerakligini aniqlang, ya'ni kimdir g'alaba qozonadi.

```
def is_over(self):  
    return self.win()
```

Ballarni qanday hisoblashni aniqlang.

```
def score(self):  
    return 100 if self.win_game() else 0
```

Uyumda qolgan tangalar sonini aniqlang.

```
def show(self):
    print(self.num_coins, 'coins left in the pile')
if __name__ == "__main__":
    tt = TT()
    LastCoin_game.ttentry = lambda self: self.num_coins
```

Quyidagi kod bloki yordamida o'yinni yechish:

```
r, d, m = id_solve(LastCoin_game,
    range(2, 20), win_score=100, tt=tt)
print(r, d, m)
```

O'yinni kim boshlashini hal qilish.

```
game = LastCoin_game([AI_Player(tt), Human_Player()])
game.play()
```

Siz ushbu o'yinning quyidagi chiqishi va oddiy o'yinini topishingiz mumkin:

```
d:2, a:0, m:1
d:3, a:0, m:1
d:4, a:0, m:1
d:5, a:0, m:1
d:6, a:100, m:4
1 6 4
15 coins left in the pile
Move #1: player 1 plays 4 :
11 coins left in the pile
Player 2 what do you play ? 2
Move #2: player 2 plays 2 :
9 coins left in the pile
Move #3: player 1 plays 3 :
6 coins left in the pile
Player 2 what do you play ? 1
Move #4: player 2 plays 1 :
5 coins left in the pile
Move #5: player 1 plays 4 :
1 coins left in the pile
Player 2 what do you play ? 1
Move #6: player 2 plays 1 :
0 coins left in the pile
```

Krestik va nollar o'yni uchun bot

Krestik va nollar o'yini - bu juda tanish va eng mashhur o'yinlardan biri. Keling, Python-dagi **easySI** kutubxonasidan foydalanib, ushbu o'yinni yarataylik . Quyidagi kod bu o'yinning Python kodi. Rasmda ko'rsatilgandek paketlarni import qiling:

```
from easyAI import TwoPlayersGame, AI_Player, Negamax
from easyAI.Player import Human_Player
```

TwoPlayerGame sinfidan sinfni meros qilib oling:

```
class TicTacToe_game(TwoPlayersGame):
    def __init__(self, players):
```

Endi o'yinchilarni va o'yinni kim boshlashini aniqlang -

```
self.players = players
self.nplayer = 1
```

To'lov turini aniqlang:

```
self.board = [0] * 9
```

Endi ba'zi ba'zi narsalarni quyidagicha belgilash kerak. Mumkin bo'lgan harakatlarni aniqlang.

```
def possible_moves(self):
    return [x + 1 for x, y in enumerate(self.board) if y == 0]
```

O'yinchi harakatni aniqlang:

```
def make_move(self, move):
    self.board[int(move) - 1] = self.nplayer
```

Sun'iy intellektni yaxshilash uchun o'yinchi qachon harakat qilganini aniqlang:

```
def umake_move(self, move):
    self.board[int(move) - 1] = 0
```

Raqib ketma-ket uchta bo‘lishi kerak bo‘lgan mag‘lubiyat holatini aniqlang:

```
def condition_for_lose(self):
    possible_combinations = [[1,2,3], [4,5,6], [7,8,9],
                             [1,4,7], [2,5,8], [3,6,9], [1,5,9], [3,5,7]]
    return any([all([self.board[z-1] == self.nopponent)
                    for z in combination]) for combination in possible_combinations])
```

O‘yinni tugatish uchun tekshiruni belgilang:

```
def is_over(self):
    return (self.possible_moves() == []) or self.condition_for_lose()
```

O‘yindagi o‘yinchilarning hozirgi holatini ko‘rsataish:

```
def show(self):
    print('\n'+'\n'.join([' '.join(['.', 'O', 'X'])[self.board[3*j + i]]
                           for i in range(3)]) for j in range(3)))
```

Ballarni hisoblash.

```
def scoring(self):
    return -100 if self.condition_for_lose() else 0
```

Algoritmi aniqlash ushuni asosiy usulini tanlang va o‘yinni boshlang:

```
if __name__ == "__main__":
    algo = Negamax(7)
    TicTacToe_game([Human_Player(), AI_Player(algo)]).play()
```

Siz ushbu o‘yinning quyidagi chiqishi va oddiy o‘yinini ko‘rishingiz mumkin:

```
...
...
...
Player 1 what do you play ? 1
Move #1: player 1 plays 1 :
O . .
...
...
```


Move #2: player 2 plays 5 :

O . .

. X .

121

...

Player 1 what do you play ? 3

Move #3: player 1 plays 3 :

O . O

. X .

...

Move #4: player 2 plays 2 :

O X O

. X .

...

Player 1 what do you play ? 4

Move #5: player 1 plays 4 :

O X O

O X .

...

Move #6: player 2 plays 8 :

O X O

O X .

. X .

O‘z-o‘zini nazorat qilish uchun savollar

1. Qidiruv algoritmlari?
2. Kombinatsiyalangan qidiruv?
3. Minimax algoritmi?
4. Alfa-beta kesish?
5. Negamax algoritmi?

III-BOB. SUN'IY INTELLEKTDAN NEYRON TARMOQLAR.

3.1-§. SUN'IY NEYRON TARMOQLAR

Neyron tarmoqlar — bu parallel hisoblash qurilmalari bo'lib, ular miyaning kompyuter modelini yaratishga bo'lgan urinishni ifodalaydi. Asosiy maqsad — an'anaviy tizimlarga qaraganda turli hisoblash vazifalarini tezroq bajarishga mo'ljallangan tizimni ishlab chiqishdir. Ushbu vazifalarga tasvirlarni tanib olish va tasniflash, approksimatsiya, optimizatsiya va ma'lumotlarni klasterlash kiradi.

Sun'iy neyron tarmoqlar (Artificial Neural Networks - ANN) nima?

Sun'iy neyron tarmoq (ANN) — bu biologik neyron tarmoqlarga o'xshashlik asosida yaratilgan samarali hisoblash tizimi. ANN shuningdek, sun'iy neyron tizimlar, parallel taqsimlangan ishlov berish tizimlari va konneksionistik tizimlar deb ham ataladi. ANN bir-biri bilan aloqani ta'minlash uchun ma'lum bir shablon bo'yicha o'zaro bog'langan modullar to'plamidan iborat. Ushbu qurilmalar, shuningdek, tugunlar yoki neyronlar deb ataladi va ular bir vaqtning o'zida parallel ishlaydigan oddiy protsessorlardir.

Har bir neyron boshqa neyron bilan ulanish kanali orqali bog'langan. Har bir ulanish kanaliga kiruvchi signal haqidagi ma'lumotni o'z ichiga oluvchi vazn qiymati birlashtirilgan. Bu vazn ma'lum bir vazifani bajarishda neyronlar uchun eng foydali ma'lumot bo'lib, u odatda uzatilayotgan signalni kuchaytiradi yoki bostiradi. Har bir neyronning o'z ichki holati bor, bu aktivatsiya signali deb ataladi. Kiruvchi signallarni birlashtirish va aktivatsiya qoidasi qo'llanilganidan so'ng hosil bo'ladigan chiqish signallari boshqa bloklarga yuborilishi mumkin

Ob'ektlar va ularning xususiyatlari

Faraz qilaylik, bir xil turdagi obyektlar mavjud (masalan, sharob shishalari, shifoxonadagi tashrif buyuruvchilar, shaxmat taxtasidagi pozitsiyalar):



Har bir obyektни $\mathbf{x}=\{x_1, x_2, \dots, x_n\}$ ko‘rinishidagi **belgilarning** to‘plami (vektor) tavsiflaydi. Belgilar quyidagi turlarda bo‘lishi mumkin:

- **Haqqoniy qiymatlar** (masalan, vazn, bo‘y);
- **Ikkilik qiymatlar** (masalan, ayol/erkak);
- **Sonli bo‘lmagan qiymatlar** (masalan, qizil, ko‘k va hokazo).

Keyinchalik belgilarni $[0...1]$ diapazonidagi haqqoniy sonlar sifatida ko‘rib chiqamiz. Buni normallashtirish orqali har doim amalga oshirish mumkin, masalan:

$x \rightarrow (x - x_{\min}) / (x_{\max} - x_{\min})$. Ikkilik belgilar mos ravishda 0 yoki 1 qiymatlarini qabul qiladi. Sonli bo‘lmagan belgilarni vektor $\mathbf{x}$ -ning o‘lchamini oshirib, ikkilik ko‘rinishga keltirish mumkin (masalan, qizil/qizil emas, ko‘k/ko‘k emas).

Bundan tashqari, hozircha obyektlarning o‘zaro sababiy bog‘lanishi yo‘qligini va ularning tartibi muhim emasligini faraz qilamiz.

Faraz qilaylik, ushbu turdagi obyektlar sinflarga bo‘linadi (masalan, inson: {sog‘lom, kasal}; sharob: {italyancha, fransuzcha, gruzincha}). Shuningdek, har bir obyekt bilan ma’lum bir son y bog‘lanishi mumkin (masalan, shaxmat pozitsiyasida oq figuralar ustunligining darajasi; sharobning ekspertlar fikriga ko‘ra o‘rtacha sifati va boshqalar).

Ko‘pincha quyidagi ikki, bir-biri bilan yaqin bog‘liq masalalar hal qilinadi:

- **Klasifikatsiya:** obyekt KKK sinflardan qaysi biriga tegishli ekanligini aniqlash.
- **Regressiya:** obyektga mos keluvchi y sonini aniqlash.

Misollar:

- 1) 3 belgi: $\mathbf{x} = \{\text{harorat, gemoglobin darajasi, xolesterin miqdori}\}$; 2 sinf: {0: sog‘lom, 1: kasal};

- 2) $w \times h$ xususiyatlari: $\mathbf{x} = \{\text{kengligi } w \text{ va balandligi } h \text{ bo'lgan tasvir piksellerining yorqinligi}\}$; 10 ta sinf: $\{0-9: \text{rasmdagi raqam}\}$.
- 3) $8 \times 8 \times 13$ belgilari: $\mathbf{x} = \{\text{shaxmat donalarining katakchalardagi kodlari}\}$; regressiya: $\{\text{qora ustidan oq ustunlik} = [-1 \dots 1]\}$.

Klassifikatsiya yoki regressiya masalalarini muvaffaqiyatli hal qilish uchun obyektни tavsiflovchi belgilar mazmunli bo'lishi kerak, belgi vektori esa to'liq bo'lishi lozim (obyektlarni tasniflash yoki regressiya kattaligi $\mathbf{y}$ -ni aniqlash uchun yetarli darajada).

Bir kuni Jed va Ned otlarini ajratmoqchi bo'lishdi. Jed otning qulog'ini tirnadi. Ammo Nedning oti xuddi shu qulog'ini tikanga tirnadi. Keyin Ned otining dumiga ko'k kamon qo'ydi, lekin Jedning oti uni yedi. Fermerlar uzoq va qattiq o'ylab, o'zgartirish oson bo'lmagan xususiyatni tanladilar. Ular otlarning bo'yini sinchkovlik bilan o'lchab ko'rishdi va ma'lum bo'ldiki, Jedning qora toychog'i Nedning oq ayg'ridan bir santimetr balandroq ekan.

O'qituvchi bilan o'qitish

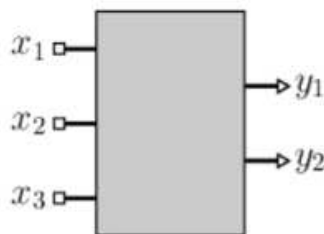
Faraz qilaylik, obyektlar to'plami mavjud bo'lib, har bir obyekt k ta raqamlangan $(0, 1, 2, \dots, k-1)$ sinflardan biriga tegishli. Bu o'qituvchi (odatda odam) tomonidan taqdim etilgan o'qitish to'plamida tizim o'rganadi. Keyin, tizimga noma'lum obyektlar (sinov to'plami) kiritilganda, ularni tasniflashni amalga oshiradi, ya'ni har bir obyekt qaysi sinfga tegishli ekanligini aniqlaydi. Bunday holatda bu o'qituvchi bilan o'rganishdan keyingi tasvirni tanib olish masalasi hisoblanadi.

Belgilar soni n belgi fazosining o'lchami deb ataladi. Faraz qilaylik, belgilar $[0 \dots 1]$ oralig'ida joylashgan. Demak, har bir obyekt belgi fazosidagi birlik n -o'lchovli kubning ichidagi nuqta sifatida tasvirlanadi.

Tasniflovchi tizimni qora quti sifatida tasavvur qilamiz. Ushbu qutining n ta kirishi bor, ularga belgi qiymatlari $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ beriladi va k ta chiqishi $\mathbf{y} = \{y_1, \dots, y_k\}$ bo'ladi (sinflar soni bo'yicha). Chiqishlarning qiymati ham $[0 \dots 1]$ oralig'idagi haqiqiy sonlar sifatida ko'rib chiqiladi.

Tizim to'g'ri o'rgatilgan deb hisoblanadi, agar i -chi sinfga tegishli belgilarga mos keladigan kirishlar taqdim etilganda, i -chi chiqishning qiymati 1 bo'lsa,

qolganlari esa 0 bo'lsa. Amalda bunday natijaga erishish qiyin bo'ladi, shuning uchun barcha chiqishlar nolga teng bo'lmagan qiymatlarga ega bo'ladi. Shunda, eng katta qiymatga ega bo'lgan chiqishning raqami sinfning raqami deb hisoblanadi, va bu qiymatning 1 ga yaqinligi tizimning "ishonch darajasi"ni bildiradi.



Agar faqat ikkita sinf bo'lsa, quti bitta chiqishiga ega bo'lishi mumkin. Agar bu chiqish 0 ga teng bo'lsa, bu birinchi sinfni bildiradi, agar 1 ga teng bo'lsa — ikkinchi sinfni. Noaniq tasniflashda esa ishonch darajasi uchun chegaralar kiritiladi. Masalan, agar chiqish qiymati $y=[0...0.3]$ oralig'ida bo'lsa, bu birinchi sinf, $y=[0.7...1]$ oralig'ida bo'lsa — ikkinchi sinf, va agar $y=(0.3...0.7)$ oralig'ida bo'lsa, tizim "qaror qabul qilishdan bosh tortadi".

Bitta chiqishli quti, shuningdek, $y=f(x_1,...,x_n)$ funktsiyasini yaqinlashtirishi mumkin, bu yerda y qiymatlari uzluksiz bo'lib, odatda 1 ga normallashtiriladi, ya'ni $y=[0...1]$. Bunday holda, regressiya masalasi hal qilinadi.

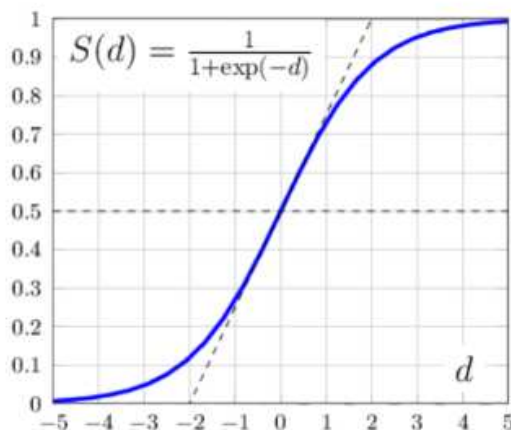
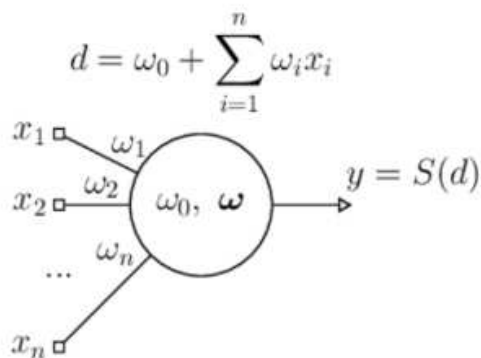
Neyron

Neyron tarmog'i — bu qora qutining mumkin bo'lgan tarkiblaridan biridir. Tarmoq tuguni — bu neyron bo'lib, u n ta kirish $\mathbf{x}=\{x_1,x_2,...,x_n\}$ va bitta chiqish y ga ega. Har bir kirish bilan *sinaptik vazn* $\boldsymbol{\omega}=\{w_1,w_2,...,w_n\}$ haqiqiy parametr bog'langan. Bundan tashqari, neyron "*siljish parametri*" w_0 ga ham ega. Shunday qilib, har qanday neyron n ta kirishga ega bo'lsa, u $n+1$ ta parametr bilan to'liq tavsiflanadi.

Neyronning chiqishi quyidagicha hisoblanadi. Har bir kirish x_i qiymati uning mos sinaptik vazni w_i bilan ko'paytiriladi va bu ko'paytmalar yig'iladi. Yig'indi qiymatiga w_0 naryad parametri qo'shiladi. Natija d ni $[0...1]$ oralig'iga sig'dirish uchun noliylik sigmoidal funksiya $y=S(d)$ qo'llaniladi:

$$d = w_0 + w_1x_1 + ... + w_nx_n, \quad y = S(d) = 1/(1+\exp(-d)).$$

Sigmoid funktsiyasi katta musbat d qiymatlarida 1 ga, katta salbiy d qiymatlarida esa 0 ga yaqinlashadi. $d=0$ bo'lganda, u $S(0)=0.5$ ga teng bo'ladi. Shunday qilib, neyron bu n ta o'zgaruvchiga ega bo'lgan chegarali turdagi noaniq funksiya hisoblanadi.



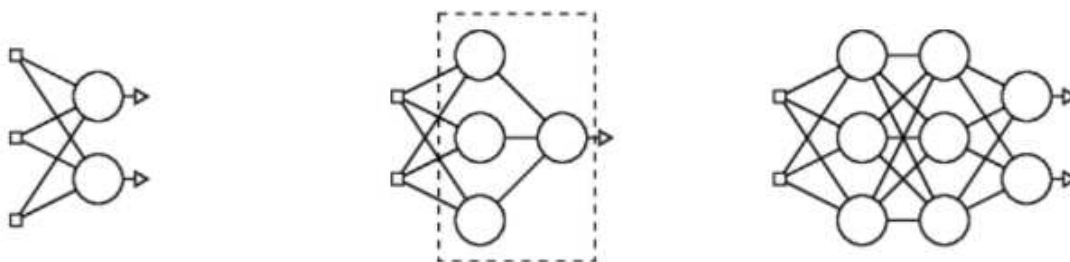
Neyronlar tarmog'i

Tarmoq — bu o'zaro bog'langan neyronlar to'plami. Ularni ulashning turli usullari mavjud, shuningdek, turli tarmoqlarni tashkil qilish mumkin. Faraz qilaylik, neyronlar qatlamlarga joylashtirilgan va i -chi qatlamning neyronlarining chiqish qiymatlari $i+1$ -chi qatlamdagi barcha neyronlarning kirishlariga taqdim etiladi. Bunday tarmoqni **to'liq bog'langan oldinga tarqatilgan tarmoq** (feedforward network) deb atashadi. Tarmoqning kirishlarini biz kvadrat shakllar bilan belgilaymiz va ularga **kirish neyronlari** deb nom beramiz. "Oddiy" neyronlardan farqli o'laroq, bu neyronlar faqatgina lineyali funksiya $y=x$ ni bajaradi. Tarmoqning oxirgi qatlamining neyronlarining chiqishlari qora qutining chiqishlari bo'lib, ular uchburchaklar bilan belgilangan.

Misol tariqasida, birinchi rasmda tarmoq uchta kirish (nol qatlam) va ikki chiqish neyronidan iborat. Bunday arxitekturaning quyidagicha kodlash mumkin: **[3,2]**, bu yerda raqamlar qatlamdagi neyronlar sonini bildiradi. Birinchi raqam — har doim kirishlar soni, oxirgi raqam esa chiqishlar soni.

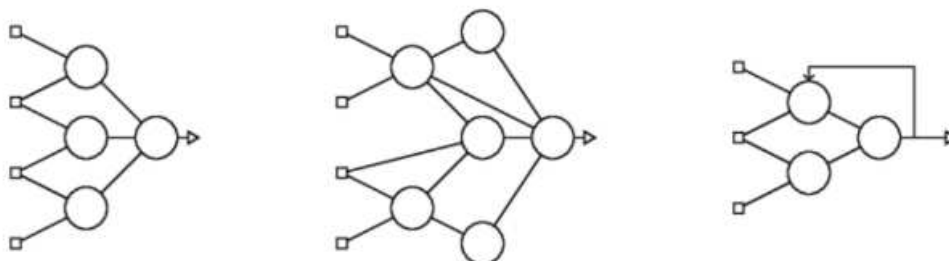
Keyingi rasmda **[2,3,1]** arxitekturasini ko'rsatilgan. Bu tarmoqda *bitta yashirin qatlam* bor va u uchta neyronni o'z ichiga oladi. Yashirin qatlam shunday nomlanadi, chunki u qora quti (chiziqli) ichida joylashgan, kirish va chiqish

qatlamlari orasida. (Shuningdek, chiqish qatlamining neyronlari ham qisman yashirilgan va faqat ularning chiqishlari tashqariga chiqqan). Uchinchi rasmda esa [2,3,3,2] tarmog‘i ko‘rsatilgan, bunda ikki yashirin qatlam mavjud.



Bu tarmoqlar **oldinga tarqatilgan tarmoqlar** deb ataladi, chunki ma’lumotlar (obyektning belgilarini) kirishiga kiritiladi va ketma-ket, halqasiz, chiqishlarga tarqatiladi. Xuddi shu turdagi tarmoqlarga **yoyilgan tarmoqlar** ham kiradi, bunda ikki qo‘shni qatlamdagi neyronlar o‘rtasida barcha ulanishlar mavjud emas (quyida birinchi rasm). Ko‘pincha bunday tarmoqlarda konvolyutsion qatlamdagi barcha neyronlarning vaznlari bir xil bo‘ladi. Bunday tarmoqlar haqida ko‘proq ma’lumotni rasmni tanib olishda muhokama qilamiz.

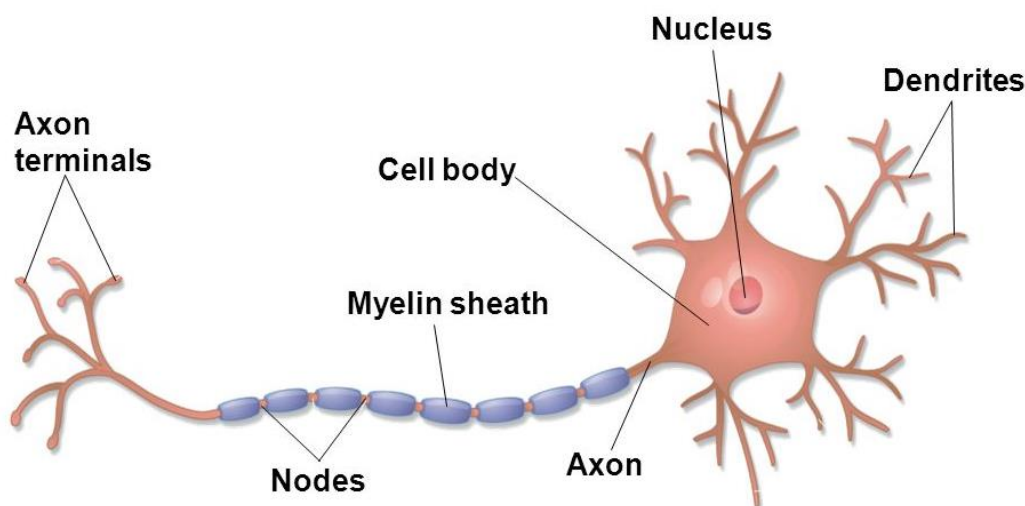
Ikkinchi rasmda esa qatlam tushunchasi mavjud bo‘lmagan tarmoqning variantlari ko‘rsatilgan, ammo bu hali ham oldinga tarqatilgan tarmoqdir.



So‘nggi rasm — bu endi **oldinga tarqatilgan tarmoq** emas, balki **rekurrent tarmoq** (recurrent network) deb ataladi. Ushbu tarmoqda bir yoki bir nechta chiqish neyronlaridan signal qaytib kirishlarga yuboriladi. Odatda bunday rekursiya bir necha tsikl davomida amalga oshiriladi, taqsimotning tarmoq chiqishlarida stasionar qiymatlar o‘rnatilmaguncha. Rekurrent tarmoqlar xotiraga ega va obyektlarning tartibi ular uchun muhimdir. Bunday xatti-harakatlar, agar obyektlar vaqt bo‘yicha tartiblangan bo‘lsa (masalan, vaqt ketma-ketligini bashorat qilishda), juda foydalidir.

Har qanday tarmoqni o'qitish neyronlarning har birining parametrlarini $w_0, w_1, \dots, w_n$ mos ravishda tanlashni o'z ichiga oladi, shunday qilib, agar tarmoqqa obyekt (ya'ni, kirishlar $x_1, \dots, x_n$) kiritilsa, tarmoq chiqishlari obyektning sinfiga mos qiymatlar olishadi.

Shuni ta'kidlash kerakki, har doim neyron faqat bitta chiqishga ega bo'lsa-da, u turli neyronlarning kirishlariga "yuborilishi" mumkin. Tabiiy neyronlar kabi, akson o'zining turli shoxlariga bo'linadi, har bir shox esa boshqa neyronlarning dendritlaridagi sinapslarga ta'sir qiladi. Agar neyron "qizib ketsa" (ya'ni, aktivlashsa), bu hayajon uning aksoni orqali qo'shni neyronlarning dendritlariga uzatiladi.



Neyron giperplan sifatida

Tanib olish tizimining qora qutisini shaffofroq qilish uchun neyronning geometrik talqinini ko'rib chiqaylik. n o'lchovli fazoda har bir nuqta n ta koordinata bilan belgilanadi (haqiqiy sonlar $\mathbf{x} = \{x_1, \dots, x_n\}$). Tekislik (oddiy 3 o'lchovli fazodagi kabi) normal vektor $\boldsymbol{\omega} = \{w_1, \dots, w_n\}$ (tekislikka perpendikulyar) va ixtiyoriy $\mathbf{x}_0 = \{x_{01}, \dots, x_{0n}\}$ nuqta bilan aniqlanadi, bu tekislikda yotgan qachon $n > 3$ tekislik odatda *giperplan deb ataladi*. $\mathbf{x} = \{x_1, \dots, x_n\}$ nuqtagacha bo'lgan masofa d formula bo'yicha hisoblanadi.

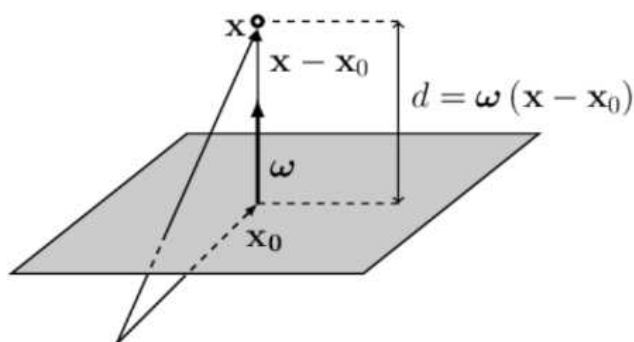
$$d = w_0 + w_1 x_1 + \dots + w_n x_n, \text{ bu erda } w_0 = -(w_1 x_{01} + \dots + w_n x_{0n}).$$

Bunda, agar x nuqta vektor ω to'g'ri keladigan tekislikning tomonida bo'lsa, $d > 0$, agar qarama-qarshi tomonda bo'lsa, $d < 0$. $d = 0$ bo'lganda - x nuqta tekislikda yotadi. Bu eslash kerak bo'lgan keyingi taqdimot uchun asosiy bayonotdir.

w_0 parametrini o'zgartirish tekislikni fazoda parallel ravishda siljitadi. Agar w_0 pasaysa, u holda tekislik **vektor** yo'nalishi bo'yicha siljiydi (masofa kichikroq), agar w_0 oshsa, tekislik ω vektoriga qarshi siljiydi. Bu to'g'ridan-to'g'ri yuqoridagi formuladan kelib chiqadi.

Ushbu formulani (o'tkazib yuborilishi mumkin) vektor yozuvida olamiz. x_0 nuqtadan (tekislikda yotgan) va x nuqtaga yo'naltirilgan $x - x_0$ vektorini yozamiz (o'ngdagi rasmga qarang; vektorlar uchburchak qoidasiga muvofiq qo'shiladi). x_0 nuqtaning pozitsiyasi ω vektorining bazasida tanlanadi, shuning uchun ω va $x - x_0$ kollinear (bir xil to'g'ri chiziqda yotadi). Agar ω vektori birlik ($\omega^2 = 1$) bo'lsa, u holda $x - x_0$ va ω vektorlarining skalyar ko'paytmasi x nuqtaning tekislikgacha bo'lgan masofasiga teng bo'ladi:

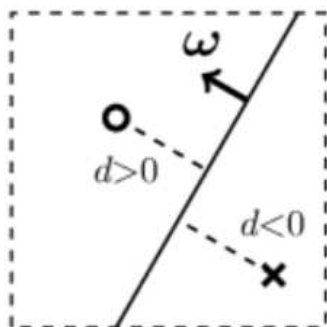
$$d = \omega \cdot (x - x_0) = -\omega \cdot x_0 + \omega \cdot x = w_0 + \omega \cdot x.$$



Agar uzunlik $w = |\omega|$ tekislikka normal ω vektori birlikdan farq qiladi, u holda d n o'lchovli fazodagi Evklid masofasidan w marta katta ($w > 1$) yoki kichik ($w < 1$) bo'ladi. $x - x_0$ va ω vektorlari qarama-qarshi yo'nalishda yo'naltirilganda: $d < 0$.

Agar fazo n ta o'lchamga ega bo'lsa, giper tekislik $(n-1)$ o'lchovli ob'ektdir. U butun makonni ikki qismga ajratadi. Aniqlik uchun 2 o'lchovli fazoni ko'rib chiqaylik. Undagi giperplane to'g'ri chiziq (bir o'lchovli ob'ekt) bo'ladi. Rasmning o'ng tomonida aylana kosmosdagi bir nuqtani, xoh esa boshqasini ifodalaydi. Ular chiziqning qarama-qarshi tomonlarida joylashgan (giperplan). Agar

ω vektorining uzunligi birdan ancha katta bo'lsa, u holda nuqtalardan tekislikgacha bo'lgan mutlaq masofalar d birdan sezilarli darajada katta bo'ladi.



Neyronga qaytaylik. Koordinatalari $x = (x_1, \dots, x_n)$ (kiritish vektori) nuqtadan gipertekislikka (w_0, ω) masofani d ni hisoblab chiqishini oson tushunish mumkin. Neyron parametrlari $\omega = (w_1, \dots, w_n)$ gipertekislik normalining yo'nalishini aniqlaydi va w_0 tekislikning ω vektori bo'ylab siljishi bilan bog'liq. Neyron chiqishi $[0 \dots 1]$ oralig'iga normallashtirilgan $S(d)$ masofasi bilan ta'minlanadi. Katta w_i uchun yuqorida aylana bilan chizilgan ob'ekt birga yaqin neyron chiqishiga, o'zaro faoliyat esa nolga olib keladi. Nisbat $w_0 / |\omega|$ tekislikdan koordinatagacha bo'lgan masofaga teng $(0, \dots, 0)$. Modulda u $n^{1/2}$ dan oshmasligi kerak.

1. Neyron giper tekislikdir. Uning chiqishi qiymati kirish vektoridan giperplanegacha bo'lgan normallashtirilgan masofaga teng. O'quv jarayonida har bir neyronning tekisligi o'z yo'nalishini o'zgartiradi va xususiyat maydonida siljiydi.

Neyron ishonchliligi

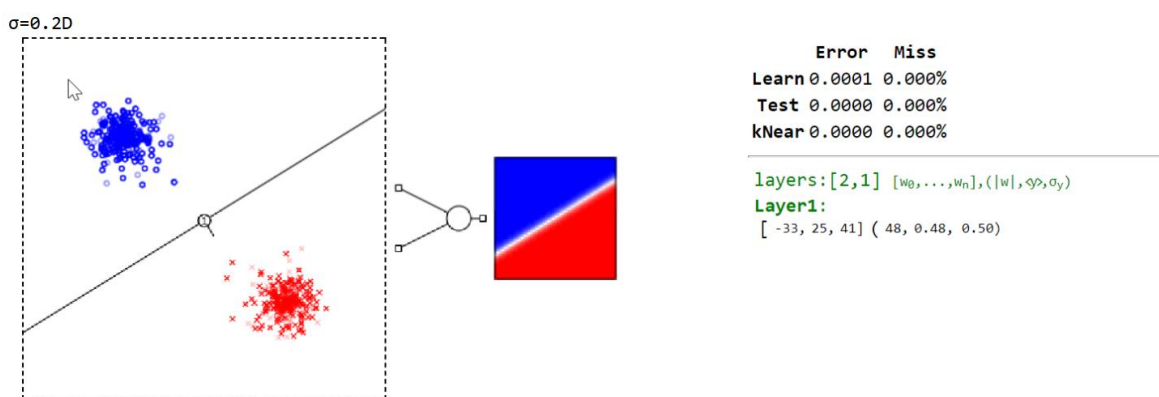
Tarmoqni o'rgatishda neyronlarning parametrlari tanlanadigan mezon talab qilinadi. Odatda, bu tarmoq chiqishlarining maqsadli qiymatlaridan chetlanish kvadrati yordamida amalga oshiriladi. Shunday qilib, ikkita sinf va bitta chiqish uchun biz xato tarmoq xatosi deb hisoblaymiz

$$Error^2 = (1/N) \sum (y - y_c)^2,$$

bu erda y - haqiqiy chiqish va y_c - uning to'g'ri qiymati, bir sinf uchun 0, ikkinchisi uchun 1. Yig'indi N ta ta'lim misollari asosida hisoblanadi. Ular barcha o'quv ob'ektlari uchun bu ildiz o'rtacha kvadrat xatosini minimal qilishga harakat

qilishadi. Xatolarni minimallashtirish usullarini (neyron parametrlarini tanlash) keyinroq muhokama qilamiz.

2 o'lchovli xususiyat maydonini ko'rib chiqing x_1 , x_2 va ikkita sinf 0 va 1. Quyidagi rasmda bir sinf ob'ektlari ko'k doiralar bilan, ikkinchi sinf ob'ektlari esa qizil xochlar bilan ifodalangan. Xususiyat maydonining o'ng tomonida bitta neyronning tarmog'i [2,1]. Uning orqasida, ko'k-qizil kvadratda, ma'lum kirishlar uchun *neyronning chiqish qiymatlari xaritasi* (x_1 , x_2 qiymatlari 0 dan 1 gacha 0,01 ga o'zgaradi). Agar $y=0$ bo'lsa - u ko'k, $y=1$ bo'lsa - qizil, oq esa $y=0,5$ qiymatiga mos keladi:

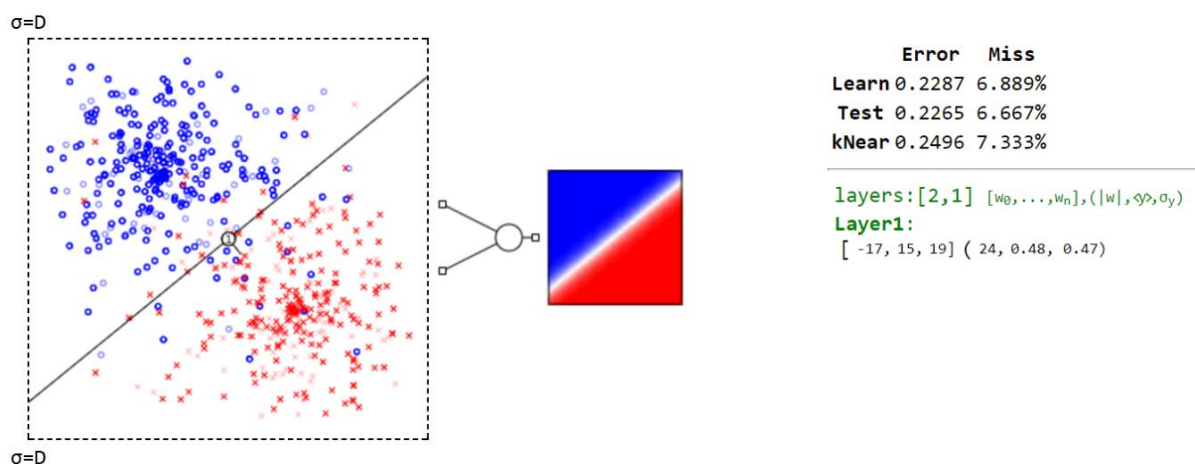


Raqamlarning o'ng tomonida, chiziq ostida, neyron parametrlari kvadrat qavs ichida berilgan: $[w_0, w_1, w_2]$. Qavslar ichida uzunligi $|w|$. normal vektor ω va o'rtacha chiqish qiymati $\langle y \rangle$ va uning o'zgaruvchanligi σ_y (pastga qarang). Xususiyat maydonining x_1 o'qi o'ngga, x_2 o'qi esa pastga yo'naltirilgan. Shuning uchun $\{w_1, w_2\}$ musbat komponentli ω vektori diagonal pastga yo'naltiriladi (u 1-sonli neyronni o'z ichiga olgan to'g'ri chiziqdagi aylana yonida chiziladi).

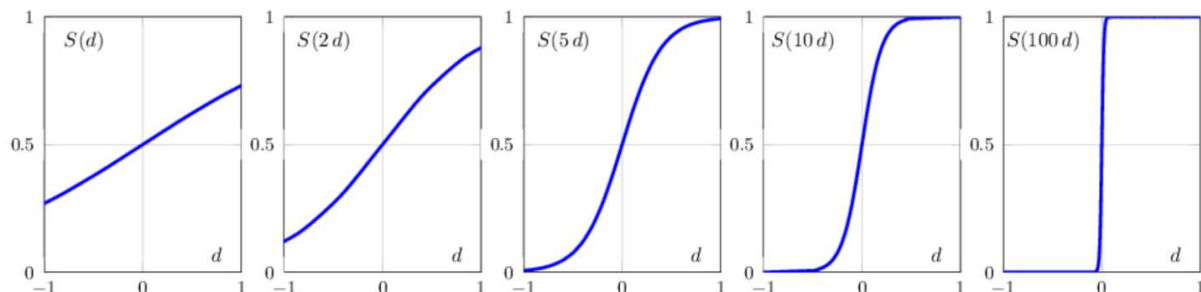
Jadvaldagi chiziq ustidagi ildiz o'rtacha kvadrat xatosi Bunday tarmoqning xatosi. Bunday holda, Learn liniyasi ob'ektlarning o'quv ketma-ketligini, Test esa mashg'ulotlarda ishtirok etmagan sinov ketma-ketligini bildiradi (xususiyatlar maydoni grafigidagi sinov ob'ektlari yarim shaffof sifatida tasvirlangan). Miss ustuni tarmoq tomonidan noto'g'ri tan olingan ob'ektlarning foizini o'z ichiga oladi (ularning sinfiga tayinlanmagan). Oxirgi satr kNear 10 ta eng yaqin qo'shnilar usulida xato va xatolar foizini bildiradi (bu haqda keyinroq tavsiflanadi).

Ushbu misolda har bir sinf ob'ektlarining xususiyatlarining tarqalishi kichikdir. Sinflar giperplan (2 o'lchamdagi chiziq) bilan osongina ajratiladi. Tarmoq chiqishda 0 yoki 1 ning maqsadli qiymatlariga xatolikni minimallashtirishga intiladi, shuning uchun vektor moduli $|w|=48$ nisbatan katta qiymatni oladi. Natijada, hatto tekislikka yaqin joylashgan ob'ektlar ham (odatdagi Evklid ma'nosida) d ning katta mutlaq qiymatini oladi. Shunga ko'ra, $y=S(d) = 0$ yoki 1 . Biz bunday neyronni *ishonchli deb ataymiz*. Ko'proq $|w|$, neyron qanchalik ishonchli bo'lsa. Uning chiqish xaritasida nozik oq chiziq (noaniqlik maydoni) quyuq ko'kni (bir sinf) quyuq qizildan (ikkinchi sinf) ajratib turadi.

Ikkinchi misolda vaziyat biroz farq qiladi, bu erda turli sinflar ob'ektlari o'rtasida keng qamrovli maydon mavjud. Endi neyron unchalik ishonchli emas va vektor uzunligi $|w|=24$ 2 marta kichik:



1,2,5,10,100 ga teng normal vektor uzunligi uchun masofa deformatsiya funksiyalarini (sigmasimon) keltiramiz:



koordinatalari $\{x_1, \dots, x_n\}$ bo'lgan x nuqtadan maksimal masofa n ning ildiziga teng. 2 o'lchovli xususiyat fazosida $d_{max}=1,4$. Agar tekislik kubning o'rtasidan o'tsa $d_{max} \sim 0,5$.

Ishonchli neyron har doim ham yaxshi neyron emas. Agar n xususiyat maydonining o'lchami katta bo'lsa va o'quv ma'lumotlari N kichik bo'lsa, ishonchli neyronlardan tashkil topgan tarmoq *haddan tashqari o'qitilishi* va sinov ob'ektlarida katta tanib olish xatosiga olib kelishi mumkin. Bundan tashqari, o'ziga ishongan neyronlar sekinroq o'rganadilar. Quyida biz ushbu masalalarni batafsil muhokama qilamiz.

Xulosa qilib, biz har qanday o'lchamdagi bo'shliqlar uchun amal qiladigan asosiy xulosani tuzamiz:

2. Xususiyat fazosidagi ikkita sinf giperplan bilan ajratilsa, ularni tanib olish uchun bitta neyron yetarli.

Neyronning foydaliligi

Agar neyronning gipertekisligi, xususiyatlar (yoki oldingi neyronlarning chiqishlari) joylashgan birlik giperkubini kesib o'tmasa, bunday neyron odatda foydasiz bo'ladi. U kirish ma'lumotlarini ikki qismga ajratmaydi (bu ma'lumotlar har doim $[0...1]/[0 \dots 1]/[0...1]$ oraliqda bo'ladi). Bunday neyron **foydasiz** deb ataladi.

Tarmoqdagi barcha neyronlar foydali bo'lishi kerak. Ba'zan gipertekislik, giperkubni kesib o'tgan bo'lsa ham, foydasiz bo'lishi mumkin, chunki barcha sinflardagi obyektlar ushbu gipertekislikning bir tomonida joylashgan bo'ladi. O'qitishdan oldin neyronlarning parametrlari tasodifiy qiymatlarga teng deb hisoblanadi. Shunday qilib, neyron darhol foydasiz bo'lishi mumkin. Bunday holatlarning oldini olish uchun quyidagi **algoritmini** qo'llash mumkin:

(Algoritmnini izohlash uchun qo'shimcha tafsilotlar kiritishingiz mumkin)

ω vektorining komponentlarini tasodifiy o'rnatamiz, masalan, $[-w \dots w]$ oralig'ida, bu erda $w \sim 1 - 10$. Keyin, birlik giperkubining ichida (yoki uning markaziy qismida) biz tanlaymiz. tasodifiy nuqta $x_0 = \{x_{01}, \dots, x_{0n}\}$. Shift parametrini quyidagicha o'rnatamiz: $w_0 = -\omega \cdot x_0 = -(w_1 x_{01} + \dots + w_n x_{0n})$. Natijada, giperplane giperkubdan o'tishi kafolatlanadi.

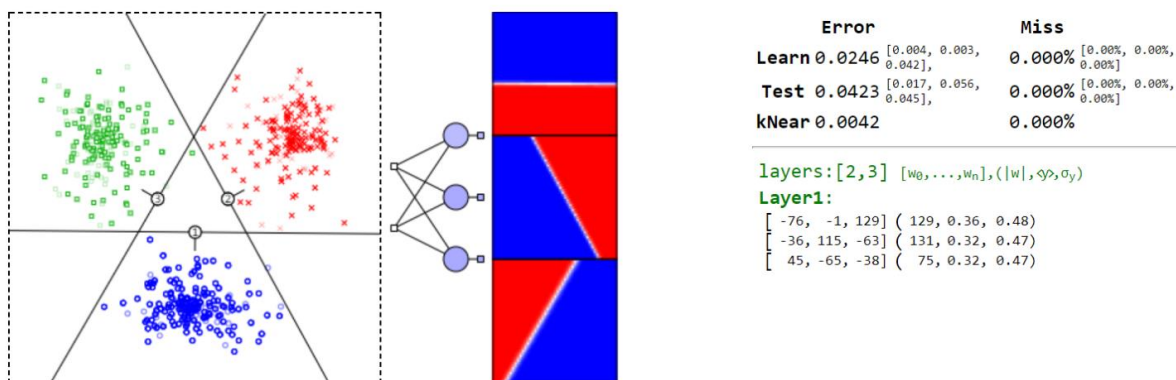
Siljish parametri o'quv jarayonida ham nazorat qilinishi kerak, shuning uchun neyron har doim foydali bo'ladi. Bu erda ikkita mumkin bo'lgan usul mavjud - geometrik va empirik. Empirikda har bir neyronning o'quv ob'ektlari bo'yicha o'rtacha chiqish qiymati hisoblanadi. Agar barcha o'quv misollari tarmog'idan o'tgandan so'ng, ba'zi neyronlarning o'rtacha $\langle y \rangle$ qiymatlari nolga yoki birga yaqin bo'lsa, ular foydasiz hisoblanadi. Bunday holda, ular tasodifiy "silkitilishi" mumkin (ehtimol ω vektorini saqlab, faqat siljish parametrini o'zgartirganda w_0).

Ushbu hujjatdagi barcha misollarda tarmoqlardagi neyronlar $\langle y \rangle$ qiymatiga qarab ranglanadi. Agar $\langle y \rangle = 0,5$ bo'lsa, neyron oq, agar $\langle y \rangle = 0$ bo'lsa - ko'k va $\langle y \rangle = 1$ bo'lsa - qizil. To'yingan ko'k yoki qizil ranglar neyronning foydasizligini anglatadi. Birinchi ikkita misolda yagona neyron juda foydali (oq) bo'lib chiqdi, chunki sinf ob'ektlari chiziqning o'ng va chap tomonida (giperplanni ajratish) teng darajada joylashishi mumkin edi.

Neyronning o'zgaruvchanligi muhim rol o'ynaydi σ_y , uning chiqishining o'rtacha qiymatidan standart og'ishiga teng $\langle y \rangle$. O'zgaruvchanlik qanchalik past bo'lsa, neyron shunchalik kam foydali bo'ladi. Darhaqiqat, bu holda, kirish qiymatlaridan qat'i nazar, u bir xil chiqish qiymatini oladi. Shuning uchun, tarmoqning chiqish qiymatlarini o'zgartirmasdan, bunday neyronni foydasiz neyron kirish bo'lgan neyronlarning parametrlarini mos ravishda o'zgartirish orqali yo'q qilish mumkin.

Bir nechta chiqish

Keling, 3 ta sinfni ko'rib chiqaylik. Bitta neyrondan foydalanish unchalik qulay emas, shuning uchun [hujjatning boshida tasvirlanganidek](#), biz uchta chiqishli tarmoqni [2,3] yaratamiz. Sinflar xususiyat maydonida quyidagi tarzda mahalliyashtirilsin:

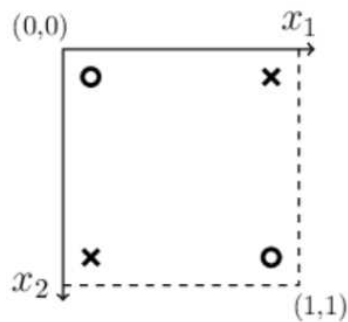


Har bir chiqish neyroni o'zining "sinfini" qolgan ikkitasidan ajratib turadi. Masalan, yuqoridan birinchi neyron (rasmdagi gorizontalk tekislik raqami 1) ko'k doiralar bilan belgilangan ob'ektlarni taniydi va agar ob'ekt $\vec{0}$ vektori yo'naltirilgan tomonda bo'lsa (tekislik yonidagi chiziqcha) bo'lsa, 1 chiqishini hosil qiladi. raqam).

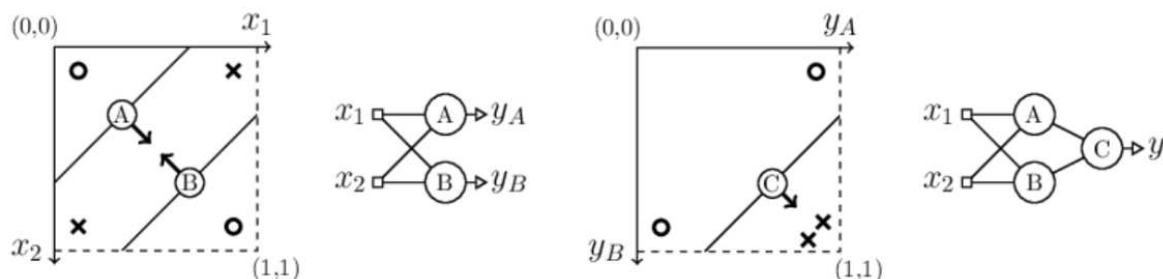
Xuddi shunday, ikkinchi neyron qizil xochlarni, uchinchisi esa yashil kvadratlarni taniydi. Har bir holatda normal vektorlar "o'z" sinflariga yo'naltiriladi. Tarmoqning barcha neyronlari juda ishonchli va juda foydali. Ularning engil ko'kligi normal vektorga qarama-qarshi bo'lgan vektor bo'ylab har doim ikki barobar ko'p ma'lumotlar (ikkita "xorijiy" sinf) mavjudligi bilan bog'liq. Shuning uchun har bir chiqishning o'rtacha qiymati neytral darajadan 0,5 pastga siljiydi.

Yashirin qatlam qachon kerak?

Endi biroz murakkabroq masalaga o'tamiz. Ikki sinf ob'ektlari (doiralar va xochlar) o'ngdagi rasmda bo'lgani kabi, xususiyat maydonining burchaklarida to'plangan bo'lsin. Bu ikki sinfni bitta giperplan (chiziq) bilan ajratib bo'lmaydi. Bu muammo ba'zan bo'linish YOKI(xor) deb ataladi. Ushbu mantiqiy operatsiya haqiqatga (bir) teng bo'ladi, agar dalillardan biri to'g'ri bo'lsa va ikkinchisi noto'g'ri bo'lsa (nol): "Masha Kolyani yoki Vasyani yaxshi ko'radi, lekin ikkalasini ham sevmaydi." Rasmda doiralar bilan belgilangan sinf uchun tarmoq nolga, xochli sinf uchun esa bitta chiqishi kerak. Agar ob'ektlar aynan burchaklarda bo'lsa, $xor(0,0) = xor(1,1) = 0$ va $xor(0,1) = xor(1,0) = 1$.

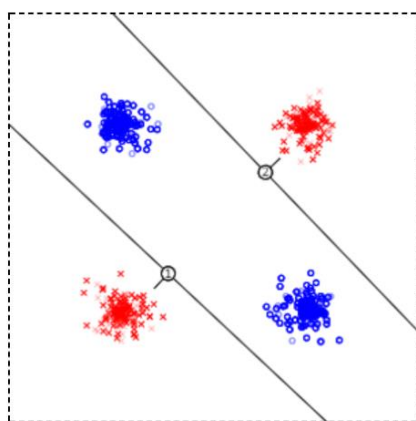


Tasniflashni amalga oshirish uchun bitta yashirin qatlamli neyron tarmoq kerak bo‘ladi [2,2,1]. Quyidagi birinchi grafik (x_1 va x_2 xususiyat o‘qlarida) ikkita giperplanni (A va B chiziqlari) ko‘rsatadi. Ular ikkita yashirin neyronlarga mos keladi A va B . Neyronlarning chiqish qiymatlari ikkinchi grafikda ko‘rsatilgan (y o‘qlari y_A va y_B).



Ikkala xoch ham A va B tekisliklarning normal vektorlari yo‘nalishida yotadi. Shuning uchun ulardan tekisliklargacha bo‘lgan masofa ijobiy bo‘ladi va agar neyronlar o‘zlariga etarlicha ishonch hosil qilsalar, ularning y_A va y_B chiqishlari bitta (tekislikning pastki o‘ng burchagi y_A, y_B).

Tekislikning yuqori chap burchagidagi x_1, x_2 belgilari $(0,0)$ bo‘lgan doira $y_A \sim 0, y_B \sim 1$ neyronlarning chiqishlarini beradi (bu ob‘ekt A tekislikning normal vektoriga qarama-qarshi yotadi va B tekislikning normal vektori bo‘ylab). $(1,1)$ belgilariga ega ikkinchi doira neyron chiqishida $y_A \sim 1, y_B \sim 0$ qiymatlarini beradi. Natijada y_A va y_B xususiyatlarining "deformatsiyalangan" maydoni (ikkinchi grafik) tarmoqning chiqish neyroni bo‘lgan bitta C tekisligiga osongina bo‘linishi mumkin. Agar uning normal vektori ikkinchi grafikda ko‘rsatilgandek yo‘naltirilgan bo‘lsa, u holda ikkala kesma uchun $y \sim 1$, aylanalar uchun esa $y \sim 0$ bo‘ladi. Quyida har biri ikkita klasterga bo‘lingan ikkita ob‘ekt sinfini tan olishga o‘rgatilgan neyron tarmog‘ining haqiqiy namunasi keltirilgan:



	Error	Miss
Learn	0.0001	0.000%
Test	0.0001	0.000%
kNear	0.0000	0.000%

```

layers:[2,2,1] [w0,...,wn],(|w|,σx,σy)
Layer1:
[ -4, -15, 16] ( 22, 0.24, 0.40)
[ -4, 16, -15] ( 22, 0.24, 0.40)
Layer2:
[ -10, 22, 22] ( 31, 0.48, 0.50)

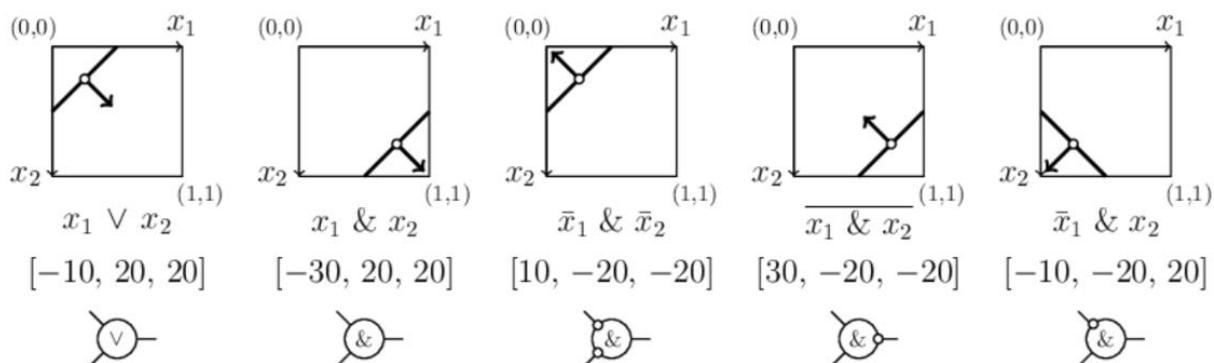
```

3. Tarmoqning har bir qatlami kirish funksiyasi maydonini boshqa fazoga, ehtimol, boshqa o'lchamga aylantiradi. Ushbu chiziqli bo'lmagan transformatsiya sinflar chiqish qatlamida chiziqli ravishda ajratiladigan neyronlarga aylanmaguncha sodir bo'ladi.

Neyronlar mantiqiy element sifatida

Neyronlarning xatti-harakatlarini tahlil qilishga matematik mantiq nuqtai nazaridan yondashish mumkin. Buning uchun xor masalalarining bir sinfiga, masalan, xochlarga e'tibor qaratamiz. Bu sinfning barcha ob'ektlari qanoatlantiradigan mantiqiy shartni yozamiz. Yuqoridagi misolda: "har qanday xoch A tekislik vektori bo'ylab va B tekislik vektori bo'ylab yotadi." Buni $A \& B$ formulasi bilan qisqacha ifodalash mumkin. "C" chiqish neyroni bunday mantiqiy "END" ni amalga oshiradi. Haqiqatan ham, uning tekisligi (1,1) koordinatalari bo'lgan xususiyat maydonida kvadratning pastki o'ng burchagiga bosiladi. Shuning uchun, birga yaqin kirishlar ("A" va "B" neyronlari tomonidan ta'minlangan) uchun ushbu neyronning chiqishi 1 bo'ladi (aniqrog'i, uning qiymati 0,5 dan katta). Shuning uchun, kutilganidek, $1 \& 1 = 1$. Agar kirishlardan kamida bittasi 1 dan farq qilsa, u holda chiqish nolga teng bo'ladi (0,5 dan kam). Bu mantiqiy "END" ni ta'minlovchi neyronning giperplaniyasi koordinatalari (1,1,...,1) bo'lgan giperkub burchagiga bosilgan (uni ixtiyoriy o'lchamdagi fazoda ham to'g'ri keladi). giperkubning qolgan qismi).

Agar tekislik burchakka (1,1) normal yo‘nalishini saqlab, burchakka (0,0) o‘tkazilsa, unda bunday neyron mantiqiy "OR" bo‘ladi. Uning $y=S(x_1, x_2)$ funksiyasi $S(0,0)=0$ va boshqa hollarda 1 ni (birinchi rasmdan past) beradi:



Umumiy holatda mantiqiy “OR” ni amalga oshiruvchi neyron tekisligi n o‘lchamli kubning burchagini $(0,0,...,0)$ kesib tashlaydi va uning normal vektori kattaroq hajmga yo‘naltiriladi. giperkub. Bundan farqli o‘laroq, standart mantiqiy "END" (ikkinchi raqam) kichikroq hajmga nisbatan normal vektorga ega. n ta kirishga ega neyron uchun mantiqiy "END" quyidagi funktsiya bilan tavsiflanadi:

$$y = S(w (x_1 + \dots + x_n + a - n)), \quad y = x_1 \& x_2 \& \dots \& x_n,$$

bu yerda a parametr $0 < a < 1$ oralig‘ida yotgan parametr. Nolga qanchalik yaqin bo‘lsa, tekislik $(1,1,...,1)$ koordinatali burchakka nisbatan kuchliroq bosiladi. Haqiqatan ham, $a=0$ va $x_1 = \dots = x_n = 1$ bo‘lganda, bizda $x_1 + \dots + x_n - n = 0$ bo‘ladi. Ushbu neyron mantiqiy END ni ta‘minlashi uchun u bitta koordinatasi nolga teng bo‘lgan giperkubning "eng yaqin" burchagiga salbiy masofani berishi kerak: $(1,...,1,0,1,...,1)$. Bu $a < 1$ cheklovini beradi. Umumiy omil w normaning uzunligini tavsiflaydi (u qanchalik katta bo‘lsa, neyron shunchalik ishonchli bo‘ladi). [Sigmasimon funktsiya](#) $S(d)$ hujjat boshida berilgan.

Mantiqiy “OR” funksiyasi ham xuddi shunday yoziladi ($0 < a < 1$)

$$y = S(w (x_1 + \dots + x_n - a)), \quad y = x_1 \vee x_2 \vee \dots \vee x_n.$$

Inkorning yana bir mantiqiy funktsiyasi ayirish yordamida amalga oshiriladi. Uni o‘zgaruvchi nomidan yuqori chiziq bilan belgilaymiz. Keyin $x=1-x$ va odatdagidek, $0=1$, $1=0$. Agar neyron kirishlaridan biri manfiy bo‘lsa, uning chiqish funktsiyasi quyidagi ko‘rinishga ega bo‘ladi:

$$y = S(w \cdot (-x_1 + \dots + x_n + a - n + 1)), \quad y = x_1 \& x_2 \& \dots \& x_n.$$

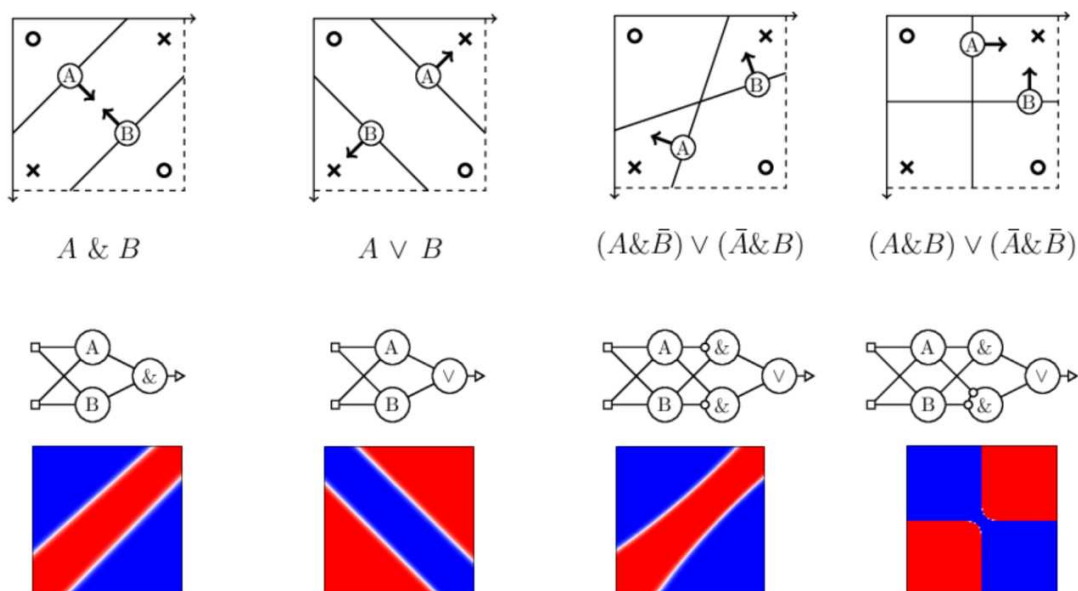
Shunday qilib, normal vektorning tarkibiy qismlaridan biri o'z belgisini o'zgartiradi va neyron tekisligi siljiydi. Yuqorida, uchinchi va to'rtinchi raqamlarda o'zgaruvchilarning turli xil inkorlari ko'rsatilgan. Bu atamalarda mantiqiy OR ni mantiqiy END de-Morgan qoidasidan foydalanishga arziydi:

$$\neg(x_1 \& x_2) = x_1 \vee \neg x_2,$$

bu erda undov belgisi mantiqiy inkorni ko'rsatishning yana bir usuli hisoblanadi.

XOR to'rtta varianti

Mashq sifatida neyron tarmoqlardan foydalangan holda ikkilik xor funksiyasining 4 xil tatbiqini tahlil qilish kerak:

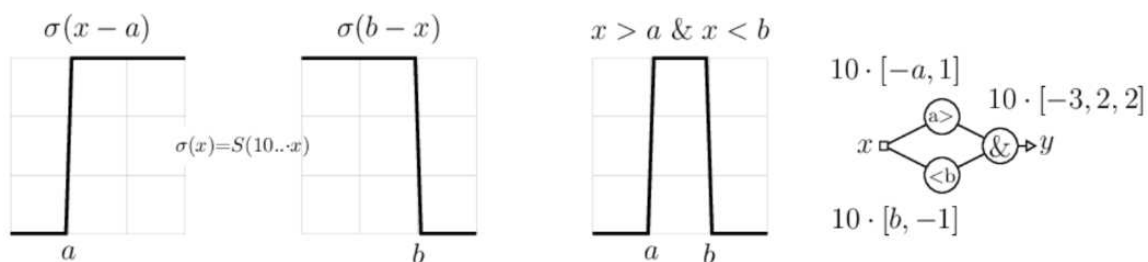


Rasmlarning oxirgi qatori birinchi yashirin qatlamda ajratuvchi tekisliklarga ega, masalan, rasmlarning birinchi qatorida ko'rsatilgandek, o'qitilgan neyron tarmoqlarning chiqish xaritalari.

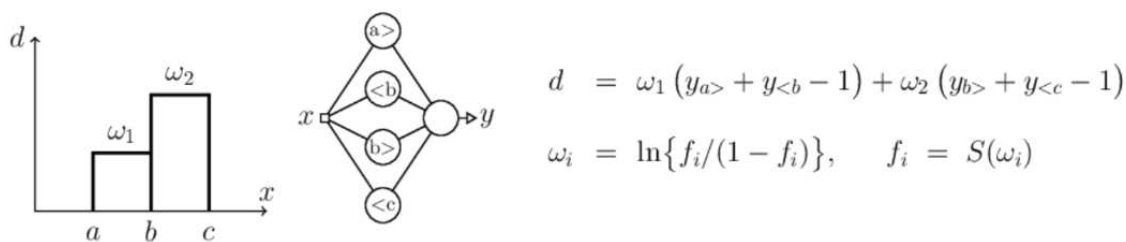
$y=f(x)$ funksiyaning yaqinlashishi

Bitta kirish, bitta chiqish va etarlicha katta yashirin qatlamga ega neyron tarmoqdan foydalanib, har qanday $y=f(x)$ funksiyani taxminiy hisoblashingiz mumkin. Buni isbotlash uchun, avvalo, agar kirish $[a...b]$ diapazonida bo'lsa, 1 ga, aks holda 0 ga teng bo'lgan chiqishni beradigan tarmoq yarataylik.

$\sigma(d) = S(\omega \cdot d)$ sigmasimon funktsiya bo'lsin, uning argumenti $\bar{0}$ katta songa ko'paytiriladi, shunda to'rtburchaklar qadam olinadi. Ikkita shunday qadamdan foydalanib, siz birlik balandligi ustunini yaratishingiz mumkin:



x argumenti va y qiymati uchun $[0...1]$ oraliqda normallashtiramiz. $x=[0...1]$ o'zgarishlar diapazonini ko'p sonli intervallarga bo'ling (teng bo'lishi shart emas). Har bir intervalda funktsiya biroz o'zgarishi kerak. Quyida ikkita shunday interval mavjud:



Yashirin qatlamdagi har bir juft neyron birlik ustunini amalga oshiradi. d ning qiymati $x \in (a, b)$ bo'lsa w_1 va $x \in (b, c)$ bo'lsa w_2 ga teng. Agar chiqish neyroni chiziqli qo'shimcha bo'lsa, u holda $w_i = f_i$ ni qo'yishimiz mumkin, bu erda f_i har bir intervaldagi funktsiya qiymatlari. Agar chiqish neyroni oddiy chiziqli bo'lmagan element bo'lsa, u holda sigmasimon teskari funktsiyadan foydalangan holda w_i og'irliklarini qayta hisoblash kerak (oxirgi formula).

Noravshan mantiq

Neyronlarni mantiqiy elementlar sifatida talqin qilish neyron tarmoqlari arxitekturasini tahlil qilishni yanada soddalashtiradi. Bundan tashqari, neyron tarmoqlar yordamida loyqa mantiq bilan ishlaydigan tizimlarni amalga oshirish uchun ajoyib imkoniyat mavjud. Bunday mantiqda haqiqat (1) va yolg'onga (0) qo'shimcha ravishda 0 dan 1 gacha bo'lgan doimiy haqiqat qiymatlari oralig'i mavjud:

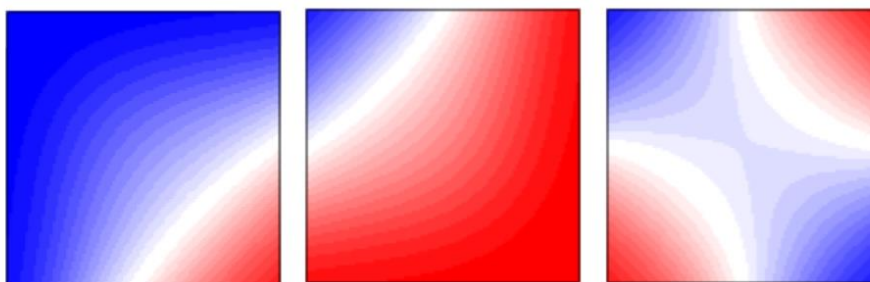
Xalta yoqimli	Qimmat	Qizil rang	Bir do'stimda shunday bor	Mening mashinam qizil	Men tushkunlikka tushdim	Men bu sumkani sotib olaman!
x ₁	x ₂	x ₃	x ₄	x ₅	x ₆	y
1.0	0,0	1.0	1.0	0,0	0,0	0,0
1.0	1.0	1.0	0,0	1.0	0,0	1.0
0,5	0,5	0,5	0,5	0,5	1.0	1.0

Ob'ektlarning xususiyatlari ikkilik qiymatlar bo'lsin ("ha", "yo'q"). Va qora qutining chiqishi y doimiy qiymatlarni qabul qilishi mumkin. Ba'zi hollarda ular 0 ("yo'q") yoki 1 ("ha") ga teng. Biroq, ba'zi kirishlar uchun chiqish, masalan, 0,5 ("ehtimol") bo'lishi mumkin. Bunday misollar bo'yicha o'qitilgan neyron tarmoq ikkala ikkilik tavsiyalarni ishlab chiqaradi va uning noaniqlik darajasini ifodalaydi. Quyida loyqa mantiqda tez-tez ishlatiladigan funksiyalar va ularni neyron tarmoq yordamida yaqinlashtirish (regressiya muammosi!) keltirilgan:

$$\text{and}(x_1, x_2) = x_1 * x_2$$

$$\text{or}(x_1, x_2) = x_1 + x_2 - x_1 * x_2$$

$$\text{xor}(x_1, x_2) = \text{yoki}(\text{va}(x_1, 1 - x_2) \text{ va } (1 - x_1, x_2))$$



Foydali paketlarni o'rnatish

NeuroLab deb nomlangan kuchli neyron tarmoq paketidan foydalanishimiz mumkin. Bu moslashuvchan tarmoq konfiguratsiyasi va Python uchun o'qitish algoritmlari bilan asosiy neyron tarmoq algoritmlari kutubxonasi. Ushbu paketni buyruq satrida quyidagi buyruq yordamida o'rnatishingiz mumkin:

```
pip install NeuroLab
```

Agar siz Anaconda muhitidan foydalanayotgan bo'lsangiz, NeuroLab-ni o'rnatish uchun quyidagi buyruqdan foydalaning:

```
conda install -c labfabulous neurolab
```

Neyron tarmoqlarni qurish

Ushbu bo'limda NeuroLab paketi yordamida Python-da bir nechta neyron tarmoqlarni quramiz.

Perseptron asosidagi klassifikator

Perseptronlar ANNning qurilish bloklari hisoblanadi. Agar Perceptron haqida ko'proq bilmoqchi bo'lsangiz, havolaga o'tishingiz mumkin - [sun'iy_neyron_tarmoq](https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_supervised_learning.htm)

(https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_supervised_learning.htm). Quyida perseptron neyron tarmog'iga asoslangan oddiy klassifikatorni yaratish uchun Python kodining bosqichma-bosqich bajarilishi keltirilgan. Rasmda ko'rsatilganidek, kerakli paketlarni import qiling:

```
import matplotlib.pyplot as plt
import neurolab as nl
```

Kirish qiymatlarini kiriting. E'tibor bering, bu nazorat ostida o'rganishga misol, shuning uchun siz maqsadli qiymatlarni ham ko'rsatishingiz kerak bo'ladi.

```
input = [[0, 0], [0, 1], [1, 0], [1, 1]]
target = [[0], [0], [0], [1]]
```

2 ta kirish va 1 neyronli tarmoq yarating -

```
net = NL.net.newp([[0, 1],[0, 1]], 1)
```

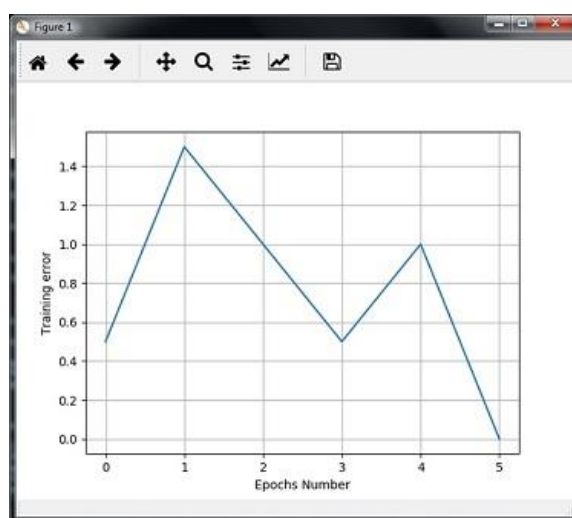
Endi tarmoqni o'rgatamiz. Bu erda trening uchun delta qoidasidan foydalanamiz.

```
error_progress = net.train(input, target, epochs=100, show=10, lr=0.1)
```

Endi natijani tasavvur qiling va grafik chizing -

```
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.grid()
plt.show()
```

Xato darajasidan foydalangan holda o'rganish jarayonini ko'rsatadigan quyidagi grafikni ko'rishingiz mumkin.



Bir qatlamli neyron tarmoqlari

Ushbu misolda biz chiqish ma'lumotlarini ishlab chiqarish uchun kirish ma'lumotlariga ta'sir qiluvchi mustaqil neyronlardan tashkil topgan bir qatlamli neyron tarmog'ini yaratmoqdamiz. Kirish sifatida biz **neural_simple.txt** nomli matn faylidan foydalanayotganimizni unutmang. Rasmda ko'rsatilganidek, foydali paketlarni import qiling.

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
```

Ma'lumotlar to'plamini quyidagi tarzda yuklang.

```
input_data = np.loadtxt("/Users/admin/neural_simple.txt")
```

Quyida biz foydalanmoqchi bo'lgan ma'lumotlar mavjud. E'tibor bering, ushbu ma'lumotlarda dastlabki ikkita ustunlar ob'ektlar va oxirgi ikki ustunlar teglardir.

```
array([[2. , 4. , 0. , 0. ],  
       [1.5, 3.9, 0. , 0. ],  
       [2.2, 4.1, 0. , 0. ],  
       [1.9, 4.7, 0. , 0. ],  
       [5.4, 2.2, 0. , 1. ],  
       [4.3, 7.1, 0. , 1. ],  
       [5.8, 4.9, 0. , 1. ],  
       [6.5, 3.2, 0. , 1. ],  
       [3. , 2. , 1. , 0. ],  
       [2.5, 0.5, 1. , 0. ],  
       [3.5, 2.1, 1. , 0. ],  
       [2.9, 0.3, 1. , 0. ],  
       [6.5, 8.3, 1. , 1. ],  
       [3.2, 6.2, 1. , 1. ],  
       [4.9, 7.8, 1. , 1. ],  
       [2.1, 4.8, 1. , 1. ]])
```

Endi ushbu to'rtta ustunni 2 ta ma'lumot ustuniga va 2 ta tegga bo'ling.

```
data = input_data[:, 0:2]  
labels = input_data[:, 2:]
```

Quyidagi buyruqlar yordamida kiritilgan ma'lumotlarni chizing.

```
plt.figure()  
plt.scatter(data[:,0], data[:,1])  
plt.xlabel('Dimension 1')  
plt.ylabel('Dimension 2')  
plt.title('Input data')
```

Endi bu erda ko'rsatilganidek, har bir o'lcham uchun minimal va maksimal qiymatlarni aniqlang:

```
dim1_min, dim1_max = data[:,0].min(), data[:,0].max()  
dim2_min, dim2_max = data[:,1].min(), data[:,1].max()
```


Keyinchalik, chiqish qatlamidagi neyronlar sonini quyidagicha aniqlang:

```
nn_output_layer = labels.shape[1]
```

Endi bir qavatli neyron tarmoqni aniqlaymiz:

```
dim1 = [dim1_min, dim1_max]
dim2 = [dim2_min, dim2_max]
neural_net =NL.net.newp([dim1, dim2], nn_output_layer)
```

Rasmda ko'rsatilganidek, neyron tarmoqni davrlar soni va o'rganish tezligiga o'rgating.

```
error = neural_net.train(data, labels, epochs = 200, show = 20, goal = 0.01)
```

Endi quyidagi buyruqlar yordamida o'rganish jarayonini tasavvur qiling va rejalashtiring:

```
plt.figure()
plt.plot(error_progress)
plt.xlabel('Number of epochs')
plt.ylabel('Training error')
plt.grid()
plt.show()
plt.title ("Progress learning errors")
plt.grid()
plt.show()
```

Endi yuqoridagi tasniflagichdagi test ma'lumotlar nuqtalaridan foydalaning:

```
print('\nTest Results:')
data_test = [[1.5, 3.2], [3.6, 1.7], [3.6, 5.7],[1.6, 3.9]]
for item in data_test:
    print(item, '-->', neural_net.sim([item])[0])
```

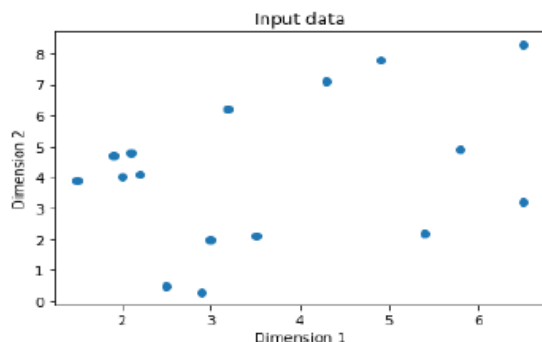
Sinov natijalarini bu yerda ko'rsatilganidek tekshirishingiz mumkin:

```
[1.5, 3.2] --> [1. 0.]
[3.6, 1.7] --> [1. 0.]
```

[3.6, 5.7] --> [1. 1.]

[1.6, 3.9] --> [1. 0.]

Siz hozirgacha muhokama qilingan kodning chiqishi sifatida quyidagi chizmalarni ko‘rishingiz mumkin.



Ko‘p qatlamli neyron tarmoqlar

Ushbu misolda biz o‘quv ma’lumotlaridan asosiy shablonlarini olish uchun bir nechta qatlamlardan iborat ko‘p qatlamli neyron tarmog‘ini yaratamiz. Ushbu ko‘p qatlamli neyron tarmoq regressor sifatida ishlaydi. Biz tenglama $y = 2x^2 + 8$ asosida ba’zi ma’lumotlar nuqtalarini generatsiya qilamiz. Quyida ko‘rsatilganidek, kerakli paketlarni import qiling.

```
import numpy as np
import matplotlib.pyplot as plt
import neurolab as nl
```

Yuqoridagi tenglama asosida ba’zi ma’lumotlar generatsiya qilamiz.

```
min_val = -30
max_val = 30
num_points = 160
x = np.linspace(min_val, max_val, num_points)
y = 2 * np.square(x) + 8
y /= np.linalg.norm(y)
```

Endi ushbu ma’lumotlar to‘plamini quyidagicha o‘zgartiring.

```
data = x.reshape(num_points, 1)
labels = y.reshape(num_points, 1)
```

Quyidagi buyruqlar yordamida kiritilgan ma'lumotlar to'plamini tasavvur qiling va chizing.

```
plt.figure()
plt.scatter(data, labels)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Data-points')
```

birinchi yashirin qatlamda **o'nta** , ikkinchi yashirin qatlamda **oltita** va chiqish qatlamida **bitta neyronli neyrolab** yordamida ikkita yashirin qatlamli neyrotarmoqni yarating.

```
neural_net = nl.net.newff([[min_val, max_val]], [10, 6, 1])
```

Endi gradient o'rganish algoritmidan foydalaning.

```
neural_net.trainf = nl.train.train_gd
```

Endi yuqorida yaratilgan ma'lumotlarni o'rganish uchun tarmoqni o'rgating:

```
error = neural_net.train(data, labels, epochs = 1000, show = 100, goal = 0.01)
```

Endi neyron tarmoqlarni o'quv ma'lumotlar nuqtalarida ishga tushiring -

```
output = neural_net.sim(data)
y_pred = output.reshape(num_points)
```

Endi syujet va vizualizatsiya vazifasi.

```
plt.figure()
plt.plot(error)
plt.xlabel('Number of epochs')
plt.ylabel('Error')
plt.title('Training error progress')
```

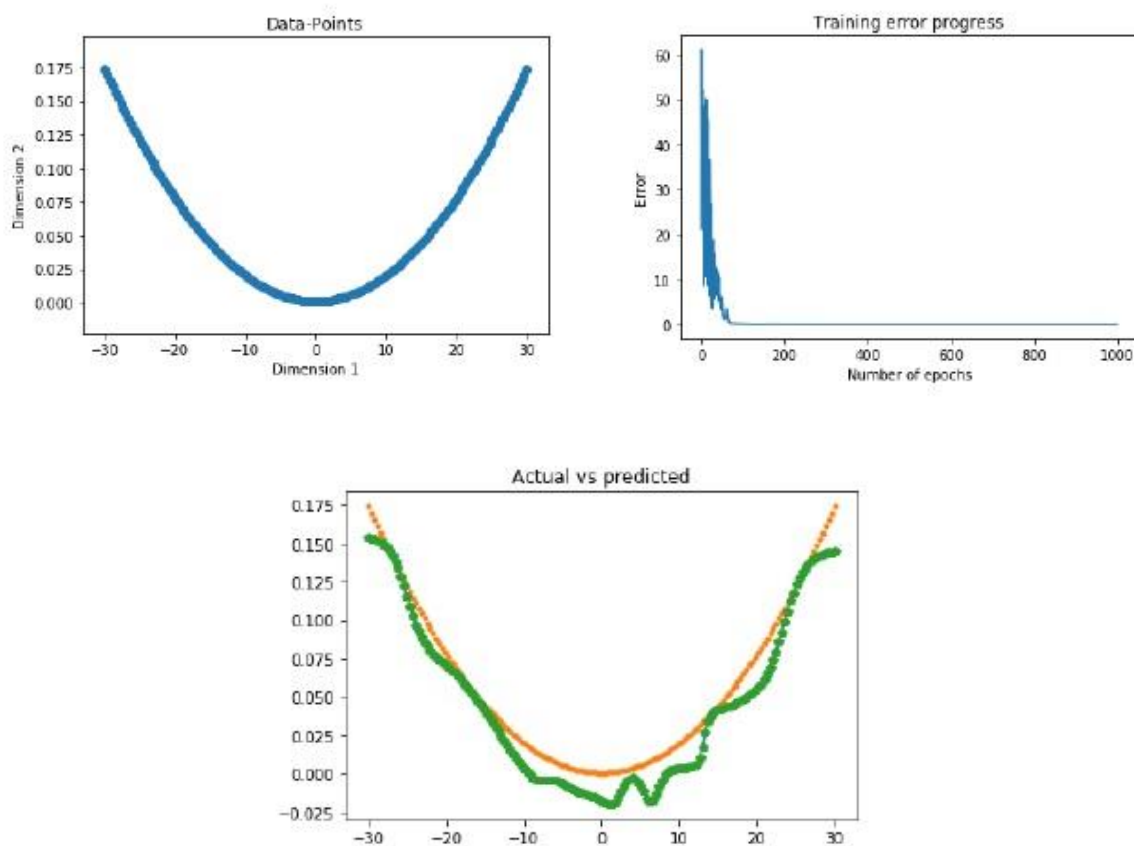
Endi biz haqiqiy va bashorat qilingan natijani chizamiz.

```

x_dense = np.linspace(min_val, max_val, num_points * 2)
y_dense_pred = neural_net.sim(x_dense.reshape(x_dense.size,1)).reshape(x_dense.size)
plt.figure()
plt.plot(x_dense, y_dense_pred, '-', x, y, '.', x, y_pred, 'p')
plt.title('Actual vs predicted')
plt.show()

```

Yuqoridagi buyruqlarni bajarish natijasida siz quyida ko‘rsatilgan grafiklarni kuzatishingiz mumkin.



Kohonen neyron tarmog‘i.

Neyron tarmoqlar sinflaridan birining tuzilishi va ishlash tamoyillarini, shuningdek, orqaga tarqalish usulidan foydalangan holda o‘qitishni batafsil ko‘rib chiqib, biz **Kohonen neyron tarmoqlari** deb ataladigan boshqa sinfga o‘tamiz va keyin, mantiqiy tarzda, biz muammosiz o‘z-o‘zini tartibga soluvchi **Kohonen xaritalariga** o‘tamiz .

Tarmoqlarni orqaga tarqalish usuli yordamida o'rganishdan asosiy farqni darhol ajratib ko'rsatishimiz mumkin, ya'ni ***Kohonen tarmoqlarida o'rganish o'qituvchisiz amalga oshiriladi***. Bu erda g'oya oddiy - siz eslaganingizdek, bizning o'quv majmuamiz kirish qiymatlari to'plamidan va ularning tegishli chiqish qiymatlaridan iborat edi. Shu bilan birga, ushbu ma'lumotlarning barchasi mashg'ulot uchun ishlatilgan, bundan tashqari, chiqish qiymatlari nafaqat ushbu faoliyatda qatnashgan, balki aslida asosiy rol o'ynagan; Bu o'qituvchi bilan o'rganish.

O'qituvchisiz mashg'ulotlarda faqat kirish ma'lumotlari to'plamidan foydalaniladi, chiqish vektorlari umuman yo'q bo'lishi mumkin, ular shunchaki talab qilinmaydi. Bu qanday amalga oshirilganini ushbu maqolada ham nazariy jihatdan, ham aniqroq hayotiy misol yordamida ko'rib chiqamiz. Shu munosabat bilan, hamma narsa klassikaga ko'ra - davom etayotgan hodisalarni yaxshiroq tushunish nazariy ma'lumotlarni birgalikda ko'rib chiqish va ularning ishini amaliy ko'rsatish orqali ta'minlanadi.

Kohonen neyron tarmog'ining tuzilishi.

Kohonen neyron tarmog'i birinchi navbatda klasterlash masalalarini echish, ya'ni ma'lum ob'ektlarni alohida guruhlariga (klasterlarga) birlashtirish uchun ishlatiladi. Ob'ektni ma'lum bir klasterga joylashtirish to'g'risidagi qaror ushbu neyron tarmoqlar sinfining kirish ma'lumotlari bo'lgan uning xususiyatlarining qiymatlari asosida qabul qilinadi .

Misol uchun, vazifa sportchilarni ular o'ynaydigan sport turlari bo'yicha tasniflash bo'lishi mumkin. Belgilar balandlik, vazn, sportchining yuz metrغا yugurish vaqti va boshqalar bo'lishi mumkin. Agar biz barcha sportchilarning "parametrlarini" Kohonen tarmog'i orqali o'tkazsak , unda chiqishda biz ma'lum miqdordagi guruhlarini olamiz. Bunday holda, quyidagi shartlar bajarilishi kerak:

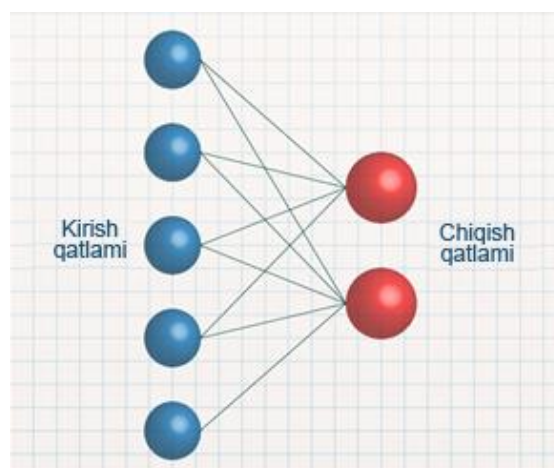
- bir guruhga mansub namunalar qaysidir ma'noda bir-biriga o'xshash bo'lishi kerak
- va bir-biriga o'xshash guruhlar, o'z navbatida, bir-biriga yaqin joylashtirilishi kerak

Bu misolda yengil atletika bilan shug'ullanuvchi barcha sportchilar bir guruhga, basketbolchilar esa boshqa guruhga tushadi. Tarmoqning keyingi mashg'ulotlari bilan yuguruvchilar guruhi yengil atletikachilar guruhidan ajralib turishi mumkin. Va keyin, sanab o'tilgan xususiyatlarning ikkinchisidan so'ng, yuguruvchilar guruhi engil atletikachilar guruhiga yaqin va basketbolchilar guruhidan uzoqda joylashgan bo'lishi kerak. Kelajakda sportchining atributlari ma'lumotlarini tarmoq kiritishiga kiritish orqali biz natijaga erishamiz, ya'ni bu sportchi qaysi guruhga (klasterga) tayinlanishi mumkinligini aniqlaymiz.

Amaliy ma'no shundaki, mavjud ob'ektlarni matematik yoki analitik usullardan foydalangan holda guruhlash har doim ham mumkin emas, ammo Kohonen neyron tarmog'i bizga buni amalga oshirish usulini beradi. Bundan tashqari, mavjud ma'lumotlar qanday bo'lishidan qat'i nazar, mexanizm o'zgarishsiz qoladi.

Ob'ektlar, masalan, tashkilotlar bo'lishi mumkin va belgilar foyda ko'rsatkichlari, aylanma mablag'lar yoki ochiq moliyaviy hisobotlarning boshqa ma'lumotlari bo'lishi mumkin. Kohonen neyron tarmog'i ushbu kompaniyalarni ko'proq yoki kamroq ishonchli kontragentlar guruhlariga bo'lishda yordam berishi mumkin, buning asosida ma'lum bir tashkilot bilan birgalikdagi faoliyatning maqsadga muvofiqligi to'g'risida qaror qabul qilinishi mumkin.

Strukturaviy ravishda tarmoq bitta kirish va bitta chiqish qatlamidan iborat bo'lib, kirish qatlamining barcha neyronlari chiqish qatlamining har bir neyroniga ulangan:



Kirish neyronlari soni ob'ekt xususiyatlari soniga aniq, chiqish neyronlari soni esa klasterlar soniga teng. Tarmoq "g'olib hamma narsani oladi" tamoyili bo'yicha ishlaydi, ya'ni eng katta chiqish signali bitta signalga aylanadi va qolgan chiqishlar nolga qaytariladi. Shunday qilib, biron bir ob'ektning belgilarini kirish sifatida taqdim etish orqali biz chiqishda bir qator nol qiymatlarni olamiz va faqat bitta neyron bittaga teng qiymatga ega bo'ladi. Bu ob'ekt ushbu klasterga tayinlanganligini bildiradi.

Bunday holda, tarmoq kiritishiga berilgan ob'ektlarni koordinatalari xususiyat qiymatlariga teng bo'lgan vektorlar sifatida ko'rib chiqish qulay:

$$x = \{x_1, x_2, x_3 \dots x_n\}$$

Xuddi shunday, chiqish elementlari ham vektorlar sifatida ko'rsatilishi mumkin, ularning koordinatalari kirish qatlamining neyronlaridan ularga keladigan ulanishlarning og'irligiga teng:

$$y_k = \{w_{1k}, w_{2k}, w_{3k} \dots w_{nk}\}$$

Bu erda x - kirish ob'ekti, y_k - k -chi chiqish neyroni, w_{1k} - qolgan og'irliklar uchun xuddi shunday 1-kirish va k -chi chiqish neyronlari o'rtasidagi bog'lanishning og'irlik koeffitsienti.

Kohonen tarmog'ini o'rgatish, boshqa turdagi tarmoqlarga o'xshash, ulanishlarning og'irlik qiymatlarini sozlashni o'z ichiga oladi. Aslida, keling, ushbu jihatga o'tamiz.

Kohonen neyron tarmog'ini o'rgatish.

Va bu erda men darhol ikkita toifaga ajrataman. Gap shundaki, Kohonen neyron tarmog'i tarmoqning boshlang'ich tuzilishi noma'lum bo'lsa ham, qanchalik g'alati tuyulmasin, o'rganishi mumkin. Ammo biz o'rganish ob'ektlarining xususiyatlarining sonini va yakuniy klasterlar sonini allaqachon bilganimizdan boshlaylik. Shunday qilib, tarmoq strukturasi to'liq aniqlangan va yuqoridagilarga mos keladi. O'quv jarayoni shunday ko'rinadi, biz aniq bosqichma-bosqich boramiz.

1-bosqich.

Og'irlik koeffitsientlarini tasodifiy qiymatlar bilan ishga tushiramiz. Bu erda neyron tarmog'i muammolari uchun umumiy bo'lgan bir nuansga to'xtalib o'tishga

arziydi . Va bu shuni anglatadiki, kirish xususiyatlari odatda normallashtiriladi, shunda ularning qiymatlari -1 dan 1 gacha yoki 0 dan 1 gacha bo‘ladi.

Bunga qarab, ulanishlarning dastlabki og‘irligiga cheklovlar kiritiladi:

- Namuna normalizatsiyasi $[-1; 1]$ - og‘irliklarning boshlang‘ich qiymatlari $w_{ij} / \leq n1$, munosabatini qondirishi kerak , bu erda n - xususiyatlar soni, bu ham kirish qatlamidagi neyronlar soni.
- Namuna normalizatsiyasi $[0; 1]$ – va bu holat uchun: $0.5 - n1 \leq w_{ij} \leq 0.5 + n1$.

2-bosqich.

Tarmoq ma’lumotlari o‘quv namunasiidagi qiymatlar, ya’ni o‘rganilayotgan ob’ektlardan birining atributlari bilan ta’minlanadi. Va bu qiymatlar uchun kirish vektoridan har bir klasterning markazlarigacha bo‘lgan [Evklid masofalari alohida](#) hisoblab chiqiladi (jami K klasterlari - chiqish neyronlari bo‘lsin):

$$R_j = \sqrt{\sum_{i=1}^n (x_i - w_{ij})^2}$$

Bu nisbat quyidagilarni o‘z ichiga oladi:

- R_j - joriy kirish vektoridan j-chi chiqish neyronigacha bo‘lgan Evklid masofasi.
- n - xususiyatlar soni, bu ham kirish neyronlarining soni.
- x_i - joriy ob’ektning kirishdagi i atributining qiymati .
- w_{ij} - i-kirish neyroni va j-chi chiqish neyroni orasidagi bog‘lanishning og‘irligi.

3-bosqich.

Ikkinchi bosqichda olingan $R_1, R_2, R_3 \dots R_K$ qiymatlari ularning minimalini topish uchun tahlil qilinadi . R_k minimal bo‘lsin , keyin k chiqish neyroni g‘olib deb e’lon qilinadi va vazn koeffitsientlarini ushbu nisbatga muvofiq sozlash boshlanadi:

$$w_{iknew} = w_{ikold} + \eta(s)(x_i - w_{ikold})$$

Keling, qahramonlarni yana bir bor ko‘rib chiqaylik:

- w_{ikold} - tortish koeffitsientining joriy qiymati.
- w_{iknew} - yangi qiymat.
- x_i - joriy ob’ektning kirishdagi i atributining qiymati .

- $\eta(s)$ - o'rganish tezligi.

neyron tarmoqning o'rganish tezligini ifodalovchi $\eta(s)\eta(s)$ funksiyasiga alohida to'xtalib o'tish kerak . Shu bilan birga, vaqt o'tishi bilan kamayishi o'rnatiladi, bu holda s_s - ta'limning joriy iteratsiyasi. Shunday qilib, g'alaba qozongan neyronning og'irlik koeffitsientlari jarayonning rivojlanishi bilan kamroq va kamroq o'zgaradi, masalan :

$$s\eta(s)=s\eta_0$$

$s=1$ bo'lganda, mashg'ulotning birinchi iteratsiyasi uchun ma'lum bir boshlang'ich qiymat η_0 .

4-bosqich.

Ushbu bosqichlar (2 dan boshlab) mashg'ulotni yakunlash sharti bajarilgunga qadar takrorlanadi. Bunday holat uchun turli xil variantlar bo'lishi mumkin. Misol uchun, bunday omil mashg'ulotlarga sarflangan vaqt bo'lishi mumkin. Ya'ni, bu belgilangan vaqtdan keyin o'quv jarayoni to'xtatiladi va yakunlanadi. O'quv davrlari soni ham belgilanishi mumkin va yana, bu qiymatga erishgandan so'ng, jarayon tugaydi.

Dastlab tarmoq strukturasi noma'lum parametr bo'lgan ikkinchi holatga o'tamiz. Ya'ni, chiqish neyronlari yo'q, keyin biz quyidagi mexanizmga muvofiq harakat qilamiz.

R_0 Yevklid masofasining ma'lum bir qiymati bo'lib , u g'olib neyrondan kirish vektorigacha bo'lgan maksimal ruxsat etilgan masofaga to'g'ri keladi.

Xuddi shunday, elementlardan biri tarmoqning kirishiga, ya'ni o'quv namunasidan kirish vektorining atributlari-koordinatalariga beriladi. Yuqorida tavsiflangan holatda bo'lgani kabi , ushbu vektordan mavjud chiqish neyronlarining har biriga Evklid masofalari hisoblab chiqiladi:

$$R_j=i\sum_n(x_i-w_{ij})^2$$

Olingan qiymatlarning minimali topiladi va g'olib chiqish neyroni aniqlanadi. Bu R_{kl} qiymati va shunga mos ravishda g'olib neyron kl bo'lsin .

Agar $R_{kl}\leq R_0$ munosabati qanoatlansa, g'olib neyronning ulanish og'irliklari (kl) bir xil formula bo'yicha o'rnatiladi :

$$W_{iknew} = W_{ikold} + \eta(s)(x_i - W_{ikold})$$

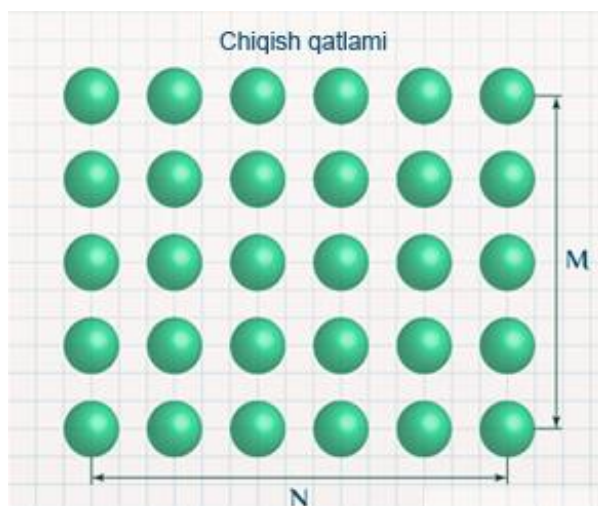
Farqi qarama-qarshi holatda, ya'ni $R_{kl} > R_0$ bo'lganda kuzatiladi. Bunday holda, neyron tarmoqqa yangi chiqish neyroni qo'shiladi. Va uning og'irlik koeffitsientlari biz tarmoq kiritishiga kiritgan joriy kirish vektoriga teng ravishda o'rnatiladi.

Bundan mantiqan kelib chiqadiki, mashg'ulotning birinchi iteratsiyasida har qanday holatda ham chiqish neyroni qo'shiladi, chunki dastlab ular yo'q va Evklid masofalarini hisoblash uchun hech narsa yo'q. Xo'sh, keyin jarayon tsiklik takrorlanadi. Bularning barchasini aniq misol bilan tasavvur qilamiz, ammo hozircha keyingi bo'limga o'tamiz.

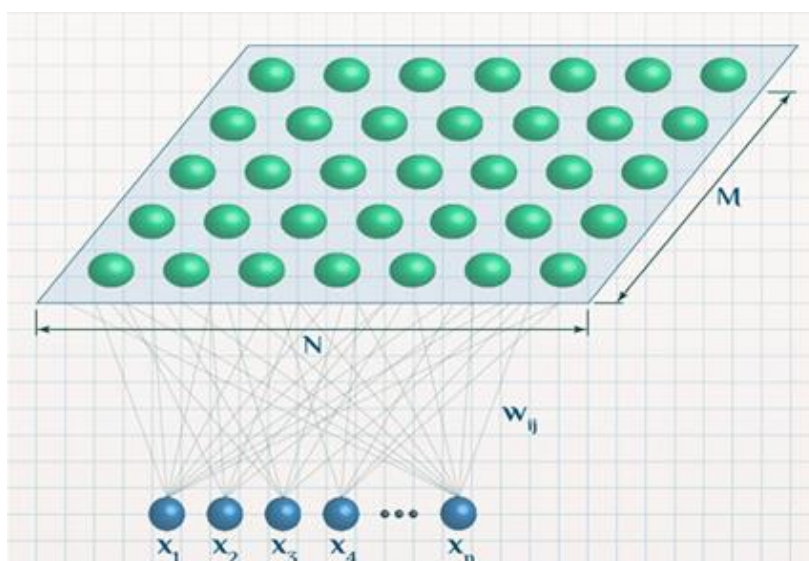
Kohonen xaritalari.

Kohonen xaritasi (**SOM** - Self-Organizing Map - O'z-o'zini tashkil qilish xaritasi) Kohonen neyron tarmoqlarining versiyasidir, shuning uchun biz muhokama qilgan hamma narsa hali ham amal qiladi. Muhim nuqta shundaki, Kohonen xaritalaridan foydalanish sizning ishingiz natijalarini tushunarli xaritalar (odatda ikki o'lchovli) ko'rinishida ko'rsatish imkonini beradi. Va allaqachon vizual grafik talqin asosida siz olingan ma'lumotlarni qulay tahlil qilishingiz, bog'liqliklarni qidirishingiz, prognoz va hk.

Shunday qilib, Kohonen xaritalari asl n o'lchovli fazoni past o'lchamli fazoga aylantirish imkonini beradi, bu bilan o'zaro aloqa qilish ancha oson. O'z-o'zini tashkil etuvchi xaritaning chiqish neyronlari ko'pincha ikki o'lchovli to'rda joylashtirilgan holda tasvirlangan:



Ammo shu bilan birga, ularning har biri hali ham, birinchi navbatda, uning og'irlik koeffitsientlarining qiymatlari bilan tavsiflanadi:



Ya'ni, mohiyat va mantiq biz muhokama qilganimizdek bir xil bo'lib qoladi, shunchaki chiqish neyronlarini tartibli joylashtirish shaklida bir nuance qo'shiladi, bu sizga tarmoq natijalarini aniq ko'rish imkonini beradi. Ko'rinib turibdiki, n o'lchovli bo'shliq, uning elementlari, aslida, chiqish qatlamining neyronlari, ularning koordinatalari - $y_k = \{w_{1k}, w_{2k}, w_{3k} \dots w_{nk}\}$, ekranda yoki qog'ozda ko'rsatish mumkin emas. Bu holatda vaziyat boshqacha, neyronlar an'anaviy tarzda grafik tarzda tasvirlanishi mumkin bo'lgan 2D panjara shaklida joylashtirilgan. Aslida, bu juda oqlangan yechim va chiroyli g'oya.

Kohonen neyron tarmoqlarining umumiy holati bilan bir xil, ya'ni biz allaqachon batafsil muhokama qilgan bosqichlarga muvofiq davom etadi, ammo bitta muhim farq bor.

Bu vazn qiymatlari nafaqat g'olib neyron uchun, balki ba'zi qo'shni neyronlar uchun ham yangilanganligidan iborat. Qaysi qo'shnilar aniq qo'shnilik funktsiyasi bilan aniqlanadi $h_{kj}(t)$, bu chiqish neyronlarining "mahalla o'lchovi" ni aniqlaydi. Eng ko'p ishlatiladigan variant bu funktsiya Gauss funktsiyasini ifodalaganida:

$$h_{kj}(s) = e^{2\sigma^2(s) - dkj^2}$$

- k - g'olib neyronning soni.
- j - boshqa chiqish neyroni.
- s - o'quv jarayonining joriy iteratsiyasi.

- $\sigma(s)$ - ta'lim omili bo'lib, uning qiymati s ortishi bilan kamayadi .
- d_{kj} - k va j neyronlari orasidagi Evklid masofasi .

Funktsiya vaqt o'tishi bilan ham, d_{kj} ortishi bilan ham kamayadi . Va vazn koeffitsientlarini yangilash quyidagi formula bo'yicha amalga oshiriladi:

$$w_{ijnew} = w_{ijold} + \eta(s) h_{kj}(s) (x_i - w_{ijold})$$

Biz shuni ta'kidlaymizki, nafaqat to'qqizinchi g'olib neyronning og'irligi , balki boshqa chiqish neyronlarining og'irliklari ham yangilanadi . Bundan tashqari, ma'lum bir neyronning og'irligi qay darajada sozlanishini aniqlaydigan yuqorida aytib o'tilgan qo'shnichilik funktsiyasi qo'shiladi . Agar j neyron g'olib bo'lgan k neyronidan juda uzoqda bo'lsa , $h_{kj}(s)$ qiymati ham kichik bo'ladi.

j ga teng k ni olaylik . Qo'shnichilik funktsiyasi quyidagicha bo'ladi:

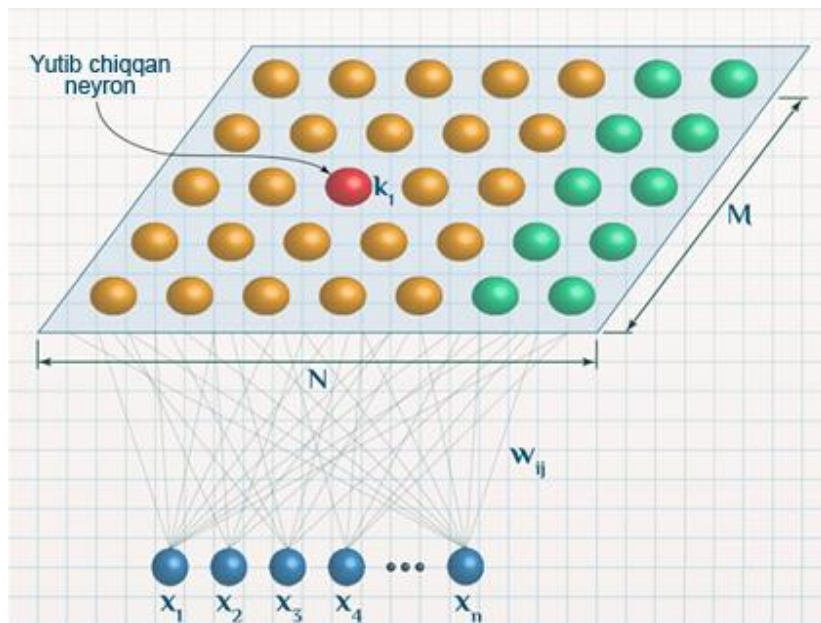
$$h_{kk}(s) = e^{2\sigma 2(s) - d_{kk}^2} = e^{2\sigma 2(s) 0} = 1$$

Shubhasiz, d_{kk} - k -neyrondan o'ziga bo'lgan masofa - 0 ga teng. Keyin:

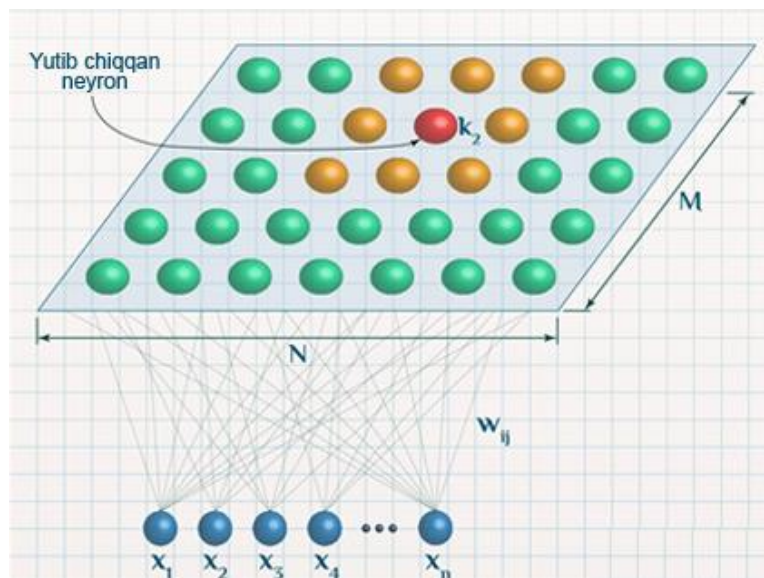
$$w_{iknew} = w_{ikold} + \eta(s) h_{kk}(s) (x_i - w_{ikold}) = w_{ikold} + \eta(s) (x_i - w_{ikold})$$

Va biz ko'rib chiqqan birinchi formulani olamiz, bu mutlaqo mantiqiy. Xulosa qilib aytamizki, o'quv jarayonidagi farq aniq qiymatlar nafaqat g'olib neyron uchun, balki uning qo'shnilari uchun ham yangilanganligidadir. Mexanizm deyarli o'zgarmaydi.

Shunday qilib, davom etamiz ... Biz ob'ektlardan birining ma'lumotlarini kirish sifatida oziqlantiramiz, g'olib neyronni aniqlaymiz, so'ngra uning va uning qo'shnilari uchun tortish koeffitsientlarini yangilaymiz:

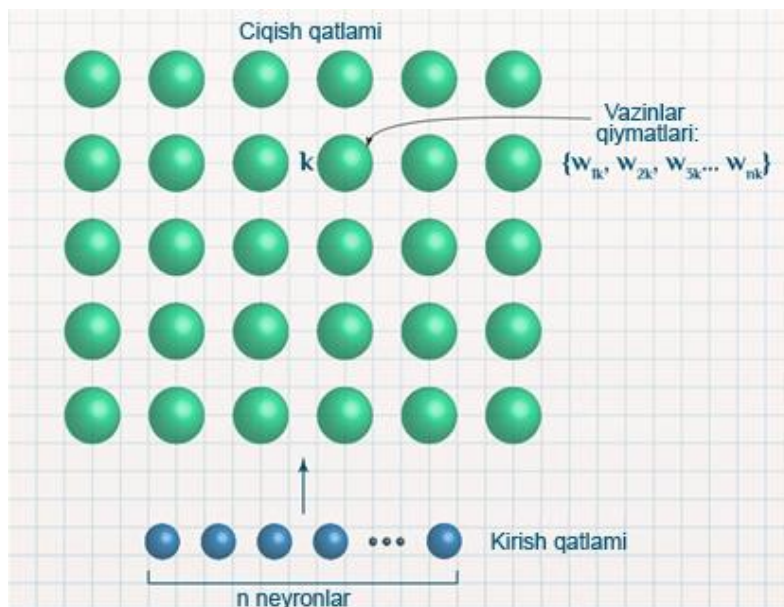


Qo'shnichilik funksiyasiga muvofiq, o'quv jarayoni davom etar ekan, qo'shnilar soni tobora kamayib boradi ("uzoq" qo'shnilar uchun mahalla funksiyasining qiymatlari kichik bo'ladi, soddaligi uchun biz ularni e'tiborsiz qoldiramiz):

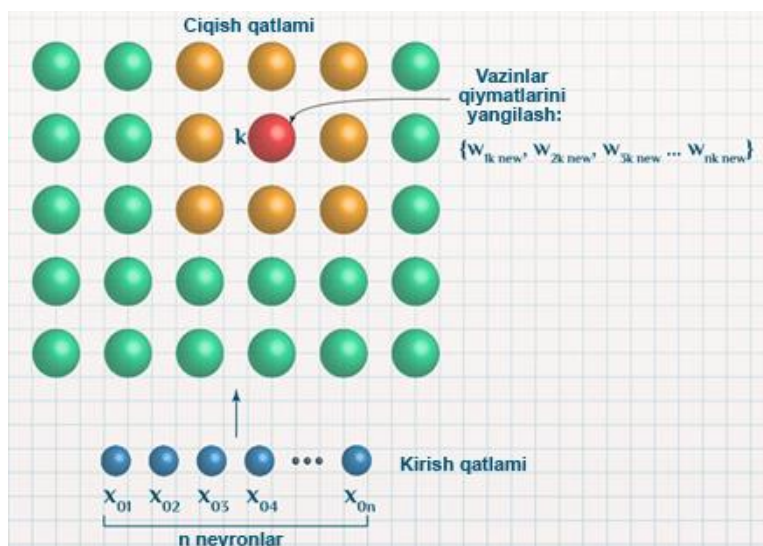


Tez-tez ishlatiladigan terminologiya shundan iboratki, og'irliklar sozlanganda, g'olib neyron va uning qo'shnilari ma'lum bir iteratsiyada kiritilgan ob'ektga yaqinlashadi. Va bu haqiqatan ham jarayonni mukammal tasvirlaydi, ammo so'zlar aniq nimani anglatishini aniq tushunmaydi. Aksincha, agar siz uni o'rganmasangiz, bu hatto chalg'itadi. Aniq bo'lmagan va savol tug'diradigan narsalarni istisno qilish uchun biz batafsilroq tahlil qilamiz.

Kohonen xaritasining versiyasini ko‘rib chiqamiz va ob‘ektlardan birini kirish sifatida yuboramiz:



Va asosiysi shundaki, bizda ikkita bo‘shliq bor - n o‘lchovli bo‘shliqda k neyronning koordinatalari $\{w_{1k}, w_{2k}, w_{3k}, \dots, w_{nk}\}$ ga teng . Bundan tashqari, ikkita o‘lchovli bo‘shliq, uning ichida biz grafik ravishda chiqish neyronlarini joylashtirdik. Taqdim etilgan ob‘ekt atributlarining qiymatlari $x_0 = \{x_{01}, x_{02}, x_{03}, \dots, x_{0n}\}$ bo‘lsin .



Aytaylik, g‘alaba qozongan neyron k neyron bo‘lib chiqadi va bu bosqichda yangilanadigan qo‘shnilari sariq rang bilan belgilangan neyronlardir.

k neyronining koordinatalari :

$$y_k = \{w_{1k}, w_{2k}, w_{3k}, \dots, w_{nk}\}$$

Trening iteratsiyasidan keyin:

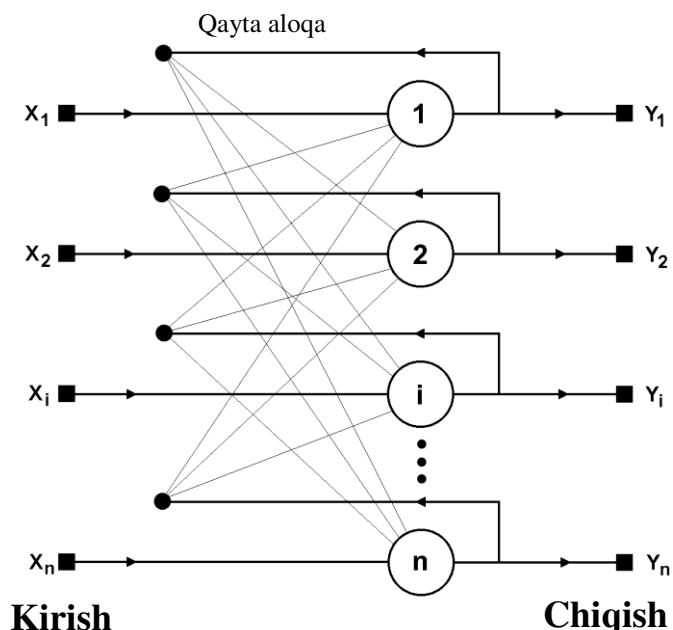
$$y_{knew} = \{ w1_{knew}, w2_{knew}, w3_{knew} \dots w_{nknew} \}$$

Natijada, u haqiqatan ham koordinatalari (xususiyatlar qiymatlari) $\{x_{01}, x_{02}, x_{03} \dots x_{0n}\}$. Shunchaki, yaqinlik bu erda aniq mavjud n o'lchovli makon kontekstida nazarda tutilgan. Ya'ni vektor $y_{knew} = \{ w1_{knew}, w2_{knew}, w3_{knew} \dots w_{nknew} \}$ Ikki o'lchovli shaklda chiqish neyronlari bir xil joylashuvida qoladi.

Amalda, men bu masala bo'yicha tez-tez chalkashliklarga duch keldim, shuning uchun bu vaziyatni batafsilroq ko'rib chiqishga qaror qilindi. Ayni paytda biz davom etamiz. Va mavzuning mohiyatiga eng katta kirish uchun, albatta, biz bir qator yorqin misollarni ko'rib chiqamiz.

Xopfild va Xemming neyron tarmoqlari

Sun'iy neyron tarmoqlarning (SN) turli xil konfiguratsiyalari orasida shundaylari borki, ular o'rganish printsipiga ko'ra tasniflanganda, qat'iy aytganda, na nazorat ostida o'rganish [1], na nazoratsiz o'rganish [2] mos kelmaydi. Bunday tarmoqlarda sinapslarning og'irlik koeffitsientlari qayta ishlangan ma'lumotlar to'g'risidagi ma'lumotlarga asoslanib, tarmoq ishlay boshlagunga qadar faqat bir marta hisoblab chiqiladi va tarmoqning barcha mashg'ulotlari shu hisob-kitobga to'g'ri keladi. Bir tomondan, aprior ma'lumotlarning taqdimotini o'qituvchining yordami deb hisoblash mumkin, ammo boshqa tomondan, tarmoq haqiqiy ma'lumotlar kiritilishidan oldin namunalarni oddiygina eslab qoladi va uning xatti-harakatlarini o'zgartira olmaydi, shuning uchun qayta aloqa havolasi haqida gapiring. "dunyo" bilan "(o'qituvchi) kerak emas. Shunga o'xshash operatsion mantiqqa ega bo'lgan tarmoqlardan eng mashhurlari Xopfild tarmog'i va Xemming tarmog'i bo'lib, ular odatda assotsiativ xotirani tashkil qilish uchun ishlatiladi. Keyinchalik ular haqida gaplashamiz.



Rasm-1. Xopfild tarmog'ining tuzilish sxemasi

Xopfild tarmog'ining blok diagrammasi 1-rasmda ko'rsatilgan. U bitta qatlamli neyronlardan iborat bo'lib, ularning soni bir vaqtning o'zida tarmoqning kirish va chiqishlari soniga teng. Har bir neyron boshqa barcha neyronlar bilan sinapslar orqali bog'langan, shuningdek, signal kiritiladigan bitta kirish sinapsiga ega. Chiqish signallari, odatdagidek, aksonlarda hosil bo'ladi.

Ushbu tarmoq tomonidan assotsiativ xotira sifatida hal qilinadigan muammo odatda quyidagicha shakllantiriladi. Ikkilik signallarning ma'lum bir to'plami (tasvirlar, ovozli raqamlashtirish, muayyan ob'ektlarni yoki jarayonlarning xususiyatlarini tavsiflovchi boshqa ma'lumotlar) ma'lum bo'lib, ular namunali hisoblanadi. Tarmoq uning kirishiga qo'llaniladigan o'zboshimchalik bilan ideal bo'lmagan signaldan mos keladigan namunani (agar mavjud bo'lsa) tanlashi ("eslab qolish") yoki kirish ma'lumotlari hech qanday ma'lumotga mos kelmaydigan "xulosa berish" imkoniyatiga ega bo'lishi kerak. namunalardan. Umuman olganda, har qanday signal vektor $\mathbf{X} = \{x_i : i=0 \dots n-1\}$, n - tarmoqdagi neyronlar soni va kirish va chiqish vektorlarining o'lchami bilan tavsiflanishi mumkin. Har bir x_i elementi +1 yoki -1 ga teng. K - nchi namunani tavsiflovchi vektorni $\mathbf{X}^k$ va uning komponentlarini mos ravishda x_i^k , $k=0 \dots m-1$, m - namunalar soni bilan

belgilaymiz . Tarmoq tanilganda unga taqdim etilgan ma'lumotlarga asoslangan hech qanday namunani (yoki "eslab qolmaydi"), uning chiqishlarida aynan uni o'z ichiga oladi, ya'ni $Y = X \cdot k$ bu erda Y tarmoq chiqish qiymatlarining vektori: $Y = \{y_i : i = 0, \dots, n-1\}$. Aks holda, chiqish vektori hech qanday mos yozuvlar vektoriga to'g'ri kelmaydi.

Agar, masalan, signallar qandaydir tasvirlar bo'lsa, u holda tarmoq chiqishidagi ma'lumotlarni grafik shaklda ko'rsatish orqali siz namunali rasmlardan biriga (muvaffaqiyatli bo'lsa) to'liq mos keladigan rasmni ko'rishingiz mumkin bo'ladi. tarmoqning "bepul improvizatsiyasi" (muvaffaqiyatsiz bo'lsa).

Tarmoqni ishga tushirish bosqichida sinapslarning og'irlik koeffitsientlari quyidagicha o'rnatiladi [3][4]:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, & i \neq j \\ 0, & i = j \end{cases} \quad (1)$$

presinaptik va postsinaptik neyronlarning indeksleri ; k-tanlama vektorining x_i^k , x_j^k i- va j- elementlari .

Tarmoqning ishlash algoritmi quyidagicha (p iteratsiya raqami):

1. Tarmoq kirishlariga noma'lum signal beriladi. Aslida, uni kiritish to'g'ridan-to'g'ri akson qiymatlarini o'rnatish orqali amalga oshiriladi:

$$y_i(0) = x_i, i = 0 \dots n-1, \quad (2)$$

shuning uchun kirish sinapslari tarmog'ining diagrammasida aniq belgilanish mutlaqo an'anaviydir. y_i ning o'ng tomonidagi qavs ichidagi nol tarmoqning ishlash siklidagi nol iteratsiyasini bildiradi.

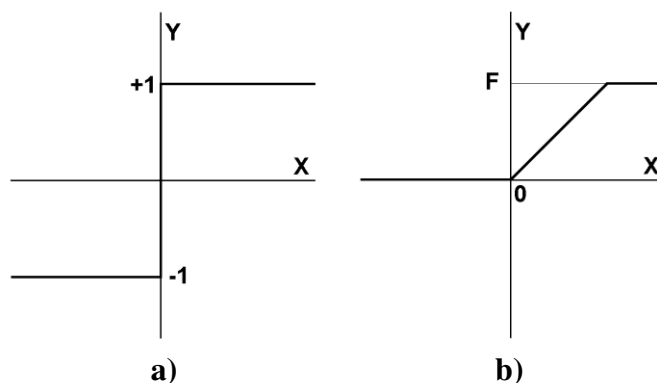
2. Neyronlarning yangi holati hisoblanadi

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p), \quad j=0 \dots n-1 \quad (3)$$

va aksonlarning yangi ma'nolari

$$y_j(p+1) = f[s_j(p+1)] \quad (4)$$

Bu erda f - sakrash ko‘rinishidagi faollashtirish funktsiyasi, 2a-rasmda



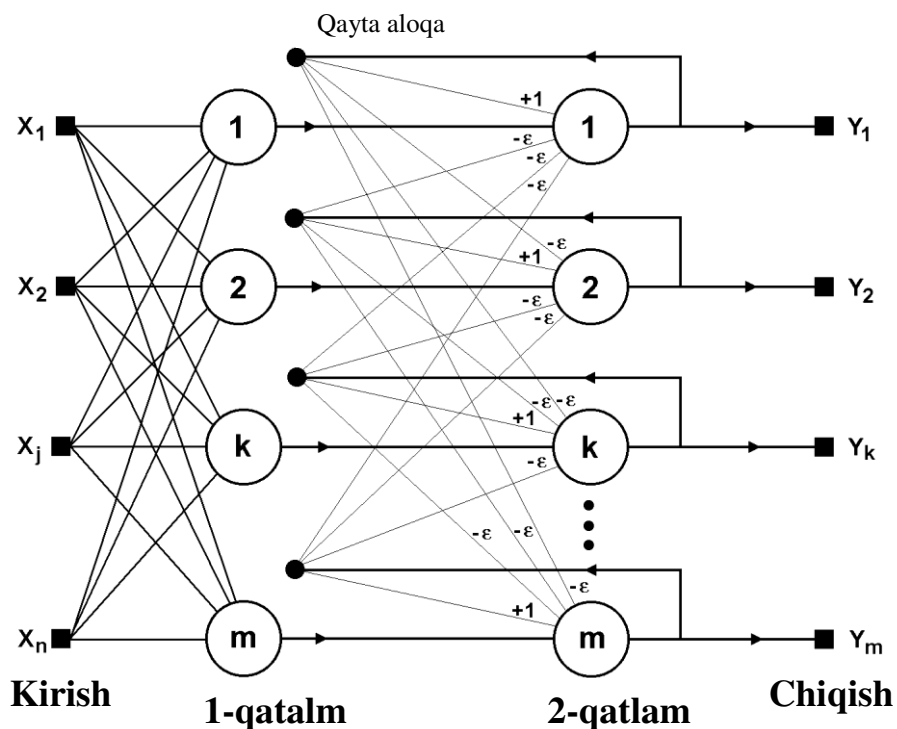
Rasm-2. Faollashtirish funktsiyasi

ko‘rsatilgan.

3. Oxirgi iteratsiyada aksonlarning chiqish qiymatlari o‘zgarganligini tekshiring. Ha bo‘lsa, 2-bandga o‘ting, aks holda (agar chiqishlar barqarorlashgan bo‘lsa) $\sqrt{}$ tugaydi. Bunday holda, chiqish vektori kirish ma’lumotlariga eng mos keladigan namunadir.

Yuqorida aytib o‘tilganidek, ba’zida tarmoq tanib olishni amalga oshira olmaydi va mavjud bo‘lmagan tasvirni chiqaradi. Bu cheklangan tarmoq imkoniyatlari muammosi bilan bog‘liq. Xopfild tarmog‘i uchun yodlangan tasvirlar soni m taxminan $0,15 \cdot n$ ga teng qiymatdan oshmasligi kerak. Bundan tashqari, agar ikkita A va B tasvirlari juda o‘xshash bo‘lsa, ular tarmoqdagi o‘zaro bog‘lanishlarni keltirib chiqarishi mumkin, ya’ni tarmoq kirishlarida A vektorini ko‘rsatish uning chiqishlarida B vektorining paydo bo‘lishiga olib keladi va aksincha.

Tarmoq namunani aniq ishlab chiqarishga hojat qolmaganda, ya'ni namuna raqamini olish uchun etarli bo'lsa, assotsiativ xotira Hamming tarmog'i tomonidan muvaffaqiyatli amalga oshiriladi. Ushbu tarmoq Xopfield tarmog'i bilan solishtirganda, xotira narxining pastligi va hisoblash hajmi bilan tavsiflanadi, bu uning tuzilishidan ko'rinib turadi (3-rasm).



Rasm-3. Hamming tarmog'ining blok-sxemasi

Tarmoq ikki qatlamdan iborat. Birinchi va ikkinchi qatlamlarning har birida m neyron mavjud, bu erda m - namunalar soni. Birinchi qavatning neyronlarining har birida tarmoq kirishlariga ulangan n ta sinaps mavjud (hayoliy nol qavatni tashkil qiladi). Ikkinchi qavatning neyronlari tormozlovchi (salbiy qayta aloqa) sinaptik aloqalar bilan o'zaro bog'langan. Har bir neyron uchun yagona ijobiy qayta aloqa sinapsi o'z aksoniga ulangan.

Tarmoqning g'oyasi sinovdan o'tgan tasvirdan barcha namunalarigacha bo'lgan Hamming masofasini topishdir. Hamming masofasi - bu ikkita ikkilik vektordagi turli bitlar soni. Tarmoq noma'lum kirish signaligacha minimal Hamming masofasi bilan naqshni tanlashi kerak, natijada ushbu naqshga mos keladigan faqat bitta tarmoq chiqishi faollashadi.

Ishga tushirish bosqichida birinchi qatlamning og'irlik koeffitsientlari va faollashtirish funktsiyasining chegarasi quyidagi qiymatlar bilan belgilanadi:

$$w_{ik} = \frac{x_i^k}{2}, i=0...n-1, k=0...m-1 \quad (5)$$

$$T_k = n/2, k = 0...m-1 \quad (6)$$

Bu yerda x_i^k i- k- tanlamaning elementi.

Ikkinchi qatlamdagi inhibitor sinapslarning og'irlik koeffitsientlari ma'lum bir qiymatga teng qabul qilinadi $0 < \varepsilon < 1/m$. O'z aksoniga ulangan neyron sinapsi +1 og'irligiga ega.

Hamming tarmog'ining ishlash algoritmi quyidagicha:

1. Tarmoq kirishlariga noma'lum $\mathbf{X} = \{x_i : i=0...n-1\}$ vektor beriladi, undan birinchi qatlam neyronlarining holati hisoblab chiqiladi (qavs ichidagi yuqori chiziq qatlam raqamini bildiradi.):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j, j=0...m-1 \quad (7)$$

Shundan so'ng, ikkinchi qatlam aksonlarining qiymatlari olingan qiymatlar bilan ishga tushiriladi:

$$y_j^{(2)} = y_j^{(1)}, j = 0...m-1 \quad (8)$$

2. Ikkinchi qavat neyronlarining yangi holatlarini hisoblang:

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), k \neq j, j = 0...m-1 \quad (9)$$

va ularning aksonlarining ma'nolari:

$$y_j^{(2)}(p+1) = f[s_j^{(2)}(p+1)], j = 0...m-1 \quad (10)$$

F faollashtirish funktsiyasi chegara ko'rinishiga ega (2b-rasm) va F qiymati etarlicha katta bo'lishi kerak, shunda argumentning har qanday mumkin bo'lgan qiymatlari to'yinganlikka olib kelmasligi kerak.

3. Ikkinchi qatlam neyronlarining chiqishlari oxirgi iteratsiya davomida o'zgarganligini tekshiring. Ha bo'lsa, 2-bosqichga o'ting. Aks holda, yakunlang.

Algoritmni baholashdan ko‘rinib turibdiki, birinchi qatlamning roli juda shartli: 1-bosqichda uning og‘irlik koeffitsientlarining qiymatlaridan bir marta foydalangan holda, tarmoq endi unga kira olmaydi, shuning uchun birinchi qatlamni butunlay chiqarib tashlash mumkin. tarmoqdan (og‘irlik koeffitsientlari matritsasi bilan almashtiriladi), bu quyida tavsiflangan uning maxsus amalga oshirilishida sodir bo‘lgan narsadir.

NeuronHN , LayerHN va NetHN maxsus sinflari to‘plami asosida qurilgan - qatorning oldingi maqolalarida muhokama qilingan sinflardan olingan [1][2]. Sinflar tavsifi 1-listingda keltirilgan. Barcha funksiyalarning amalga oshirilishi NEURO_HN faylida (2-ro‘yxat) mavjud. NeuronHN va LayerHN sinflari asosiy sinflardan ko‘pgina usullarni meros qilib oladi.

NetHN klassi quyidagi elementlarni belgilaydi:

Nin va Nout $\sqrt{\text{mos ravishda}}$, ma’lumotlar bilan kirish vektorining o‘lchami va namunalar soni;

dx va dy $\sqrt{\text{ikkita koordinatada kiritilgan tasvirning o‘lchamlari}}$ (uch o‘lchovli tasvirlar uchun siz dz o‘zgaruvchisini qo‘shishingiz kerak), dx * dy Nin ga teng bo‘lishi kerak , bu o‘zgaruvchilar ma’lumotlarni yuklash funktsiyasi tomonidan ishlatiladi LoadNextPattern faylidan ;

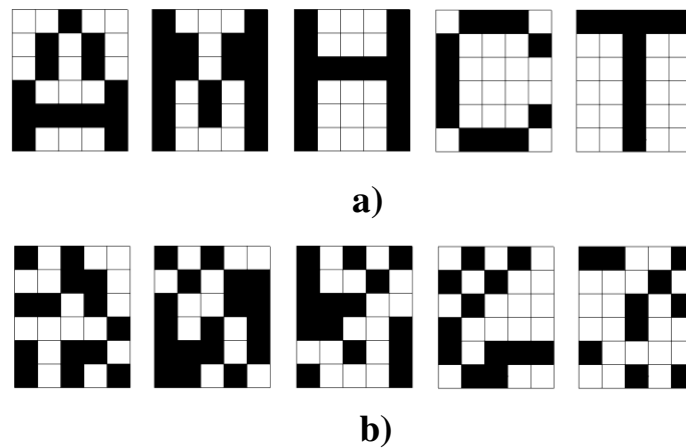
Chiqish qatlamining DX va DY $\sqrt{\text{o‘lchamlari}}$ (Show - ko‘rsatish funktsiyasidan foydalangan holda faqat chiqish qatlamining ko‘rinishiga ta’sir qiladi); har ikkala o‘lcham juftligi SetDxDy funktsiyasi tomonidan o‘rnatiladi ;

Cass $\sqrt{\text{SetClasses}}$ funktsiyasi bilan to‘ldirilgan namunalar haqidagi ma’lumotlarga ega massiv, bu funksiya tarmoqni umumiy ishga tushirishni amalga oshiradi, bu esa namuna ma’lumotlarini saqlashgacha boradi.

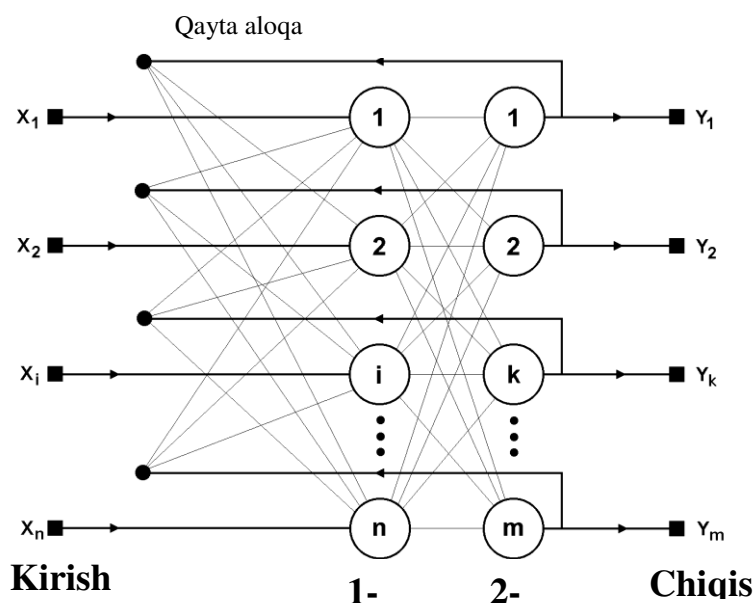
Initialize usuli tekshirilayotgan ma’lumotlar darajasida qo‘shimcha ishga tushirishni amalga oshiradi (algoritmning 1-bosqichi). Cycle usuli 2-bosqichni amalga oshiradi va IsConverged usuli neyronlarning holati barqarorlashganligini tekshiradi (3-bosqich).

Global funksiyalardan $\sqrt{\text{SetSigmoidAlfaHN}}$ faollashtirish funksiyasining F parametrini o'rnatishga imkon beradi, SetLimitHN esa noldan birgacha bo'lgan va ε tashkil etuvchi $1/m$ qiymatining ulushini aniqlovchi koeffitsientni o'rnatadi. .

Ro'yxat 3 tarmoqni sinab ko'rish uchun sinov dasturini ko'rsatadi. Bu erda beshta kirish tasvirini taniydigan beshta neyronning ikkinchi qatlami bilan tarmoq quriladi, bu 5 dan 6 gacha nuqta o'lchamdagi harflarning sxematik tasvirlaridir (4a-rasmga qarang). Tarmoqni o'rgatish aslida 4-listda ko'rsatilgan "charh.img" faylida yozilgan ideal tasvirlarni yuklash va saqlash bilan bog'liq. Keyin "charh h" faylidan 8/30 shovqinli tasvirlar (4b-rasmga qarang) uning ichiga yuboriladi. img ni birma-bir kiriting, u muvaffaqiyatli ajratib turadigan 5-listdan.



Rasm-4. Namuna va sinov rasmlari



Rasm-5. BAMning blok diagrammasi

NEURO_HN va NEUROHAM fayllariga qo‘shimcha ravishda, loyiha [1] da tasvirlangan SUBFUN va NEURO_FFni ham o‘z ichiga oladi. Dastur Borland C++ 3.1 da sinovdan o‘tkazildi.

Taklif etilgan sinflar katta Hamming tarmoqlarini simulyatsiya qilish imkonini beradi. Tan olingan tasvirlar soni va murakkabligining ortishi aslida faqat operativ xotira miqdori bilan cheklanadi. Shuni ta’kidlash kerakki, Hamming tarmoq treningi ushbu maqolalar seriyasida hozirgacha muhokama qilingan barcha algoritmlarning eng oddiy algoritmini ifodalaydi.

Ikki tomonlama assotsiativ xotira (BAM) haqida hech bo‘lmaganda qisqacha eslatib o‘tmasdan, assotsiativ xotirani amalga oshiradigan tarmoqlarni muhokama qilish to‘liq bo‘lmaydi. Bu Hopfield tarmog‘i paradigmasining mantiqiy rivojlanishi bo‘lib , unga ikkinchi qatlamni qo‘shish kifoya. BAMning tuzilishi 5-rasmda ko‘rsatilgan. Tarmoq bir-biri bilan bog‘langan juft tasvirlarni eslab qolishga qodir. Juft tasvirlar $\mathbf{X}^k = \{ x_i^k : i = 0 \dots n-1 \}$ va $\mathbf{Y}^k = \{ y_j^k : j = 0 \dots m-1 \}$, $k=0$ vektorlar shaklida yozilsin . $r-1$, bu erda $r \sqrt{\text{juftlar soni}}$. Birinchi qatlamning kirishiga qandaydir $\mathbf{P} = \{ p_i : i = 0 \dots n-1 \}$ vektorini berish boshqa $\mathbf{Q} = \{ q_j : j = 0 \dots m-1 \}$ vektorining hosil bo‘lishiga olib keladi. ikkinchi qavatning kirishi, keyin yana birinchi qatlamning kirishiga o‘tadi. Har bir bunday tsikl bilan ikkala qatlamning chiqishidagi vektorlar bir juft namuna vektoriga yaqinlashadi, ulardan birinchisi $\sqrt{\mathbf{X}} \sqrt{\mathbf{P}}$ ga eng o‘xshash bo‘lib , u eng boshida tarmoqning kirishiga oziqlangan va ikkinchisi $\sqrt{\mathbf{Y}} \sqrt{\mathbf{u}}$ bilan bog‘langan. Vektorlar orasidagi assotsiatsiyalar birinchi qatlamning $\mathbf{W}^{(1)}$ vazn matritsasida kodlangan . Ikkinchi qatlamning og‘irlik matritsasi $\mathbf{W}^{(2)}$ birinchisiga ko‘chirilgan $(\mathbf{W}^{(1)})^T$ ga teng. O‘quv jarayoni, Xopfield tarmog‘ida bo‘lgani kabi , formula bo‘yicha $\mathbf{W}$ (va shunga mos ravishda $\mathbf{W}^T$) matritsasining elementlarini dastlabki hisoblashdan iborat :

$$w_{ij} = \sum_k x_i y_j^k, i = 0 \dots n-1, j = 0 \dots m-1 \quad (11)$$

Bu formula matritsa tenglamasining kengaytirilgan ko‘rinishidir

$$\mathbf{W} = \sum_k \mathbf{X}^T \mathbf{Y}^k \quad (12)$$

maxsus holat uchun tasvirlar vektorlar ko‘rinishida yozilganda va $[n*1]$ va $[1*m]$ o‘lchamdagi ikkita matritsaning ko‘paytmasi (11) ga olib keladi.

Xulosa qilib, quyidagi umumlashtirishni amalga oshirish mumkin. Xopfield , Xemming va BAM tarmoqlari to‘liq bo‘lmagan va buzilgan ma’lumotlardan tasvirlarni qayta tiklash muammosini sodda va samarali hal qilish imkonini beradi. Tarmoqlarning past sig‘imi (esda saqlangan tasvirlar soni) tarmoqlar shunchaki tasvirlarni eslab qolmasligi, balki ularni umumlashtirishga imkon berishi bilan izohlanadi , masalan, Hamming tarmog‘idan foydalangan holda, maksimal ehtimollik mezoniga ko‘ra tasniflash mumkin. [3]. Shu bilan birga, dasturiy ta’minot va apparat modellarini yaratish qulayligi ushbu tarmoqlarni ko‘plab ilovalar uchun jozibador qiladi .

Xopfield neyron tarmog‘ini amalga oshirish

Yuqorida aniqlangan Xopfield neyron tarmog‘ini amalga oshirish C++tilida amalga oshiriladi. Tajribalarni soddalashtirish uchun biz neyron turi va uning uzatish funktsiyasi bilan bevosita bog‘liq bo‘lgan turlarning asosiy ta’riflarini *simple_neuron* sinfiga qo‘shamiz va hosilalarni keyinroq aniqlaymiz. Neyron bilan bevosita bog‘liq bo‘lgan eng asosiy turlari:

1. vazin koeffitsienti turi (*float* tomonidan tanlangan);
2. neyronning holatini tavsiflovchi tur (2 ta ruxsat etilgan qiymat bilan sanab o‘tilgan tur kiritilgan).

Ushbu turlarga asoslanib, qolgan asosiy turlarni kiritish mumkin:

1. t momentidagi tarmoq holatini tavsiflovchi tur(standart *vector* konteyneri tanlangan);
2. neyron aloqalarining og‘irlik koeffitsientlari matritsasini tavsiflovchi tur(*vector* konteynerlarining *vektor* konteyneri tanlangan).

Yangi turlarni aniqlash kodi:

```
struct simple_neuron {  
    enum state {LOWER_STATE=-1, UPPER_STATE=1};  
  
    typedef float coeff_t;    <<(1)
```



```

typedef state state_t;    <<(2)
...
};

typedef simple_neuron neuron_t;
typedef neuron_t::state_t state_t;

typedef vector<state_t> neurons_line;    <<(3)
typedef vector<vector<neuron_t::coeff_t>> link_coeffs;    <<(4)

```

Tarmoqni o‘qitish yoki $J_{i,j} = \frac{1}{N} \cdot \sum_{\mu=1}^M [\zeta_{i,\mu}^{in} \cdot \zeta_{j,\mu}^{in}]$ ga muvofiq $J_{i,j}=J_{j,i}$ matritsa elementlarini hisoblash *learn_neuro_net* funktsiyasi tomonidan amalga oshiriladi, u o‘quv rasmlari ro‘yxatini oladi va *link_coeffs_t* turidagi ob’ektni qaytaradi. $J_{i,j}$ qiymatlari faqat pastki uchburchak elementlar uchun hisoblanadi. Yuqori uchburchak elementlarning qiymatlari $S_i = \overline{\zeta_{\mu}^{in}}$ ga muvofiq hisoblanadi. *learn_neuro_net* usulining umumiy ko‘rinishi quyida ko‘rsatilgan.

Neyron tarmog‘ini o‘qitish:

```

link_coeffs learn_neuro_net(const list<neurons_line> &src_images) {
    link_coeffs result_coeffs;
    size_t neurons_count = src_images.front().size();

    result_coeffs.resize(neurons_count);
    for (size_t i = 0; i < neurons_count; ++i) {
        result_coeffs[i].resize(neurons_count, 0);
    }

    for (size_t i = 0; i < neurons_count; ++i) {
        for (size_t j = 0; j < i; ++j) {
            neuron_t::coeff_t val = 0;
            val = std::accumulate(
                begin(src_images),
                end(src_images),
                neuron_t::coeff_t(0.0),
                [i, j] (neuron_t::coeff_t old_val, const neurons_line &image) ->
neuron_t::coeff_t{
                    return old_val + (image[i] * image[j]);
                });
        }
    }
}

```

```

        result_coeffs[i][j] = val;
        result_coeffs[j][i] = val;
    }
}

return result_coeffs;
}

```

Neyron holatini yangilash *neuro_net_system* funktsiyasi yordamida amalga oshiriladi. Funktsiyaning *_do* usulining argumenti $S_i(0)$ ning boshlang'ich holati bo'lib, u tanib olinadigan tasvir — *neurons_line* tipidagi ob'ektga havola.

Funktsiya usuli uzatilayotgan *neurons_line* tipidagi ob'ektni t vaqtida neyron tarmoq holatiga o'zgartiradi. qiymat qat'iy belgilanmagan va ifoda bilan aniqlanadi:

$$S_i(T - 1) = S_i(T)$$

Ya'ni, har bir neyronning holati 1 " takt " da o'zgarmaganida.

$$S_i(t + 1) = \text{sign}\left[\sum_{j=1}^N J_{i,j} S_j(t)\right], \quad i, j \in 1, \dots, N$$

hisoblash uchun 2 STL

algoritmi qo'llaniladi:

1. *std::inner_product* og'irlik koeffitsientlari va neyron holatlari hosilalari yig'indisini hisoblash uchun (ya'ni, ma'lum bir i uchun);

2. *std::transform* har bir neyron uchun yangi qiymatlarni hisoblash uchun (ya'ni har bir mumkin bo'lgan i uchun yuqoridagi nuqtani hisoblash)

neurons_net_system funktsiyasi va *simple_neuron* sinfining *calculate* hisoblash usuli uchun manba kodi ko'rsatilgan.

Neyron tarmog'ining ishlashini amalga oshiradigan funktsiya:

```

struct simple_neuron {
...
    template <typename _Iv, typename _Ic>
    static state_t calculate(_Iv val_b, _Iv val_e, _Ic coeff_b) {
        auto value = std::inner_product(
            val_b,
            val_e,
            coeff_b,
            coeff_t(0)

```

```

        );

    return value > 0 ? UPPER_STATE : LOWER_STATE;
}
};

struct neuro_net_system {
    const link_coeffs &_coeffs;
    neuro_net_system(const link_coeffs &coeffs): _coeffs(coeffs) {}
    bool do_step(neurons_line& line) {
        bool value_changed = false;

        neurons_line old_values(begin(line), end(line));
        link_coeffs::const_iterator it_coeffs = begin(_coeffs);

        std::transform(
            begin(line),
            end(line),
            begin(line),
            [&old_values, &it_coeffs, &value_changed] (state_t old_value) -> state_t
        {
            auto new_value = neuron_t::calculate(
                begin(old_values),
                end(old_values),
                begin(*it_coeffs++)
            );

            value_changed = (new_value != old_value) || value_changed;

            return new_value;
        });

        return value_changed;
    }
};

size_t _do(neurons_line& line) {
    bool need_continue = true;
    size_t steps_done = 0;

    while (need_continue) {
        need_continue = do_step(line);
        ++steps_done;
    }

    return steps_done;
}

```

```
}  
};
```

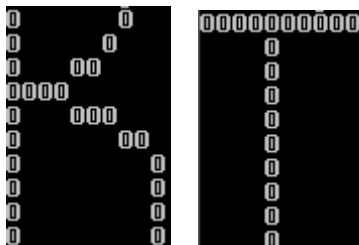
Kirish va chiqish obrazlarini konsolga chiqarish uchun *neurons_line_print_descriptor* turi yaratilgan bo‘lib, u obraz va formatlash formatiga havolani saqlaydi (obraz joylangan to‘rtburchakning kengligi va balandligi). Ushbu tur uchun << operatori bekor qilinadi. *neurons_line_print_descriptor* tipidagi manba kodi va oqim chiqishi operatori quyida ko‘rsatilgan.

Neyron tarmoq holatini oqimga chiqarish:

```
struct neurons_line_print_descriptor {  
    const neurons_line &_line;  
    const size_t _width;  
    const size_t _height;  
  
    neurons_line_print_descriptor (  
        const neurons_line &line,  
        size_t width,  
        size_t height  
    ): _line(line),  
        _width(width),  
        _height(height)  
    {}  
};  
  
template <typename Ch, typename Tr>  
std::basic_ostream<Ch, Tr>&  
operator << (std::basic_ostream<Ch, Tr>&stm, const  
neurons_line_print_descriptor &line) {  
    neurons_line::const_iterator it = begin(line._line), it_end = end(line._line);  
  
    for (size_t i = 0; i < line._height; ++i) {  
        for (size_t j = 0; j < line._width; ++j) {  
            stm << neuron_t::write(*it);  
            ++it;  
        }  
        stm << endl;  
    }  
  
    return stm;  
}
```

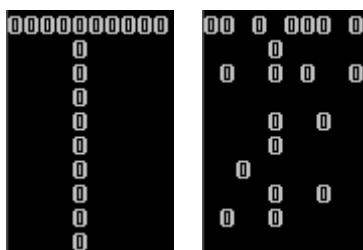
Neyron tarmog'ining ishlashiga misol.

Amalga oshirish samaradorligini tekshirish uchun neyron tarmoq 2 ta asosiy tasvirga o'rgatilgan:

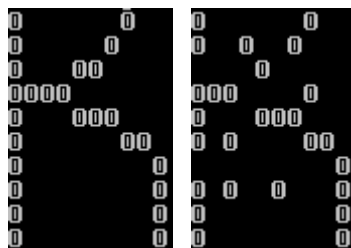


Rasm-3. Kaliy tasvirlar.

Kirishda buzilgan tasvirlar berildi. Neyron tarmoq asl tasvirlarni to'g'ri tanidi. Buzilgan tasvirlar va tanilgan tasvirlar 4, 5 rasmlarda keltirilgan



Rasm-4. 1 chi tasvieni tanish.



Rasm-5. 1 chi tasvirni tanish.

Dastur buyruq satridan quyidagi satr bilan ishga tushiriladi: APPNAME WIDTH HEIGHT SOURCE_FILE [LEARN_FILE_N], bu erda:

- *AppNaame* -bajariladigan fayl nomi;
- *WIDTH, HEIGHT* - chiqish va asosiy tasvirlar mos keladigan to'rtburchakning kengligi va balandligi;
- *SOURCE_FILE* - dastlabki rasm bilan manba fayli;

- *[LEARN_FILE_N]* - asosiy rasmlarga ega bo'lgan bir yoki bir nechta fayl (probel orqali).

O'z-o'zini nazorat qilish uchun savollar

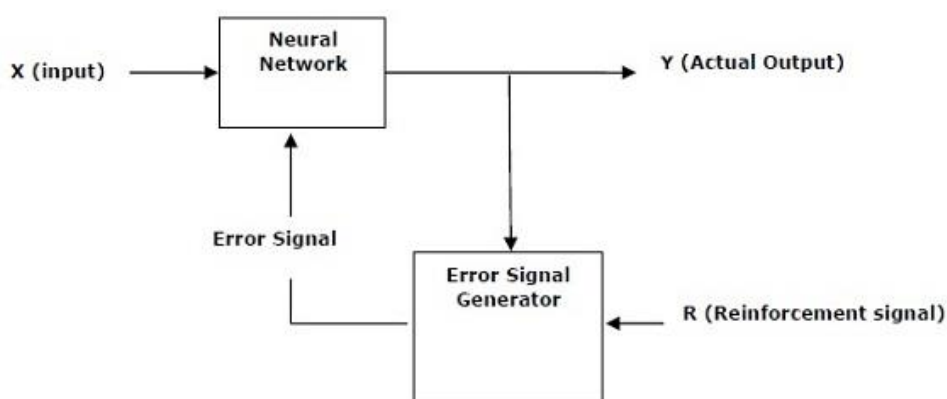
1. Sun'iy neyron tarmoqlar (Artificial Neural Networks - ANN) nima?
2. O'qituvchi bilan o'qitish?
3. Neyron?
4. Neyron ishonchi?
5. Neyronning foydaliligi?
6. Yashirin qatlam?
7. Neyronlar mantiqiy element sifatida?
8. Noravshan mantiq?
9. Bir qatlamli neyron?
10. Ko'p qatlamli neyron?
11. Kohonen neyron tarmog'i?
12. Xopfild va Xemming neyron tarmoqlari?

3.2-§. MUSTAXKAMLASH YORDAMIDA O‘RGANISH

Ushbu bo‘limda Python yordamida sun‘iy intellektda o‘rgatish bilan bog‘liq tushunchalar haqida batafsil ma’lumot olasiz.

Mustaxkamlash yordamida o‘rgatish asoslar

Ushbu turdagi o‘rgatish tarmoqni muhim ma’lumotlar asosida kuchaytirish uchun qo‘llaniladi. Ya’ni, o‘rgatilayotgan tarmoq atrof-muhitdan qandaydir qayta aloqa oladi. Biroq, bu qayta aloqa baholash xususiyatiga ega bo‘lib, mustaxkamlash asosidagi o‘rgatishdagi kabi yo‘riqnoma sifatida emas. Ushbu qayta aloqa asosida tarmoq kelajakda tanqidchilardan yanada batafsil ma’lumot olish uchun vaznlarni tuzatadi. Ushbu o‘rgatish jarayoni nazorat ostida o‘rgatish bilan o‘xshash, ammo bizda ma’lumotlar ancha kam bo‘lishi mumkin. Quyidagi rasmda o‘rgatish bilan bog‘liq blok-sxemasi keltirilgan.



Qurilish bloklari: atrof-muhit va agent

Atrof-muhit va agent sun‘iy intellektda o‘rgatish bilan bog‘liq asosiy qurilish bloklari hisoblanadi. Ushbu bo‘limda ular batafsil ko‘rib chiqiladi.

Agent

Agent – bu atrof-muhitni sensorlar yordamida sezadigan va bu atrof-muhitga effektorlar orqali ta’sir qiladigan har qanday ob’ekt.

- Inson agentida sezgi organlari, masalan, ko‘zlar, quloqlar, burun, til va teri, sensorlar bilan parallel ravishda joylashgan, shuningdek, qo‘llar, oyoqlar, og‘iz kabi organlar effektorlar sifatida xizmat qiladi.

- Robot agenti sensorlar uchun kameralar va infraqizil masofani o'lchovchi qurilmalarni, effektorlar uchun esa turli dvigatellar va haydovchilarni almashtiradi.

- Dasturning agentida o'zining dasturlari va harakatlari bitlar qatorlari sifatida kodlangan.

Agentlar terminologiyasi

Quyidagi terminlar sun'iy intellektda o'rgatish bilan bog'liq eng ko'p ishlatiladigan tushunchalardir:

- **Agentning samaradorlik ko'rsatkichi** – bu agentning muvaffaqiyatini aniqlovchi mezon.

- **Agentning xulq-atvori** – bu agentning berilgan sezgi ketma-ketligidan keyin amalga oshiradigan harakati.

- **Sezgi** – bu agentning berilgan misoldagi sezgi kirish ma'lumotlari.

- **Sezgi ketma-ketligi** – bu agentning hozirgi vaqtgacha sezgan barcha ma'lumotlar tarixi.

- **Agentning funksiyasi** – bu ko'rsatmalar ketma-ketligini harakatga aylantirish.

Atrof-muhit

Ba'zi dasturlar to'liq sun'iy atrof-muhitda ishlaydi, bu atrof-muhit faqat klaviaturadan kiritish, ma'lumotlar bazasi, kompyuter fayl tizimlari va ekranda belgilarning chiqarilishi bilan cheklanadi. Boshqa tomondan, ba'zi dastur agentlari, masalan, dasturiy robotlar yoki softbotlar, keng va cheksiz domenda mavjud. Simulyator juda batafsil va murakkab atrof-muhitga ega. Dastur agenti real vaqtda katta harakatlar to'plamidan tanlashni talab qiladi.

Masalan, mijozning onlayn afzalliklarini skanerlash va buyurtmachiga qiziqarli mahsulotlarni ko'rsatish uchun mo'ljallangan dasturiy bot, haqiqiy va sun'iy atrof-muhitda ham ishlaydi.

Atrof-muhitning xususiyatlari

Atrof-muhit ko'plab xususiyatlarga ega, quyida ular keltirilgan:

- **Diskret / uzluksiz** – Agar atrof-muhitda cheklangan miqdordagi turli aniq belgilangan holatlar mavjud bo'lsa, atrof-muhit diskret hisoblanadi, aks holda u

uzluksizdir. Masalan, shaxmat diskret atrof-muhit, boshqaruv esa uzluksiz atrof-muhitdir.

- **Kuzatiladigan / qisman kuzatiladigan** – Agar atrof-muhitning to‘liq holatini har bir vaqtda uning tasvirlari asosida aniqlash mumkin bo‘lsa, u kuzatiladigan atrof-muhit bo‘ladi; aks holda, u faqat qisman kuzatiladigan bo‘ladi.

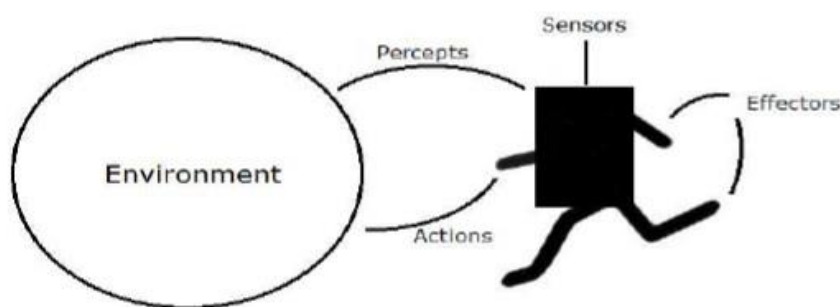
- **Statik / dinamik** – Agar atrof-muhit agentning harakati davomida o‘zgarmasa, u statik bo‘ladi; aks holda, u dinamik bo‘ladi.

- **Bitta agent / bir nechta agentlar** – Atrof-muhitda agentga o‘xshash yoki boshqacha turdagi boshqa agentlar bo‘lishi mumkin.

- **Mavjud / mavjud emas** – Agar agentning sensor qurilmasi atrof-muhitning to‘liq holatiga kirish huquqiga ega bo‘lsa, atrof-muhit agent uchun mavjud bo‘ladi; aks holda, u mavjud emas.

- **Deterministik / no-deterministik** – Agar atrof-muhitning keyingi holati hozirgi holat va agentning harakatlari bilan to‘liq aniqlansa, atrof-muhit deterministik hisoblanadi; aks holda, u no-deterministikdir.

- **Episodik / no-episodik** – Episodik atrof-muhitda har bir epizod agentning sezgilaridan keyin amalga oshiradigan harakatidan iborat. Uning harakatining sifati faqat o‘sha epizodga bog‘liq. Keyingi epizodlar oldingi epizodlardagi harakatlarga bog‘liq emas. Episodik atrof-muhitlar ancha sodda, chunki agentga oldindan o‘ylash zarurati yo‘q.



Python yordamida muhit yaratish

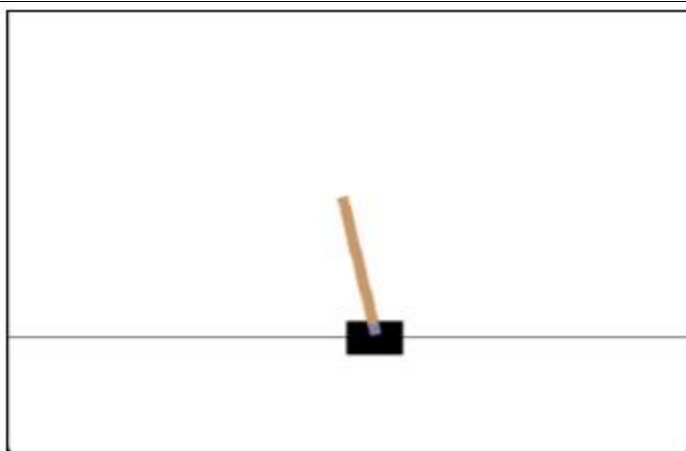
OpenSI Gym to‘plamidan foydalanamiz , uni quyidagi buyruq yordamida o‘rnatish mumkin.

```
pip install gym
```

OpenSI sport zali turli maqsadlarda ishlatilishi mumkin bo'lgan turli muhitlarga ega. Ulardan ba'zilar **Cartpole-v0**, **Hopper-v1** va **MsPacman-v0**. Ular turli xil dvigatellarni talab qiladi.

Quyidagi kod cartpole-v0 muhiti uchun namuna Python kodini ko'rsatadi.

```
import gym
env = gym.make('CartPole-v0')
env.reset()
for _ in range(1000):
    env.render()
    env.step(env.action_space.sample())
```

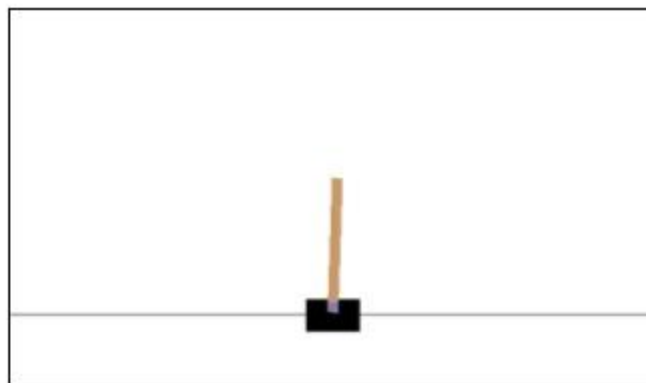


Xuddi shu tarzda boshqa muhitlarni yaratishingiz mumkin.

Python yordamida o'quv agentini yaratish

Rasmda ko'rsatilganidek, **OpenSI Gym to'plamidan** foydalanamiz

```
import gym
env = gym.make('CartPole-v0')
for _ in range(20):
    observation = env.reset()
    for i in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
    if done:
        print("Episode finished after { } timesteps".format(i+1))
        break
```



E'tibor bering, ustun ustuni o'z-o'zidan muvozanatlasha oladi.

O'z-o'zini nazorat qilish uchun savollar

1. Mustaxkamlash yordamida o'rgatish asoslar?
2. Agent bu nima?
3. Agentlar terminologiyasi?
4. Atrof-muhitning xususiyatlari?
5. Python yordamida o'quv agentini yarating?

3.3-§. GENETIK ALGORITMLAR

Genetik algoritmlar nima?

Genetik algoritmlar (GA) – bu tabiiy tanlov va genetika tushunchalariga asoslangan qidiruv algoritmlari. GAlar yanada kengroq hisoblash sohasining kichik bir qismi bo‘lib, bu soha "evolyutsion hisoblash" deb ataladi. GAlar Jon Holland va uning talabalarining, shu jumladan Michigan universitetidan Devid E. Goldberg tomonidan ishlab chiqilgan. Shundan beri ular yuqori darajada muvaffaqiyatli optimallashtirish vazifalarida sinovdan o‘tkazilgan. GAda bizda ma’lum bir muammo uchun mumkin bo‘lgan yechimlar to‘plami mavjud. So‘ngra, bu yechimlar rekombinatsiya va mutatsiyaga (tabiiy genetika kabi) duchor bo‘ladi, yangi avlodlar ishlab chiqiladi, va jarayon turli avlodlar uchun takrorlanadi. Har bir jonzotga (yoki yechim-kandidatlarga) moslashuvchanlik qiymati beriladi (uning maqsad funksiyasiga asoslangan holda), va ko‘proq moslashuvchan jonzotlarga ko‘proq imkoniyatlar beriladi, ular juftlashib, ko‘proq moslashuvchan avlodlar tug‘adi. Bu Darvinning "eng moslashuvchanlar omon qoladi" degan nazariyasiga mos keladi. Shunday qilib, u har bir avloddan yaxshiroq odamlarni yoki yechimlarni rivojlantiradi, shu tariqa to‘xtash mezoniga erishmaguncha davom etadi. Genetik algoritmlar tabiiy ravishda tasodifiy bo‘lsa-da, ular tasodifiy lokal qidiruvdan (faqat tasodifiy yechimlarni sinab ko‘rish va hozirgi vaqtda eng yaxshi yechimlarni kuzatish) ancha samarali ishlaydi, chunki ular tarixiy ma’lumotlardan ham foydalanadi.

GAni optimallashtirish vazifalarida qanday ishlatish mumkin?

Optimallashtirish – bu dizayn, holat, resurslar va tizimni maksimal samarador qilishga qaratilgan harakatdir. Quyidagi blok-sxema optimallashtirish jarayonini ko‘rsatadi.



GA mexanizmining optimallashtirish jarayonidagi bosqichlari

Quyida GA mexanizmining optimallashtirish vazifalarini hal qilish uchun qo'llaniladigan bosqichlar ketma-ketligi keltirilgan:

- **1-bosqich** – Boshlang'ich populatsiyani tasodifiy ravishda yaratish.
- **2-bosqich** – Eng yaxshi moslashuvchanlik ko'rsatkichiga ega bo'lgan boshlang'ich yechimni tanlash.
- **3-bosqich** – Tanlangan yechimlarni mutatsiya va krossover operatorlari yordamida rekombinatsiya qilish.
- **4-bosqich** – Yangi avlodni populyatsiyaga kiritish.
- **5-bosqich** – Agar to'xtash sharti bajarilgan bo'lsa, eng yaxshi moslashuvchanlik qiymatiga ega yechimni qaytarish. Aks holda, 2-bosqichga qaytish.

Kerakli paketlarni o'rnatish

Python'da genetik algoritmlar yordamida muammoni hal qilish uchun biz GA uchun kuchli paketlardan biri bo'lgan **DEAP** kutubxonasidan foydalanmoqchimiz. Bu evolyutsion hisoblashlar uchun yangi dasturiy ramka bo'lib, tezkor prototip yaratish va g'oyalarni sinovdan o'tkazish imkonini beradi. Ushbu paketni quyidagi komandani ishlatib o'rnatishimiz mumkin:

```
pip install deap
```

anakonda muhitidan foydalanayotgan bo'lsangiz, deapni o'rnatish uchun.

```
conda install -c conda-forge deap
```

Genetik algoritmlar yordamida yechimlarni amalga oshirish

Genetik algoritmlar yordamida yechimlarni amalga oshirish tushuntiriladi.

Bitli shablonlarni generatsiya qilish

One Max topshirig'iga asoslangan holda 15 tadan iborat bit qatorini qanday generatsiya qilish ko'rsatadi. Ko'rsatilganidek, kerakli paketlarni import qiling.

```
import random  
from deap import base, creator, tools
```

Hisoblash funksiyasini aniqlang. Bu genetik algoritmlar yaratish yoʻlidagi birinchi qadamdir.

```
def eval_func(individual):  
    target_sum = 15  
    return len(individual) - abs(sum(individual) - target_sum),
```

Endi kerakli parametrlarga ega asboblarni yaratish.

```
def create_toolbox(num_bits):  
    creator.create("FitnessMax", base.Fitness, weights=(1.0,))  
    creator.create("Individual", list, fitness=creator.FitnessMax)
```

Asboblarni toʻplamini ishga tushirish.

```
toolbox = base.Toolbox()  
toolbox.register("attr_bool", random.randint, 0, 1)  
toolbox.register("individual", tools.initRepeat, creator.Individual,  
    toolbox.attr_bool, num_bits)  
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

Roʻyxatdan oʻtish operator hisob-kitoblari.

```
toolbox.register("evaluate", eval_func)
```

Hozir roʻyxatdan oʻtish krossover operator.

```
toolbox.register("mate", tools.cxTwoPoint)
```

Roʻyxatdan oʻtish operator mutatsiyalar.

```
toolbox.register("mutate", tools.mutFlipBit, indpb = 0.05)
```

Aniqlash operator uchun koʻpaytirish.

```
toolbox.register("select", tools.selTournament, tournsize = 3)  
return toolbox  
if __name__ == "__main__":
```

```

num_bits = 45
toolbox = create_toolbox(num_bits)
random.seed(7)
population = toolbox.population(n = 500)
probab_crossing, probab_mutating = 0.5, 0.2
num_generations = 10
print('\nEvolution process starts')

```

Baho hammasi umumiylik ma'lumotlar.

```

fitnesses = list(map(toolbox.evaluate, population))
for ind, fit in zip(population, fitnesses):
    ind.fitness.values = fit
print('\nEvaluated', len(population), 'individuals')

```

Yaratish va takrorlang avlodlar.

```

for g in range(num_generations):
    print("\n- Generation", g)

```

Mutaxassislarning keyingi avlodini tanlash.

```

offspring = toolbox.select(population, len(population))

```

Endi tanlangan foydalanuvchilarni klonlash.

```

offspring = list(map(toolbox.clone, offspring))

```

Nasllarga chatishtirish va mutatsiyani qo'llash.

```

for child1, child2 in zip(offspring[::2], offspring[1::2]):
    if random.random() < probab_crossing:
        toolbox.mate(child1, child2)

```

Bola elementning fitnes qiymatini olib tashlang.

```

del child1.fitness.values

```

```
del child2.fitness.values
```

Hozir murojaat qiling mutatsiya.

```
for mutant in offspring:  
    if random.random() < probabab_mutating:  
        toolbox.mutate(mutant)  
        del mutant.fitness.values
```

Jismoniy tayyorgarligi etarli bo'lmagan odamlarni baholang.

```
invalid_ind = [ind for ind in offspring if not ind.fitness.valid]  
fitnesses = map(toolbox.evaluate, invalid_ind)  
for ind, fit in zip(invalid_ind, fitnesses):  
    ind.fitness.values = fit  
print('Evaluated', len(invalid_ind), 'individuals')
```

Endi populyatsiyani keyingi avlod individiga almashtiring.

```
population[:] = offspring
```

Joriy avlodlar uchun statistikasi chop etish.

```
fits = [ind.fitness.values[0] for ind in population]  
length = len(population)  
mean = sum(fits) / length  
sum2 = sum(x*x for x in fits)  
std = abs(sum2 / length - mean**2)**0.5  
print('Min =', min(fits), ', Max =', max(fits))  
print('Average =', round(mean, 2), ', Standard deviation =',  
      round(std, 2))  
print("\n- Evolution ends")
```

Yakuniy natijani chop eting.

```
best_ind = tools.selBest(population, 1)[0]  
print('\nBest individual:\n', best_ind)
```



```
print('\nNumber of ones:', sum(best_ind))
```

Following would be the output:

Evolution process starts

Evaluated 500 individuals

- Generation 0

Evaluated 295 individuals

Min = 32.0 , Max = 45.0

Average = 40.29 , Standard deviation = 2.61

- Generation 1

Evaluated 292 individuals

Min = 34.0 , Max = 45.0

Average = 42.35 , Standard deviation = 1.91

- Generation 2

Evaluated 277 individuals

Min = 37.0 , Max = 45.0

Average = 43.39 , Standard deviation = 1.46

... ..

- Generation 9

Evaluated 299 individuals

Min = 40.0 , Max = 45.0

Average = 44.12 , Standard deviation = 1.11

- Evolution ends

Best individual:

[0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1,
1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1]

Number of ones: 15

Simvolik regressiya muammosi

Bu genetik dasturlashning eng mashhur vazifalaridan biridir. Barcha simvolik regressiya vazifalarida tasodifiy ma'lumotlar taqsimoti ishlatiladi va eng aniq ma'lumotlarni simvolik formulaga moslashtirishga harakat qilinadi. Odatda, insonning jismoniy tayyorgarligini o'lchash uchun bunday ko'rsatkich sifatida

RMSE (o'rtacha kvadrat xatolik) ishlatiladi. Bu klassik regressiya muammosi bo'lib, biz quyidagi tenglamani ishlatamiz:

$$5x^3 - 6x^2 + 8x = 1.$$

Biz yuqorida keltirilgan barcha bosqichlarni bajarishimiz kerak, ammo asosiy qism bu **primitivlar to'plamini** yaratish bo'ladi, chunki ular foydalanuvchilar uchun qurilish bloklari bo'lib xizmat qiladi va baholashni boshlashga imkon beradi. Bu yerda biz klassik primitivlar to'plamidan foydalanamiz. Quyidagi Python kodi bu jarayonni batafsil tushuntiradi:

```
import operator
import math
import random
import numpy as np
from deap import algorithms, base, creator, tools, gp
def division_operator(numerator, denominator):
    if denominator == 0:
        return 1
    return numerator / denominator
def eval_func(individual, points):
    func = toolbox.compile(expr=individual)
    return math.fsum(mse) / len(points),
def create_toolbox():
    pset = gp.PrimitiveSet("MAIN", 1)
    pset.addPrimitive(operator.add, 2)
    pset.addPrimitive(operator.sub, 2)
    pset.addPrimitive(operator.mul, 2)
    pset.addPrimitive(division_operator, 2)
    pset.addPrimitive(operator.neg, 1)
    pset.addPrimitive(math.cos, 1)
    pset.addPrimitive(math.sin, 1)
    pset.addEphemeralConstant("rand101", lambda: random.randint(-1,1))
    pset.renameArguments(ARG0 = 'x')
    creator.create("FitnessMin", base.Fitness, weights = (-1.0,))
    creator.create("Individual", gp.PrimitiveTree, fitness=creator.FitnessMin)
```

```

toolbox = base.Toolbox()
toolbox.register("expr", gp.genHalfAndHalf, pset=pset, min_=1, max_=2)
toolbox.expr)
toolbox.register("population",tools.initRepeat,list, toolbox.individual)
toolbox.register("compile", gp.compile, pset = pset)
toolbox.register("evaluate", eval_func, points = [x/10. for x in range(-10,10)])
toolbox.register("select", tools.selTournament, tournsize = 3)
toolbox.register("mate", gp.cxOnePoint)
toolbox.register("expr_mut", gp.genFull, min_=0, max_=2)
toolbox.register("mutate", gp.mutUniform, expr = toolbox.expr_mut, pset = pset)
toolbox.decorate("mate", gp.staticLimit(key = operator.attrgetter("height"), max_value = 17))
toolbox.decorate("mutate", gp.staticLimit(key = operator.attrgetter("height"), max_value =
17))
return toolbox
if __name__ == "__main__":
    random.seed(7)
    toolbox = create_toolbox()
    population = toolbox.population(n = 450)
    hall_of_fame = tools.HallOfFame(1)
    stats_fit = tools.Statistics(lambda x: x.fitness.values)
    stats_size = tools.Statistics(len)
    mstats = tools.MultiStatistics(fitness=stats_fit, size = stats_size)
    mstats.register("avg", np.mean)
    mstats.register("std", np.std)
    mstats.register("min", np.min)
    mstats.register("max", np.max)
    probab_crossover = 0.4
    probab_mutate = 0.2
    number_gen = 10
    population, log = algorithms.eaSimple(population, toolbox,
        probab_crossover, probab_mutate, number_gen,
        stats = mstats, halloffame = hall_of_fame, verbose = True)

```

E'tibor bering, barcha asosiy qadamlar bit naqshlarini yaratish uchun ishlatiladiganlar bilan bir xil. Ushbu dastur bizga 10 avloddan keyin min, maks, std (standart og'ish) ko'rinishidagi chiqishni beradi.

O'z-o'zini nazorat qilish uchun savollar

1. Genetik algoritmlar nima?
2. GAni optimallashtirish vazifalarida qanday ishlatish mumkin?
3. GA mexanizmining optimallashtirish jarayonidagi bosqichlari?
4. Bitli shablonlarni generatsiya qilish?
5. Simvolik regressiya nima?

3.4-§. KOMPYUTERLI KO'RISHI

Kompyuterli ko'rishi – bu inson ko'rishining kompyuter dasturiy va apparatli ta'minot yordamida modellashtirilishi va takrorlanishi bilan bog'liq fan. Ushbu bo'limda siz bunga alohida to'liq ma'lumot olasiz.

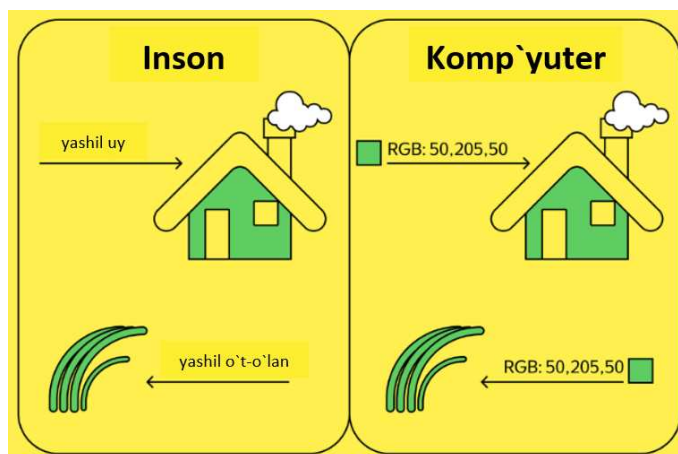
Kompyuterli ko'rishi (inglizcha **computer vision**, CV) - bu sun'iy intellektning bir sohasi bo'lib, uning maqsadi 2D tasvirlardan 3D sahnani qayta tiklash, tahlil qilish va tushunishni o'rganishdir, bunda sahnadagi tuzilma xususiyatlari inobatga olinadi.

Kompyuter ko'rishi - bu **chuqur o'qitish** (inglizcha **deep learning**) yordamida amalga oshiriladigan mashinada o'qitishning bir turi bo'lib, bu usulda neyron tarmoqlari yordamida muammolarni hal qilishga erishiladi.

Kompyuter ko'rish mutaxassislari uchun vazifa - kompyuterni tasvirlarni inson kabi idrok qilishni o'rgatishdir.

Inson ko'rish tizimi yuqori darajadagi xususiyatlarni aniqlash imkonini beradi, ya'ni tajribaga asoslanib mushukni jonsiz qushdan ajratish oson: mushukda quloq, mo'ylov va dum bor, qushda esa qanotlar va tumshuq mavjud.

Biroq, kompyuter ob'ektni aniqlash uchun dastlabki ma'lumotlarga ega bo'lmasdan bu ishni bajarolmaydi va muhimni ahamiyatsizdan ajratishni bilmaydi. Kompyuter uchun rangli tasvir - bu raqamlar to'plamiga o'xshaydi: u **RGB** ranglar tizimida har bir pikselni ifodalovchi uchta raqamdan iborat (qora va oq tasvirlar holatida esa bitta raqam). Yashil rangli piksel maydon yoki xuddi shu rangdagi uyga tegishli bo'lishi mumkin, ammo kompyuter uchun kengroq rasmni ko'rish imkoniyati yo'q.



Inson tasvirdagi ob'ektlarni yuqori darajadagi xususiyatlar asosida tanib oladi, hatto agar ular bir xil rangda bo'lsa. Kompyuter uchun esa tasvir - bu piksel to'plami bo'lib, u qaysi ob'ekt bu pikselalardan tashkil topganini aniqlay olmaydi.

Buning sababi shundaki, mashinaga **rezeptiv maydon** — bu kattaroq hududni butunlay qayta ishlash va yuqori darajadagi xususiyatlar bilan ishlash qobiliyati yetishmaydi. Kompyuter buni o'rganish uchun ob'ektning xususiyatlarini tanib olish kerak: uning chegaralari (gorizontal va vertikal chiziqlar), rang, tekstura, gradientlar, bu xususiyatlar pikselalardan tashkil topadi va tasvirni tasniflashga yordam beradi.

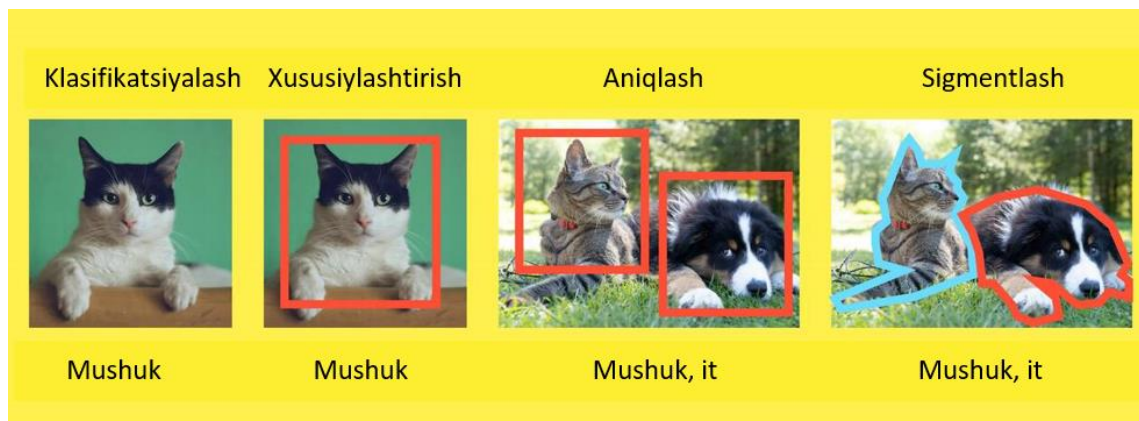
2000-yillarning boshlarida olimlar xususiyatlarni qo'lda aniqlashga harakat qilishgan, shundan "hand-crafted features" (qo'lda tayyorlangan xususiyatlar) atamasi paydo bo'ldi. Bu juda mehnat talab qiladigan jarayon bo'lib, oxir-oqibat neyron tarmoqlarini ulashga qaror qilindi, chunki ular o'rgatish mumkin, ya'ni ularga qanday qilib qonuniyatlarni topishni ko'rsatish mumkin edi.

To'liq bog'langan neyron tarmoqlari raqamli ma'lumotlarni qayta ishlashda 28% dan ortiq xatolik bilan muammoni hal qilishga qodir edi. Kompyuter ko'rish sohasida inqilob 2012 yilda tasvirlar bilan ishlashga mo'ljallangan neyron tarmoqlarini qo'llash bilan yuz berdi. Bu neyron tarmoqlari **svyortochniy** (convolutional) deb ataladi. 2012-yilda svyortochniy neyron tarmoqlari tasnifatsiya xatoliklarini sezilarli darajada kamaytirib, tasvirlarni tanib olishda insonning natijasiga yaqinlashdi — taxminan 5% xatolik bilan. 2015-yilda esa, neyron tarmoqlari insonni aniqlikda ortda qoldirib, 3,6% xatolik bilan natijalarini ko'rsatdilar.

Kompyuter ko‘rishining asosiy vazifalari

Kompyuter ko‘rishi quyidagi to‘rt asosiy vazifaga qarshi qo‘llaniladi:

1. **Klassifikatsiya** – bu ob‘ektga ma’lum bir sinfni tayinlash. Oddiy so‘z bilan aytganda, neyron tarmog‘i ob‘ektni o‘ziga ma’lum bo‘lgan guruhga ajratadi: odam, skameyka, chamadon va h.k.
2. **Lokalizatsiya** – ob‘ektning joylashuvini aniqlash.
3. **Detektsiya** – masalan, metropolitenning real vaqtda unutib qo‘yilgan narsasini aniqlash. Kompyuter lokalizatsiya o‘tkazadi va uni klassifikatsiya qiladi.
4. **Segmentatsiya** – bu ikki turga bo‘linadi. Birinchi turi - **semantik segmentatsiya**, bu tasvirlarni fonidan ajratib, ularga maskalar qo‘llash imkonini beradi. Bu ob‘ektlarni sinflarga ajratishga va har birini rangli maskalar bilan ajratishga imkon beradi: mushuklar — qizil, itlar — yashil. Boshqa bir misol — odamning orqasida tumanli fon. Keyingi qadamda ob‘ektlar segmentatsiyasi bo‘ladi. Mushuklar va itlarni sinflarga ajratadi, lekin qo‘shimcha ravishda ularning farqlarini ko‘rsatib, ularni alohida ob‘ektlar sifatida ajratadi: it № 1 va it № 2.



Kompyuter ko‘rish texnologiyalari

Birinchi navbatda, neyron tarmog‘i tasvirni o‘rganishi kerak. **Svyortochniy neyron tarmoqlari** filterni ishlatadi, uni **svyortochniy yadro** deb atashadi. **Svyortochniy yadro** - bu oyna ichidagi raqamlar bilan kvadrat bo‘lib, asosan 3x3 o‘lchamda, lekin 5x5 yoki 7x7 ham bo‘lishi mumkin. Yadro ichidagi raqamlar

neyron tarmog'i tomonidan o'zini-o'zi moslashtiriladi — bu uning o'rganish jarayonining o'ziga xos xususiyati hisoblanadi. **Svyortochlash jarayoni** — bu yadroning tasvirning chap yuqori burchagidan o'ng pastki burchagiga harakatlanishi, chapdan o'nga va yuqoridan pastga qarab.

Agar **3x3** o'lchamdagi svyortochniy yadroni misol sifatida olsak, jarayon quyidagicha amalga oshadi:

1. **Svyortochniy filtr raqamlari** 9 pikseldan iborat maydonga qo'yiladi va har bir juft raqam bir-biriga ko'paytiriladi, keyin esa bu natijalar yig'iladi. Yig'ilgan qiymat faqat markaziy pikselga kiradi, shuning uchun yakuniy tasvirda chetki pikselardan bittasi kesib tashlanadi. Misol sifatida, agar maydonda har bir juftda nol bo'lsa, faqat o'ng pastki burchakdagi juftda 2 va -4 ko'paytiriladi, natijada -8 qiymati olinadi.

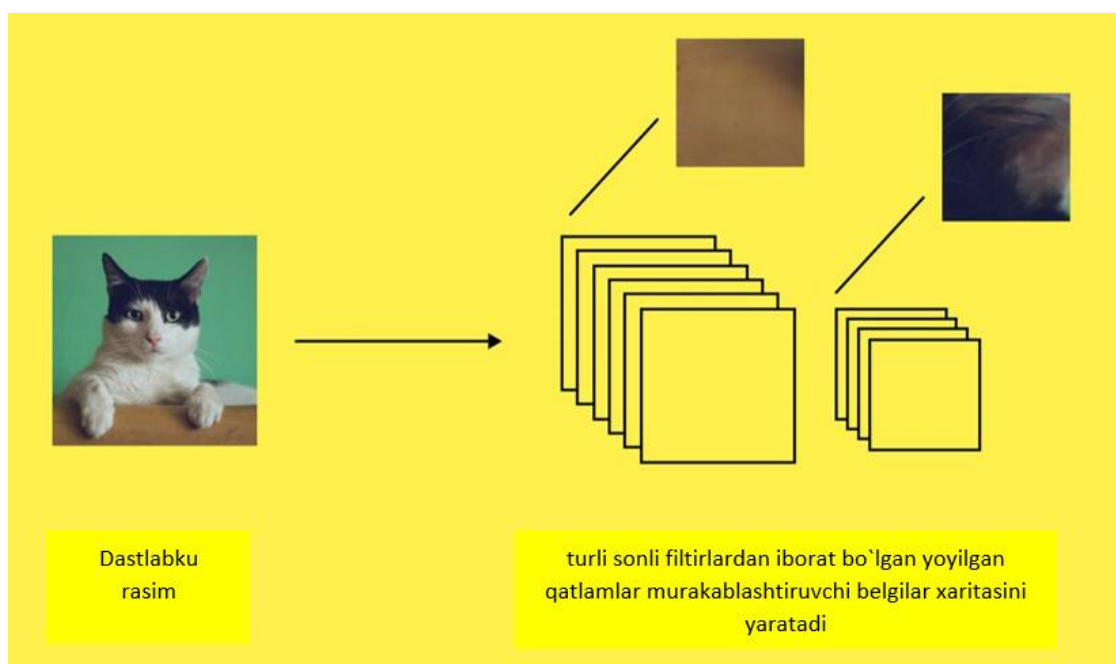


Svyortochniy filtr jarayoni: Svyortochniy filtr tasvirni tahlil qilishda birinchi bosqichni tashkil qiladi. Agar filtrning o'lchami **3x3** bo'lsa, natijada tasvirning chetki qismlari bir pikselga qisqaradi. Bu jarayon tasvirni kichraytiradi, chunki yadro tasvirning faqat markaziy qismini tahlil qiladi.

2. **Chegaralar va gradientlar:** Agar filtr **chegaralarni aniqlash** uchun ishlasa, bu chegaralar **musbat** va **manfiy** qiymatlar o'rtasidagi o'tish joylarida paydo bo'ladi. Musbat qiymatlar tasvirning chap qismidagi yorqinlik o'ng tomonga

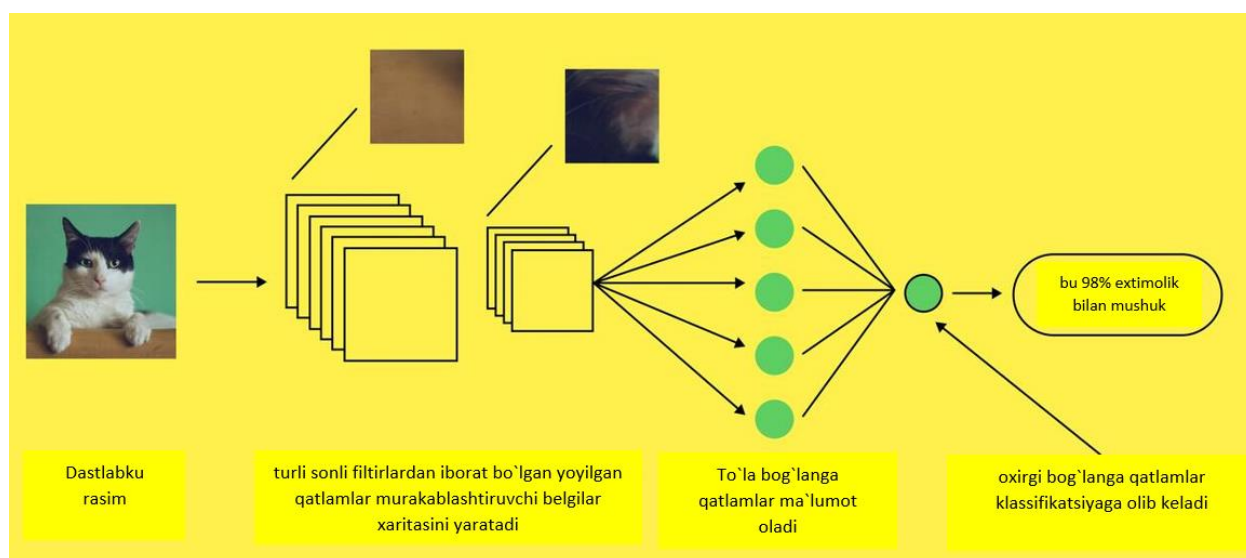
qaraganda ko‘proq bo‘lganda paydo bo‘ladi, va manfiy qiymat esa aksincha, o‘ng tomonda yorqinlik chap tomondan ko‘ra ko‘proq bo‘lsa, paydo bo‘ladi. **Bir xil tonli maydonlar** esa **nol** qiymat bilan belgilanadi, ya’ni cheksiz yoki aniq bo‘lmagan joylarni ko‘rsatadi.

3. **Boshqa svyortochki va yuqori darajadagi xususiyatlar:** Keyinchalik, boshqa svyortochniy filtrlar yordamida tasvirda **gorizontal** va **vertikal chegaralar**, **gradientlar**, **teksturalar** va **ranglar** aniqlanadi. Murakkabroq yuqori darajadagi xususiyatlar, masalan, **mushuk qulog‘i**, keyingi svyortochli qatlamlarda aniqlanadi. Bu qatlamlarning asosiy maqsadi — neyron tarmog‘iga kerakli bo‘lgan muhim xususiyatlarni ajratib olish. Yadro o‘lchami va har bir svyortochli qatlamdagi filtrlar soni dasturchi tomonidan aniqlanadi, chunki u tarmoqning arxitekturasini va ish tartibini belgilaydi.
4. **Xususiyatlar xaritalari (feature maps):** Svyortochki ishlagach, natijada **xususiyatlar xaritalari** (feature maps) hosil bo‘ladi. Bu xaritalar tasvirni raqamli formatga o‘zgartiradi. Masalan, fon qorong‘i bo‘lishi mumkin, ammo ob’ektning o‘zi yorqin oq rangda ajralib turadi. Endi bu raqamlashtirilgan xususiyatlar bo‘lib, kompyuter ulardan foydalangan holda tasvirni aniqlashi mumkin.



Neyron tarmog'i yordamida ob'ektning xususiyatlari chiqariladi. Har bir konvolyutsion qatlam arxitekturasida yangi xususiyatlar xaritasi yaratiladi. Keyingi harakatlar tarmoqning vazifasiga qarab farq qiladi:

- Agar **tasniflash** amalga oshirilsa, xususiyatlar haqida ma'lumot qo'shimcha qatlamlar yordamida "chiqariladi" va vektor — bir o'lchamli massiv, ya'ni sonlar qatori hosil qilinadi. Shundan so'ng, bu ma'lumotlarni neyron tarmog'ining chiqishidagi to'liq ulanishli qatlamga uzatish mumkin. Oxirgi to'liq ulanishli qatlamda tasvirning o'sha yoki boshqa sinfga mansub bo'lish ehtimoli aniqlanadi.



Agar neyron tarmog'iga tasniflash vazifasi topshirilgan bo'lsa, xususiyatlar xaritalari to'liq ulanishli qatlamlarga uzatiladi. Illyustratsiyada neyron tarmog'i ikkilamchi tasniflashni amalga oshiradi: masalan, mushukni itdan ajratadi. Shuning uchun oxirgi bosqichda faqat bitta neyron qoladi, u esa tasvirning biror sinfga mansub bo'lish ehtimolini aniqlaydi.

- **Segmentatsiyani amalga oshirish.** Agar segmentatsiya amalga oshirilsa, fonni xiralashtirish yoki tasvirda maska chizish uchun to'liq ulanishli qatlamlar kerak emas. Bu vazifani konvolyutsion neyron tarmog'i boshqa tuzilishga ega bo'lib, unda to'liq ulanishli qatlamlar o'rniga dekonvolyutsion konvolyutsion qatlamlar mavjud. Umuman olganda, bu qatlamlar konvolyutsiya jarayonining teskari operatsiyasini bajaradi. Bu qadamlar konvolyutsion neyron tarmoqlari asosida qurilgan kompyuter ko'rish

algoritmlarining asosi hisoblanadi. Xususiyatlarni aniqlash jarayoni oddiydan murakkabga qarab rivojlanadi: gradientlar va chegara chizig'idan yuqori darajadagi xususiyatlargacha. Agar hammasi to'g'ri amalga oshsa, neyron tarmog'i topilgan xususiyatlar asosida, masalan, attraksionlar fonidagi havo sharini tasniflashga qodir bo'ladi.

Kompyuter ko'rish modelini qo'shimcha o'qitish

Agar ma'lumotlar to'plamida (dataset) faqat 50 ta mushuk yoki it suratlari bo'lsa, tasniflash sifati past bo'ladi. Qoidaga ko'ra, ko'proq tasvirlar bo'lsa, natijalar yaxshiroq bo'ladi.

Neyron tarmoqlari tasniflashni muvaffaqiyatli amalga oshirishlari uchun, ilgari ko'rilgan va yangi tasvirlarni ajratish uchun katta o'quv bazasi kerak. Masalan, agar bir startap ilova ishlab chiqayotgan bo'lsa, foydalanuvchi kiyimga kamerani yo'naltirganda, ilova kiyim markalarini ko'rsatishi kerak. Kompyuter ko'rish modelini o'rgatgan bo'lsa ham, testlash jarayonida, modellar bir brendni boshqasidan ajratishda yaxshi ishlamayapti.

Bunday holatda yangi ma'lumotlarni qo'shish yoki tasvirlar ma'lumotlar bazasini sun'iy ravishda ko'paytirish (augmentatsiya) mumkin. Masalan, turli kiyimlar suratlarini yaqinlashtirish yoki vertikal tarzda aks ettirish. Kompyuter uchun bu boshqa tasvirlar bo'ladi. Bunday usullar neyron tarmog'ining aniqligini ko'proq ma'lumotlar bilan yaxshilashga yordam beradi.

Agar bu yordam bermasa, boshqa yo'ldan borish mumkin: ochiq manbalardan o'rgatilgan kompyuter ko'rish modelini topib, uni aniq vazifaga moslab sozlash. Buning uchun ikki usul mavjud: fine-tuning va transfer learning. Transfer learning (inglizcha "o'rgatishni ko'chirish") da, oxirgi to'liq ulanishli qatlamni o'rgatish, bu qatlam vazifa uchun sinflar sonini belgilaydi. Fine-tuning (inglizcha "nozik sozlash") esa, shu qatlamdan tashqari, mavjud qatlamlarning har qandayini o'rgatish imkonini beradi. Shunday qilib, modelni zero-dan yig'ish va o'rgatishning o'rniga, uni ma'lum bir vazifa uchun ma'lumotlar bilan moslab, faqat o'rgatish kerak bo'ladi.

Kompyuter ko'rishining iyerarxiyasi

Kompyuter ko‘rishi uchta asosiy toifaga bo‘linadi:

- **Past darajadagi ko‘rish** – obyektlarni chiqarish uchun jarayonning tasvirini o‘z ichiga oladi.
- **O‘rta darajadagi ko‘rish** – obyektlarni aniqlash va 3D-sahnalarning talqinini o‘z ichiga oladi.
- **Yuqori darajadagi ko‘rish** – sahnaning kontseptual tasvirini, masalan, faoliyat, niyat va xatti-harakatlarni o‘z ichiga oladi.

Kompyuter ko‘rishi va tasvirlarni qayta ishlash

Tasvirlarni qayta ishlash tasvirni tasvirga aylantirishni o‘rganadi. Tasvirlarni qayta ishlashning kirish va chiqish ma’lumotlari ikkala tasvirdir.

Kompyuter ko‘rishi – bu tasvir asosida fizik obyektlarning aniq, ma’noli tavsiflarini qurishdir. Kompyuter ko‘rishining natijasi 3D-sahnadagi strukturalarning tasviri yoki talqini bo‘ladi. Kompyuter ko‘rishi quyidagi sohalarda qo‘llaniladi –

Robototexnika

- **Lokalizatsiya** – robotning joylashuvini avtomatik aniqlash.
- **Navigatsiya** – robotning ma’lum bir joyga borishini boshqarish.
- **To‘siqlarni oldini olish** – robotning yo‘liga chiqadigan to‘siqlarni aniqlash va ulardan qochish.
- **Yig‘ish** (masalan, teshikka o‘rnatish, payvandlash, bo‘yash) – robotlar tomonidan turli yig‘ish jarayonlarini bajarish.
- **Manipulyatsiya** (masalan, PUMA robot manipulyatori) – robot tomonidan ob’ektlarni ushlab, harakatlantirish yoki boshqa manipulyatsiyalarni bajarish.
- **Inson va robot o‘rtasidagi o‘zaro ta’sir (HRI)** – insonlar bilan muloqot qilish va ularni xizmat ko‘rsatish uchun aqlli robototexnika.

Tibbiyot

- **Tasniflash va aniqlash** (masalan, zararlanish yoki hujayralarni tasniflash va o‘smi aniqlash).

- **2D / 3D segmentatsiya** – tasvirlarni ikki yoki uch o‘lchamli bo‘lib, muhim qismlarni ajratish.
- **3D-rekonstruksiya** – inson organlarining 3D modelini yaratish (masalan, MRT yoki UTT yordamida).
- **Robototexnik jarrohlik vizual boshqaruv bilan** – jarrohlik amaliyotlarini robotlar yordamida bajarish, vizual ma’lumotlar asosida boshqarish.

Xavfsizlik

- **Biometriya** (qorachiq, barmoq izlari, yuzni aniqlash) – shaxsni identifikatsiya qilish uchun biometrik xususiyatlar.
- **Nazorat** – ma’lum bir shubhali harakatlar yoki xatti-harakatlarni aniqlash.

Transportirovka

- **Avtonom transport vositasi** – haydovchiga ehtiyoj bo‘lmagan va o‘zi harakatlana oladigan transport vositasi.
- **Xavfsizlik**, masalan, haydovchining hushyorligini monitoring qilish – haydovchining holatini va ehtimoliy charchoqlarini kuzatish.

Sanoat avtomatlashtirish dasturi

- Sanoat tekshiruv (nuqsonlarni aniqlash)
- Assambleya
- Shtrix-kodlar va qadoqlash yorliqlarini o‘qish
- Ob’ektlarni saralash
- Hujjatlarni tushunish (masalan, OCR)

Foydali paketlarni o‘rnatish

Kompyuter ko‘rish uchun Python-da siz OpenCV nomli mashhur kutubxonadan foydalanishingiz mumkin (ochiq manba kodli kompyuter ko‘rishi). Bu asosan real vaqtda kompyuter ko‘rishi uchun mo‘ljallangan dasturiy funktsiyalar kutubxonasi. U C++ tilida yozilgan bo‘lib, asosiy interfeysi ham C++ tilida. Ushbu paketni quyidagi buyruq orqali o‘rnatishingiz mumkin.

```
pip install opencv_python-X.X-cp36-cp36m-winX.whl
```

Bu yerda X kompyuteringizda o'rnatilgan Python versiyasini, shuningdek, sizda mavjud bo'lgan win32 yoki 64-bit versiyasini anglatadi. Agar siz anaconda muhitidan foydalansangiz, OpenCV ni o'rnatish uchun quyidagi buyruqni ishlatib

```
conda install -c conda-forge opencv
```

OpenCV nima

OpenCV (Open Source Computer Vision Library) — bu kompyuter ko'rishi bilan ishlash uchun ochiq manba kodli kutubxona. Dastlab C++ tilida yozilgan, lekin Python, C, Java va MATLAB tillarida foydalanish uchun moslashtirilgan. OpenCV kutubxonasida turli funktsional imkoniyatlarga ega alohida modullar shaklida mashinaviy o'rganish asosida kompyuter ko'rish algoritmlari mavjud. Quyidagi ba'zi modullarni ko'rib chiqamiz:

- **Core Functionality (Asosiy funktsionallik)** — kutubxonaning asosiy ma'lumotlar strukturalari va funksiyalarini belgilaydi, ular boshqa modullarda ishlatiladi.
- **Image Processing (Tasvirni qayta ishlash)** — statik tasvirlar bilan ishlash imkonini beradi: oddiy rasm formatlari, masalan, PNG, JPG va boshqalar.
- **Video Analysis (Video tahlili)** — ob'ektlarning harakatlarini kuzatish va fon bilan ishlash uchun ishlatiladi.
- **Camera Calibration and 3D Reconstruction (Kamerani kalibrlash va 3D rekonstruktsiya)** — ob'ektlarning geometriyasi bilan ishlaydi, bir nechta tasvirlar yoki videolar asosida ularning 3D modellarini yaratishga imkon beradi.
- **2D Features Framework (Ikki o'lchovli xususiyatlar ramkasi)** — tasvirning boshqa tasvirlardan farq qiladigan bo'laklarini aniqlaydi, ularning konturlarini saqlaydi va ular orasida o'xshashlarini topishi mumkin.
- **Object Detection (Ob'ektlarni aniqlash)** — ob'ektlarni topish, masalan, yuzlar, mashinalar, qushlar va boshqalar.

- **High-level GUI (Yuqori darajadagi grafik interfeys)** — oddiy operatsiyalarni bajarish uchun grafik interfeyslarni yaratishga imkon beradi.
- **Video I/O (Video kirish-chiqish)** — video fayllarni o‘qish va qayta ishlash imkonini beradi.

Bu OpenCV modullarining faqat bir qismi. Hujjatlarda ular va boshqa kutubxona imkoniyatlari haqida ko‘proq o‘qishingiz mumkin .

Kompyuter ko‘rish va OpenCV qaerda qo‘llaniladi

Kompyuter ko‘rishi bizning hayotimizda allaqachon keng tarqalgan: robot-changyutgichlar va avtomobillar to‘siqlardan o‘tib ketadi, ijtimoiy tarmoqlar esa sizni tasvirlardan topib, bu sizmi deb so‘raydi. Kompyuter ko‘rishi turli vazifalarni hal qilishda ishlatiladi.

- **Matnni tanib olish.** Masalan, yorliqni tarjima qilish uchun telefoningizda tarjimon ilovasini ishga tushirib, uni matn ustiga yo‘naltirishingiz mumkin. Ilova matnni tanib olib, kerakli tilga tarjima qiladi.
- **Tasvirlar bo‘yicha qidiruv.** Tanish bo‘lmagan gulni suratga olib, uning nomini bilish uchun Google yoki "Yandex" qidiruv tizimiga yuklashingiz mumkin.
- **Kontentni moderatsiya qilish.** Ko‘plab forumlar va ijtimoiy tarmoqlar kontentni tahlil qilib, agar u platforma qoidalarini buzsa, avtomatik ravishda o‘chirishadi.
- **Biometriyani qo‘llash.** Telefonni blokirovkadan chiqarish yoki barmoq izi yoki yuz orqali tizimga kirish uchun sizning ma’lumotlaringiz kompyuter ko‘rish algoritmlari orqali qayta ishlanadi.
- **3D-modellarni yaratish.** Kompyuter ko‘rishi bir nechta surat yoki video asosida ob’ekt yoki makonning 3D modelini yaratishga imkon beradi. Bu ob’ektlar va manzaralarni tiklash uchun, shuningdek robototexnikada robotlarga xonalarni tasavvur qilishda ishlatiladi.
- **Qishloq xo‘jaligini rivojlantirish.** Kompyuter ko‘rish yordamida chorva sonini hisoblash va ekinlarni kuzatib borish mumkin: ularning holatini

baholash, zararkunandalar va kasalliklarni odam ularni sezishdan oldin aniqlash.

- **Ishlab chiqarishni boshqarish.** Rutinni bajarish uchun insonlarga tayanish oʻrniga, kompyuter koʻrish mahsulot va jarayonlarni mustaqil ravishda tahlil qilishi mumkin. Masalan, konveyerda notoʻgʻri mahsulotni aniqlaydigan kamera oʻrnatish mumkin. Insonlardan farqli oʻlaroq, bunday tizim charchamaydi va diqqatini yoʻqotmaydi.
- **Kompyuter koʻrishi tizimlarini yaratish uchun** turli xil vositalar qoʻllaniladi. Ulardan eng mashhuri — OpenCV. Uni Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota va boshqa texnologik kompaniyalar turli vazifalarni hal qilish uchun ishlatadi.
- **OpenCV tasvirlarni qanday koʻradi**
- Amaliyotga oʻtishdan oldin, OpenCV qanday tasvirlarni qabul qilishini koʻrib chiqamiz. Kutubxona tasvirlar bilan NumPy massivi sifatida ishlaydi.
- Agar tasvir kulrang rangdagi boʻlsa, u holda massiv ikki oʻlchovli boʻladi. Har bir piksel tasvirda 0 dan 255 gacha boʻlgan raqam bilan ifodalanadi, bu yerda 0 — qora, 255 — oq, va qolganlar — ular orasidagi kulranglarning har xil tonlari.
- Har bir piksel satri bir oʻlchovli massivga birlashtirilgan. Masalan, [255, 255, 77, 77, 77, 255, 255]. Bu yetti piksel satri: ikki oq, uch kulrang va yana ikki oq.
- Har bir bunday piksel satri ikkinchi darajadagi massivga birlashtirilgan.

Masalan:

```
[[ 255 , 255 , 255 , 77 , 255 , 255 , 255 ],  
 [ 255 , 255 , 77 , 77 , 77 , 255 , 255 ],  
 [ 255 , 77 , 77 , 255 , 77 , 77 , 255 ],  
 [ 255 , 255 , 77 , 77 , 77 , 255 , 255 ]  
 [ 255 , 255 , 255 , 77 , 255 , 255 , 255 ]]
```

Natijada atigi 100 marta kichikroq rasm paydo boʻladi:



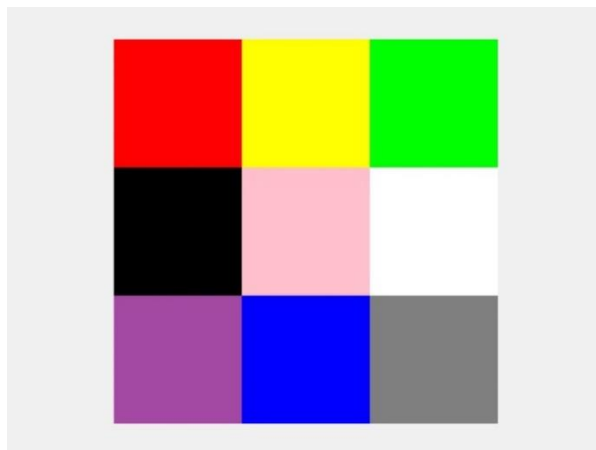
Agar tasvir rangli bo'lsa, uch o'lchovli massiv kerak bo'ladi. Har bir pikselning rangi bitta raqamdan emas, balki 0 dan 255 gacha bo'lgan uchta raqamdan iborat — bu RGB modeli bo'yicha. Faqat qizil va ko'k ranglar o'z o'rinlarini almashtirgan, shuning uchun birinchi raqam ko'k rangni, ikkinchisi yashil, uchinchi esa qizil rangni bildiradi: ya'ni BGR formati hosil bo'ladi. Mana, OpenCV tilida turli rangdagi piksellarni ifodalovchi misollar:

- **[0, 0, 0]** — qora.
- **[255, 255, 255]** — oq.
- **[255, 0, 0]** — ko'k.
- **[0, 255, 0]** — yashil.
- **[0, 0, 255]** — qizil.
- **[0, 255, 255]** — sariq.
- **[203, 192, 255]** — pushti.

Kulrang tasvirlar bilan bo'lganidek, keyingi darajadagi massiv piksel satrlarini birlashtiradi, va yuqori darajadagi massiv butun tasvirni tashkil qiladi. Mana, OpenCV'da to'qqizta rangli pikseldan iborat tasvir uchun massiv qanday ko'rinishda bo'ladi:

```
[[[ 0 , 0 , 255 ], [ 0 , 255 , 255 ], [ 0 , 255 , 0 ]],  
[[ 0 , 0 , 0 ], [ 203 , 192 , 255 ], [ 255 , 255 , 255 ]],  
[[ 164 , 73 , 163 ], [ 255 , 0 , 0 ], [ 127 , 127 , 127 ]]]
```

Va rasmning o‘zi, qulaylik uchun kattalashtirilgan:



Rasmlarni o‘qish, yozish va ko‘rsatish

Aksariyat rezyume ilovalari tasvirlarni kirish sifatida olishi va chiqish sifatida tasvirlarni yaratishi kerak. Ushbu bo‘limda siz OpenCV tomonidan taqdim etilgan funksiyalar yordamida tasvir faylini o‘qish va yozishni o‘rganasiz.

Rasm faylini o‘qish, ko‘rsatish va yozish uchun OpenCV funktsiyalari

OpenCV shu maqsadda quyidagi funktsiyalarni taqdim etadi -

- **imread() funktsiyasi** tasvirni o‘qish funktsiyasi. OpenCV imread() PNG, JPEG, TIFF va boshqalar kabi turli xil tasvir formatlarini qo‘llab-quvvatlaydi.
- **imshow() funktsiyasi** tasvirni oynada aks ettirish funktsiyasi. Oyna avtomatik ravishda tasvir hajmiga moslashadi. OpenCV imshow() PNG, JPEG, TIFF va boshqalar kabi turli xil tasvir formatlarini qo‘llab-quvvatlaydi.
- **imwrite() funktsiyasi** tasvirni yozish funktsiyasi. OpenCV imwrite() PNG, JPEG, TIFF va boshqalar kabi turli xil tasvir formatlarini qo‘llab-quvvatlaydi.

Misol

Ushbu misolda tasvirni bir formatda o‘qish, uni oynada ko‘rsatish va bir xil tasvirni boshqa formatda yozish uchun Python kodi ko‘rsatilgan. Quyida ko‘rsatilgan qadamlarni ko‘rib chiqaylik. Rasmda ko‘rsatilganidek, OpenCV paketini import qiling.

```
import cv2
```

Endi ma’lum bir rasmni o‘qish uchun imread() funktsiyasidan foydalaning.

```
image = cv2.imread('image_flower.jpg')
```

Rasmni ko'rsatish uchun **imshow()** funksiyasidan foydalaning. Rasmni ko'rishingiz mumkin bo'lgan oynaning nomi **image_flower** bo'ladi.

```
cv2.imshow('image_flower',image)  
cv2.destroyAllWindows()
```



Endi biz `imwrite()` funksiyasidan foydalanib, xuddi shu tasvirni boshqa formatga, masalan, .png formatiga yozishimiz mumkin.

```
cv2.imwrite('image_flower.png',image)
```

Chiqish qiymati `True` tasvirning xuddi shu papkadagi .png fayliga muvaffaqiyatli yozilganligini bildiradi.

True

Eslatma - `destroyAllWindows()` funksiyasi biz yaratgan barcha oynalarni shunchaki yo'q qiladi.

Rang maydonini aylantirish

OpenCV-da tasvirlar odatiy RGB rangi yordamida saqlanmaydi, lekin teskari tartibda, ya'ni BGR tartibida. Shuning uchun tasvirni o'qishda standart rang kodi BGR hisoblanadi. **cvtColor()** rangni o'zgartirish funksiyasi tasvirni bir rang kodidan boshqasiga aylantirish uchun mo'ljallangan.

Misol

Tasvirni BGR dan kulrang rangga aylantirishning ushbu misolini ko‘rib chiqing. Rasmda ko‘rsatilganidek, **OpenCV paketini** import qiling.

```
import cv2
```

Endi ma’lum bir rasmni o‘qish uchun `imread()` funksiyasidan foydalaning -

```
image = cv2.imread('image_flower.jpg')
```

`imshow()` funksiyasidan foydalangan holda ko‘rsak , bu BGR tasviri ekanligini ko‘ramiz.

```
cv2.imshow('BGR_Penguins',image)
```



Endi bu tasvirni kulrang rangga aylantirish uchun **`cvtColor()` funksiyasidan** foydalaning.

```
image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)  
cv2.imshow('gray_penguins',image)
```



Chetni aniqlash

Qo‘pol eskizni ko‘rgan odamlar ob’ektlarning ko‘p turlarini va ularning pozitsiyalarini osongina taniy oladilar. Shuning uchun qirralar inson hayotida, shuningdek, kompyuterni ko‘rish dasturlarida muhim rol o‘ynaydi. OpenCV chekkalarni aniqlash uchun **Canny()** deb nomlangan juda oddiy va foydali funksiyani taqdim etadi .

Misol

Quyidagi misol qirralarning aniq identifikatsiyasini ko‘rsatadi. Rasmda ko‘rsatilganidek, OpenCV paketini import qiling.

```
import cv2
import numpy as np
```

Endi ma’lum bir rasmni o‘qish uchun **imread()** funksiyasidan foydalaning .

```
image = cv2.imread('Penguins.jpg')
```

o‘qilgan tasvirning qirralarini aniqlash uchun **Canny()** funksiyasidan foydalaning .

```
cv2.imwrite('edges_Penguins.jpg',cv2.Canny(image,200,300))
```

Endi tasvirni qirralar bilan ko‘rsatish uchun **imshow()** funksiyasidan foydalaning.

```
cv2.imshow('edges', cv2.imread('edges_Penguins.jpg'))
```

edges_penguins.jpg nomli tasvirni yaratadi .



Yuzni tanish

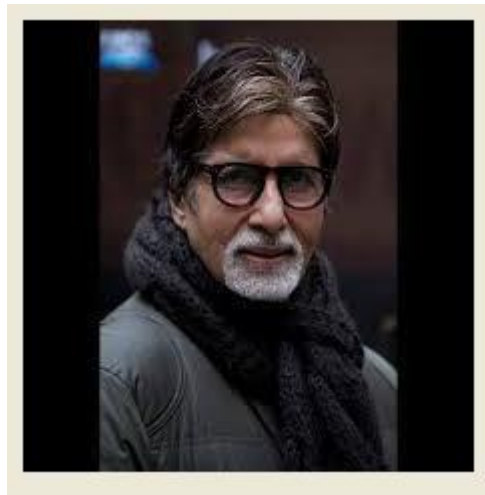
Yuzni tanib olish kompyuter ko‘rishning qiziqarli ilovalaridan biri bo‘lib, uni yanada real va futuristik qiladi. OpenCV-da yuzni tanishni amalga oshirish uchun o‘rnatilgan qurilma mavjud. Biz yuzni tanib olish uchun **Haar** kaskad klassifikatoridan foydalanmoqchimiz .

Haar kaskad ma’lumotlari

Haar klassifikatori uchun ma’lumotlar kerak. Ushbu ma’lumotlarni bizning OpenCV paketimizda topishingiz mumkin. OpenCV ni o‘rnatganingizdan so‘ng siz papka nomini ko‘rasiz **haarcascades** . Boshqa dastur uchun .xml fayllari bo‘ladi. Endi ularning barchasini boshqa foydalanish uchun nusxalang va keyin ularni joriy loyiha ichidagi yangi papkaga joylashtiring.

Misol

Quyida Amitabh Bachanning yuzini aniqlash uchun Haar kaskadidan foydalanadigan Python kodi quyidagi rasmda ko‘rsatilgan.



Rasmda ko‘rsatilganidek, **OpenCV** paketini import qiling.

```
import cv2
import numpy as np
```

Endi yuzni aniqlash uchun **HaarCascadeClassifier**-dan foydalaning.

```
face_detection= cv2.CascadeClassifier('D:/ProgramData/cascadeclassifier/
haarcascade_frontalface_default.xml')
```

Endi ma’lum bir rasmni o‘qish uchun **imread()** funksiyasidan foydalaning.

```
img = cv2.imread('AB.jpg')
```

Endi uni kulrang rangga aylantiring, chunki u kulrang rangdagi tasvirlarni qabul qiladi.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

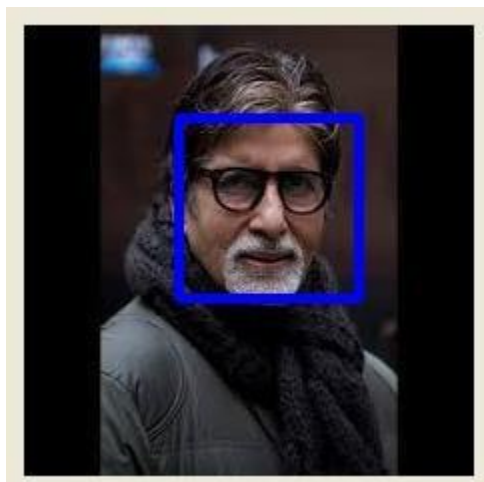
Endi **face_detection.detectMultiScale** dan foydalanib, haqiqiy yuzni aniqlashni bajaring.

```
faces = face_detection.detectMultiScale(gray, 1.3, 5)
```

Endi butun yuz bo‘ylab to‘rtburchaklar chizing.

```
for (x,y,w,h) in faces:
    img = cv2.rectangle(img,(x,y),(x+w, y+h),(255,0,0),3)
cv2.imwrite('Face_AB.jpg',img)
```

Rasmda ko'rsatilganidek, yuzni tanish bilan **Face_AB.jpg** nomli tasvirni yaratadi.



Ko'zni aniqlash

Ko'zni aniqlash kompyuter ko'rishning yana bir qiziqarli ilovasi bo'lib, uni yanada real va futuristik qiladi. OpenCV ko'zni aniqlashni amalga oshirish uchun o'rnatilgan qurilmaga ega. Ko'zni aniqlash uchun biz **Haar cascade** klassifikatoridan foydalanamiz.

Misol

Quyidagi misol, Amitabh Bachanning quyidagi rasmda ko'rsatilgan yuzini aniqlash uchun Haar kaskadidan foydalanadigan Python kodi.



Rasmda ko'rsatilganidek, OpenCV paketini import qiling.

```
import cv2  
import numpy as np
```

Endi yuzni aniqlash uchun **HaarCascadeClassifier**-dan foydalaning.

```
eye_cascade = cv2.CascadeClassifier('D:/ProgramData/cascadeclassifier/haarcascade_eye.xml')
```


Endi ma'lum bir tasvirni o'qish uchun **imread()** funksiyasidan foydalaning.

```
img = cv2.imread('AB_Eye.jpg')
```

Endi uni kulrang rangga aylantiring, chunki u kulrang rangdagi tasvirlarni qabul qiladi.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Endi yuzni haqiqiy aniqlashni amalga oshirish uchun **eye_cascade.detectMultiScale** dan foydalaning.

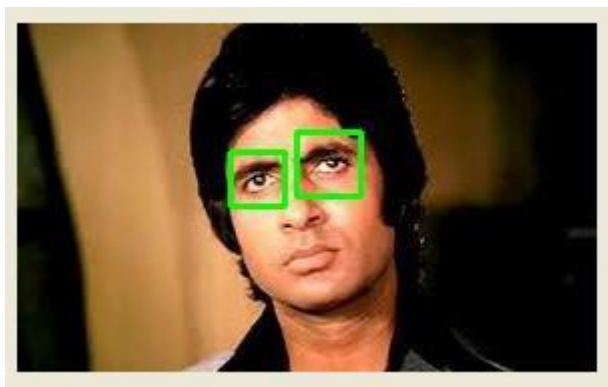
```
eyes = eye_cascade.detectMultiScale(gray, 1.03, 5)
```

Endi butun yuz bo'ylab to'rtburchak chizing.

```
for (ex,ey,ew,eh) in eyes:
```

```
    img = cv2.rectangle(img,(ex,ey),(ex+ew, ey+eh),(0,255,0),2)  
cv2.imwrite('Eye_AB.jpg',img)
```

Rasmda ko'rsatilganidek, ko'zni tanishi bilan **Eye_AB.jpg** nomli tasvirni yaratadi.



Kompyuter ko'rinishi uchun MEDIAPIPE

Hech kimga sir emaski, video yoki fotosuratda odamni topish vazifasi doimo dolzarb bo'lib kelgan. Ammo bitta vosita odamni aniqlashdan tashqari, inson yuzining to'liq niqobini, qo'llar va barmoqlarning joylashishini va umuman odamning holatini to'liq qidirishni birlashtirsa nima bo'ladi? Aynan shunday ochiq manba vositasi taniqli google tomonidan yaratilgan.

Mediapipe allaqachon juda faol va eng muhimi, fotosuratdagi ko‘plab yuzlarni aniqlash, hissiyotlarni tanib olish modellarini o‘rgatish, sport paytida mashqlarni sifatli bajarish, imo-ishora tilini yozma tilga aylantirish va boshqalar uchun samarali ishlatiladi!

Mediapipe vositasini o‘rganish bo‘yicha tajribamni siz bilan baham ko‘rmoqchiman. Va boshlash uchun men uchta modelning ishlashini kuzatishingiz mumkin bo‘lgan xulosani ko‘rsataman: qo‘llarni topish, yuzning nuqtalarini topish, odamning holatini aniqlash va oxir-oqibat, barchasi birgalikda.

Shunday qilib, har bir vosita haqida bir oz ko‘proq.

```
import cv2
import numpy as np
import mediapipe as mp
import time
import os
# Подключаем камеру
cap = cv2.VideoCapture(0)
cap.set(3, 640) # Width
cap.set(4, 480) # Length
cap.set(10, 100) # Brightness

mpHands = mp.solutions.hands
hands = mpHands.Hands(False)
npDraw = mp.solutions.drawing_utils

pTime = 0
cTime = 0

while True:
    success, img = cap.read()
    img = cv2.flip(img,1) # Mirror flip

    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = hands.process(imgRGB)
    if results.multi_hand_landmarks:
        for handLms in results.multi_hand_landmarks:
            for id, lm in enumerate(handLms.landmark):
                h,w,c = img.shape
                cx, cy = int(lm.x*w), int(lm.y*h)
                # print(id, lm)
                if id == 8 or id == 12:
                    cv2.circle(img, (cx,cy),10,(255,0,255),cv2.FILLED)

            npDraw.draw_landmarks(img, handLms, mpHands.HAND_CONNECTIONS)

    cTime = time.time()
    fps = 1/(cTime-pTime)
    pTime = cTime
    cv2.putText(img, str(int(fps)),(10,30), cv2.FONT_HERSHEY_PLAIN, 2, (255,0,0), 2)
# ФреймПейт

cv2.imshow('python', img)
```

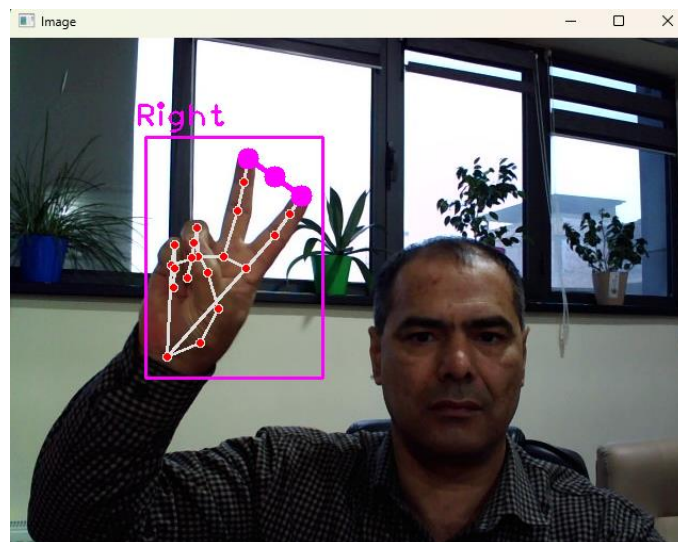
```

if cv2.waitKey(20) == 27: # exit on ESC
    break

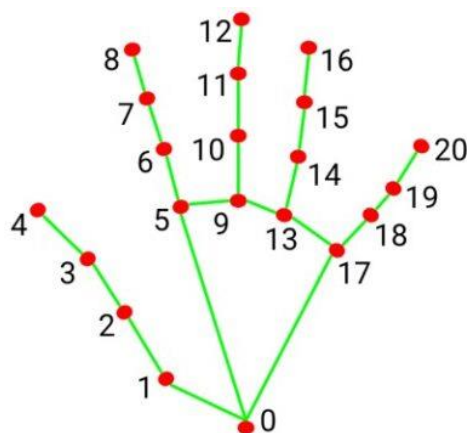
cv2.destroyAllWindows("python")
cap.release()
cv2.waitKey(1)

```

Qo'lni Aniqlash (Hand Detection)



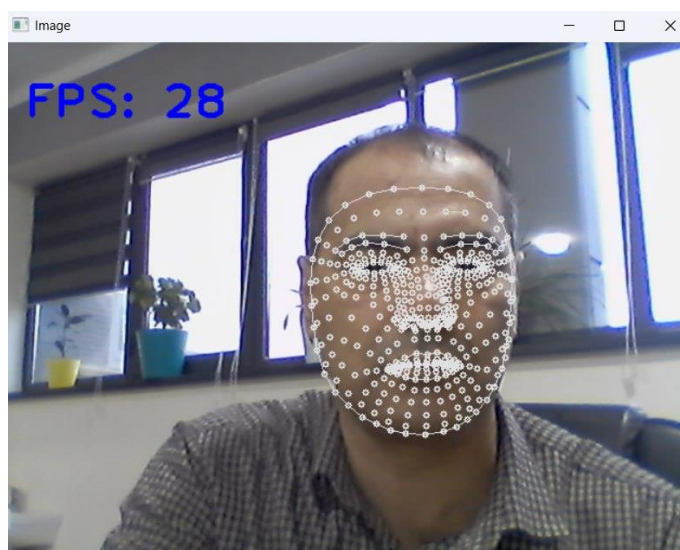
Aslida, butun nuqta fazoda 3 ta qiymatni topishdir har biri 21 nuqta qo'llar.



0. WRIST	11. MIDDLE_FINGER_DIP
1. THUMB_CMC	12. MIDDLE_FINGER_TIP
2. THUMB_MCP	13. RING_FINGER_MCP
3. THUMB_IP	14. RING_FINGER_PIP
4. THUMB_TIP	15. RING_FINGER_DIP
5. INDEX_FINGER_MCP	16. RING_FINGER_TIP
6. INDEX_FINGER_PIP	17. PINKY_MCP
7. INDEX_FINGER_DIP	18. PINKY_PIP
8. INDEX_FINGER_TIP	19. PINKY_DIP
9. MIDDLE_FINGER_MCP	20. PINKY_TIP
10. MIDDLE_FINGER_PIP	

Ushbu nuqtalarning aniq joylashishini bilib, siz turli xil modellarni yaratishingiz mumkin. Qo'lni aniqlashning o'zi juda moslashuvchan tarzda sozlanishi mumkin – qo'llarni aniqlash uchun minimal va maksimal aniqlikdan tortib, ramkadagi qo'llar soniga qadar. Yuqoridagi rasmlarda siz ikkita barmoq ustidagi tanlangan nuqtalarni ko'rishingiz mumkin va eng muhimi, yashirin barmoqlarning nuqtalari juda yaxshi topilgan.

Yuz nuqtalarini aniqlash (Face Mesh)



Ular aytganidek-natija shunday ko‘rinib turibdi! Ishning mohiyati bir xil-ma’lum nuqtalar oralig‘ida 3 koordinatani toping. Faqat endi qo‘llarda emas, balki odamning yuzida va nuqtalarning o‘zi 21 emas, balki 468! Aynan shu texnologiya yordamida Instagram foydalanuvchilari sevadigan barcha niqoblar tayyorlanadi.

```
import cv2
import mediapipe as mp
import time

cap = cv2.VideoCapture(0)
pTime = 0

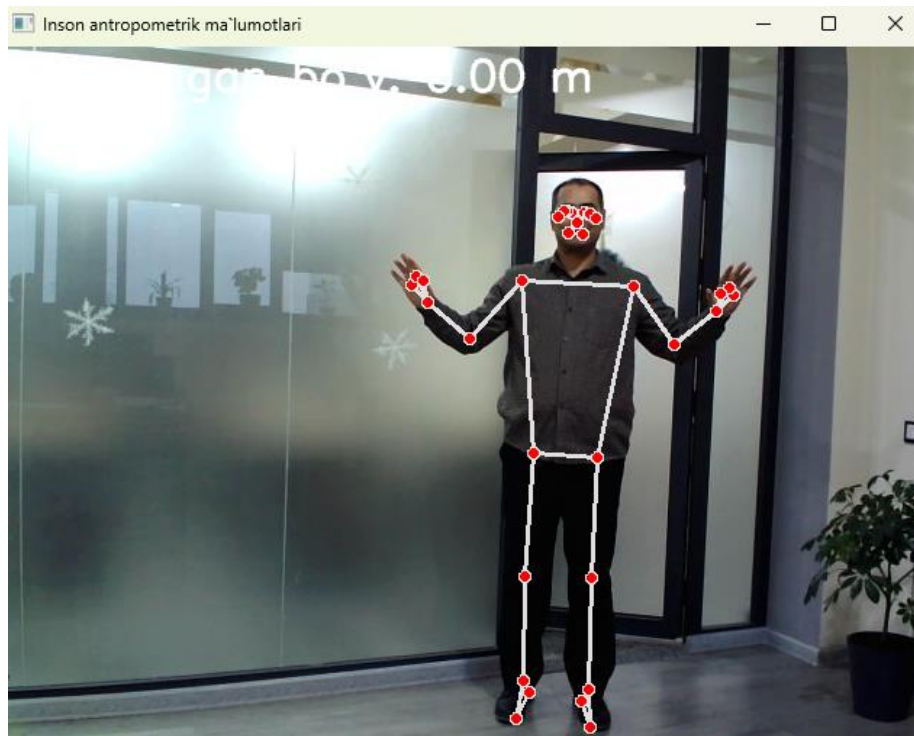
mpDraw = mp.solutions.drawing_utils
mpFaceMesh = mp.solutions.face_mesh
faceMesh = mpFaceMesh.FaceMesh(max_num_faces=2)
drawSpec = mpDraw.DrawingSpec(thickness=1, circle_radius=2)

while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = faceMesh.process(imgRGB)
    if results.multi_face_landmarks:
        for faceLms in results.multi_face_landmarks:
            mpDraw.draw_landmarks(img, faceLms, mpFaceMesh.FACEMESH_CONTOURS,
                                  drawSpec, drawSpec)
            for id, lm in enumerate(faceLms.landmark):
                #print(lm)
                ih, iw, ic = img.shape
                x, y = int(lm.x*iw), int(lm.y*ih)
                print(id, x, y)

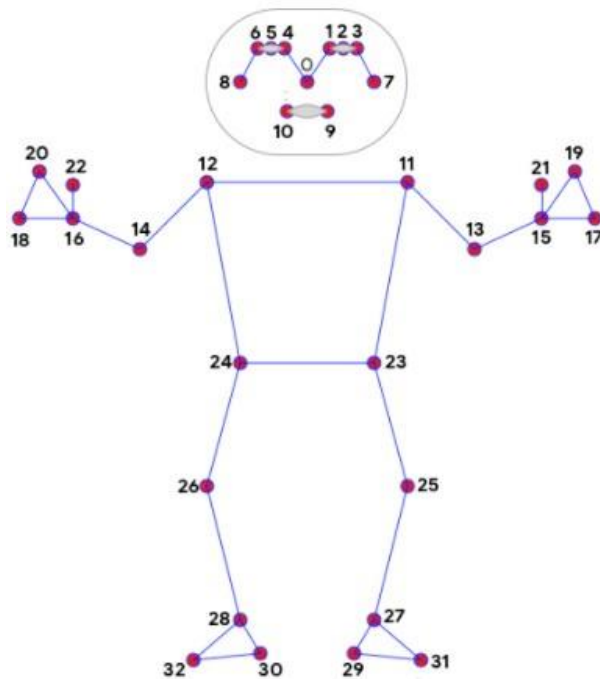
    cTime = time.time()
    fps = 1 / (cTime - pTime)
    pTime = cTime
```

```
cv2.putText(img, f'FPS: {int(fps)}', (20, 70), cv2.FONT_HERSHEY_PLAIN,  
            3, (255, 0, 0), 3)  
cv2.imshow("Image", img)  
if cv2.waitKey(1) & 0xFF == ord('q'):  
    break
```

Tanani aniqlash (Poseestimation + Holistic)



Menimcha, butun tanani topish printsipi siz uchun allaqachon aniq! 33 nuqta butun tanada uch o'lchovli bo'shliqda. Tana skleti sifatida men bir vaqtning o'zida 3 ta modelni ishga tushirdim.



0. nose	17. left_pinky
1. left_eye_inner	18. right_pinky
2. left_eye	19. left_index
3. left_eye_outer	20. right_index
4. right_eye_inner	21. left_thumb
5. right_eye	22. right_thumb
6. right_eye_outer	23. left_hip
7. left_ear	24. right_hip
8. right_ear	25. left_knee
9. mouth_left	26. right_knee
10. mouth_right	27. left_ankle
11. left_shoulder	28. right_ankle
12. right_shoulder	29. left_heel
13. left_elbow	30. right_heel
14. right_elbow	31. left_foot_index
15. left_wrist	32. right_foot_index
16. right_wrist	

```
import cv2
import numpy as np
import mediapipe as mp
import time
import os

cap = cv2.VideoCapture(0)
cap.set(3, 640) # Width
cap.set(4, 480) # Length
cap.set(10, 100) # Brightness

mp_drawing = mp.solutions.drawing_utils
mp_holistic = mp.solutions.holistic

pTime = 0
cTime = 0

while True:
    with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:
        ret, frame = cap.read()
        # Recolor Feed
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        # Make Detections
        results = holistic.process(image)
        # print(results.face_landmarks)
        # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks
        # Recolor image back to BGR for rendering
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Draw face landmarks
        mp_drawing.draw_landmarks(image, results.face_landmarks,
mp_holistic.FACEMESH_TESSELATION,
                                mp_drawing.DrawingSpec(color=(80,110,10),
thickness=1, circle_radius=1),
                                mp_drawing.DrawingSpec(color=(80,256,121),
thickness=1, circle_radius=1)
                                )

        # Right hand
```



```

        mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS)

        # Left Hand
        mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS)

        # Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS)

        cTime = time.time()
        fps = 1/(cTime-pTime)
        pTime = cTime
        cv2.putText(image, str(int(fps)),(10,30), cv2.FONT_HERSHEY_PLAIN, 2,
(255,0,0), 2) # ФреймРейт

        cv2.imshow('python', image)

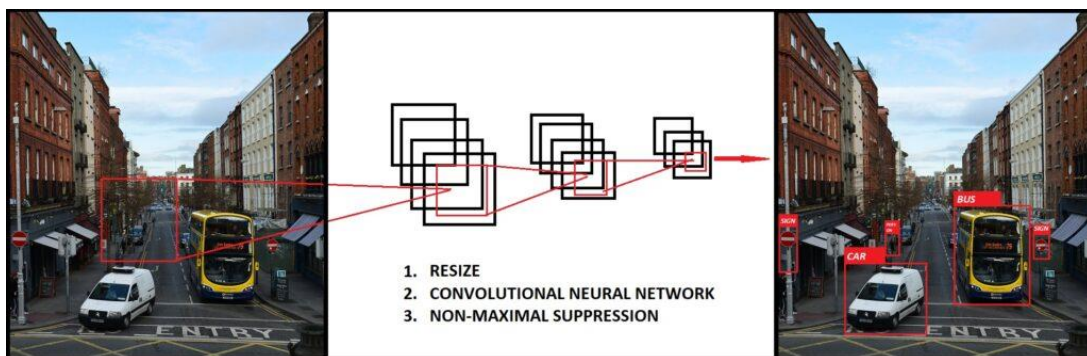
        if cv2.waitKey(1) == 27: # exit on ESC
            break

cv2.destroyAllWindows("python")
cap.release()
cv2.waitKey(1)

```

YOLO nima?

YOLO-bu ob'ektlarni aniqlashning eng yangi tizimi (tarmog'i). U Jozef Redmon tomonidan ishlab chiqilgan. YOLO-ning boshqa arxitekturalarga nisbatan eng katta afzalligi-bu tezlik. YOLO oilasining modellari juda tez va R-CNN (*Region-Based Convolutional Neural Network*) va boshqa modellardan ancha ustundir. Bu Real vaqt rejimida ob'ektlarni aniqlashga imkon beradi.

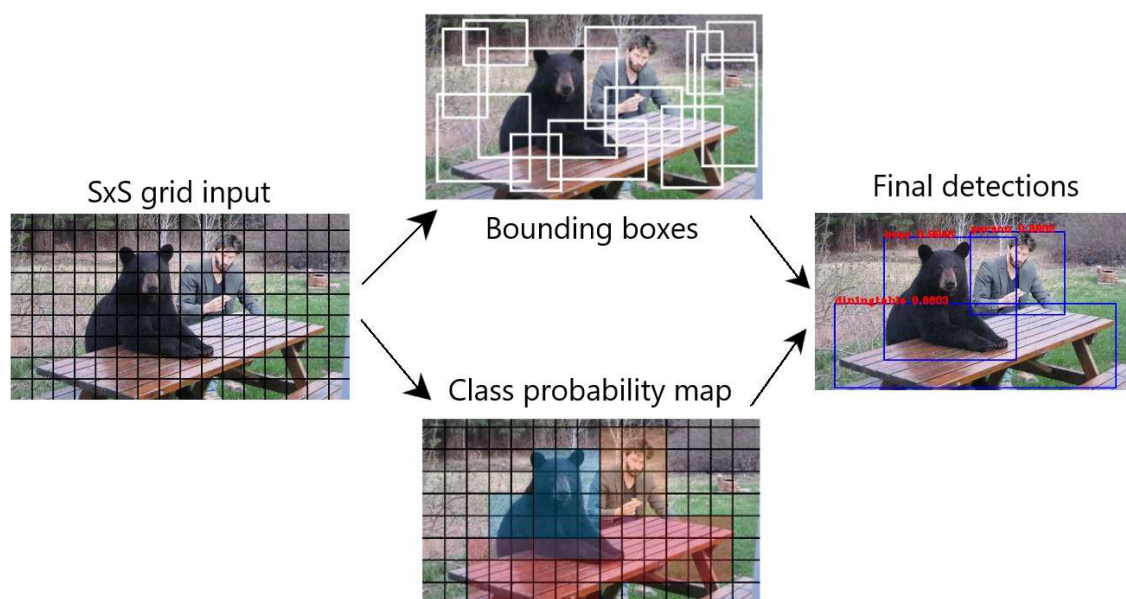


YOLO ob'ektlarni aniqlash vazifasini regressiya vazifasiga aylantirdi. U tasvir piksellaridan bounding box'lar koordinatalari va sinf ehtimolliklariga o'tadi. Shunday qilib, bitta konvolyutsion tarmoq bir nechta bounding box'larni va ushbu sohalarni o'z ichiga olgan sinflar ehtimolini bashorat qiladi.

YOLO rasmga faqat bitta qarashga muhtoj bo'lganligi sababli, toymasin oyna usuli bu vaziyatda mos emas. Buning o'rniga, rasm $S \times S$ o'lchamdagi katakchalarga bo'linadi, har bir katakchada tanib olish uchun bir nechta turli xil narsalar bo'lishi mumkin.

Birinchidan, har bir hujayra bounding box'lar sonini bashorat qilish uchun javobgardir. Shuningdek, har bir katak bounding box'm bilan cheklangan har bir maydon uchun ishonch qiymatini (*confidence value*) bashorat qiladi. Boshqacha qilib aytganda, bu qiymat ma'lum bir hududda ma'lum bir ob'ektni topish ehtimolini aniqlaydi. Ya'ni, agar biron bir panjara katakchasida ma'lum bir ob'ekt bo'lmasa, bu maydon uchun ishonch qiymati past bo'lishi muhimdir.

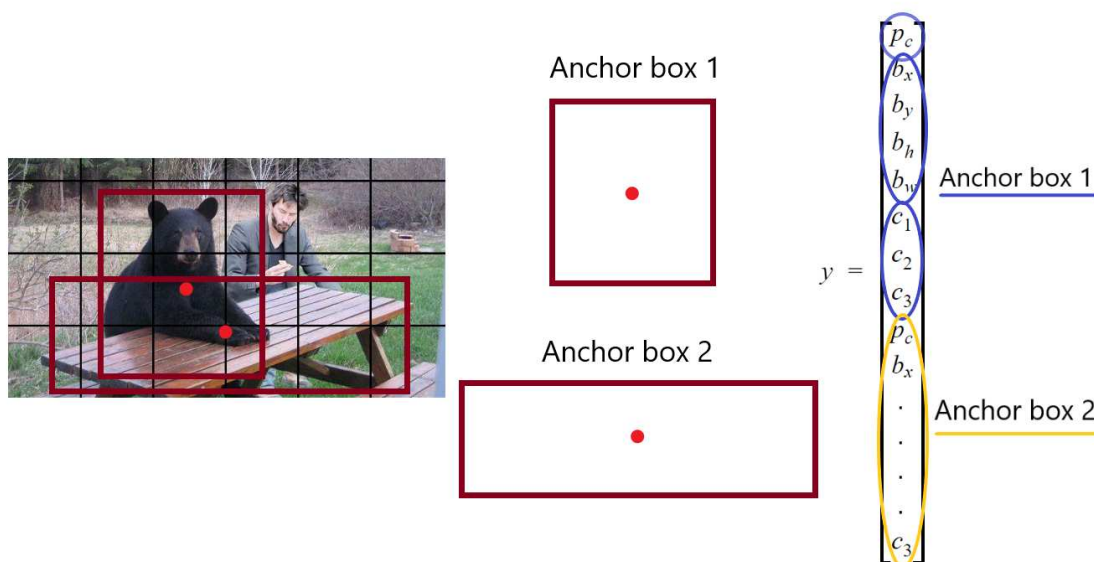
Barcha bashoratlarni tasavvur qilganimizda, biz ob'ektlar xaritasini olamiz va ishonch bilan tartibga solinadi, ramkalar.



Ikkinchidan, har bir yacheyka sinflarning ehtimolligini bashorat qilish uchun javobgardir. Bu ba'zi bir katakchada biron bir ob'ekt borligini anglatmaydi, faqat ob'ektni topish ehtimoli. Aytaylik, agar hujayra mashinani bashorat qilsa, bu mashina aslida unda mavjudligiga kafolat bermaydi. Bu shuni ko'rsatadiki, agar ob'ekt mavjud bo'lsa, unda bu ob'ekt avtomobil bo'lishi mumkin. Keling, modelning chiqishini batafsil bayon qilaylik.

YOLO bounding box'sni bashorat qilish uchun anchor boxes (Anchor Frames / fixed Frames) dan foydalanadi. Anchor box's g'oyasi ikki xil shaklni oldindan

aniqlashga to'g'ri keladi. Shunday qilib, biz ikkita bashoratni ikkita anchor box'lar bilan birlashtira olamiz (umuman olganda, biz undan ham ko'proq anchor box'lardan foydalanishimiz mumkin). Ushbu langarlar COCO DataSet (*Common Objects in Context*) va k-o'rta klasterlash (*k-means clustering*) yordamida hisoblab chiqilgan.



Bizda har bir katakcha bashorat qiladigan yacheyka mavjud:

- Xar bir bounding box' ga:
 - 4 kordinatalar (t_x , t_y , t_w , t_h)
 - 1 objectness error ob'ekt xatosi), bu ma'lum bir ob'ekt mavjudligiga ishonch ko'rsatkichidir
- Sinflarning ba'zi ehtimolliklari Agar yuqori chap burchakdan c_x , c_y ga biroz siljish bo'lsa, unda prognozlar mos keladi:

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

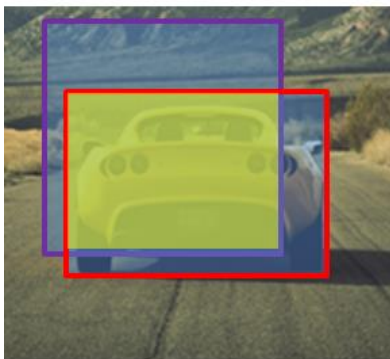
bu erda p_w (kenglik) va p_h (balandlik) bounding box ' a ning kengligi va balandligiga mos keladi.yolov2 ning oldingi versiyasida bo'lgani kabi ofsetni

taxmin qilish o'rniga, mualliflar hujayra joylashgan joyga nisbatan joylashuv koordinatalarini bashorat qilishadi.

Bu xulosa bizning neyron tarmog'imizning xulosasidir. Umuman olganda, bu erda $SxSx[B * (4+1+C)]$ xulosalar, bu erda B - ob'ektlar xaritasidagi hujayra bashorat qilishi mumkin bo'lgan chegara qutisi soni, C - sinflar soni, 4- bounding box' uchun, 1- objectness prediction (bashorat qilish ob'ektivlik). Bir o'tishda biz rasmdagi topilgan narsalarga mos keladigan kirish tasviridan chiqish tensoriga o'tishimiz mumkin. Shuni ham ta'kidlash kerakki, YOLOv3 uch xil miqyosda bounding box-ni bashorat qilmoqda.

Endi, agar biz ehtimollikni olsak va ularni ishonchli qiymatlarga ko'paytirsak, biz ushbu ob'ekt tarkibidagi ehtimollik bilan tortilgan barcha chegara qutilarini olamiz. Faqat chegarani topish bizni past ishonch bilan bashorat qilishdan xalos qiladi. Keyingi qadam uchun IOU (*Intersection over Union* / birlashma ustidagi kesishma) metrikasini aniqlash juda muhim. Ushbu metrik kesishgan hududlar maydonining Birlashgan hududlar maydoniga nisbatiga teng.

площади пересекающихся областей к площади областей объединенных.



Intersection over union (IoU)

$$= \frac{\text{size of } \text{yellow box}}{\text{size of } \text{blue box}}$$

Shundan so'ng, dublikatlar hali ham qolishi mumkin va ulardan xalos bo'lish uchun siz "maksimal bo'lmagan bostirish" dan foydalanishingiz kerak. Yuqori bo'lmagan darajalarni bostirish quyidagicha: algoritm ob'ektga tegishli bo'lish ehtimoli yuqori bo'lgan bounding box-ni oladi, so'ngra ushbu sohadagi boshqa chegara Bounding box'lar orasida eng yuqori IoU bilan birini oladi va uni bostiradi.

Hamma narsa bitta yugurishda amalga oshirilganligi sababli, ushbu model tasniflash kabi deyarli tez ishlaydi. Bundan tashqari, barcha kashfiyotlar bir vaqtning

o‘zida bashorat qilinadi, ya’ni model global kontekstni bevosita hisobga oladi. Oddiy qilib aytganda, model bilib olishi mumkin qaysi ob’ektlar odatda birga topiladi, ularning nisbiy kattaligi va ob’ektlarning joylashuvi va boshqalar.

	Type	Filters	Size	Output
	Convolutional	32	3×3	256×256
	Convolutional	64	$3 \times 3 / 2$	128×128
1x	Convolutional	32	1×1	
	Convolutional	64	3×3	
	Residual			128×128
	Convolutional	128	$3 \times 3 / 2$	64×64
2x	Convolutional	64	1×1	
	Convolutional	128	3×3	
	Residual			64×64
	Convolutional	256	$3 \times 3 / 2$	32×32
8x	Convolutional	128	1×1	
	Convolutional	256	3×3	
	Residual			32×32
	Convolutional	512	$3 \times 3 / 2$	16×16
8x	Convolutional	256	1×1	
	Convolutional	512	3×3	
	Residual			16×16
	Convolutional	1024	$3 \times 3 / 2$	8×8
4x	Convolutional	512	1×1	
	Convolutional	1024	3×3	
	Residual			8×8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Yolov3

YOLO qanday o‘rnatiladi?

Keling, bir nechta narsalarni aniqlaylik. YOLO-bu chuqur o‘rganish algorithm, shuning uchun o‘zi hech qanday o‘rnatishga muhtoj emas, buning o‘rniga bizga te algorithm-ni ishlatish uchun chuqur o‘rganish doirasi kerak. Bu erda men YOLO bilan mos keladigan eng ko‘p ishlatiladigan va ma’lum bo‘lgan 3 ta ramkani va ularning har birining afzalliklari va kamchiliklarini tasvirlab beraman:

* **Darknet** : bu YOLO ishlab chiqaruvchisidan qurilgan va yolo uchun maxsus yaratilgan ramka. Afzalligi: bu tez, u GPU yoki protsessor bilan ishlashi mumkin
Kamchilik: u olny Linux os bilan ishlaydi

* **Darkflow**: bu darknetning tensor oqimga moslashishi (boshqa chuqur moyil ramka). Afzalligi: bu tez, u GPU yoki CPU bilan ishlashi mumkin, shuningdek

Linux, oyna va Mac bilan mos keladi. Kamchilik: o'rnatish bu haqiqatan ham murakkab, ayniqsa derazalarda

* **Opencv**: shuningdek, opencv YOLO bilan ishlaydigan chuqur o'rganish tizimiga ega. Faqat siz opencv bor ishonch hosil qiling 3.4.2 kamida. Afzallik: opencv-dan boshqa hech narsa o'rnatmasdan ishlaydi. Kamchilik: u faqat protsessor bilan ishlaydi, shuning uchun siz Real vaqtda videolarni qayta ishlash uchun juda yuqori tezlikka erisha olmaysiz.

YOLO algoritmi nima uchun muhim YOLO algoritmi quyidagi sabablarga ko'ra muhimdir:

* **Speed**: ushbu algoritm aniqlash tezligini yaxshilaydi, chunki u ob'ektlarni Real vaqtda bashorat qila oladi.

* **High accuracy**: YOLO-bu minimal fon xatolari bilan aniq natijalarni beradigan bashorat qilish usuli.

* **Learning capabilities**: algoritm ob'ektlarning tasvirlarini o'rganish va ularni ob'ektni aniqlashda qo'llash imkonini beruvchi mukammal o'rganish qobiliyatiga ega.

YOLO-ni Opencv bilan qanday ishlatish kerak Biz ushbu blogda YOLO-ni Opencv bilan qanday ishlatishga e'tibor qaratamiz. Bu yangi boshlanuvchilar uchun algorythm-ni murakkab o'rnatmasdan tezda ishlash uchun eng yaxshi yondashuv. OpenCV va numpy kutubxonalarini import qilishdan boshlaymiz va keyin algorythm-ni yuklaymiz. Biz sinflarni import qilamiz:

import	cv2	as	cv
import numpy as np			

CV2 ni cv sifatida import qilish numpy ni np sifatida import qilish Biz algorythmni yuklaymiz. Algorythmni ishga tushirish bizga uchta fayl kerak:

* **Weight file**: bu o'qitilgan model, ob'ektlarni aniqlash uchun algorythm yadrosi.

* **Cfg file:** bu algorythm-ning barcha sozlamalari mavjud bo'lgan konfiguratsiya fayli.

* **Name files:** algorythm aniqlay oladigan ob'ektlarning nomini o'z ichiga oladi.

```
from wandb import Classes
# load yolo
net = cv.dnn.readNet("D:\PycharmProjects\pythonProject\YOLO-object-detection-using-Opencv-with-Python-main\yolov3.weights",
                    "D:\PycharmProjects\pythonProject\YOLO-object-detection-using-Opencv-with-Python-main\yolov3.cfg")
clasees = []
with open("coco.names", 'r') as f:
    classes = [line.strip() for line in f.readlines()]
# print(classes)
layer_name = net.getLayerNames()
output_layer = [layer_name[i - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))
```

Keyin biz tasvirni ob'ektni aniqlashni xohlagan joyga yuklaymiz va biz uning kengligi va balandligini ham olamiz.

```
# Load Image
img = cv.imread("D:/PycharmProjects/pythonProject/YOLO-object-detection-using-Opencv-with-Python-main/room_ser-01.jpg")
img = cv.resize(img, None, fx=0.4, fy=0.4)
height, width, channel = img.shape
```

Endi bizda algorythm ishlashga tayyor, shuningdek rasm, tasvirni tarmoqqa o'tkazish va aniqlashni amalga oshirish vaqti keldi.

Shuni yodda tutingki, biz tarmoqdagi to'liq tasvirni darhol ishlata olmaymiz, lekin avval uni blob-ga aylantirish uchun kerak. Blob u rasmdan xususiyatni ajratib olish va ularning hajmini o'zgartirish uchun ishlatiladi. YOLO uchta o'lchamni qabul qiladi:

- 320 320 bu kichik, shuning uchun kamroq aniqlik, lekin yaxshi tezlik
- 609 609 bu juda katta aniqlik va sekin tezlik

- 416 416 bu o‘rtada va siz ikkalasidan ham bir oz olasiz.

21-qatordagi chiqishlar bu aniqlash natijasidir. Outs-bu aniqlangan ob’ektlar, ularning joylashuvi va aniqlashga bo‘lgan ishonch haqidagi barcha ma’lumotlarni o‘z ichiga olgan massiv.

```
# Detect Objects
blob = cv.dnn.blobFromImage(
    img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
net.setInput(blob)
outs = net.forward(output_layer)
# print(outs)
```

Bu vaqtda aniqlash amalga oshiriladi va biz faqat natijani ekranda ko‘rsatishimiz kerak. Biz keyin ezilgan qator orqali halqa, biz ishonch hisoblash va biz bir ishonch Pol tanlang.

32-satrdan biz 0,5 chegara ishonchini o‘rnatdik, agar u kattaroq bo‘lsa, biz ob’ektni to‘g‘ri aniqlangan deb hisoblaymiz, aks holda biz uni o‘tkazib yuboramiz.

Ostona 0 dan 1 gacha boradi. 1 ga qanchalik yaqin bo‘lsa, aniqlashning aniqligi shunchalik katta bo‘ladi, 0 ga yaqin bo‘lsa, aniqlik shunchalik kam bo‘ladi, lekin aniqlangan ob’ektlar soni ham katta bo‘ladi.

```
# Showing Information on the screen
class_ids = []
confidences = []
boxes = []
for out in outs:
    for detection in out:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            # Object detection
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)
            # cv.circle(img, (center_x, center_y), 10, (0, 255, 0), 2 )
```

```

# Rectangle Coordinate
x = int(center_x - w/2)
y = int(center_y - h/2)
boxes.append([x, y, w, h])
confidences.append(float(confidence))
class_ids.append(class_id)

# print(len(boxes))
# number_object_detection = len(boxes)

```

Aniqlashni amalga oshirganimizda, bizda bir xil ob'ekt uchun ko'proq qutilar bo'ladi, shuning uchun biz ushbu "shovqin"ni olib tashlash uchun boshqa funktsiyadan foydalanishimiz kerak. Bu **Non maximum suppression** deb ataladi.

```

indexes = cv.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
print(indexes)

```

Biz nihoyat barcha ma'lumotlarni chiqaramiz va ularni ekranda ko'rsatamiz.

- **Box:** aniqlangan ob'ekt atrofidagi to'rtburchak koordinatalarini o'z ichiga oladi.
- **Label:** bu aniqlangan ob'ektning nomi
- **Confidence:** dan aniqlash haqida ishonch 0 uchun 1.

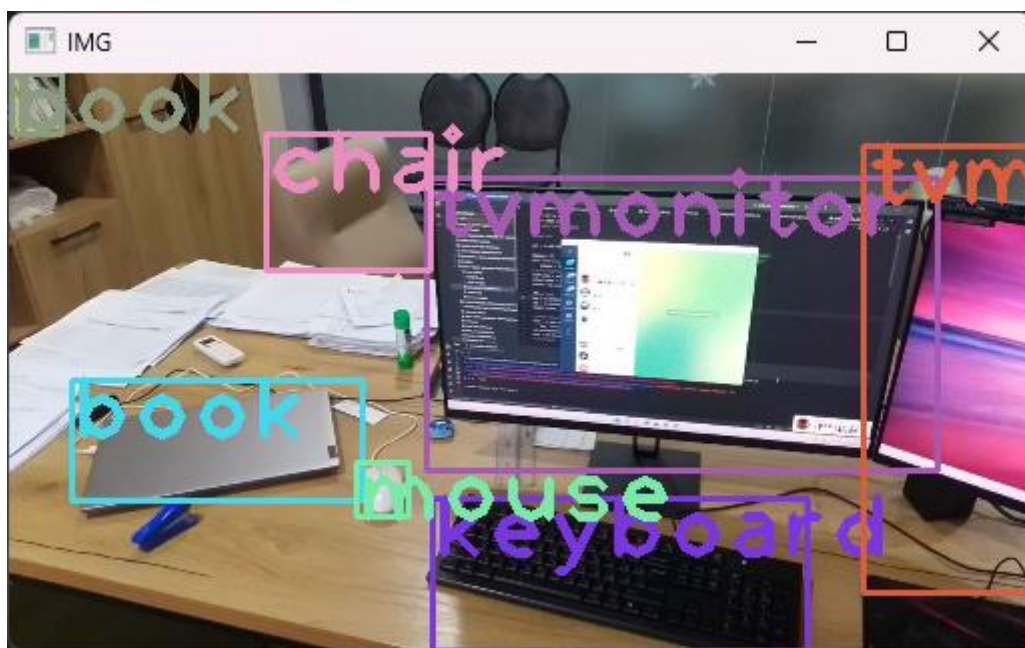
```

font = cv.FONT_HERSHEY_PLAIN
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i]
        label = str(classes[class_ids[i]])
        # print(label)
        color = colors[i]
        cv.rectangle(img, (x, y), (x + w, y + h), color, 2)
        cv.putText(img, label, (x, y + 30), font, 3, color, 3)

cv.imshow("IMG", img)
cv.waitKey(0)
cv.destroyAllWindows()

```

Ushbu blogda YOLO algoritmi va uning ob'ektni aniqlashda qanday ishlatilishi haqida umumiy ma'lumot berilgan. Ushbu uslub tezkor r-CNN va Retina-Net kabi boshqa ob'ektlarni aniqlash texnikalariga nisbatan yaxshilangan aniqlash natijalarini beradi.



O‘z-o‘zini nazorat qilish uchun savollar

1. Kompyuterli ko‘rishi?
2. Kompyuter ko‘rishining asosiy vazifalari?
3. Kompyuter ko‘rish texnologiyalari?
4. Kompyuter ko‘rish modelini qo‘shimcha o‘qitish?
5. Kompyuter ko‘rishining iyerarxiyasi?
6. Kompyuter ko‘rishi va tasvirlarni qayta ishlash?
7. OpenCV nima?
8. Kompyuter ko‘rinishi uchun MEDIAPIPE?
9. YOLO nima?

3.5-§. SUN'IY INTELLEKTD A CHUQUR O'QITISH

Sun'iy neyron tarmog'i (SNT) — bu samarali hisoblash tizimi bo'lib, uning markaziy mavzusi biologik neyron tarmoqlari bilan bog'liq analogiyadan olingan. Neyron tarmoqlari mashinada o'rganishning modellaridan biri hisoblanadi. 1980-yillarning o'rtalarida va 1990-yillarning boshlarida neyron tarmoqlarda bir qator muhim arxitektura yutuqlari erishilgan. Ushbu bobda siz chuqur o'qitish va sun'iy intellektning ushbu yondashuvi haqida ko'proq ma'lumotga ega bo'lasiz.

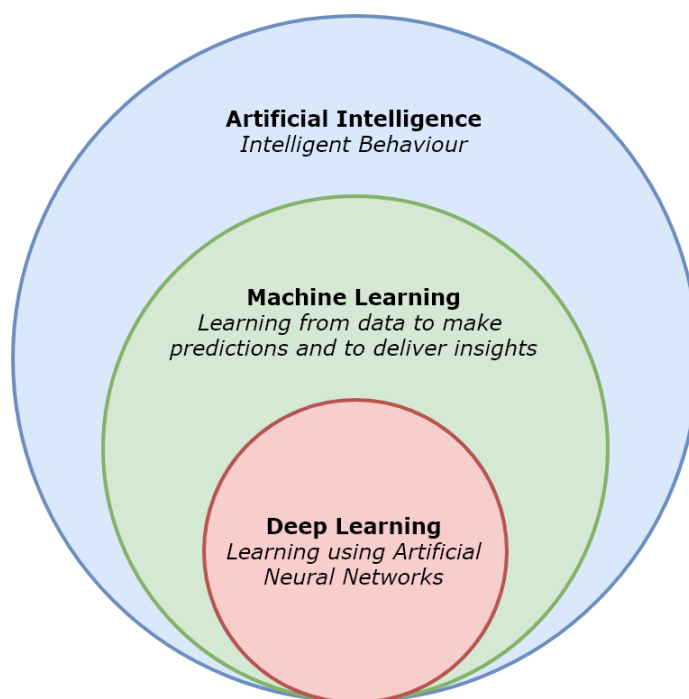
Chuqur o'rganish (deep learning)

Chuqur o'rganish (chuqur o'rganish) - bu aniq vazifalar uchun ixtisoslashtirilgan algoritmlarga emas, balki xususiyatlarni / vakillikni o'rganishga asoslangan mashinani o'rganish usullari to'plami (o'qituvchi bilan, qisman o'qituvchi bilan, o'qituvchisiz, mustahkamlash bilan). Ko'pgina chuqur o'rganish usullari 1980-yillarda (va hatto undan oldin ham) ma'lum bo'lgan, ammo natijalar sun'iy neyron tarmoqlar nazariyasidagi yutuqlarga qadar (cheklangan deb ataladigan yo'naltirilmagan grafik modelning maxsus holatidan foydalangan holda neyron tarmoqlarni oldindan tayyorlash) ta'sirchan emas edi. Boltsman mashinasi) va 2000-yillarning o'rtalarida hisoblash quvvati (shu jumladan grafik tezlatgichlar, maydonda dasturlashtiriladigan eshik massivlari va neyron protsessorlarining turli shakllari) neyron tarmoqlarning murakkab texnologik arxitekturasini yaratishga imkon bermadi. Etarli ko'rsatkichlarga ega bo'lgan va ilgari samarali hal qilib bo'lmaydigan keng ko'lamli muammolarni hal qilishga imkon beradigan, masalan, kompyuterda ko'rish, mashina tarjimai, nutqni aniqlash va ko'p hollarda hal qilish sifati hozirda solishtirish mumkin, ba'zi hollarda inson samaradorligidan oshadi.

Chuqur o'qitish inqilobi

2012-yilda Jorj E. Dahl boshchiligidagi jamoa bir dorining biomolekulyar maqsadini bashorat qilish uchun ko'p vazifali chuqur neyron tarmoqlardan foydalangan holda Merck Molecular Activity Challenge tanlovida g'olib chiqdi. 2014-yilda Xochreiter guruhi atrof-muhit, ozuqa moddalari, uy-ro'zg'or buyumlari va dori vositalarida mavjud bo'lgan kimyoviy moddalarning maqsaddan tashqari va toksik ta'sirini aniqlash uchun chuqur o'rganishdan foydalangan va AQSh Milliy

Sogʻliqni saqlash instituti, Sogʻliqni saqlash idorasi tomonidan "Tox21 Data Challenge" gʻolibi boʻlgan. Jamoat salomatligi oziq-ovqat va farmatsevtika idorasi va NCATS [23].



Mashinani oʻrganishning kichik toʻplami sifatida chuqur oʻrganishning kontseptual tasviri va sunʼiy intellektning kichik toʻplami sifatida mashinani oʻrganish (AI).

2011-2012 yillarda tasvir yoki obʼektni aniqlashda sezilarli oʻzgarishlar sezildi. Orqa tarqalish boʻyicha oʻqitiladigan konvolyutsion neyron tarmoqlari (CNN) oʻnlab yillar davomida mavjud boʻlsa-da va GPUlar koʻp yillar davomida neyron tarmoqlarni, shu jumladan CNNlarni amalga oshirayotgan boʻlsa-da, kompyuter koʻrishni rivojlantirish uchun CNN-larning GPU-larda tezkor amalga oshirilishi ishlatilgan. 2011-yilda bu yondashuv vizual naqshni aniqlash tanlovida birinchi marta gʻayritabiiy ishlashga erishdi. Shuningdek, u 2011-yilda ICDAR qoʻlyozma tanlovida va 2012-yil may oyida ISBI tasvir segmentatsiyasi tanlovida gʻolib chiqdi. 2011 yilgacha CNN kompyuterlarni koʻrish konferentsiyalarida katta rol oʻynamagan, ammo 2012 yil iyun oyida Tsiresanning CVPR yetakchi konferentsiyasidagi nutqi GPU-larda maksimal CNN sintezi koʻplab benchmark natijalarini sezilarli darajada yaxshilashi mumkinligini koʻrsatdi. 2012 yil oktyabr oyida shunga oʻxshash tizim Krizhevskiy tomonidan ishlab chiqilgan boʻlib, uning

jamoasi keng ko‘lamli ImageNet tanlovida sayoz mashinani o‘rganish usullariga nisbatan sezilarli ustunlik bilan g‘olib chiqdi. 2012 yil noyabr oyida Tsiresan jamoasi saraton kasalligini aniqlash uchun yirik tibbiy tasvir tahlili bo‘yicha ICPR tanlovida va keyingi yili xuddi shu mavzu bo‘yicha MICCAI Grand Challenge tanlovida g‘olib chiqdi. 2013 va 2014-yillarda ImageNet chuqur o‘rganish topshirig‘idagi xatolik darajasi keng ko‘lamli nutqni tanib olishdagi o‘xshash tendentsiya tufayli yanada kamaydi. Stiven Volfram tasvirni aniqlash loyihasining bir qismi sifatida ushbu yaxshilanishlarni nashr etdi.

Keyinchalik tasvir tasnifi ko‘pincha CNN va LSTM kombinatsiyasi sifatida tasvirlar uchun tavsiflarni (taglavhalarni) yaratish kabi yanada murakkab vazifaga kengaytirildi.

Ayrim tadqiqotchilarning fikricha, ImageNetning 2012-yil oktabrdagi g‘alabasi sun‘iy intellekt sohasini o‘zgartirgan “chuqur o‘rganish inqilobi”ning boshlanishi bo‘ldi.

2019-yil mart oyida Yoshua Benjio, Jefri Xinton va Yann LeKun chuqur neyron tarmoqlarni hisoblashning muhim tarkibiy qismiga aylantirgan kontseptual va muhandislik yutuqlari uchun Tyuring mukofoti bilan taqdirlandilar.

Chuqur o‘rganish algoritmlari

Ular turli vazifalar uchun o‘zlarining chuqur o‘rganish modellaridan foydalanadilar. Keling, asosiylarini ko‘rib chiqaylik:

Tasvirni qayta ishlash uchun konvolyutsion neyron tarmoqlari (CNN) ishlatiladi. Bunday tarmoqlarda konvolyutsiya deb ataladigan matematik operatsiya yordamida kirish ma’lumotlaridan asosiy xususiyatlarni chiqaradigan qatlamlar mavjud. Shuning uchun ism. Tarmoq tasvirlardagi ob’ektlar va naqshlarni ajratishni o‘rganadi, bu tasniflash, ob’ektlarni aniqlash va tasvirni segmentatsiyalash uchun foydalidir. Misol uchun, mushuk tasvirida konvolyutsion qatlamlar yuzni, quloqlarni va chiziqlarni ta’kidlashi mumkin.

Takroriy neyron tarmoqlar (RNN) oldingi holatlarni eslab qolishi mumkin, bu ularni kontekst va oldingi bosqichlarga bog‘liqlik muhim bo‘lgan vazifalar uchun mos qiladi. Xuddi shu tamoyilga ko‘ra, odam gapdagi har bir so‘zni o‘qiydi, uni

eslab qoladi va shuning uchun ma'nosini tushunadi. RNNlar mashina tarjimai, matn tahlili, nutqni yaratish va izchil ma'lumotlar tuzilishi muhim bo'lgan boshqa sohalarda qo'llaniladi.

Generativ raqib tarmoqlari (GAN) ikki qismdan iborat: generator va diskriminator. Generator rasm yoki matn kabi yangi ma'lumotlarni yaratadi va diskriminator bu ma'lumotlarni haqiqiy ma'lumotlar bilan taqqoslaydi. Bunday neyron tarmoqni o'rgatish rassom va tanqidchi o'rtasidagi doimiy kurashdir. Generator ko'proq va ko'proq real ma'lumotlarni yaratishga intiladi va diskriminator ularni haqiqiylardan ajratib olishga intiladi. GANlar real tasvirlarni yaratish, ma'lumotlar sifatini yaxshilash, matn yaratish va haqiqatga yaqin kontent zarur bo'lgan boshqa vazifalarni bajarish uchun ishlatiladi.

Ba'zida bitta algoritmi vazifani bajarish uchun etarli emas. Misol uchun, Midjourney kabi vositani ishlab chiqish uchun sizga GAN, RNN, shuningdek, tasvirni vektor va orqaga aylantiradigan maxsus dekodeklar kerak bo'ladi.

Deep Learning qayerda qo'llaniladi?

Chuqur mashinalarni o'rganishni qo'llash doirasi texnologiya va tadqiqotlar rivojlanishi bilan birga doimiy ravishda kengayib bormoqda. U qo'llaniladigan bir nechta sohalari:

Marketing. Chuqur o'rganish orqali kompyuterlar inson tilini tushunadi va qayta ishlaydi. Misol uchun, ChatGPT va Alice matn yaratishi, savollarga javob berishi va hatto san'at asarini yaratishi mumkin. Deyarli odam kabi muloqot qiladigan chatbotlardan qo'ng'iroq markazlari, veb-saytlar va ilovalarda yordamchi agentlar o'rniga foydalanish mumkin. Chuqur o'rganish iste'molchi xatti-harakatlari, afzalliklari va xaridlar tarixi haqidagi ma'lumotlarni tahlil qilish uchun ishlatiladi. Ushbu ma'lumotlar asosida shaxsiylashtirilgan tavsiyalar va reklama takliflari yaratiladi. Masalan, axborot byulletenlari, bannerlar yoki push-bildirishnomalar. Ma'lumotlar kompaniyalarga mahsulotlarga bo'lgan talabni bashorat qilish va marketing kampaniyalarini rejalashtirishga yordam beradi. Murakkab neyron tarmoqlar ijtimoiy tarmoqlardagi sharhlar va sharhlar kabi matnli

ma'lumotlarning fikrlari va hissiy ohangini tahlil qilishi mumkin. Kompaniya ma'lumotlarni qo'lda qayta ishlashga vaqt sarflamaydi.

Xavfsizlik. Chuqur o'rganish kompyuterlarning tasvirlarni ko'rish va tushunish qobiliyatini yaxshiladi. Chuqur o'rganish aeroportlarda yoki tadbirlarda yuzni tanish xavfsizlik tizimlaridan foydalanadi.

Tibbiyot. Chuqur o'rganish algoritmlari rentgen tasvirlarini tahlil qilishi, diagnostika aniqligi va tezligini oshirishi mumkin. Masalan, fotosuratlarda xavfli kasalliklarning dastlabki bosqichlarida belgilarini aniqlash.

Ommaviy axborot vositalari va ijodkorlik. Neyron tarmoqlardan olingan matnlar haqida hali ham ko'plab shikoyatlar mavjud, ammo Midjourneydagi rasmlarni ba'zan rassomlar asarlaridan ajratib bo'lmaydi. Neyron tarmoqlar bannerlar va afishalar, jurnal muqovalari, o'yin qahramonlari, video va komikslar yaratish uchun ishlatiladi.

O'z-o'zidan boshqariladigan avtomobillar. Chuqur o'rganish neyron tarmoqlari avtomobillarni atrofdagi dunyo bilan o'zaro munosabatda bo'lishni o'rgatish uchun ishlatiladi. Masalan, piyodalar va boshqa yo'l harakati qatnashchilarining harakatini bashorat qilish.

Moliya. Chuqur mashinani o'rganish bozor tahlili, tendentsiyalarni bashorat qilish, firibgarlikka qarshi tizimlar va shaxsiylashtirilgan moliyaviy tavsiyalar uchun ishlatiladi. Masalan, birja narxlarining o'zgarishini bashorat qiluvchi va portfelni optimallashtirish bo'yicha tavsiyalar beradigan algoritmlar.

Chuqur o'rganishsiz mashinani o'rganish

Chuqur o'qitish hisoblash texnologiyalarining o'n yillik portlovchi rivojlanishi natijasida paydo bo'ldi va bu sohada jiddiy raqobatchi bo'ldi. Shunday qilib, chuqur o'qitish — bu mashinada o'rganishning maxsus turi bo'lib, uning algoritmlari inson miyasi tuzilishi va funktsiyalariga asoslanadi. Mashinada o'rganish va chuqur o'qitish o'rtasidagi farq chuqur o'qitish — bu hozirgi kunda eng kuchli mashinada o'rganish usuli. U juda samarali, chunki u muammoni qanday hal qilishni o'rganar ekan, uni taqdim etishning eng yaxshi usulini ham o'rganadi. Quyida chuqur o'qitish va mashinada o'rganish taqqoslanadi:

Ma'lumotlarga bog'liqlik - Birinchi farq, bu DL (chuqur o'qitish) va ML (mashinada o'rganish) algoritmlarining ma'lumotlar hajmi oshgan sari qanday ishlashida ko'rinadi. Ma'lumotlar katta bo'lganda, chuqur o'qitish algoritmlari juda yaxshi ishlaydi.

Mashinaga bog'liqlik - Chuqur o'qitish algoritmlari samarali ishlash uchun yuqori unumdorlikka ega mashinalarga ehtiyoj sezadi. Boshqa tomondan, mashinada o'rganish algoritmlari kam quvvatli mashinalarda ham ishlashi mumkin.

Funktsiyalarni ajratib olish - Chuqur o'rganish algoritmlari yuqori darajadagi xususiyatlarni ajratib olishi va ulardan o'rganishga harakat qilishi mumkin. Boshqa tomondan, mashinani o'rganish yordamida olingan xususiyatlarning aksariyatini aniqlash uchun mutaxassis talab qilinadi.

Chuqur o'qitish ishlash vaqti - Ishlash vaqti algoritmda ishlatiladigan ko'plab parametrlar bilan bog'liq. Chuqur o'qitish algoritmlari mashinada o'rganish algoritmlaridan ko'proq parametrlarni talab qiladi. Shunday qilib, DL algoritmlarining ish vaqti, ayniqsa o'rgatish vaqti, ML algoritmlaridan ancha uzoq. Lekin, DL algoritmlarini test qilish vaqti ML algoritmlaridan kamroq.

Muammolarni hal qilish yondashuvi - Chuqur o'qitish muammoni boshidan oxirigacha hal qiladi, shu bilan birga mashinada o'rganish an'anaviy muammoni hal qilish usulini qo'llaydi, ya'ni uni qismlarga ajratib, har birini alohida o'rganadi.

Konvolyutsion neyron tarmog'i (Convolutional Neural Network - CNN).

Konvolyutsion neyron tarmoqlari odatiy neyron tarmoqlaridan farq qilmaydi, chunki ular ham o'qitiladigan og'irliklar va qiyoslashlarga ega neyronlardan iborat. Odatdagi neyron tarmoqlari kirish ma'lumotlarining tuzilishini e'tiborga olmaydi va barcha ma'lumotlar tarmoqqa uzatilishdan oldin bir o'lchovli massivga aylantiriladi. Bu jarayon oddiy ma'lumotlar uchun mos bo'lsa-da, agar ma'lumotlarda rasmlar bo'lsa, bu jarayon noqulay bo'lishi mumkin.

CNN bu muammoni oson hal qiladi. Ular rasmning 2D tuzilishini hisobga olgan holda, tasvirga xos xususiyatlarni chiqarib olish imkonini beradi. Shunday qilib, CNN ning asosiy maqsadi - rasmning xom ma'lumotlaridan kirish qatlamidan

to'g'ri sinfga o'tish. Oddiy NNs va CNNs orasidagi yagona farq - bu kirish ma'lumotlarini qayta ishlash va qatlamlar turidir.

CNN arxitekturasining umumiy ko'rinishi

Arxitektura nuqtai nazaridan, an'anaviy neyron tarmoqlar kirish ma'lumotlarini qabul qiladi va ularni bir qator yashirin qatlamlar orqali o'zgartiradi. An'anaviy neyron tarmoqlarning asosiy kamchiligi shundaki, ular to'liq hajmli tasvirlar uchun yaxshi miqyosda emas.

CNN arxitekturasida neyronlar kenglik, balandlik va chuqurlik deb ataladigan uch o'lchamda joylashtirilgan. Joriy qatlamdagi har bir neyron oldingi qatlam chiqishining kichik qismiga ulanadi. Bu kiritilgan rasmga $N \times N$ **filtr** qo'llashga o'xshaydi. U barcha tafsilotlarni olishiga ishonch hosil qilish uchun **M filtrlaridan foydalanadi**. Ushbu **M** filtrlari qirralar, burchaklar va h.k. kabi xususiyatlarni ajratuvchi xususiyatli ekstraktorlardir.

Arxitektura nuqtai nazaridan odatiy neyron tarmoqlari kirish ma'lumotlarini qabul qiladi va ularni bir nechta yashirin qatlamlar orqali o'tkazadi. Har bir qatlam boshqa qatlam bilan neyronlar yordamida bog'lanadi. Odatiy neyron tarmoqlarining asosiy kamchiligi shundaki, ular to'liq tasvirlarga nisbatan yomon miqyoslanadi.

CNN arxitekturasida neyronlar uch o'lchovda joylashgan bo'lib, ular kenglik, balandlik va chuqurlik deb ataladi. Hozirgi qatlamdagi har bir neyron oldingi qatlamning kichik bir qismini chiqarish bilan bog'langan. Bu kirish tasviriga $N \times N$ filtrini qo'llashga o'xshaydi. Barcha detallarga ishonch hosil qilish uchun u **M** ta filtrni ishlatadi. Ushbu **M** filtrlar ob'ektlarni ajratib olish vositalaridir, ular chetlar, burchaklar va boshqalarni chiqaradi.

CNN Qurish uchun ishlatiladigan qatlamlar

CNN yaratishda quyidagi qatlamlar ishlatiladi:

- **Kirish qatlam** – bu qatlam rasmning xom ma'lumotlarini o'z holida qabul qiladi.
- **Konvolyutsion qatlam** – bu qatlam CNNning asosiy qurilish blokidir va hisoblashlarning katta qismini bajaradi. Ushbu qatlam neyronlar o'rtasida

konvolyutsiyalarni hisoblaydi va kirish ma'lumotlarining turli qismlarini o'rganadi.

- **Tuzatishli chiziqli birlik qatlam** – bu qatlam oldingi qatlamning chiqish ma'lumotlariga faollashtirish funksiyasini qo'llaydi. U tarmoqda noxatolikni qo'shadi, shunda u har qanday funksiya turiga yaxshi moslashadi.
- **Birlashtirish qatlam** – birlashtirish tarmoqda ilgarilab borar ekan, faqat muhim qismlarni saqlashga yordam beradi. Birlashtirish qatlamlari har bir kirish ma'lumotlari chuqurlik kesiti uchun mustaqil ishlaydi va uning fazoviy o'lchamini o'zgartiradi. U **MAX** funksiyasidan foydalanadi.
- **To'liq bog'lanish qatlam / chiqish qatlam** – bu qatlam oxirgi qatlamdagi chiqish ballarini hisoblaydi. Natijada olingan natija $1 \times 1 \times L$ o'lchamga ega bo'ladi, bu yerda L – o'quv to'plamining sinflar soni.

Kerakli Python paketlarini o'rnatish

Kerasdan foydalanishingiz mumkin, bu Python tilida yozilgan yuqori darajadagi neyron tarmoqlari uchun API bo'lib, TensorFlow, CNTK yoki Theano ustida ishlay oladi. U Python 2.7-3.6 bilan mos keladi. Keras haqida batafsil ma'lumotni <https://keras.io/> topishingiz mumkin. Kerasni o'rnatish uchun quyidagi komandalarni ishlating: Keras-ni o'rnatish uchun quyidagi buyruqlardan foydalaning.

```
pip install keras
```

Conda muhitida siz quyidagi buyruqdan foydalanishingiz mumkin -

```
conda install -c conda-forge keras
```

ANN yordamida chiziqli regressorni qurish

Ushbu bo'limda siz sun'iy neyron tarmoqlardan foydalangan holda chiziqli regressorni qanday qurishni o'rganasiz. Bunga erishish uchun **KerasRegressor** dan foydalanishingiz mumkin . Ushbu misolda biz Bostondagi uylar uchun 13 raqamli

qiymatga ega Bostondagi uy narxlari ma'lumotlar to'plamidan foydalanamiz. Mana bir xil ob'ekt uchun Python kodi. Quyida ko'rsatilganidek, barcha kerakli paketlarni import qiling.

```
import numpy
import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

Endi mahalliy katalogda saqlangan ma'lumotlar to'plamimizni yuklab oling.

```
dataframe = pandas.read_csv("/Usrrs/admin/data.csv", delim_whitespace = True, header = None)
dataset = dataframe.values
= pandas.read_csv("/Usrrs/admin/data.csv", delim_whitespace = True, header = None)
dataset = dataframe.values
```

Endi ma'lumotlarni kirish va chiqish o'zgaruvchilariga ajrating, ya'ni. X va Y.

```
X = dataset[:,0:13]
Y = dataset[:,13]
```

Biz asosiy neyron tarmoqlardan foydalanayotganimiz uchun - modelini aniqlang.

```
def baseline_model():
    baseline_model():
```

Endi modelni quyidagi tarzda yarating.

```
model_regressor = Sequential()
model_regressor.add(Dense(13, input_dim = 13, kernel_initializer = 'normal',
    activation = 'relu'))
model_regressor.add(Dense(1, kernel_initializer = 'normal'))
```

Keyingisi kompilyatsiya qilaylik model.

```
model_regressor.compile(loss='mean_squared_error', optimizer='adam')  
return model_regressor
```

Endi takrorlanish uchun tasodifiy urug'ni quyidagicha tuzating.

```
seed = 7  
numpy.random.seed(seed)
```

Scikit-learn - da regressiya hisoblagichi sifatida foydalanish uchun Keras o'rash ob'ekti **KerasRegressor** deb ataladi. Ushbu bo'limda biz ushbu modelni standartlashtirilgan ma'lumotlar to'plamidan foydalangan holda baholaymiz.

```
estimator = KerasRegressor(build_fn = baseline_model, nb_epoch = 100, batch_size = 5,  
verbose = 0)  
kfold = KFold(n_splits = 10, random_state = seed)  
baseline_result = cross_val_score(estimator, X, Y, cv = kfold)  
print("Baseline: %.2f (%.2f) MSE" % (Baseline_result.mean(),Baseline_result.std()))
```

Modelani baholash natijasi

Yuqorida ko'rsatilgan kod natijasida modelning ko'rinmas ma'lumotlar bo'yicha ishlash samaradorligi baholanadi. Bu o'rtacha kvadrat xatolikni, shu jumladan barcha 10 bosqichli kesma tekshiruvining o'rtacha qiymati va standart og'ishlarini o'z ichiga oladi.

Tasvirlarni tasniflash: Chuqur o'qitishni qo'llash

Konvolyutsion neyron tarmoqlari (CNN) tasvirlarni tasniflash muammosini hal qiladi, ya'ni kirish tasvirining qaysi sinfga mansub ekanligini aniqlaydi. Siz Keras kutubxonasini ishlatishingiz mumkin. E'tibor bering, biz quyidagi havolada keltirilgan ma'lumotlar to'plamidan foydalanib, mushuk va itlarning tasvirlarini o'rgatish va test qilish uchun foydalanamiz: <https://www.kaggle.com/c/dogs-vs-cats/data>. Rasmda ko'rsatilganidek, muhim keras kutubxonalari va paketlarini import qiling. Sequential paketi neyron tarmoqlarini **Sequential Network** sifatida ishga tushiradi. Bu arxitektura neyron tarmoqlarini qatlamma-qatlam joylashtirib,

tarmoqlarni ketma-ket bog‘lashni osonlashtiradi. Bu usul ko‘plab oddiy tarmoq arxitekturalari uchun ideal hisoblanadi.

```
from keras.models import Sequential
```

Conv2D deb nomlangan quyidagi paket CNN ning birinchi bosqichi bo‘lgan konvolyutsiya operatsiyasini bajarish uchun ishlatiladi.

```
from keras.layers import Conv2D
```

MaxPoling2D deb nomlangan quyidagi paket CNN ning ikkinchi bosqichi bo‘lgan birlashma operatsiyasini bajarish uchun ishlatiladi.

```
from keras.layers import MaxPooling2D
```

Flatten deb nomlangan keyingi paket barcha hosil bo‘lgan 2D massivlarni bitta uzun, uzluksiz chiziqli vektorga aylantirish jarayonidir.

```
from keras.layers import Flatten
```

Dense deb nomlangan navbatdagi paket CNN ning to‘rtinchi bosqichi bo‘lgan to‘liq neyron tarmoq ulanishini amalga oshirish uchun ishlatiladi.

```
from keras.layers import Dense
```

Hozir yaratish ob‘ekt sinf ketma-ketligi.

```
S_classifier = Sequential()
```

Endi keyingi qadam konvolyutsiya qismini kodlashdir.

```
S_classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))
```

Bu erda **relu** - rektifikator funktsiyasi.

Endi CNN ning navbatdagi bosqichi konvolyutsiya qismidan keyin olingan xususiyat xaritalarini birlashtirish operatsiyasidir.

```
S_classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

Endi barcha birlashtirilgan tasvirlarni yaltiroq vektorga aylantiring.

```
S_classifier.add(Flatten())
```

Keyinchalik, to'liq bog'langan qatlamni yarating.

```
S_classifier.add(Dense(units = 128, activation = 'relu'))
```

Bu erda 128 - yashirin birliklar soni. Yashirin birliklar sonini 2 kuch sifatida aniqlash odatiy amaliyotdir. Endi chiqish qatlamini quyidagicha ishga tushiring.

```
S_classifier.add(Dense(units = 1, activation = 'sigmoid'))
```

Endi CNN kompilyatsiya qiling, bizda o'rnatilgan.

```
S_classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Bu erda optimallashtiruvchi parametr stokastik gradient tushish algoritmini tanlash uchun, yo'qotish parametri yo'qotish funktsiyasini tanlash uchun, metrika parametri esa ishlash ko'rsatkichini tanlash uchundir. Endi tasvirni kengaytiring va keyin uni neyron tarmoqlarga moslang.

```
train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2,  
zoom_range = 0.2,  
horizontal_flip = True)  
test_datagen = ImageDataGenerator(rescale = 1./255)  
  
training_set =  
train_datagen.flow_from_directory("../Users/admin/training_set", target_size =  
(64, 64), batch_size = 32, class_mode = 'binary')
```

Endi ma'lumotlarni biz yaratgan modelga moslang.

```
classifier.fit_generator(training_set, steps_per_epoch = 8000, epochs =  
25, validation_data = test_set, validation_steps = 2000)
```

Bu yerda step_per_epoch trening tasvirlari sonini bildiradi.

Endi model o'qitilgandan so'ng, biz undan quyidagi tarzda bashorat qilish uchun foydalanishimiz mumkin.

```
from keras.preprocessing import image
test_image = image.load_img('dataset/single_prediction/cat_or_dog_1.jpg',
target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = classifier.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
prediction = 'dog'
else:
prediction = 'cat'
```

O'z-o'zini nazorat qilish uchun savollar

1. Chuqur o'rganish (deep learning)?
2. Chuqur o'rganish algoritmlari qanday?
3. Deep Learning qayerda qo'llaniladi?
4. Chuqur o'rganishsiz mashinani o'rganish?
5. Konvolyutsion neyron tarmog'i CNN?
6. Qurish uchun ishlatiladigan qatlamlar?

GLOSSARIY

SI (Sun'iy Intellekt) - bu kompyuter tizimlari yoki dasturiy ta'minotlar tomonidan amalga oshiriladigan intellektual vazifalar, odatda, inson aql-idroki bilan bog'liq bo'lgan vazifalar. SI o'ziga o'qish, muammolarni hal qilish, qarorlar qabul qilish kabi faoliyatlarni oladi.

Superintellekt - bu sun'iy intellektning inson aqlidan ancha yuqori darajada bo'lgan shakli. Superintellekt nafaqat insonlardan tez va samarali fikrlashda ustun, balki o'zini o'zgartirish va yangi bilimlarni olishda ham ustun bo'lishi mumkin.

Eksponensial o'sish - bu tez rivojlanish yoki o'sishning tezlashuvi, bunda o'sish har bir bosqichda o'zgaradi. Bostrom bu o'sishni sun'iy intellektning rivojlanishida muhim xavf sifatida ko'radi, chunki u nazoratsiz rivojlanishiga olib kelishi mumkin.

Narvonli rivojlanish (Staircase Scenario) - bu siyoiy va texnologik rivojlanishning bosqichma-bosqich tarzda, har bir yangi avlod texnologiyalarining avvalgi tizimlarga integratsiyalanishi orqali amalga oshirilishi.

Samarali boshqarish (Value Alignment) - bu sun'iy intellekt tizimlarining insoniyatning axloqiy va ijtimoiy qadriyatlariga mos kelishini ta'minlash jarayoni. Bostrom buni SI rivojlanishining eng muhim jihati sifatida ko'radi.

Intellektual bohon (Intelligence Explosion) - bu sun'iy intellektning o'zini takomillashtirishda tezda va kutilmagan o'sishni boshlashi, bu esa insoniyatni nazoratsiz holda qoldirishi mumkin. Bu faza o'z-o'zini o'rgatish jarayonining tezlashishi bilan bog'liq.

Kvant kompyuterlar - bu klassik kompyuterlardan farqli o'laroq, kvant mexanikalari asosida ishlaydigan kompyuterlar. Bostromning fikricha, kvant kompyuterlar sun'iy intellektni yuqori darajaga olib chiqishi mumkin.

Xavf va xavfsizlik (AI Safety) - bu sun'iy intellektni rivojlantirishda uning insoniyatga zarar yetkazmasligi uchun ko'riladigan choralarning yig'indisi. SI xavfsizligini ta'minlash uchun turli xavfsizlik protokollari va strategiyalar ishlab chiqilishi zarur.

Etik masalalar (Ethical Issues) - sun'iy intellektni rivojlantirishda va undan foydalanishda yuzaga keladigan axloqiy savollar, masalan, SI tomonidan qabul qilinadigan qarorlar, insoniyat uchun qanday zarar yoki foyda keltirishi mumkinligi.

Boshqariluvchi sun'iy intellekt (Controlled AI) - bu sun'iy intellekt tizimlari, ularning faoliyati to'liq nazorat ostida bo'lib, insonlar tomonidan qarorlar qabul qilishda yordam beradi va xavfsiz rivojlanishini ta'minlaydi.

Tengsizlik (Inequality) - bu sun'iy intellektning rivojlanishi bilan ijtimoiy tengsizlikning kuchayishi mumkinligi haqida gap boradi. Bostrom buni muhokama qilib, yangi texnologiyalar va ish o'rinlarining avvalgilariga nisbatan farqini ta'kidlaydi.

Etik "Hayotiy xususiyatlar" (Moral Considerations) - bu sun'iy intellekt tizimlarining axloqiy prinsiplar asosida ishlashini va insoniyatning yuksalishiga mos kelishini ta'minlash uchun ko'riladigan choralardir. Bu prinsiplar quyidagilarni o'z ichiga olishi mumkin: inson huquqlari, erkinlik, tenglik, adolat.

Boshqarishdan chiqish (Control Problem) - bu sun'iy intellekt tizimlarining insoniyat nazoratidan chiqish va o'zaro aloqasiz rivojlanish ehtimoli. Bu asosan superintellektni yaratishdagi xavflarni anglatadi.

Maxsus superintellekt tizimlari (Artificial General Intelligence, AGI) - bu odamning intellektual qobiliyatlariga to'liq mos keladigan sun'iy intellekt tizimlari. AGI sun'iy intellektning umumiy aqlga ega bo'lishini anglatadi.

Uchun xavf-xatarlar va natijalar (Existential Risk) - bu sun'iy intellektning rivojlanishidan kelib chiqadigan uzoq muddatli xavf-xatarlar, ular insoniyatning mavjudligini yo'q qilish yoki jiddiy salbiy natijalar keltirish ehtimolini anglatadi.

Takroriy neyron tarmoqlari (RNS) — bu dinamik ma'lumotlar yoki ketma-ketliklar to'plamini qayta ishlash uchun sozlangan mashinani o'rganish modeli. RNS ma'lumotlar ketma-ketligi bilan bog'liq muammolarni hal qilish uchun ishlatiladi. Masalan, matn yoki tovushni tahlil qilish va tanib olish uchun. RNS katta hajmdagi ma'lumotlardan o'rganishi va keyin yangi ketma-ketliklarni bashorat qilishi mumkin.

Bashorat qilish algoritmi sun'iy intellektidagi asosiy tushunchalardan biri bo'lib, ma'lumotlarni tahlil qilish natijasida kerakli ma'lumotlarni bashorat qilish va taklif qilish usulini belgilaydi. Algoritmilar yordamida ovozli yordamchi sizning savolingizga javob beradi, onlayn kinoteatr qiziqarli filmni taklif qiladi va monitoring tizimi mumkin bo'lgan nosozliklarni bashorat qiladi.

Neyron tarmoq arxitekturasini neyron tarmoqdagi neyronlarning bir-biri bilan qanday bog'lanishi va o'zaro ta'sirini aniqlaydigan tuzilma. U ma'lumotlarni qayta ishlashning har bir bosqichida qanday funktsiyalar bajarilishini va tarmoqning turli qatlamlari o'rtasida ma'lumot qanday uzatilishini aniqlaydi. Arxitektura neyron tarmoqning maqsadiga va ishlatilgan o'rganish algoritmlariga qarab farq qilishi mumkin. Masalan, konvolyutsion neyron tarmoqlar naqshni aniqlash uchun, takroriy neyron tarmoqlar esa tabiiy tilni qayta ishlash uchun ishlatiladi.

Bayes tarmoqlari - bu ko'plab o'zgaruvchilar va ularning ehtimoliy bog'liqliklari bo'lgan model. Masalan, Bayes tarmog'idan bir qator alomatlar mavjudligi yoki yo'qligi sababli kasallik ehtimolini hisoblash uchun foydalanish mumkin. Shunday qilib, faqat Afrikadan qaytib kelgan isitma va tana og'rig'i bo'lgan bemorga og'ir alomatlar asosida bezgak tashxisi qo'yilishi mumkin.

Ma'lumotlarni chuqur tahlil qilish - bu inson faoliyatining turli sohalarida qaror qabul qilish uchun zarur bo'lgan ilgari noma'lum va amalda foydali bilimlarni aniqlash jarayoni. Ma'lumotlarni qazib olishning asosini tasniflash, modellashtirish va bashorat qilishning barcha usullari tashkil etadi.

Gradient tushish ko'plab mashinalarni o'rganish modellarida qo'llaniladigan eng keng tarqalgan va birinchi optimallashtirish algoritmlaridan biridir. Matematik formuladan foydalanib, algoritim to'g'ri naqsh va to'g'ri javobni izlash uchun "fikrlash" ni to'g'rilaydi. Masalan, siz dasturni odamning oyoq hajmini uning balandligiga qarab bashorat qilishni o'rgatmoqchisiz. Algoritim tasodifan parametrlar o'rtasidagi munosabatni taxmin qiladi, so'ngra uning va test ma'lumotlar jadvalidagi haqiqiy natijalar o'rtasidagi xatolarni hisoblab chiqadi. Kelajakda model parametrlari xato minimallashtirilgunga qadar tuzatiladi.

Ma'lumotlar to'plami - bu turli xil tahlil va mashinalarni o'rganishda ishlatiladigan ma'lumotlar to'plami. Bundan tashqari, ikkinchisining muvaffaqiyati to'g'ridan-to'g'ri dastlabki ma'lumotlarning hajmiga bog'liq deb ishoniladi: qancha ko'p bo'lsa, AI shunchalik yaxshi rivojlanadi.

Kompyuterni ko'rish - bu kompyuter fanlari, sun'iy intellekt va mashinani o'rganishning kichik sohasi bo'lib, u kompyuterlarga tasvirlar va videolar haqida tez, yuqori darajadagi va insonga yaqin tushuncha berishga intiladi. So'nggi yillarda kompyuterni ko'rish chuqur o'rganish va neyron tarmoqlari tufayli aniqlik va tezlikda ajoyib natijalarni ko'rsatmoqda.

Logistik regressiya - bu ikki ma'lumot omili o'rtasidagi munosabatlarni izlash uchun matematikadan foydalanadigan ma'lumotlarni tahlil qilish usuli. Keyinchalik bu munosabatlar ushbu omillardan birining qiymatini boshqasiga qarab bashorat qilish uchun ishlatiladi. Bashorat odatda ma'lum bir natijaga ega, masalan, "ha" yoki "yo'q".

Sun'iy intellekt modeli - ma'lum turdagi naqshlarni tanib olish uchun ma'lumotlar to'plamida o'qitilgan dastur. AI modellari biznes muammolarini hal qilish uchun ushbu ma'lumotlarni tahlil qilish va o'rganish uchun turli xil algoritmlardan foydalanadi.

Neyromorfik kompyuterlar dasturlashtiriladigan sun'iy neyronlardan foydalangan holda miya funksiyalarini taqlid qiladi va naqshni yanada samarali aniqlashni ta'minlaydi, ayniqsa ba'zi ma'lumotlar aniqlanmagan yoki ma'lumot etishmayotgan holatlarda. Ular, shuningdek, ma'lumotni ketma-ket qayta ishlaydigan klassik kompyuterlardan farqli o'laroq, haqiqiy ma'lumotlarga yanada samarali javob berish qobiliyati bilan ajralib turadi.

NLP (Natural Language Processing) — sun'iy intellekt va matematik tilshunoslikning umumiy yo'nalishi. U tabiiy tillardagi matnlarni kompyuter tahlili va sintezi muammolarini o'rganadi.

Neyron tarmoqlarni o'qitish - bu muammoni hal qilish uchun neyron tarmoqni o'qitish jarayoni. Neyron tarmoqlari belgilangan yoki ajratilmagan ma'lumotlarning

katta to'plamlarini birlamchi qayta ishlash orqali o'qitiladi. Ushbu misollar asosida tarmoqlar noma'lum kirishlarni aniqroq boshqarishi mumkin.

Perceptron neyron tarmoqlari sohasidagi birinchi ishlanmalardan biridir. Bu inson miyasidagi neyronlarning ishlashini taqlid qilishga harakat qiladigan algoritmdir. Perceptronning asosiy printsipli kirish ma'lumotlarini qayta ishlash va olingan ma'lumotlar asosida qaror qabul qilishdir.

Qora quti zamonaviy sun'iy intellektning asosiy muammolaridan biri bo'lib, ishlab chiquvchi modelni ko'rib chiqa olmaydi va tizim nima uchun u yoki bu echimni taklif qilganini tushunolmaydi.

Til modellari - bu oldingi kontekstni hisobga olgan holda matndagi keyingi so'zni (yoki so'zning bir qismini) bashorat qilishni o'rganadigan katta neyron tarmoqlar. Ko'rinib turgan soddaligiga qaramay, muammoning bunday bayoni juda ko'p funktsional neyron tarmog'ining paydo bo'lishiga olib keladi.

ADABIYOTLAR

1. SH.A.Sadullayev, D.F.Yusupov, F.Yusupov “Sun’iy intellekt va neyron to‘rli texnologiyalar” Oliy o‘quv yurtlari uchun darslik . Mahalla va Oila nashriyoti. -T., 2022.-194 b.
2. Q.A.Bekmuratov “Sun’iy intellekt va neyron tarmoqlar”, SamDU nashriyoti, S., 2021. — 356b.
3. Aurélien Géron. “Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow” 2nd edition. copyright © 2019 Kiwisoft S.A.S. All rights reserved. printed in Canada. Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472
4. “Artificial Intelligence Technology”. ofcial textbooks for Huawei ICT Academy. Huawei Technologies Co., Ltd. Hangzhou, China. Posts & Telecom Press 2023
5. Рутковский Л. Методы и технологии искусственного интеллекта ./ Пер. с полск. М.: Горячая линия –Телеком, 2010. -520 с.
6. Пенькова, Т. Г. Модели и методы искусственного интеллекта : учеб. пособие / Красноярск : Сиб. федер. ун-т, 2019. – 116 с.
7. Першина Е.Л., Попова О.А., Чуканов С.Н. Интеллектуальные системы поддержки принятия решений: комплексы программ, модели, методы, приложения: монография. – Омск: СибАДИ, 2010. – 204с.
8. N.R.Yusupbekov Texnologik jarayonlarni avtomatlashtirish va boshqarish darslik Toshkent 2011.
9. Лонца А. Л76 Алгоритмы обучения с подкреплением на Python / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 286 с.: ил.
10. Протоdjяконов А.В., Пылов П.А., Садовников В.Е. Алгоритмы Data Science и их практическая реализация на Python / Учебное пособие. Волгоград. Инфа-Инжинерия, 2022. – 392 с.
11. Казанцев Т. Искусственный интеллект и Машинное обучение. Основы программирования на Python / Т. Казанцев — «ЛитРес: Самиздат», 2020

12. Боровская Е. В. Основы искусственного интеллекта : учебное пособие / изд. Электрон. Ч М. : Лаборатория знаний, 2020. Ч 130 с.
13. Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.
14. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
15. Ng, A. (2018). Machine Learning Yearning. Self-published. Retrieved from <https://www.mlyearning.org>
16. Bostrom, N. (2014). Superintelligence: Paths, Dangers, Strategies. Oxford University Press.
17. Raschka, S. (2015). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow (1st ed.). Packt Publishing.
18. Nilsson, N. J. (1998). Artificial Intelligence: A New Synthesis. Morgan Kaufmann.
19. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
20. Kaku, M. (2014). The Future of the Mind: The Scientific Quest to Understand, Enhance, and Empower the Mind. Doubleday.
21. Богустов А. А. Искусственный интеллект как субъект права: аргументы к дискуссии / А. А. Богустов // Хозяйство и право. - 2021. - № 9. - С. 114-121.
22. Бродская М. Доверенный ИИ: начало пути / М. Бродская // Bis journal. - 2023. - № 3. - С. 16-20.
23. Шальнева М. С. Внедрение технологии искусственного интеллекта в бизнес-процессы торговых корпораций = Introduction of Artificial Intelligence Technology into the Business Processes of Trading Corporations / М. С. Шальнева, Н. В. Шпильман // Страховое дело. - 2022. - № 4. - С. 54-62.
24. Эртель В. Введение в искусственный интеллект / В. Эртель; пер. с англ. А. Горман. - Москва: Эксмо, 2019. – 448 с.

25. Барский А. Б. Искусственный интеллект и интеллектуальные системы управления: монография / А. Б. Барский. - Москва: РУСАЙНС, 2024.- 186 с.
26. Косаренко Н. Н. Система искусственного интеллекта: понятие, теория, право и перспективы развития: монография/Н. Н. Косаренко. - Москва: РУСАЙНС, 2024.- 176 с.
27. Кузнецов А. В. Искусственный интеллект и информационная безопасность общества: монография / А. В. Кузнецов, С. И. Самыгин, М. В. Радионов. - Москва: РУСАЙНС, 2024.- 118 с.
28. Фисун В. В. Искусственный интеллект управления информационной безопасностью объектов критической информационной структуры: монография / В. В. Фисун. - Москва: РУСАЙНС, 2023.- 360 с.
29. https://translated.turbopages.org/proxy_u/en-ru.ru.695a2d92-673ebf2c-351d12fc-74722d776562/https/www.tutorialspoint.com/artificial_intelligence_with_python_ht ht ht_quick_artificial_guckwick
30. https://en.wikipedia.org/wiki/Artificial_intelligence
31. <https://www.geeksforgeeks.org/python-ai/>
32. <https://realpython.com/python-ai-neural-network/>
33. <https://intuit.ru/studies/courses/3691/933/info>
34. <https://opencv-tutorial.readthedocs.io>
35. <https://www.tensorflow.org/?hl=ru>
36. <https://pytorch.org/>
37. <https://keras.io/>
38. <https://caffe.berkeleyvision.org/>
39. <https://scikit-learn.org/stable/>
40. <https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov8-object-detection-on-custom-dataset.ipynb>

Ilmiy nashr

A.A.ABDUMANONOV, A.A.QUCHQAROV

SUN'IY INTELLEKT ASOSLARI

darslik

Mas'ul muharrir: L.K. MAMADALIYEVA

Muharrir: Elyorbek Rahmonov

Dizayner: Marg'uba Qurbanova

Musahhih: Madinaxon Mamajanova

Terishga berildi: 30.06.2025-yilda.

Bosishga ruxsat etildi: 31.07.2025-yilda.

Bichimi: 60x84 $\frac{1}{16}$. Hajmi: 21,5 bosma taboq.

Adadi: 100 nusxa. Bahosi kelishilgan narxda.

«SUNRISE-PRO» MAS'ULIYATI CHEKLANGAN JAMIYATI

“FAZILAT ORGTEX SERVIS”

xususiy korxonasi bosmaxonasida chop etildi.

Manzil: Namangan sh. Amir Temur ko'chasi, 97-uy.

