B.B. MO‘MINOV

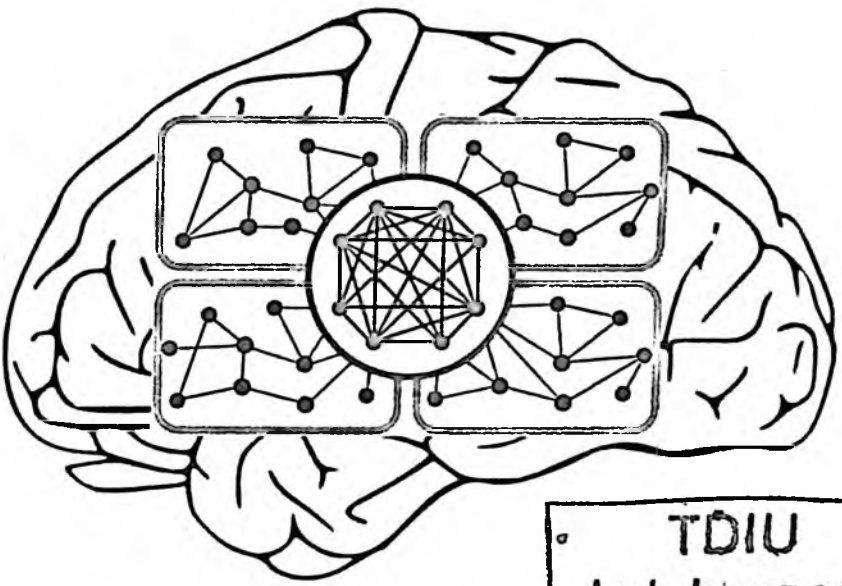
SUN'IY INTELLEKT

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

TOSHKENT DAVLAT IQTISODIYOT UNIVERSITETI

B.B. MO‘MINOV

SUN’IY INTELLEKT



TOSHKENT – 2023

UO'K: 004.8.
KBK 32.813.5.
S 61

B.B. Mo'minov. Sun'iy intellekt. (O'quv qo'llanma). –T.: «Innovatsion rivojlanish nashriyot-matbaa uyi», 2023. 175 bet.

ISBN 978-9910-735-27-1

Mazkur o'quv qo'llanmada sun'iy intellekt asosida ma'lumotlarga ishlov berish yo'nalishida ish olib borayotgan izlanuvchilar uchun sun'iy intellekt borasida tushuncha hosil qilish, uning ko'rinishi bilan tanishish, ishlash tamoyillari, imkoniyatlari, qo'llash sohasi bo'yicha tushunarli va ixcham tilda ma'lumot olishlari mumkin. Shuningdek, sun'iy intellektning modellari, usullari va algoritmlari, qo'llash uslublari keltirilgan.

UO'K: 004.8.
KBK 32.813.5.

Taqrizchi: **N.O. Raximov** – Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti ATDT kafedrasi mudiri, **t. f. d.**

O'quv qo'llanma Toshkent davlat iqtisodiyot universiteti rektorining 2023 yil 14 iyuldagi 228-son buyrug'iga asosan nashr qilindi.

ISBN 978-9910-735-27-1

QISQACHA MAZMUNI

Sun'iy intellektdan foydalangan holda muammolarni hal qilishda standart neyron tarmoq konfiguratsiyasi tanlanadi, ammo muammoning murakkabligi va xususiyatlarini hisobga olgan holda mavjud konfiguratsiyalarni tanlash muammoli bo'lishi mumkin. Agar masalaning yechimi ma'lum turdagi neyron tarmoqlardan biri asosida yechimga ega bo'lmasa, yangi konfiguratsiyani shakllantirish muammosi hal qilinishi kerak bo'ladi.

Sun'iy intellekt tuzilishini aniqlash uchun bir nechta muammolarni hal qilish kerak: sun'iy intellekt tasnifini yaratish; mavjud sun'iy intellektni tahlil qilish; modelni yaratish uchun asosiy mezonlarni ishlab chiqish; sun'iy intellektga asoslangan modelning sifatini aniqlash uchun asosiy xususiyatlarni aniqlash.

O'quv qo'llanmaning maqsadi: asosiy sun'iy intellektning batafsil tasnifi, sun'iy intellektning asosiy xususiyatlari va mavjud modellarini tanlash mezonlarini aniqlashda ko'makchi bo'lib xizmat qiladi.

Keltirilgan misollar asosida esa umumiy ilmiy-amaliy tamoyillar, tasnifni qurish uchun tizimli va kompleks yondashuvlar nazariyasiga asos sifatida qabul qilish mumkin.

Ushbu o'quv qo'llanmada sun'iy intellektning tasnifi keltirilgan va identifikatsiya qilish muammolarini qayta ishlash, tarmoqlaridan foydalanish va xatolarni teskari tarqatish usuli bo'yicha o'qitiladigan tarmoqlar – tashxislash vazifasi uchun hal qilish taklif etiladi. Keltirilgan sun'iy intellektning keng qo'llaniladigan algoritmlariga, modellariga alohida urg'u berilib, aniq misollar ko'rinishida izohlangan.

Tasvirlarni aniqlash, ular asosida o'rganish, ularni tasniflash, ma'lumotlarni yig'ib borish kabi masalalarni yechishda asosan konvolyutsiyalash neyron tarmoqlar CNN (Convolutional neural networks) qo'llanilishi borasida aniq misollar keltirilgan.

Rekurrent neyron tarmoqlar RNN (Recurrent neural network) ko‘rinishlari, ular asosida qo‘llaniladigan algoritmlar LSTM, Word2Sec asosida mashinali tarjima borasida aniq misollar keltirilgan.

Kohonen neyron tarmog‘i KN (Kohonen network) berilgan katta hajmdagi ma’lumotlarni klasterlash, ular ichida qonuniyatlarni aniqlash, o‘z-o‘zini o‘rganish tizimlari (o‘qituvchili, o‘qituvchisiz) masalalarini yechishda qo‘llanilishi va aniq algoritmlariga misollar keltirilgan.

Keng tarqalgan Bayes algoritmlarining izohi va uning qo‘llanilishi aynan ehtimolliklar nazariyasiga asoslanishi borasida misollar keltirilgan.

Turli sohalarda katta hajmdagi ma’lumotlarni qo‘llash jarayonida texnik holatni aniqlash va ma’lumotlar asosida tashxislash muammolarini hal qilish, ma’lum bir qonuniyatlarni aniqlash, tizimlarni avtomatlashtirishni yo‘lga qo‘yish, boshqaruvni sun’iy intellekt asosida amalga oshirilishi bilan bog‘liq masalalarni hal qilish uchun sun’iy intellektdan yoki oddiygina turli xil sun’iy intellektdan foydalanish mumkin. SI funksiyalarining maksimal ta’siri bilan amalga oshirish uchun odatda SI parametrlarining optimal to‘plami mavjud.

Bundan tashqari, adabiyot manbayi amalda qo‘llanilgan va nazariy ma’lumotlarning to‘plami sifatida qabul qilinishi tavsiya etiladi.

Shunday qilib, SI ishlab chiquvchisi oldida turgan asosiy vazifalardan biri, bu tarmoq turini aniqlaydigan ushbu to‘plamni tanlashdir. O‘quv qo‘llanma so‘ngida tavsiya sifatida chizma keltirilgan bo‘lib, algoritm ko‘rinishi va qo‘llanilishi mumkin bo‘lgan yo‘nalish keltirilgan.

АННОТАЦИЯ

При решении задач с использованием искусственного интеллекта подбирают стандартную конфигурацию нейросети, но с учетом сложности и особенности задачи подбор существующих конфигураций может быт проблематичен. Если же задача не может быть сведена ни к одному из известных типов нейросети, приходится решать сложную проблему синтеза новой конфигурации.

Для определения структуры модели искусственного интеллекта необходимо решит несколько задач: построит классификацию искусственных интеллектов; провести анализ искусственных интеллектов; разработать основные критерии отбора искуственных интеллектов для построения модели; определить основные характеристики для определения качества модели на основе искусственного интеллекта.

Цель написания учебного пособия подробная характеристика искусственного интеллекта позволит выявит основные характеристики искусственного интеллекта и критерии отбора существующих моделей.

Методологической основой работы явились общенаучные принципы проектирования, теория системно-комплексных подходов к построению классификации.

В учебном пособие приводится характеристики искусственного интеллекта и предлагается для решения задач идентификации исполлзовать рекуррентные сети, а сети с обучением по методу обратного распространения ошибок - для задачи прогнозирования. Для решения задач идентификации и прогнозирования технического состояния в процессе эксплуатации могут быт исползованы искусственные нейронные сети (ИНС) или, просто, нейронные сети (НС) различного рода. Для реализации с максимальсным эффектом фунций НС, как правило, существует

оптимальная совокупност параметров НТ. Следовательно, одной из основных задач, стоящих перед разработчиком НС, является выбор этой совокупности, определяющей, в конечном итоге, вид сети. Основным элементом сети является искусственный нейрон (далее нейрон). Нейроны делятся на три типа в соответствии с функсиями, выполняемыми ими в сети. Входные нейроны (нейроны входного слоя) принимают данные из внешней среды и определенным образом распределяют их далее по сети. На промежуточные нейроны (нейроны скрытого слоя) возлагается рол основных участников процесса решения задачи. Выходные же нейроны (нейроны выходного слоя) передают резултаты работы сети во внешнюю среду (потребителю). Типы нейронов в зависимости от их функсий в сети. В зависимости от механизма обработки получаемых данных можно выделит селый ряд математических моделей нейронов. Существуют две группы моделей нейронов, которые принадлежат, соответственно, двум типам сетей: классическим и нечетким. Каждая из моделей нейронов обладает рядом присущих ей свойств, однако имеются и общие черты, к которым можно отнести наличие входного и выходного сигналов, а также блока их обработки.

Кроме описания основных нейронных сетей, особое внимание уделяется широко исползуемым алгоритмам, моделям нейронных сетей, приведены конкретные применения:

При решении таких задач, как распознавание изображений, определение не дополненных образов на основе базы обучения, тасниф и группировка изображений и т.д., применяется конволюсионные нейронные сети CNN (convolutional neural networks). После описания представлены конкретные примеры применения этой сети.

Ресуррент нейронные сети РНН (ресуррент неурал нетwорк) представляют собой рекуррентную нейросет, приведен пример на

основе алгоритмов как LSTM, Word2Sec сети исползуются в машинном переводе.

Нейронная сет Кохонена CN (Cohonen network) которая работает с болшим объемом данных, для решения таких задач как определения закономерностей, взаимосвязи определенных параметров, кластеризация данных. Она удобна для работы в системах с самообучением (с учителем, или без учителя). В энциклопедии можно ознакомиця с примерами конкретных алгоритмов.

Приведены примеры, которые описывают принцип работы алгоритмов Баес. Баес это алгоритм классификации, с допущением о независимости признаков.

В процессе исползования болших объемов данных в различных областях можно исползоват искусственные нейронные сети или просто различные нейронные сети для решения проблем технического состояния и прогнозирования на основе данных, выявления определенных закономерностей, автоматизации систем, решения проблем, связанных с осуществлением управления на основе искусственного интеллекта. Для реализации с максимальным эффектом функций NN обычно существует оптимальный набор параметров НН. Таким образом, одной из основных задач, стоящих перед разработчиком NN, является выбор этого набора, определяющего тип сети. В конце энциклопедии в реферальном просмотре приводится схема, в которой приводится алгоритм работы и рекомендуемое направление, в котором он может быт применен.

Литературные ресурсы, представленные в конце энциклопедии, применены практически и рекомендуются в качестве сборника теоретической информации.

ABSTRACT

When solving problems using a neural network, a standard configuration of the neural network is selected, but taking into account the complexity and features of the problem, the selection of existing configurations can be problematic. If the problem can not be reduced to any of the known types of neural network, it is necessary to solve the complex problem of synthesis of the new configuration.

To determine the structure of the neural network model, it is necessary to solve several tasks: to build a classification of neural networks; to analyze existing neural networks; to develop the main criteria for selecting neural networks for building a model; to determine the main characteristics for determining the quality of the model based on the neural network.

The purpose of writing an encyclopedia detailed classification of neural networks will reveal the main characteristics of neural networks and selection criteria of existing models.

The methodological basis of the work was the General scientific principles of design, the theory of system-integrated approaches to the construction of classification.

The classification of neural networks is given in the encyclopedia and it is proposed to use recurrent networks to solve identification problems, and networks with training by the method of back propagation of errors - for the forecasting problem. Artificial neural networks (ins) or, simply, neural networks (NS) of various kinds can be used to solve the problems of identification and prediction of technical condition during operation. As a rule, there is an optimal set of NT parameters for the implementation of NS functions with maximum effect. Therefore, one of the main tasks facing the NS developer is the choice of this set, which ultimately determines the type of network. The main element of the network is an artificial neuron (hereinafter referred to as the neuron). Neurons are divided into three types according to the

functions they perform on the network. Input neurons (neurons of the input layer) receive data from the external environment and distribute them further along the network in a certain way. Intermediate neurons (neurons of the hidden layer) are assigned the role of the main participants in the process of solving the problem. Output neurons (neurons of the output layer) transmit the results of the network to the external environment (consumer). Types of neurons depending on their functions in the network Depending on the processing mechanism of the data can be identified a number of mathematical models of neurons. There are two groups of neuron models, which belong, respectively, to two types of networks: classical and fuzzy. Each of the models of neurons has a number of inherent properties, but there are common features, which include the presence of input and output signals, as well as their processing unit.

In addition to the description of the main neural networks, special attention is paid to widely used algorithms, neural network models, specific applications are given:

In solving problems such as image recognition, the definition of non-augmented images based on the training base, classification and grouping of images, etc., used convolutional neural networks cNN (convolutional neural networks). The description is followed by specific examples of the use of this network.

Recurrent neural networks RNN (recurrent neural network) are a recurrent neural network, an example based on algorithms like LSTM, Word2Sec networks are used in machine translation.

Neural network Kohonen (KN) which works with a large amount of data to solve problems such as determining patterns, the relationship of certain parameters, data clustering. It is convenient to work in self - learning systems (with or without a teacher). In the encyclopedia you can find examples of specific algorithms.

Examples that describe the principle of Bayes algorithms are given. Bayes is a classification algorithm, with the assumption of feature independence.

In the process of using large amounts of data in various fields, you can use artificial neural networks or simply different neural networks (NT) to solve problems of technical condition and forecasting based on data, identify certain patterns, automate systems, solve problems related to the implementation of management based on artificial intelligence. For the implementation of the maximum effect of NN functions, there is usually an optimal set of NN parameters. Thus, one of the main challenges facing the NN developer is the choice of this set, which determines the type of network. At the end of the encyclopedia in the referral view is a diagram that shows the algorithm and the recommended direction in which it can be applied.

The literary resources presented at the end of the encyclopedia are practically used and are recommended as a collection of theoretical information.

I bob. SUN'IY INTELLEKTNING NAZARIY ASOSLARI

1.1. Sun'iy intellektning rivojlanish tarixi

Inson ongiga o'xshash fikrlash qobiliyatiga ega sun'iy ongni yaratish borasidagi g'oya ancha yillar oldin paydo bo'lgan edi. R.Lulliy XIV asrlarda turli masalalarni mulohazalar asosida yechadigan mashinani yaratishga harakat qildi.

XVIII asrlarda *G.Leybnis* (1646–1716) va *R.Dekart* (1596–1650) bir-biridan xolis holda universal tilni ishlab chiqish g'oyasini olg'a surdi. Ushbu g'oyalar sun'iy intellektning nazariy asosi bo'lib qoldi.

Sun'iy intellektni yaratish yo'nalishi birinchi EHMlar yaratilgandan so'ng, XX asrning 40-yillarida kuchayib ketdi. Shu vaqtning o'zida *I.Viner* (1894–1964) yangi yo'nalish fani – kibernetika bo'yicha o'z tadqiqotlarini olib bordi.

Sun'iy intellekt tushunchasi (Artificial intelligence) 1956-yil Stanford universiteti (AQSH) seminarida birinchi marta e'lon qilindi. Ushbu seminar hisoblash ishlariga emas balki, mantiqiy ishlanmalarga yo'naltirilgan edi. Sun'iy intellekt yangi fan yo'nalishi sifatida tan olingandan keyin u asosiy ikkita yo'nalishga ajratildi: neyrokibernetika va «qora yashik» kibernetikasi. Faqat hozirga kelib ular bitta yagona yo'nalish sifatida birlashtirildi.

Neyrokibernetikaning asosiy g'oyasini quyidagicha ifodalash mumkin. Fikrlash qobiliyatining yagona obyekti inson ongidir, shuning uchun barcha «fikrlash» imkoniyatiga ega qurilma aynan ong faoliyatini takrorlashi lozimdir.

Demak, neyrokibernetika inson miyasi strukturasiga o'xshash model strukturasining apparat ko'rinishga yo'naltirilgan. Fiziologlar tomonidan shu narsa aniqlashtirilganki, inson miyasi asosini o'zaro bog'langan nerv hujayralari – neyronlar tashkil qiladi. Shuning uchun neyrokibernetiklar ishi neyronlarga mos bo'lgan elementlarni yaratish

va ularni yagona tizimga birlashtirishga yo'naltirilgan. Bunday tizimlar ***neyron tarmoqlar*** yoki ***neyrotarmoqlar*** deb nom olgan.

Eng birinchi neyrotarmoqlar 50-yillar oxirida amerikalik olimlar *G.Rozenblat* va *P.Mak-Kigyuk* tomonidan ishlab chiqilgan. Bu inson ko'zi modeli va uning inson miyasi bilan hamkorlikda ishlashini ta'minlovchi tizimni yaratishga harakat edi. Ular yaratgan qurilma ***perseptron*** degan nomni oldi. U alifbo harflarini farqlash qobiliyatiga ega bo'lib, biroq ular yozilishiga ta'sir qilardi. Masalan, *A*, A va ***A*** harflari yozilishi uchta turli belgi sifatida qabul qilinar edi. 70–80-yillarga kelib, sun'iy intellekt yo'nalishi bo'yicha izlanishlar soni kamayib bordi. Eng birinchi natijalar samarasiz bo'lib chiqdi, mualliflar fikricha, o'sha davr uchun kompyuterlar xotira hajmi va ish kuchi tezligi yetarlicha bo'lmaganligi sabab bo'ldi.

Biroq, 80-yillar o'rtalarida Yaponiyada yangi avlod kompyuterlarini ishlab chiqarish sohasida bilimlari asosida ishlovchi neyrokompyuterlari ishlab chiqildi. O'sha davrda xotira hajmi va tezkorlik ko'rsatkichlaridagi cheklanmalar yo'qolib, ***transpyuterlar*** paydo bo'ldi, katta hajmdagi protsessorlarga ega parallel kompyuterlar va transpyuterlardan ***neyrokompyuterlar*** inson miyasi strukturasi modelini yaratishga bir qadam qolgan edi. Neyrokompyuterlarni qo'llashning asosiy sohasi bu obrazlarni aniqlashda qo'llash hisoblandi.

Hozirgi kunda neyrotarmoqlarni yaratishning uchta yo'nalishi qo'llanilmoqda:

apparat – maxsus kompyuterlarning, kengaytma platalari, mikrosxemalar to'plami yaratilishi bo'lib, ular qo'yilgan algoritm bo'yicha ishlashga yo'naltiriladi;

dasturiy – yuqori tezkorlikka ega kompyuterlar uchun dasturlar va instrumentlarni yaratish. Neyrotarmoqlar kompyuter xotirasida yaratiladi, barcha amallarni esa ularning protsessorlari amalga oshiradi;

gibrid – ikki qismning muvofiqlashuvi. Hisoblashlarning bir qismini maxsus kengaytirilgan platalar amalga oshirsa, (soprotsessorlar), qolgan qismini dasturiy vositalar tashkil qiladi.

«Qora yashik» kibernetikasi asosida neyrokibernetika asosiga qarama-qarshi bo'lgan prinsip yotadi. Bunda «fikrlovchi» qurilma qanday joylashganligiga ahamiyatsiz holda, asosiy inson aqli kabi berilganlarni to'g'ri qayta ishlashi muhim hisoblanadi.

Sun'iy intellektning ushbu yo'nalishi mavjud kompyuter modellarida intellektual masalalarni yechish algoritmlarini izlab topishga yo'naltirilgan. 1956–1963-yillarda inson fikrlash qobiliyatining modellari va algoritmlarini aniqlash hamda birinchi dasturlarni ishlab chiqish ishlari olib borilgan. Biroq, mavjud fanlardan birortasi – falsafa, psixologiya, lingvistika aynan algoritmni taklif eta olmaydi. Turli yo'nalishlarda tadqiqotlar olib borilgan.

50-yillar oxirida labirintli izlash modeli yaratildi. Bu yo'nalishda masala ma'lum bir grafik ko'rinishda keltirilib, holatlar muhitini ifodalab, ushbu grafika asosida kiruvchi berilganlar va olinadigan natijalargacha bo'lgan yo'llarning optimalini izlab topish imkonini beradi. Ushbu modelni ishlab chiqish bo'yicha ancha ishlar olib borilgan, biroq amaliy masalalarni yechishda bu usul o'z aksini topmadi.

60-yillar boshi – ***evristik dasturlash davri*** bo'lib, **evristika** – nazariy jihatdan asoslanmagan qonuniyat, biroq izlash keng muhitida holatlar sonini qisqartirish imkonini beradi. Evristik dasturlash – oldindan berilgan evristika asosida strategik amallarni ishlab chiqish degan ma'noni anglatadi.

1963–1970-yillarda masalalarni yechishda matematik mantiq usullarini qo'llash ishlari boshlandi. Qonuniyatlardan foydalanish usuli, ya'ni mavjud aksiomalar negizida teoremalarning isbotlanishi asosida 1973-yilda ***Prolog*** dasturlash tili yaratildi.

Sun'iy intellekt amaliy dasturlash sohasida 70-yillar o'rtasida ahamiyatli o'zgarish yuz berdi, bunda fikrlash universal algoritmini izlash o'rniga mutaxassis-ekspertlar aniq bilimlarini modellashtirish g'oyasi paydo bo'ldi. AQSHda bilimlarga tayangan holda ishlovchi tijorat tizimlari yoki boshqacha qilib aytganda, ***ekspert tizimlar***

yaratildi. Sun'iy intellekt masalalarini yechishning yangi yondashuvi – ***bilimlarni ifodalash*** usuli kirib keldi. Tibbiyot va kimyo yo'nalishlari uchun klassik bo'lib qolgan ekspert tizimlari – MYCIN va DENDRAL yaratildi. Intellektual texnologiyalarni rivojlantirishning bir necha global loyihalari taklif qilingan. ESPRIT (Yevropa Ittifoqi). DARPA (AQSH Mudofaa vazirligi), V avlod mashinalari bo'yicha yapon loyihasi hisoblanadi.

80-yillar o'rtalaridan boshlab sun'iy intellekt tijorat loyihalarida qo'llanila boshlandi. Yillar davomida ekspert tizimlarini ishlab chiqarishga katta hajmdagi mablag' ajratilib, o'rganish qobiliyatiga ega sun'iy intellekt tizimlari bo'yicha izlanishlar olib borilmoqda va ishlab chiqilmoqda.

1.2. Sun'iy intellekt rivojlanish bosqichlari

1-bosqich (50-yillar) (Neyron va neyron tarmoqlar)

Bu davr ketma-ket amallarni bajaruvchi hozirgi vaqt uchun o'rtacha quvvatga va ma'lum bir resursli xotira, tezkorlikka, masalalar yechimiga ega bo'lgan mashinalar paydo bo'lishi bilan bog'liq. Ushbu masalalar faqat hisoblash amallari bilan bog'liq bo'lib, buning uchun masala yechimi sxemasi ma'lum bir rasmiy dasturlash tilida izohlangan bo'lgan. Bunday masalalarga adaptatsiya masalalari kiradi.

2-bosqich (60-yillar) (Evristik izlanish)

Mashina «intellekt» qismiga ma'lumotlarni izlash, saralash, oddiy usullarda umumlashtirish mexanizmlari qo'shilib, unda qayta ishlanadigan ma'lumotlar mohiyat jihatdan bog'liqsizdir. Aynan shu davr inson faoliyatini avtomatizatsiyalash masalalarini hal qilish va rivojlantirish uchun turtki nuqtasi bo'lib xizmat qildi.

3-bosqich (70-yillar) (Bilimlarni ifodalash)

Bu davrda olimlar uchun aynan masalalar yechimi yangi algoritmlarini sintez qilish uchun bilimlar (ularning hajmi va mohiyati) muhimligi aniqlandi. Ushbu bilimlar matematik nuqtayi nazardan

izohga ega bo‘lmasdan, to‘plangan tajriba asosida olingan bilimning rasmiy xususiyatiga ega bo‘lmasdan, faqat izohlar to‘plami sifatida shakllangan bilimlar inobatga olinmoqda. Bu bilimlar turli soha mutaxassislari, shifokorlar, kimyogarlar, tadqiqotchilar va boshqalar erishgan yutuqlar, bilimlar to‘plamidir. Ushbu bilimlar ekspert bilimlar deb nom olgan bo‘lib, ular asosida ishlovchi tizimlar konsultant (maslahatchi) yoki ekspert tizimlar deb nomlandi.

4-bosqich (80-yillar) (Bilim oluvchi mashinalar)

SI rivojlanishining to‘rtinchi bosqichi ilg‘or qadam bilan olg‘a surilishga turtki bo‘ldi. Ekspert tizimlar paydo bo‘lganidan boshlab, intellektual texnologiyalar rivojlanishining yangi bosqichi – intellektual tizimlar – konsultantlar erasi boshlandi, bunga qo‘yilgan masala yechim yo‘llarini ko‘rsatib, ularni asoslab beradi, o‘rganish va bilim sohasini kengaytirish imkoniyatiga ega, inson bilan cheklangan tabiiy tilda muloqot qilish imkoniga ega tizimlardir.

5-bosqich (90-yillar) (Ma’lumotlarni qayta ishlashning avtomatizatsiyalashgan markazlari)

Aloqa tizimlarining va ular yordamida bajariladigan masalalarning murakkablashuvi dasturiy ta’minotning «intellektualligini» oshiruvchi darajaga o‘tishni taqozo etdi, bunda tizimdan chetdan turib huquqsiz foydalanishdan himoyalash, axborot resurslarini himoyalash, tahdidlarni oldini olish, tahlil o‘tkazish hamda kerakli ma’lumotni izlash va boshqa ishlarni ta’minlashi kerak. Himoya tizimlarini yaratishning yangi ko‘rinishi bu intellektual tizimlarni qo‘llash bo‘lib qoldi. Aynan ular asosda oson o‘zgartirish mumkin bo‘lgan muhitni yaratish va kerakli masalalar yechimini ta’minlash mumkin.

6-bosqich (2000-yillar) (Robototexnika)

Robotlarni qo‘llash sohasi juda keng bo‘lib, oddiy atrofni tozalovchi robotlardan boshlab, to zamonaviy harbiy va kosmik texnika ko‘rinishlarga ega bo‘lgan. Modellarda navigatsiya tizimlari va periferiyali datchiklar o‘rnatilgan.

7-bosqich (2008-yillar) (Singulyarlik)

Sun'iy intellekt va robotlarni yaratuvchi sun'iy intellektlarning yaratilishi, insonlarning hisoblash mashinalari bilan integratsiyalanishi, inson aqliy faoliyatining biotexnologiya hisobidan yuqori ko'rsatkichga oshirilishi shu bosqich yutuqlari hisoblanadi.

Olimlarning bashorat qilishlaricha, texnologik singulyarlik 2030-yillarda yetishi mumkin, texnologik singulyarlik nazariyasini qo'llab-quvvatlovchi mutaxassislarning fikricha, inson ongidan farqli o'laroq ong yaratilsa, sivilizatsiyaning kelajagini inson harakati belgilamay qoladi.

1.3. Sun'iy intellektning rivojlanish yo'nalishlari

Sun'iy intellekt – bu informatikaning bir qismi bo'lib, asosiy mohiyati apparat dasturiy vositalarni ishlab chiqishga yo'naltirilgan va dasturchi bo'lmagan foydalanuvchiga masalani qo'yish va uni yechish imkonini beradi. Bunda intellektual masalalarning belgilangan tilda EHM yordamida hal qilinishi tushuniladi.

1.3.1. Bilimlarga asoslangan tizimlarni ishlab chiqish va ularni ifodalash

Bu sun'iy intellektning asosiy yo'nalishi bo'lib, bilimlarni ifodalash modelini, ma'lumotlar bazasini yaratish va ular asosida ekspert tizim ishlab chiqish bilan bog'liq. Oxirgi yillarda bilimlarni strukturalash, ajratib olish modeli va usullari ishlab chiqilmoqda va ular bilimlarni strukturali qayta ishlashga olib kelmoqda.

1.3.2. O'yin va ijod

Sun'iy intellekt odatiy tusda intellektual o'yin masalalarni shaxmat, shashka va boshqalarni o'z ichiga oladi. Uning asosida boshlang'ich yo'nalishdagi yondashuv, ya'ni labirint model va evristika yotadi. Hozirda bu yondashuv tijorat yo'nalish, chunki ilmiy jihatdan bu yondashuv samarasiz hisoblanadi.

1.3.3. Tabiiy tilda va mashinali tarjima tizimini ishlab chiqish

50-yillarda sun'iy intellekt tadqiqot yo'nalishlaridan biri mashinali tarjima sohasi bo'lib keldi. Eng birinchi tarjimon dastur – ingliz tilidan rus tiliga tarjima qiluvchi bo'lib, g'oya bo'yicha so'zma-so'z tarjima va u samarasiz hisoblandi. Hozirgi zamonda murakkabroq model qo'llanilmoqda, bunda tabiiy tillardagi ma'lumolar analiz va sintez asosida tarjima qilinadi. Tahlil jarayonida quyidagilar amalga oshiriladi:

Morfologik analiz – matnda so'zlarning analizi.

Sintaksik analiz – gaplarning grammatikasi va so'zlar o'rtasidagi bog'lanish analizi.

Semantik analiz – har bir gapni mohiyatga – yo'naltirilgan bilimlar bazasiga asoslangan holda ma'nosini tahlil qilish.

Pragmatik analiz – xususiy bilimlar bazasiga asoslangan holda gaplar ma'nosini kontekstli analiz qilish hisoblansa, sintez esa xuddi shu amallarning o'zgacha ketma-ketlikda bajarilishi hisoblanadi.

1.3.4. Tasvirlarning aniqlanishi

Sun'iy intellektning keng tarqalgan yo'nalishi bo'lib, tarixiy asosni hosil qiladi. Bunda har bir obyektga xossalar matritsasi belgilanib, u asosida obrazni aniqlash ishlari olib boriladi. Ushbu yo'nalish mashinali o'rganishga juda yaqin bo'lib, neyrokibernetika bilan yaqin bog'langan.

1.3.5. Intellektual robotlar

Robotlar – bu elektromexanik mashinalar bo'lib, inson mehnatini avtomatizatsiyalash uchun mo'ljallangan.

Robotlarni yaratish g'oyasi azaldan kelib chiqqan bo'lib, iboraning o'zi 20-yillarda paydo bo'lgan, uning muallifi – chex yozuvchisi Karel Chapek. O'sha davrdan boshlab bir necha robot avlodlari yaratildi.

Qattiq sxemali boshqaruvga ega robotlar. Zamonaviy ishlab chiqarish sohasida qo'llanilib kelinayotgan robotlar birinchi avlod robotlari hisoblanadi, ular dasturlanadigan manipulyatorlardir.

Sensorli qurilmaga ega adaptiv robotlar. Ularning namunaviy ko'rinishi mavjud, biroq ishlab chiqarishda qo'llanilmaydi.

O'z-o'zini boshqaradigan yoki intellektli robotlar. Bu robototexnikaning so'nggi holati bo'lib, unda asosiy muammo – mashina axborot qabul qilish (ko'rish) qobiliyatini hosil qilish hisoblanadi

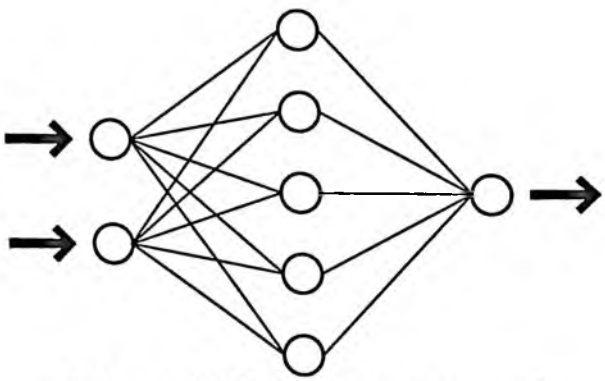
1.3.7. Maxsus dasturiy ta'minot

Ushbu yo'nalish uchun maxsus dasturlash tillari ishlab chiqilgan bo'lib, aynan hisoblash qonuniyatlariga bo'ysunmaydigan masalalarni yechishga yo'naltirilgan. Ushbu tillar ma'lumotlarni simvolli qayta ishlashga yo'naltirilgan bo'lib, ularga LISP, PROLOG, SMALLTALK, REFAL va boshqalar kiradi. Ishlab chiqarishda qo'llaniladigan intellektual tizimlar yoki sun'iy intellekt dasturiy instrumenti uchun amaliy dasturlar paketi ishlab chiqilmoqda, masalan, KEE, ARTS. Eng keng tarqalgan ko'rinishi bu bo'sh ekspert tizim yoki «qobiq»lar bo'lib, masalan, BXSYS, Ml va boshqalar, ular tarkibini bilimlar bazasi bilan to'ldirib, turli ko'rinishdagi tizimlarni yaratishi mumkin.

1.3.8. O'rganuvchi va mustaqil o'rganuvchi

Sun'iy intellekt faol rivojlanayotgan soha bo'lib, ma'lumotlarni tahlil qilgan holda bilimlarni avtomatik to'plab borish va umumlashtirishga yo'naltirilgan modullar, usullar va algoritmlar kiradi. Keltirilgan misollar asosida o'rganib borish va kiritilayotgan obrazlarni tahlil qilgan holda aniqlash usullarini o'z ichiga oladi.

1.4. Sun'iy neyron tarmog'i



1.1-rasm. Sodda neyrotarmoq sxemasi

Sun'iy neyron tarmoq (SNT) — dasturiy va apparat ko'rinishga keltiriladigan matematik model bo'lib, biologik neyronlar tarmog'i ishlash prinsipi asosida yaratilgan. Eng birinchi neyron tarmoqlar U.Makkalok va U. Pitts tomonidan yaratilgan edi. O'rganish algoritmlari ishlab chiqilgandan so'ng yaratilgan modellar amaliyotda, ya'ni obrazlarni aniqlashda bashorat qilish, boshqaruvda va boshqa masalalarda qo'llanila boshlandi.

SNT o'zaro bog'langan hamkorlikda ishlaydigan sodda protsessorlar (sun'iy neyronlar) tizimini hosil qiladi. Odatda, bu protsessorlar kompyuter protsessorlariga nisbatan juda sodda hisoblanadi. Har bir protsessor faqat signallar bilan ishlab, bu signallarni hosil qiladi va davriy ravishda boshqa protsessorlarga uzatib turadi. Ma'lum bir boshqaruvga ega yirik tarmoqqa ulanganligi sabab, ushbu sodda protsessorlar juda murakkab masalalarni hamjihatlikda yechish imkonini beradi

Mashinali o'rganish sohasi bo'yicha qaralganda neyron tarmoq obrazlarni aniqlashda, diskriminant tahlilda, klasterlash usullarida xususiy hol bo'lib hisoblanadi.

- Matematika nuqtayi nazaridan neyron tarmoqlar bu ko'p parametrli chiziqsiz optimizatsiya masalasi hisoblanadi.
- Kibernetika nuqtayi nazaridan neyron tarmoqlar adaptiv boshqaruv masalalarida va robototexnikada qo'llaniladi.
- Hisoblash texnikasi va dasturlash sohasi rivojlanishi nuqtayi nazaridan qaraganda neyron tarmoq — samarali parallellik masalalarini hal qilish usuli hisoblanadi.
- Sun'iy intellekt nuqtayi nazaridan SNT konnektivizm falsafasining asosi hisoblanadi va tabiiy intellektni modellashtirishda kompyuterli algoritmlarni qo'llash bo'yicha asosiy yo'nalish bo'lib keladi.

Neyron tarmoqlar ma'lum ma'noda dasturlanmaydi, balki ular o'rganish jarayonini o'tadi. O'rganish faoliyati neyron tarmoqlarning algoritmlashda asosiy afzalligi hisoblanadi. O'rganish jarayoni texnik

nuqtayi nazaridan neyronlar o‘rtasida koeffitsiyentlarni aniqlash hisoblanadi. O‘rganish jarayonida neyron tarmoqqa kiruvchi va chiquvchi ma’lumotlar o‘rtasida murakkab bog‘lanishlarni aniqlab topish hamda umumlashtirish imkoniyati mavjud. Natijada, samarali o‘rganish asosida yangi noaniq va buzilgan ma’lumotlar qabul qilinganda to‘g‘ri javob qaytarish imkonini beradi.

1.4.1. Neyron tarmoqlarni qo‘llash sohalari. Obrazlarni aniqlash. Tasnif masalalari

Obrazlar sifatida turli ko‘rinishdagi obyektlar: matn simvollari, tasvirlar, tovushli obyektlar hisoblanadi. Tarmoqni o‘rganish jarayonida turli ko‘rinishdagi obrazlar namunasi ularning qaysi sinfga mansubligi bilan birga beriladi. Namuna, odatda xususiyatlar ko‘rsatkichlari vektori ko‘rinishida ifodalaniladi. Bunda barcha xususiyatlar birlashmasi keltirilgan namuna sinfini belgilashi lozim. Xususiyatlar yetarlicha bo‘lmasa keltirilgan namuna bir necha sinflarga mansub deb qabul qilinadi. Tarmoq o‘rganish jarayoni tugaganidan so‘ng yangi obrazlar ifodalanganda u mansub bo‘lgan sinf aniqlanib beriladi.

Bu ko‘rinishdagi tarmoq topologiyasi chiqish qatlamida neyronlar soni aniqlangan sinflar soniga teng bo‘ladi. Bunda neyron tarmoq chiqish qismi va ifodalanilayotgan sinf o‘rtasida moslashuv o‘rnatiladi. Tarmoqqa ma’lum bir obraz ifodalanganda uning chiqish qismiga obraz mansub bo‘lgan sinfni belgilovchi ma’lumot paydo bo‘lishi lozim, shu bilan birga qolgan chiqish qismlarida unga mos bo‘lgan sinfga mansub emasligi to‘g‘risida ma’lumot paydo bo‘lishi lozim. Agar birdan chiqish qismlarda aynan unga mos sinfga mansubligi borasida ma’lumot paydo bo‘lsa, u holda tarmoq javobi «ishonchsiz» deb qabul qilinadi.

Yangi neyron tarmoqli arxitekturalari yaratilgandan boshlab, ular ko‘pligi tufayli barchasini o‘rganib chiqish, tartiblash juda murakkab bo‘lib kelmoqda.

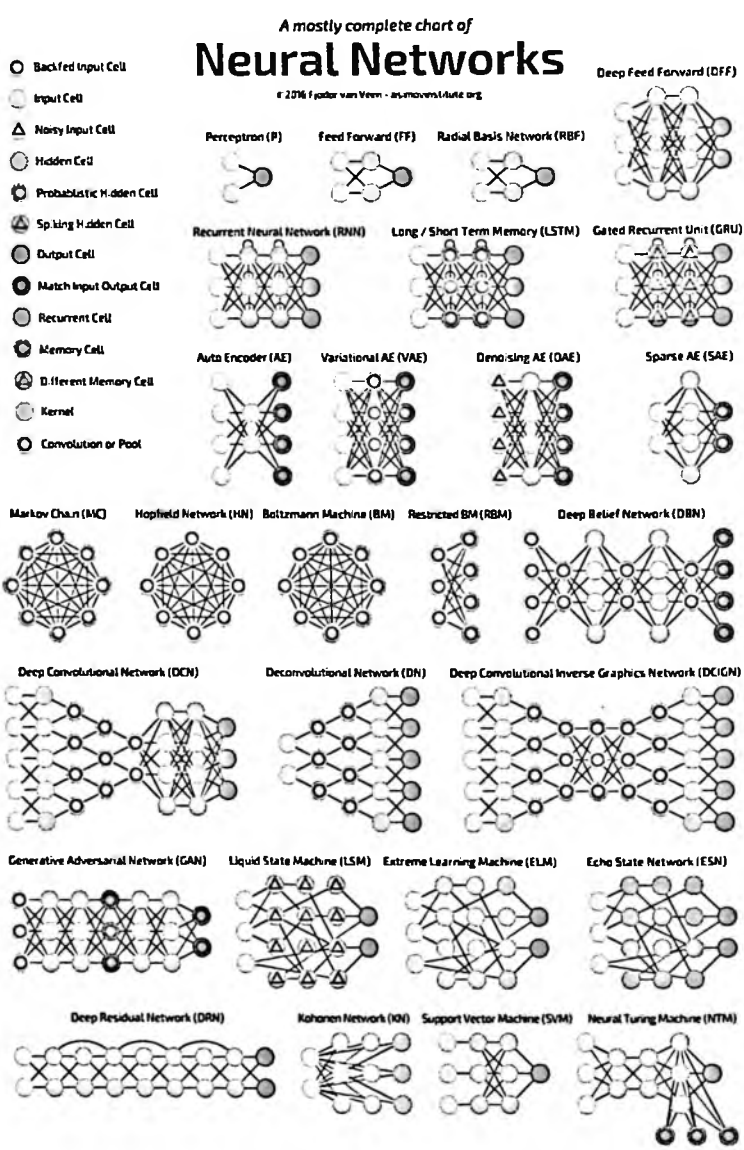
2-rasmda neyron tarmoqlarning grafik ko‘rinishlari ma’lum bo‘lgan arxitekturaning izohi keltirilgan.

1.4.2. Neyron tarmoq tasnifi

Neyron tarmoqlarning xaritasini tuzishda asosiy yechilmagan masala bu ularni qo‘llash holatini ifodalamasligidir, masalan, variatsion avtoshifratorlar, (VAE) avtoshifratorlar (AE) kabi ko‘rinishi mumkin, biroq o‘rganish jarayoni mutlaqo boshqa hisoblanadi. O‘rgatilgan tarmoqlar ko‘rinishlarida qo‘llash holatlari juda farqlanadi, chunki bu holatlarda VAE generator sifatida xizmat qiladi, signalni yangi tarmoqlar hosil qilinadi. AEda esa kiruvchi ma’lumotlarni o‘zi «xotirasida» mavjud namunalar bilan taqqoslagan holda qabul qiladi va o‘rganadi.

Shuni ta’kidlash lozimki, barcha qo‘llaniladigan abbreviatura umumiy holda qabul qilingan bo‘lsa-da, barchasini birdek qabul qilib bo‘lmaydi. Masalan: RNNni ko‘pincha rekursiv neyron tarmoq deb qabul qilishadi, biroq ko‘pincha rekurrent neyron tarmoqlarni belgilaydi. Shu bilan birga, RNN ko‘p takrorlanadigan arxitektar-moqlarda qo‘shimcha tarmoq holatda qo‘llanilishini uchratish mumkin, bunga LSTM, GRU va ikki tomonlama yo‘naltirilgan holatlar kiradi. Xuddi shunga o‘xshash AE tarmoqlar ham mavjud, ularda ham VAE, DAE va boshqalarni umumlashgan holda AE deb nomlashadi, biroq ish prinsipi farqlanadi. Ko‘p abbreviaturalar «N» harfi soni bilan ham farqlanadi, masalan, Convolutional Neural Network so‘zida neyron so‘zini ishlatmasak CNN abbreviatura CN deb nomlanadi, biroq mohiyat o‘zgarmaydi.

Keltirilgan arxitektura ro‘yxatini so‘nggi deb qabul qilishni tavsiya etmaymiz. Chunki kundan kunga yangi ko‘rinishdagi arxitekturalar yaratilmoqda, ular haqida to‘liq ma’lumot topish murakkab bo‘lib qoladi, shu sababli ushbu ro‘yxatni SI dunyosi haqida ma’lum bir ma’lumot olish ko‘rsatmasi deb qabul qilishingiz lozim bo‘ladi.



1.2-rasm. Asosiy neyron tarmoqlarning grafik ko‘rinishlari.

1-jadval

O‘rganish xususiyati bo‘yicha neyron tarmoqlar tasnifi

USUL	IZOH
O‘qituvchi yordamida o‘rganish	Kompyuterga kiruvchi ma’lumotlar va taxminiy chiquvchi natijalar «o‘qituvchi» yordamida ifodalaniladi. Asosiy maqsad umumiy kiruvchi va chiquvchi ma’lumotlar o‘rtasida umumiy qonuniyatni aniqlash.
O‘qituvchisiz o‘rganish	O‘rganish jarayonida algoritmga kutilayotgan natija ifodalanilmaydi, balki algoritmning o‘zi chiquvchi natijani belgilaydi. O‘qituvchisiz o‘rganish asosiy maqsad (ma’lumotlar o‘rtasida yashirin qonuniyatni aniqlash) bo‘lib qoladi.
Mustahkamlash orqali o‘rganish	Kompyuter dastur dinamik muhit bilan o‘zaro bog‘langan holda ma’lum bir masalani bajarishi lozim bo‘ladi, bunda o‘qituvchi dastur maqsadga yaqinlashganlik darajasini belgilamaydi.

2-jadval

Ma’lumotlarni tahlil qilish bo‘yicha neyron tarmoqlar tasnifi

KATEGORIYA	IZOH
Tasnif	O‘qituvchi yordamida o‘rganish algoritmlari kirish qismiga uzatilayotgan ma’lumotlar tegishli bo‘lgan sinfni aniqlaydi.
Klasterlash	O‘qituvchisiz o‘rganish algoritmlari turi bo‘lib, kiruvchi ma’lumotlar bir yoki bir necha klasterlarga taqsimlanadi.

KATEGORIYA	IZOH
Regressiya	O‘qituvchi yordamida o‘rganish algoritmlari turi bo‘lib, chiquvchi qiymatlar uzluksiz hisoblanadi.
O‘lchovni kamaytirish	Kiruvchi ma’lumotlarni qisqartirilgan fazaviy o‘lchamda ifodalash usulida soddalashtirish algoritmlari.

3-jadval

Neyron tarmoqlar modellari

ALGORITM	USUL	KATE-GORIYA	IZOH
Ordinary Least Squares Regression (OLSR)	O‘qituvchi yordamida o‘rganish	Regressiya	Chiziqli regressiya algoritmi.
Linear Regression	O‘qituvchi yordamida o‘rganish	Regressiya	Bashorat qilishda chiziqli funksiyani qo‘llovchi regressiya algoritmi sinfi.
Logistic Regression	O‘qituvchi yordamida o‘rganish	Regressiya	Ehtimollik modeliga asoslangan regressiya algoritmi.

ALGORITM	USUL	KATE-GORIYA	IZOH
Stepwise Regression	O‘qituvchi yordamida o‘rganish	Regressiya	Regressiya algoritmi.
Multivariate Adaptive Regression Splines (MARS)	O‘qituvchi yordamida o‘rganish	Regressiya	Chiziqli bo‘lmagan holatlarni yechishda qo‘llaniladigan regressiya algoritmi.
Locally Estimated Scatterplot Smoothing (LOESS)	O‘qituvchi yordamida o‘rganish	Regressiya	Bashorat etish uchun qo‘llaniladigan silliq tekislikni quruvchi regressiya algoritmi.
k-Nearest Neighbour (kNN)	O‘qituvchi yordamida o‘rganish	Regressiya, tasnif	Tasnif va regressiya turiga mansub algoritm bo‘lib, natijani bashorat etishi uchun xususiyatlar tekisligida k masofadagi qo‘shni ma’lumotlarni qo‘llaydi.
Learning Vector Quantization (LVQ)	O‘qituvchi yordamida o‘rganish	Tasnif	Berilgan namunalar asosida o‘qituvchi yordamida

ALGORITM	USUL	KATE-GORIYA	IZOH
			o‘rganish usulini qo‘llovchi algoritm tasnifi.
Self-Organizing Map (SOM)	O‘qituvchisiz o‘rganish	Tasnif	Raqobatli o‘rganish usulini qo‘llovchi sun’iy neyron tarmoq turi.
Locally Weighted Learning (LWL)	O‘qituvchi yordamida o‘rganish	Regressiya, tasnif	Sust o‘rganish algoritmi.
Ridge Regression	O‘qituvchi yordamida o‘rganish	Regressiya	Tixonov sozlanmasi deb nomlanadi.
Least Absolute Shrinkage and Selection Operator (LASSO)	O‘qituvchi yordamida o‘rganish	Regressiya	Xususiyatlarni ajratib olish va nazorat etish regressiya algoritmi.
Elastic Net	O‘qituvchi yordamida o‘rganish	Regressiya	Turli ko‘rinishdagi boshqaruv usullarini muvofiqlashtiruvchi regressiya algoritmi.

ALGORITM	USUL	KATE-GORIYA	IZOH
Least-Angle Regression (LARS)	O‘qituvchi yordamida o‘rganish	Regressiya	Katta hajmdagi ma’lumotlarga mo‘ljallangan regressiya algoritmi.
Classification and Regression Tree (CART)	O‘qituvchi yordamida o‘rganish	Regressiya, tasnif	Tasnif va regressiya masalalarini yechishda daraxtsimon modelni qo‘llovchi rekursiv algoritm.
Iterative Dichotomiser 3 (ID3)	O‘qituvchi yordamida o‘rganish	Tasnif	Tabiiy tilda berilgan ma’lumotlarni qayta ishlash masalalarida keng qo‘llanuvchi daraxtsimon algoritm.
C4.5 and C5.0	O‘qituvchi yordamida o‘rganish	Tasnif	Ma’lumotlarni entropiyalashga mo‘ljallangan daraxtsimon algoritm.
Chi-squared Automatic Interaction	O‘qituvchi yordamida o‘rganish	Tasnif	Marketing sohasida keng qo‘llaniladigan

ALGORITM	USUL	KATE-GORIYA	IZOH
Detection (CHAID)			daraxtsimon algoritm ko'rinishi.
Decision Stump	O'qituvchi yordamida o'rganish	Tasnif	Daraxtsimon masalalarni yechish ko'rinishi.
M5	O'qituvchi yordamida o'rganish	Regressiya, tasnif	Daraxtsimon masalalarni yechish ko'rinishi.
Conditional Decision Trees	O'qituvchi yordamida o'rganish	Tasnif	Daraxtsimon masalalarni yechish ko'rinishi.
Naive Bayes	O'qituvchi yordamida o'rganish	Tasnif	Sodda tasodifiy klassifikator.
Gaussian Naive Bayes	O'qituvchi yordamida o'rganish	Tasnif	Sodda tasodifiy klassifikator.
Multinomial Naive Bayes	O'qituvchi yordamida o'rganish	Tasnif	Matnli hujjatlarga mo'ljallangan Bayes tarmoq ko'rinishi
Averaged One-Dependence Estimators (AODE)	O'qituvchi yordamida o'rganish	Tasnif	Tasodifiy klassifikator.

ALGORITM	USUL	KATE-GORIYA	IZOH
Bayesian Belief Network (BBN)	O‘qituvchi yordamida o‘rganish	Tasnif	Neyron tarmoq ko‘rinishi bo‘lib, tasodifiylikni hisoblash uchun qo‘llanishi mumkin.
Random Forest	O‘qituvchi yordamida o‘rganish	Regressiya, tasnif	Regressiya va tasniflash amallarini muvofiqlashtirgan holda o‘rganish usuli.
k-Means	O‘qituvchisiz o‘rganish	Klasterlash	O‘qituvchisiz o‘rganishning sodda algoritmlaridan biri.
k-Medians	O‘qituvchisiz o‘rganish	Klasterlash	Klasterlash algoritmi.
Expectation Maximisation (EM)	O‘qituvchisiz o‘rganish	Klasterlash	Biologik ma’lumotlarni qayta ishlashda keng qo‘llaniladi.
Hierarchical Clustering	O‘qituvchisiz o‘rganish	Klasterlash	Klasterlar iyerarxiyasini yaratuvchi

ALGORITM	USUL	KATE-GORIYA	IZOH
			klasterlash algoritmi.
Perceptron	O‘qituvchi yordamida o‘rganish	Regressiya, tasnif	Sun’iy neyron tarmoqning sodda turi.
Back-Propagation	O‘qituvchi yordamida o‘rganish	Regressiya, tasnif	Sun’iy neyron tarmoqning o‘rganish algoritmi.
Hopfield Network	O‘qituvchi yordamida o‘rganish	Tasnif	Sun’iy neyron tarmoqning rekurrent ko‘rinishi. Assotsiativ xotira.
Radial Basis Function Network (RBFN)	O‘qituvchi yordamida o‘rganish	Regressiya, tasnif	Sun’iy neyron tarmoqning ko‘rinishi bo‘lib, radial bazoviy funksiyalarni qo‘llaydi va vaqt ko‘rsatkichli bashoratlarda qo‘llaniladi.
Deep Boltzmann Machine (DBM)	O‘qituvchi yordamida o‘rganish	Tasnif	Tasodifiy Yashirin o‘zgaruvchilardan iborat qatlamlarga ega binar tasodifiy Markov ketma-

ALGORITM	USUL	KATE-GORIYA	IZOH
			ketligiga o‘xshash Bolsman mashinasi.
Deep Belief Networks (DBN)	O‘qituvchi yordamida o‘rganish	Tasnif	Chuqur neyron tarmoqning ko‘rinishi.
Convolutional Neural Network (CNN)	O‘qituvchi yordamida o‘rganish	Tasnif	Jonzotlarning ko‘rish sistemasidan olingan strukturaga ega sun’iy neyron tarmoq ko‘rinishi.
Stacked Auto-Shifrators	O‘qituvchisiz o‘rganish	Tasnif	O‘qituvchisiz o‘rganish uchun sun’iy neyron tarmoq ko‘rinishi.
Principal Component Analysis (PCA)	O‘qituvchisiz o‘rganish	O‘lchovni kamaytirish	O‘lchovni kamaytirish algoritmi.
Principal Component Regression (PCR)	O‘qituvchi yordamida o‘rganish	Regressiya	Asosiy komponentlar usuliga asoslangan regressiya algoritmi.

ALGORITM	USUL	KATE-GORIYA	IZOH
Partial Least Squares Regression (PLSR)	O‘qituvchi yordamida o‘rganish	Regressiya	Asosiy komponentlar regressiyasi bilan bog‘liq bo‘lgan regressiya algoritmi.
Sammon Mapping	O‘qituvchisiz o‘rganish	O‘lchovni kamaytirish	O‘lchovni kamaytirish algoritmi.
Multidimensional Scaling (MDS)	O‘qituvchisiz o‘rganish	O‘lchovni kamaytirish	O‘lchovni kamaytirish algoritmi.
Projection Pursuit	O‘qituvchisiz o‘rganish	O‘lchovni kamaytirish	O‘lchovni kamaytirish algoritmi
Linear Discriminant Analysis (LDA)	O‘qituvchisiz o‘rganish	O‘lchovni kamaytirish	O‘lchovni kamaytirish algoritmi.
Mixture Discriminant Analysis (MDA)	O‘qituvchi yordamida o‘rganish	Tasnif	Aralash ko‘rinishdagi modellarga asoslangan tasnif usuli.

ALGORITM	USUL	KATE-GORIYA	IZOH
Quadratic Discriminant Analysis (QDA)	O‘qituvchi yordamida o‘rganish	Tasnif	Aralash ko‘rinishdagi modellarga asoslangan tasnif usuli.
Flexible Discriminant Analysis (FDA)	O‘qituvchi yordamida o‘rganish	Tasnif	Tasnif usuli.
Boosting	O‘qituvchi yordamida o‘rganish	Tasnif	Mashinali o‘rganish modellariga bashorat aniqligini oshirish maqsadida qo‘llaniladigan algoritmlar.
Bootstrap aggregating (Bagging)	O‘qituvchi yordamida o‘rganish	Tasnif	Mashinali o‘rganish algoritmlari aniqlik va barqarorlikni oshirish maqsadida yaratilgan algoritm.
AdaBoost	O‘qituvchi yordamida o‘rganish	Tasnif	Busting algoritmi.
Stacked Generalization (blending)	O‘qituvchi yordamida o‘rganish	Tasnif	Turli ko‘rinishdagi mashinali o‘rganish modellarini

ALGORITM	USUL	KATE-GORIYA	IZOH
			kombinatsiyalash uchun qo‘llaniladi.
Gradient Boosting Machines (GBM)	O‘qituvchi yordamida o‘rganish	Tasnif	Sodda bashorat qiluvchi modellar ko‘rinishida tasniflash yondashuvini taklif etadi.
Gradient Boosted Regression Trees (GBRT)	O‘qituvchi yordamida o‘rganish	Regressiya	Sodda bashorat qiluvchi modellar ko‘rinishlar asosida bashorat etuvchi modelni taklif etuvchi regressiya modell

4-jadval

"Kompyuterli o'rganish" masalalarini dasturlashda keng qo'llaniladigan dasturiy kutubxonalar ro'yxati

KUTUBXONA	DASTURLASH TILI	IZOH
Shogun	C++	Ushbu maxsus paket mashinali o'rganish uchun mo'ljallangan keng qamrovli obyektlar va o'rganish sozlanmalariga ega bo'lgan paket bo'lib, ma'lumotlarni tasniflash, regressiya yoki chuqur o'rganishga mo'ljallangan.
Weka	Java	Umumiy qo'llaniladigan paket.
Kernlab	R	Yadro asosida tasniflash va o'lchovni kamaytirish.
Dlib	C++	Portlarga ajratish, tahrirlash.
NLTK	Python	Chiziqli regressiya, tartiblash, tasniflash.
Orange	Python	Yangi foydalanuvchilar va ekspertlar uchun ma'lumotlarni vizuallashtirish va tahlil etish uchun ochiq kodli kutubxona. Katta hajmdagi uskunalar yordamida interaktiv jarayon ta'minlanadi.

KUTUBXONA	DASTURLASH TILI	IZOH
Java-ml	Java	Mashinali o‘rganish algoritmining kolleksiyasi.
pyML	C++; Python	Mashinali o‘rganish uchun Python tilida yozilgan interaktiv obyektga mo‘ljallangan muhit. Yadroga asoslangan SVMga yo‘naltirilgan PyML boshqa usullar. Kutubxona Linux va Mac OS X tomonidan qo‘llaniladi.
Mlpy	Python	Mlpy Phyton uchun modul hisoblanib, NumPy/SciPy va GNU Scientific kutubxonasi uchun sozlanma hisoblanadi.
Pybrain	Python	pybrain – Python-Based Reinforcement Learning ning qisqartmasi bo‘lib, Artificial Intelligence and Neural Network Library (Phyton tilida yozilgan kutubxona, mustahkamlash asosida o‘rganish, sun’iy intellekt va sun’iy neyron tarmoqlar).
Torch	C++; Lua	Mashinali o‘rganish uskunalari to‘plamidan iborat uskunalarning ilmiy to‘plami.

KUTUBXONA	DASTURLASH TILI	IZOH
scikit-learn	Python; Cython	Keng qo'llaniladigan kutubxona. Imkoniyati keng va qo'llanishda sodda hisoblanadi.
Theano	Python	GPU usuli yordamida ishlaydigan samarali hisoblash kutubxonasi. Chuqur o'rganishda qulaylik yaratadi.
Pylearn2	Python	Theano yaratilgan mashinali o'rganish uskunalarining to'plami.
MDP	Python	Ma'lumotlarni qayta ishlash uchun uskunalar modulli to'plami.
Spark	Java	Katta hajmdagi ma'lumotlarni qayta ishlash uchun tezkor va universal maket.
Mahout	Java	Hadoop asosida yaratilgan mashinali o'rganish muhiti.
Mallet	Java	Tabiiy til sohasida statistik qayta ishlash uchun Java paketi.
JSAT	Java	Mashinali o'rganish va statistik qayta ishlash uskunalarining java paketi.

KUTUBXONA	DASTURLASH TILI	IZOH
Accord.NET	.NET	.NET asosida ilmiy hisoblashlarga mo'ljallangan paket.
Vowpal Wabbit	C++	Tez o'rganish uchun mo'ljallangan, BSD litsenziyasi asosida ishlab chiqilgan.
MultiBoost	C++	Busting algoritmlarini qo'llovchi S++ da yaratilgan paket.
TensorFlow	Python, C++	Ochiq kodli Google kutubxonasi.

1.5. To'g'ri chiziqli neyron tarmoqlari

(Feed forward neural networks, FF or FFNN) va perseptronlar (perceptrons, P) eng sodda chiziqli taqsimlangan tarmoq bo'lib, ular kirish qismidan chiqish qismiga uzatadi. Neyron tarmoqlar odatda, kirish qatlami yashirin qatlam va chiqish qatlami neyronlaridan iborat. Bir qatlam neyronlari o'zaro bog'lanishga ega bo'lmaydi. Biroq har bir qatlam neyroni qo'shni qatlam neyroni bilan bog'langan bo'ladi. Eng sodda tarmoq ikkita kirish va bitta chiqish neyronidan iborat bo'ladi, u oddiy mantiqiy ifodani izohlashi mumkin. FFNN da odatda teskari bog'lanishda xatolikni aniqlash usulida o'rganish olib boriladi, ya'ni kirish qatlami juftligiga kutilayotgan natija uzatilib model yaratiladi. Xatolik deganda kutilayotgan chiquvchi ma'lumotlarning kiruvchi ma'lumotlarga nisbatan nomuvofiqligi tushuniladi (masalan: o'rta kvadrat qiymatning farqlanishi). Tarmoqda yetarlicha yashirin qatlam neyronlar mavjud bo'lsa, u holda kiruvchi va chiquvchi berilganlar

o‘rtasida bog‘lanishni o‘rnatishi mumkin. Amalda esa bu ko‘rinishdagi chiziqli taqsimlangan tarmoqlardan foydalanish faqat boshqa ko‘rinishdagi tarmoqlar bilan hamkorlikda qo‘llaniladi.

1943-yili «Nerv faolligi bilan bog‘liq fikrlarni mantiqiy aniqlash» maqolasida U. Mak-Kallok va U. Pitts sun’iy neyron tarmoq tushunchasini birinchi marta olg‘a surdi. Ular tomonidan sun’iy neyron modeli taklif etildi. 1949-yili D. Xebb o‘zining «Holatni tashkil etish» nomli ishida neyronlarni o‘qitishning asosiy prinsiplarini izohladi.

Bir necha yil o‘tgach bu g‘oyalar asosida amerikalik neyrofiziolog Frenk Rozenblatt inson ongi modelini tashkil qiluvchi qurilma sxemasini taklif etdi va unga perseptron deb nom berdi. Perseptron elektromexanik xotira yacheykasida fotoelementlardan signallarni uzatib, sensorli maydonni hosil qiladi. Yacheykalar konnektivizm asosida tasodifiy o‘zaro bog‘lanadi. Perseptron obrazlarni tasniflashini «o‘rganish» uchun maxsus iteratsion usul ishlab chiqilgan bo‘lib, inson o‘rganish jarayoni kabi o‘z xatoliklarini to‘g‘rilangan holda o‘rganish usuli qo‘llanilgan. Bundan tashqari u yoki bu harfni aniqlash uchun perseptron harf o‘ziga xos xususiyatini statistik usulda ajratib bergan, biroq individual ahamiyatsiz qismlari bundan mustasno. Demak, shu usulda perseptron turli ko‘rinishda yozilgan harflarni bitta yagona obrazga umumlashtirish qobiliyatiga ega bo‘lgan. Shunday bo‘lsa-da, perseptron imkoniyatlari cheklangan, bir tomoni ko‘rinmas bo‘lgan harflarni, yoki hajmi jihatdan to‘g‘ri kelmaydigan, burilgan, surilgan harflarni o‘rgangan harflari bilan solishtirgan holda ishonchli aniqlab bermagan.[7]

Perseptron yaratilishining asosiy maqsadi obrazlarni aniqlash mashinasini yaratish emas, balki inson ongining ishlash prinsipi modelini yaratish, ya’ni intellekt ishini chuqur o‘rganish va tatbiq etish hisoblanadi.

Elementar perseptron 3 turdagi elementlardan: S A va R elementlardan iborat. S elementlar bu — retseptorlar qatlami. Ushbu retseptorlar A elementlar bilan qo‘zg‘alish holatida bog‘lanadi. Har bir

retseptor ikki holatdan birida bo'lishi mumkin— tinch va qo'zg'alish. A elementlar chegaralangan qiymatda summator ko'rinishda ifodalanadi. Ya'ni, retseptorlardan keluvchi qo'zg'alishlar darajasining yig'indisi qo'zg'algan A elementlar signallari summatorga R uzatiladi, bunda i elementdan keluvchi signal w_i koeffitsiyent bilan uzatiladi **[10]**.

Ko'rib chiqilgan oddiy elementlarning barchasi sodda hisoblanadi, chunki ular keskin o'zgarib turuvchi funksiyalarni qo'llaydi. Murakkab masalalarni hal qilish uchun boshqa ko'rinishdagi funksiyalardan foydalanishni talab etadi, masalan, chiziqli funksiyalar.

Natijada Rozenblatt quyidagi g'oyalarni olg'a surdi:

Perseptron bu S, A, R elementlardan iborat tarmoq bo'lib, o'zaro ta'sirni ta'minlovchi o'zgaruvchi matritsaga W (elementlari W_{ij} — og'irlik koeffitsiyentlari)ga ega va u tarmoqning oldingi faollik holati bilan aniqlanadi.

Ketma-ketlik bog'lanishlarga ega perseptron bu S elementga yaqin bir elementdan d mantiqiy masofada S elementga yaqin d+1 mantiqiy masofada joylashgan element bilan tugaydigan tizimga aytiladi.

Kesishmali bog'lanishga ega perseptron bir turdagi (S, A yoki R) elementlar o'rtasida bog'lanish mavjud bo'lgan tizim bo'lib, ushbu elementlar S elementlardan bir xil mantiqiy masofada joylashgan, qolgan bog'lanishlar esa ketma-ket ko'rinishda teskari bog'lanishli perseptron mantiqiy uzoq bo'lgan elementdan mantiqiy yaqin bo'lgan elementgacha bog'lanish bo'lgan tizimga aytiladi.

Oddiy perseptron deb quyidagi talablarga javob beruvchi tizimga aytiladi:

tizimda faqat bitta R element mavjud (tabiiyki, ular barcha A element og'irliklari bilan bog'langan);

ketma-ket bog'lanishga ega perseptron bo'lib, u faqat S elementdan A elementga va A elementdan R elementga harakatlangan;

S elementlardan A elementlargacha bo'lgan barcha bog'lanishlarning (S—A bog'lanishlar) og'irliklari o'zgarmasdir;

har bir bog‘lanishning uzatish vaqti yoki nolga teng yoki belgilangan o‘zgarmas τ teng bo‘ladi;

barcha S, A, R elementlarni faollashtiruvchi funksiyalar qo‘yidagi kurinishga teng:

$$U_i(t)=f(a_i(t)),$$

Bunda, $a_i(t)$ – u_i elementlar kirish qismiga bir vaqtning o‘zida keluvchi barcha signallarning algebraik yig‘indisidir.

Elementar perseptron bu sodda perseptron bo‘lib, unda barcha elementlar – sodda. Bu holatda faollashtirish funksiyasi

$$C_{ij}(t)=U_i(t-\tau)W_{ij}(t)$$

ko‘rinishga ega.

Olimlar perseptronlarning xususiyatlarini inobatga olgan holda 5 sinfini ajratgan:

Diametri bo‘yicha cheklangan perseptron xususiy mantiqqa ega aniqlangan X o‘lcham, ma’lum bir belgilangan qiymatdan oshmaydi.

Cheklangan ketma-ketlikka ega perseptron har bir xususiy bog‘lanish cheklangan sondagi X nuqtadan iborat.

Gamba perseptroni har bir xususiy bog‘lanish chegaralangan chiziqli funksiya ko‘rinishda, ya’ni miniperseptron bo‘lishi lozim.

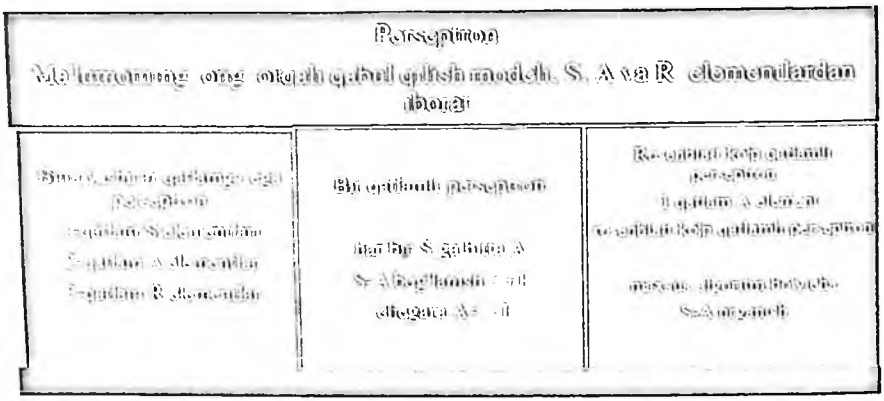
Tasodifiy perseptronlar cheklangan perseptronlar bo‘lib, xususiy bog‘lanishlar tasodifiy mantiqiy (bool) funksiyalar ko‘rinishida bo‘ladi.

Cheklangan perseptron — xususiy bog‘lanishlar to‘plami cheksiz, biroq aniq qiymatlar a_i ketma-ketligi esa cheklangan.

O‘rganish algoritmi

Perseptronni o‘rganishning klassik usuli — bu xatolikni to‘g‘rilash usuli. Bu shunday usulki, joriy perseptron ta’sirlanishi belgilangan holatda bo‘lgunga qadar bog‘lanish og‘irlik ko‘rsatkichi o‘zgarmas qoladi. Perseptron ta’sirlanishida o‘zgarish sezilganda bog‘lanish og‘irlik ko‘rsatkichi 1 ga aylanadi, xatolik ko‘rsatkichiga mos holda uning aksiga o‘zgaradi (+/-).

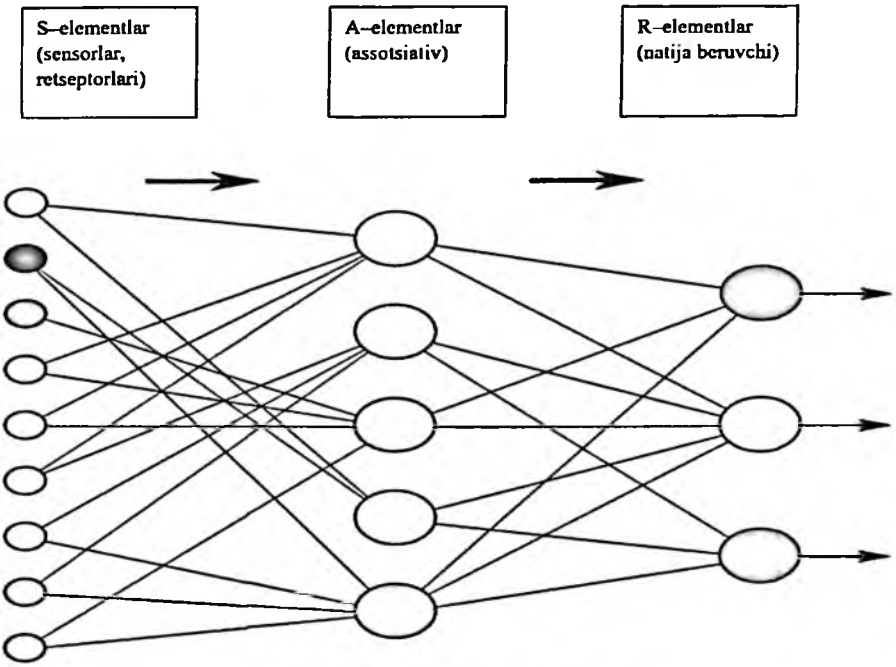
Quyidagi chizmaga e’tibor beramiz:



1.3-rasm. Perseptron berilganlarni qabul qilish modeli.

Bitta yashirin qatlamli perseptron bittadan S, A, R qatlamlardan iborat. Qatlamlardagi barcha neyron keyingi qatlam har bir neyroni bilan bog‘liq.

Bir qatlamli perseptron birtadan S, A, R qatlamlardan iborat,biroq har bir S neyronga bir A neyron bog‘langan, S-A bog‘lanish og‘irlik ko‘rsatkichi +1 ga teng,
A cheklanmasi (porog)+1 ga teng.



1.4-rasm. Perseptron tarmog‘i.

Ko‘p qatlamli perseptron ikki ko‘rinishga ega: Rozenblatt ko‘p qatlamli perseptron va Rumelxart ko‘p qatlamli perseptron.

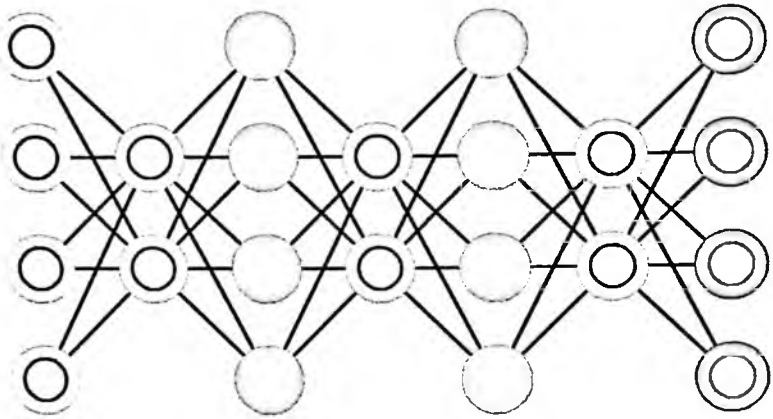
Rozenblatt ko‘p qatlamli perseptron bittadan oshiq A-elementlar qatlamiga ega.

Rumelxart ko‘p qatlamli perseptroni bu Rozenblatt ko‘p qatlamli perseptronning xususiy holati bo‘lib, ikki xususiyatga ega:

1. S-A bog‘lanishlar ixtiyoriy og‘irlikka ega bo‘lishi mumkin va A-R bog‘lanish bilan birgalikda o‘rganishi mumkin.

2. O‘rganish maxsus algoritm bo‘yicha olib borilib, teskari bog‘lanish orqali xatolikni tekshirish usuli deb nomlanadi.

1.6. Chuqur ishonchli tarmoqlar



Yashirin hujayrasi

Kirish chiqish hujayrasiga o‘xshashlik

Kirish hujayrasiga qayta mutojat

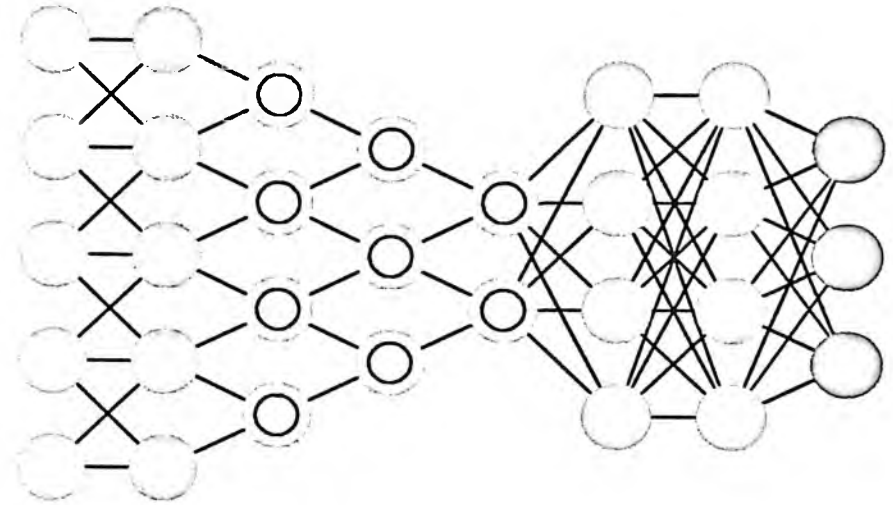
Ehtimolli yashirin hujayra

1.5-rasm. Chuqur ishonchli tarmoqlar.

Chuqur ishonchli tarmoqlar (Deep belief networks, DBN) — ushbu tarmoqlar, bir necha RBM yoki VAE kompozitsiyalaridan iborat. Ushbu tarmoqlar samaraliligi shundan iboratki, har bir tarmoq birma-bir o‘rganish jarayonidan o‘tib, keyingi tarmoq oldingi tarmoqni kodlashtirishni o‘rganishi lozim. Unda aynan joriy holatda to‘g‘ri keladigan yechimlardan optimalini topadi, biroq bu natija optimalligini kafolatlamaydi. DBN lar contrastive divergence usullari asosida yoki xatolikni teskari yo‘nalishda aniqlash usulida o‘rganish jarayonini olib boradi, RBM yoki VAE kabi berilganlarni ehtimollik modeli ko‘rinishda aniq ifodalashni o‘rganadi. Keyinchalik o‘rgatilgan va

statsionar holatga keltirilgan modelni yangi berilganlarni yaratishda qo'llash mumkin [7].

1.6.1. Konvolyutsion neyron tarmoqlar



1.6-rasm. Konvolyutsion neyron

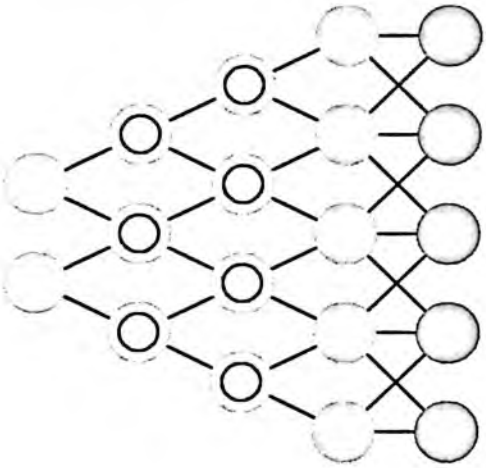
Konvolyutsion neyron tarmoqlar **(convolutional neural networks, CNN)** va chuqurlashgan konvolyutsion neyron tarmoqlar, **(deep convolutional neural networks,** DCNN) boshqa tarmoqlardan ancha farqlanadi, ular asosan tasvirlarni qayta ishlashda, shu bilan birga boshqa ko'rinishdagi ma'lumotlarni ayni damda audio ma'lumotlarni

qayta ishlashda qo‘llanilishi mumkin. Odatiy misol sifatida CNNni qo‘llab, bir necha ko‘rinishdagi tasvirlarni uzatish va ularni tasniflash, mushuk rasmini berganda «mushuk» deb, itning rasmini berganda «it» deb javob chiqarilishini keltirish mumkin. CNN odatda kirish qismi «skaner» holatda bo‘lib, olingan ma’lumotlarni barchasini qayta ishlashga mo‘ljallanmagan. Masalan: 200 x 200 piksel hajmdagi tasvirni chiqarish uchun 40 000 tugundan iborat qatlam yaratish shart emas. Buning uchun 20x20 hajmdagi skanerlash qatlami yaratiladi, unga tasvirning 20 x 20 pikseli yuklanadi (odatda, yuqori chap burchakdan boshlab yuklanadi). So‘ngra, kirish qismini o‘rganish uchun ham qo‘llash mumkin va skanerni bir piksel o‘ngga surib keyingi 20x20 pikselni o‘rganadi.

E’tiborlisi shuki, kirish 20 piksellarini ko‘chirish kerak emas, balki tasvirni 20x20 hajmdagi bloklarga ajratib, u bo‘yicha harakat qilinadi. Kirish qatlamidagi ma’lumotlar konvolyutsion qatlamdan o‘tkazilib bunda har bir tugun barcha tugun bilan bog‘lanish holatida bo‘lmaydi. Har bir tugun faqat qo‘shni yacheykalar bilan bog‘lanadi (bog‘lanish darajasi qo‘llash davomida aniqlanadi, odatda bir nechtadan oshmaydi). Konvolyutsion tarmoqlar qisqarib borib chuqurlashadi va kirish ma’lumotlar omillarini ajratib oladi (demak, 20 tali qatlam, avval 10 tali qatlam, so‘ngra 5 qatlamga o‘tadi). Bu yerda baravarga qisqarish qoidasi qo‘llanilib, u aniq belgilangan: 32, 16, 8, 4, 2, 1.

Konvolyutsion urdan tashqari pul mavjud bo‘lib, birlashma – bu datallarni filtrlab olish: qo‘llaniladigan birlashma texnologiyasi bu 2x2 piksellar bo‘yicha eng qizil bo‘lgani o‘tkaziladi. Audio uchun CNN qo‘llashda asosan tovush to‘lqinlari va ular uzunligi bo‘yicha segmentlangan holatda kirish qatlamiga uzatiladi. Amalda asosan CNN da ko‘pincha FFNN qo‘llagan holda chiziqsiz abstraksiyalangan funksiyalarni qo‘llash imkonini beradi. Ushbu tarmoqlar DCNN deb nomlanadi, biroq nomlanishi va abbreviaturasi ko‘pincha bir-birining o‘rnini bosadi [8].

1.7. Dekonvolyutsion neyron tarmoqlar



Yadro

Konvolyutsiya yoki pulning

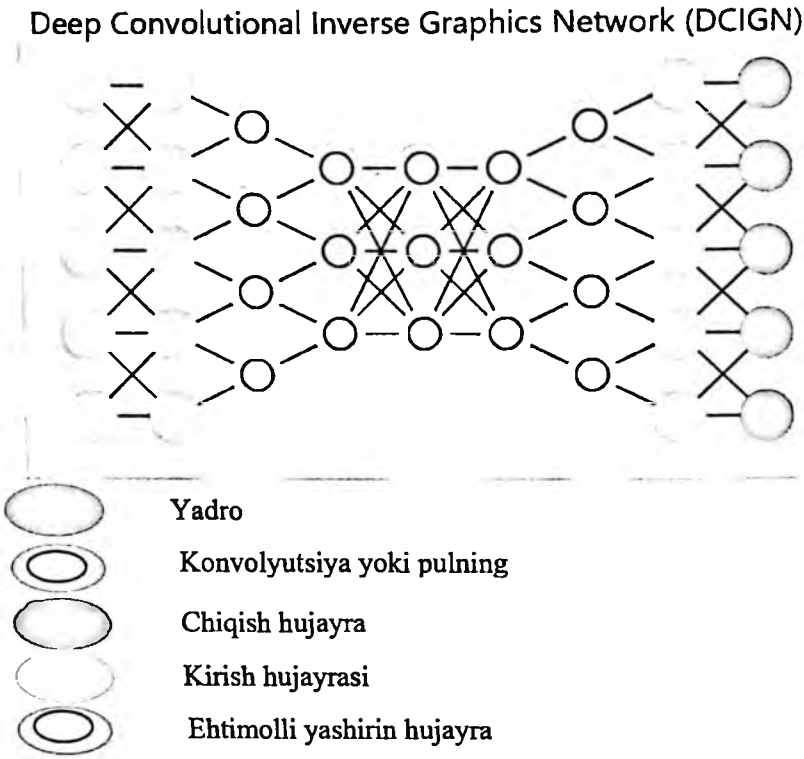
Chiqish hujayra

Kirish hujayrasi

1.7-rasm Dekonvolyutsion neyron

Dekonvolyutsion neyron tarmoqlar (deconvolutional networks, DN), teskari grafik tarmoqlar deb nomlanib, ular konvolyutsion tarmoqlarning aksi hisoblanadi. Faraz qilaylik, «mushuk» so‘zini o‘rganish uchun mushuklar tasvirini qayta ishlash asosida beriladigan tasvirlarni taqqoslash orqali o‘rganish olib boriladi. DNN tarmog‘ini FFNN bilan birlashtirish mumkin. Ko‘p hollarda tarmoqlarga ma’lumotlar satr ko‘rinishda emas, balki binar tasniflangan vektor ko‘rinishda: masalan: <0, 1> — bu mushuk, <1, 0> — bu it, <1, 1> esa — ham mushuk, ham it tushuniladi. CNNda uchraydigan birlashtirish qatlami o‘rniga uning aksi bo‘lmish interpolyatsiya yoki ekstrapolyatsiya amallari mavjud [9].

1.7.1. Chuqurlashtirilgan teskari konvolyutsion grafik tarmog‘i

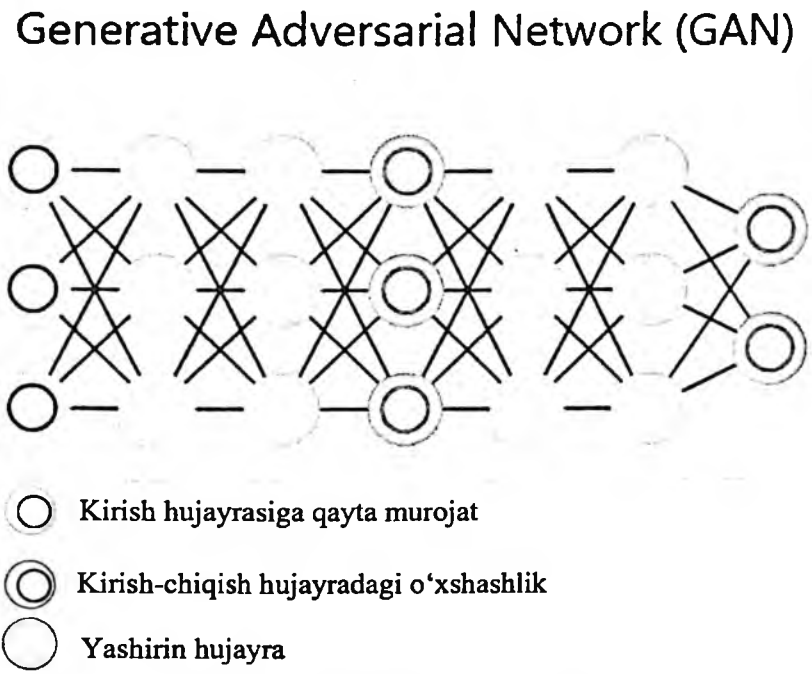


1.8-rasm. Chuqurlashtirilgan teskari konvolyutsion grafik tarmog‘i

Chuqurlashtirilgan teskari konvolyutsion grafik tarmog‘i (deep convolutional inverse graphics network, DCIGN) variatsion avtokodlashtiruvchi tarmoq bo‘lib, kodlashtiruvchi va deodlashtiruvchi qismlari mos ravishda konvolyutsion va dekonvolyutsion bo‘lib xizmat qiladi. DCIGN ehtimolliklar ko‘rinishida obyekt belgilarini modellashtiradi: mushuk va it tasvirlangan obyektni yaratishda o‘rganish mumkin,

holbuki, tarmoq faqat mushuk va it ko'rsatilgan tasvirlarni ko'rib o'rgangan. Shu bilan birga ikki obyektdan birini olib tashlash imkoni ham mavjud. Chuqurlashtirilgan teskari konvolyutsion grafik tarmog'i xatolikni teskari harakatda aniqlash usullni qo'llaydi [9].

1.7.2. Generativ-raqobatli tarmoq (generative adversarial network, GAN)



1.9-rasm. Generativ-raqobatli tarmoq

Generativ-raqobatli tarmoq (generative adversarial network, GAN) ikki neyrotarmoqlardan iborat: kontent yaratuvchi generator, uni baholovchi diskriminator. Ko'pincha, bu mos ravishda FFNN va CNN tarmoqlaridir. Diskriminator tarmog'i generator tomonidan yaratilgan

yoki o‘rgatuvchi ma’lumotlarni qabul qiladi. Diskriminatorga kelayotgan ma’lumot manbayiga asoslanib xatolik shakllanadi. Demak, generator va diskriminator o‘rtasida «musobaqa» paydo bo‘ladi: birinchisi ikkinchisini aldashni o‘rganadi, ikkinchisi esa yolg‘onni yechishni o‘rganadi. Ushbu ko‘rinishdagi tarmoqlarni o‘rganish mushkul, chunki ularning har birini o‘rganish bilan bir qatorda ularning o‘zaro muvofiqlashgan holda ishlashini sozlashni ham talab etadi [10].

1.8. Konvolyutsion neyron tarmoqlari (SNS, CNN)

Konvolyutsion neyron tarmoqlar (SNS, CNN) oddiy neyron tarmoqlarga o‘xshaydi: o‘z og‘irlik ko‘rsatkichini va o‘rnini o‘zgartiruvchi neyronlar asosida shakllantiriladi. Har bir neyron ma’lum bir berilganlarni qabul qiladi, ularning skalyar ko‘paytmasini olib boradi, ba’zi hollarda chiziqli bo‘lmagan tenglamalarni qo‘llaydi. Oddiy neyronlar kabi CNN to‘liqligicha bitta differensiyalanadigan to‘ldirish funksiyani ifodalaydi (samarali to‘ldirish): bir tomondan piksellari qayta ishlanmagan tasvirlar, ikkinchi tomondan tasvirni tavsiflovchi sinflarini yoki tasodifiy sinflar guruhini chiqaradi. Bunda to‘liq bog‘lanishga ega so‘nggi qatlamda yo‘qotish funksiyasi mavjud bo‘lib, neyron tarmoq bilan bog‘lanish jarayonida ishlab chiqilgan barcha qonuniyatlar bog‘lanish funksiyalari CNNda paydo bo‘ladi. Undagi farqni ko‘rib chiqamiz.

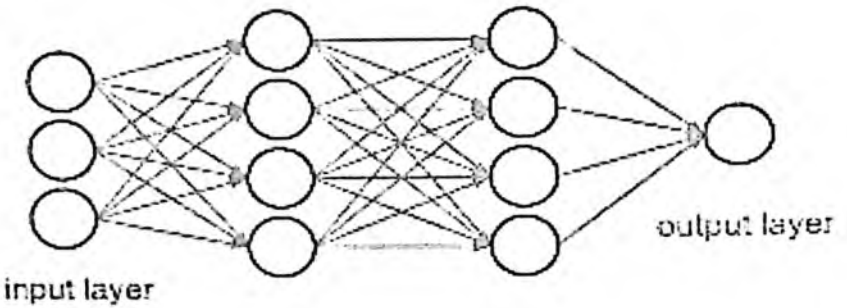
Konvolyutsion neyron tarmoqlar arxitekturasi «kirish qismidagi berilganlar tasvirini hosil qiladi» prinsipini taxmin qiladi, so‘ngra ma’lum bir xususiyatlarini arxitekturasiga mos qilib aniqlaydi. Ushbu xususiyatga asoslangan holda boshlang‘ich ma’lumotlarni taxmin qilish tarmoq parametrlaridan qisqartirilgan holda samarali foydalanish mumkin.

1.8.1. CNN arxitekturasining to‘liq izohi

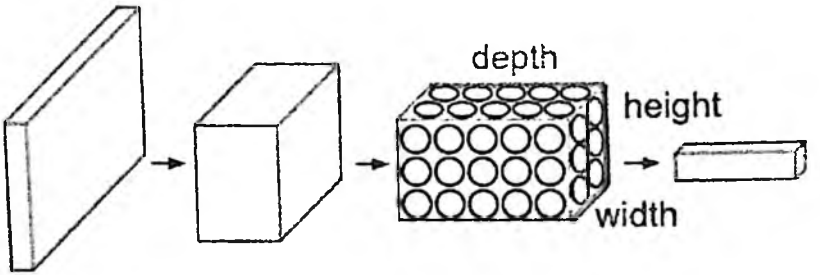
Ma’lumki, neyron tarmoqlar boshlang‘ich ma’lumotlarni qabul qiladi (bitta vektor), ma’lumotlarni bir necha yashirin qatlamlardan o‘tkazib ma’lumotlar siljishiga erishadi. Har bir yashirin qatlam bir necha neyronlardan iborat bo‘lib, har bir neyron oldingi qatlam neyronlari bilan bog‘lanishga ega bo‘ladi va bitta qatlam funksiyasida o‘zaro mustaqil bog‘lanishga ega bo‘lmaydi. So‘nggi to‘liq bog‘langan qatlam chiqish qatlami deb nomlanib, tasnifni aniqlashda sinflar sonini ifodalaydi.

Oddiy neyron tarmoqlar katta hajmdagi tasvirlar bilan ishlashda masshtablanmaydi. CIFAR-10 kompyuterli ko‘rish tizimida tasvir [32×32×3] (32 – kengligi, 32 – balandligi, 3 – ranglar kanali) hajmga ega, shuning uchun oddiy neyron tarmog‘idagi birinchi yashirin qatlamga ulangan neyron 3 072 (32*32*3) og‘irlikka ega bo‘ladi. Bir ko‘rishda o‘zgartirish mumkin deb bo‘ladi, biroq katta hajmdagi tasvirlarda to‘liq bog‘langan struktura masshtablanmaydi. Katta ruxsat ko‘rsatkichiga ega tasvirlar, masalan, [200×200×3], to‘liq bog‘langan neyron 120 000 og‘irlikka ega bo‘ladi. Bundan tashqari, yangi parametrlarni qo‘shish imkonini beruvchi bir necha neyronlar mavjud bo‘lishi talab etiladi. To‘liq bog‘lanish – juda ko‘p resursni talab etmoqda va katta hajmdagi parametrlar tez qayta o‘rganishdan o‘tadi.

Konvolyutsion neyron tarmoqlarda kiruvchi ma’lumotlar tasvir bo‘lib, boshqacha tarmoqni yaratish uchun cheklanishlar hosil qiladi. Oddiy neyron tarmoqlardan farqli ravishda CNN qatlamlari neyronlar uch o‘lchovli: eniga, bo‘yiga va chuqurlikka, ya’ni hajmni shakllantiruvchi o‘lchamlar asosida shakllanadi. Masalan, CIFAR-10 kirish qismida faollashgan kiruvchi hajm sifatida qabul qilinadi, hajmi esa 32×32×3 o‘lchovda shakllantiriladi. Neyronlar esa qatlamning ma’lum bir qismiga faollashgan holda bog‘lanadi. Bundan tashqari, chiquvchi natija qatlami aynan shu tizimda 1x1x10ga teng bo‘ladi, CNN to‘liq shakllantirish natijasida to‘liq tasvirni sinflarni belgilovchi vektorga aylantiriladi, bu o‘z navbatida chuqurlik o‘lchovi ko‘rinishda izohlanadi. Quyida ushbu jarayonning vizualizatsiyasi keltirilgan.



1.10-rasm. Oddiy neyron tarmoq



1.11-rasm. Konvolyutsion neyron tarmoq

1.10-rasmda oddiy uch o'lchovli neyrotarmoqda joylashgan neyronlar joylashuvi berilgan.

1.11-rasmda konvolyutsion uch o'lchovli neyrotarmoqda joylashgan bitta qatlamida neyronlar joylashuvi ko'rsatilgan. CNN har bir qatlami kiruvchi 3D hajmli ma'lumotni aktivlashgan chiquvchi 3D hajmli neyronlarga o'giradi. Ushbu misolda qizil kiruvchi qatlam tasvir ko'rinishda bo'lganligi sababli eni va bo'yi tasvir o'lchami bilan belgilanadi, chuqurligi esa 3 ga teng (qizil, yashil, ko'k kanallar) bo'ladi.

Natija: konvolyutsion neyrotarmoqlarning asosini qatlamlar tashkil qiladi. Har bir qatlam oddiy API bilan tavsiflanadi: **u** kiruvchi

3D hajmli ma'lumotlarni chiquvchi 3D hajmli ma'lumotlarga o'giradi, bunda o'zgartiruvchi funksiya parametriga ega bo'lgan va ega bo'lmagan ko'rinishda o'zgartiradi.

1.8.2. CNN qo'llaniladigan qatlamlar

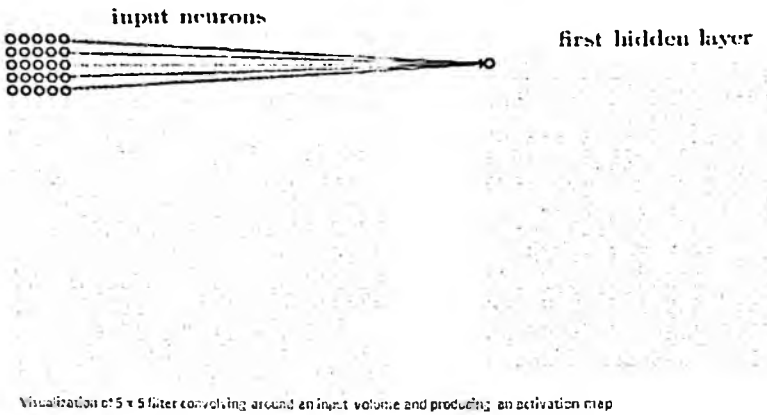
Yuqorida aytilgandek, sxematik ko'rinishda CNN – bu qatlamlar ketma-ketligi hisoblanadi. Har bir qatlam faollashgan hajmni differensiallashgan funksiya orqali boshqa hajm ko'rinishiga o'giradi. Konvolyutsion neyron tarmoqlarini shakllantirish uchun 3 ta asosiy qatlam qo'llaniladi:

1. Konvolyutsion.
2. Puling (quyi tanlanma yoki subdiskretizatsiya).
3. To'liq bog'langan qatlam.

Ushbu qatlamlar to'liq CNN arxitekturasini shakllantirish uchun qo'llaniladi.

CNN tarmog'i shakllantirishni ma'lum bir misolda tasvirni aniqlash tizimi tasnifini CIFAR-10 ko'rinishda keltirib, [INPUT — CONV — RELU — POOL — FC] arxitekturaga ega bo'ladi.

- INPUT (kiruvchi ma'lumotlar) [32×32×3]da tasvir haqida boshlang'ich ma'lumotlar saqlanadi (bu misolda 32 — eni, 32 — bo'yi, 3 — ranglar kanali R, G, B).
- CONV qatlami (konvolyutsiya qatlami) filtr ko'rsatkichi qiymatlarini tasvir piksellarining boshlang'ich qiymatlariga ko'paytiradi (elementlar bo'yicha ko'paytirish), so'ngra bu ko'paytmalar summalanadi. Kiritilgan tasvirning har bir unikal pozitsiyasi qiymatga ega bo'ladi. Masalan: 12 talik filtr qo'llanilsa, hajmi 32*32*12 [32×32×12] ga teng bo'ladi.



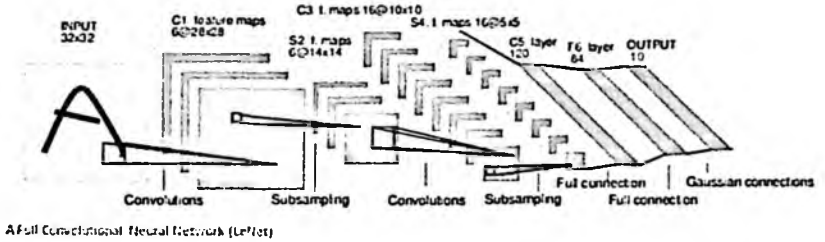
Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

1.12-rasm. Faollashgan xaritada 5x5 filtr orqali kiritilgan tasvirning konvolyutsiya jarayoni ko'rinishi

▪ RELU qatlam (chiziqli ajratish bloki) elementlar bo'yicha faollashtirish funksiyasini **f(x) = max(0,x)** qo'llaydi, bunda boshlang'ich ko'rsatkich 0 deb o'rnatiladi. Boshqacha qilib aytganda, RELU quyidagi amallarni bajaradi: agar x > 0, hajm o'zgarmas qoladi ([32×32×12]), agar x < 0, keraksiz qismlar 0 ga o'girish orqali qirqiladi.

▪ POOL qatlami (puling qatlami) fazoviy o'lchamlarni (eni va bo'yini) qisqartirish amallarini bajaradi, natijada hajmi [16×16×12] gacha qisqartiriladi. Ya'ni bu bosqichda xususiyatlar xaritasida chiziqli bo'lmagan zichlanish bajariladi. Ishning mantiqan bajarilishi quyidagicha: agar oldingi konvolyutsiya amalida ma'lum bir xususiyatar aniqlangan bo'lsa, keyingi qayta ishlashlar uchun to'liq tasvir uzatilishi shart emas, shuning uchun zichlangan holda tasvir uzatiladi.

▪ FC qatlami (to'liq bog'langan qatlam) kerakli sinfni aniqlash uchun N-o'lchamli vektorni chiqaradi (N – sinflar soni). Oldingi qatlam chiqish qismiga (xususiyatlar xaritasi) murojaat orqali va aniq sinfga mansub bo'lgan xususiyatlarni aniqlash orqali amalga oshiriladi.



A full Convolutional Neural Network (LeNet)

1.13-rasm. To‘liq konvolyutsion neyron tarmoq

Aynan shu usulda CNN qatlamlar bo‘yicha o‘tish orqali boshlang‘ich tasvirni ko‘chiradi, bunda boshlang‘ich piksellar qiymatlariga asoslangan holda sinflarni aniqlash amali bajariladi. Ahamiyatli bo‘lmagan qatlamlar ham parametrlarga ega bo‘ladi. Xususan, konvolyutsiya qatlami va to‘liq bog‘langan qatlam o‘zgartirish amalini bajaradi, bu esa boshlang‘ich faollashgan hajm ko‘rsatkich bilan bir qatorda parametr (og‘irlik va siljish ko‘rsatkichi)ga asoslangan holda bajariladi. Ikkinchi tomondan chiziqli ajratish bloki va puling qatlami aniq belgilangan funksiyani qo‘llaydi. CONV va FC qatlamdagi parametrlar gradiyentli o‘tish usuli yordamida o‘rganishni amalga oshirishadi, shuning uchun konvolyutsion neyrotarmoq orqali sinfni aniqlash amali har bir tasvirga mos ravishda o‘rganish uchun belgilangan ko‘rsatkichlarga bog‘liq bo‘lib qoladi.

Natijada:

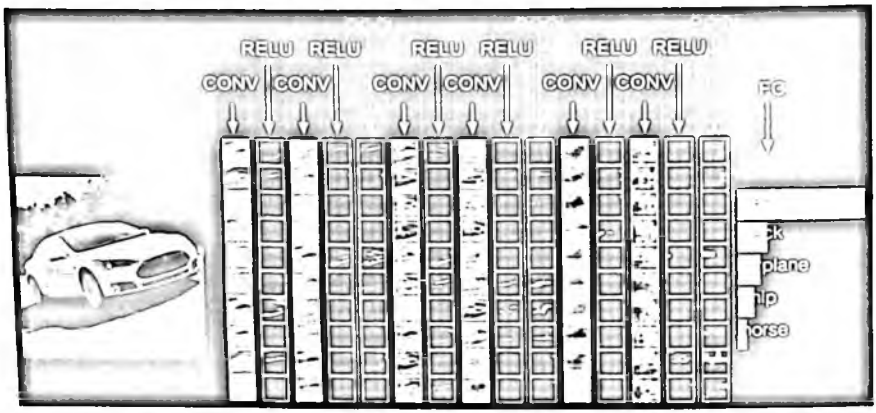
1. Eng sodda hollar uchun CNN arxitekturasi — qatlamlar jamlanmasi bo‘lib, berilgan obrazni chiquvchi (masalan, sinfni aniqlash ko‘rinishidagi) obrazga o‘girish.

2. Har bir qatlam tasvirni qayta ishlash jarayonining ma’lum bir bosqichi uchun javob beradi (konvolyutsiya qatlami, chiziqli taqsimlash, puling va to‘liq bog‘langan qatlam, bular barchasi hozirda keng tarqalgan).

3. Har bir qatlam kirish qismida 3D hajmli ma’lumotni qabul qiladi va 3D hajmni saqlagan holda zichlash funksiyasi orqali o‘zgartiradi.

4. Qatlam parametrlarga ega bo'lmasligi mumkin (CONV va FC da mavjud, RELU va POOLda esa yo'q).

5. Qatlam giperparametrlarga ega bo'lmasligi mumkin (masalan, CONV, FC va POOL da mavjud, RELU da esa yo'q).



1.14-rasm. CNN qatlamlari asosida tasvirni qayta ishlash.

1.14-rasmda ko'rsatilgandek, boshlang'ich hajm qayta ishlanmagan piksellarga ega (chapda), so'nggi hajmda esa aniqlangan sinf (o'ngda). Har bir faollashgan hajm tasvirni qayta ishlash jarayonida ustun holatda ko'rsatilgan. 3D hajmni vizuallashtirish mushkul bo'lganda har bir qatlam hajm ko'rinishi qator ko'rinishga keltiriladi. So'nggi qatlamning hajmi har bir sinfning ehtimollik ko'rsatkichini belgilaydi, FC sinflarni saralangan ketma-ketlikda vizuallashtiradi. Bu yerda izohlangan arxitektura qisqartirilgan VGG (Visual Geometry Group) tarmoqni ifodalaydi.

Endi har bir qatlamni birma-bir ko'rib chiqamiz, ular o'rtasidagi bog'lanishni, giperparametrlik xususiyatlarini o'rganib boramiz.

1.8.3. Konvolyutsiya qatlami

Konvolyutsiya qatlami – CNN qatlam asosini yaratuvchi qatlam bo‘lib, asosiy vazifani amalga oshiradi.

«Aqliy» xususiyatlarni inobatga olmagan holda ishning mantig‘i va izohi.

Faraz qiling, CONV qatlam «aqliy» yoki neyron yondashuvsiz ishlaydi. Konvolyutsiya qatlamining parametrlari o‘rganish filtrlari to‘plamidan iborat. Har bir filtr o‘ziga xos kichik o‘lchamli (eni va bo‘yiga) fazoga ega, biroq kiruvchi to‘liq hajm bo‘yicha o‘tadi. Masalan: konvolyutsion neyron tarmoqning birinchi qatlam standart filtri [5x5x3] o‘lchamga teng bo‘lishi mumkin. Oldinga qarab harakatlanganda kiruvchi ma’lumotlarni filtrning eni va bo‘yiga qarab o‘tkazamiz va filtr yozuvi bo‘yicha skalyar ko‘paytmasini kiruvchi ma’lumot bo‘yicha amalga oshiramiz. Tasvirning eniga va bo‘yiga qarab filtr o‘tishi davomida 2-o‘lchovli faollashish xaritasini tuzamiz va uning asosida fazoning xohlagan qismida akslanishni aniqlaydi va keyinchalik ifodalaydi. Ma’lum bir vizual ko‘rsatkich aniqlanganda faollashadigan filtrlarni tarmoq o‘rganib chiqadi. Bu birinchi qatlamda ma’lum bir o‘tish chegarasi, aniq rang to‘plami bo‘lishi mumkin. Tarmoqning yuqori bosqichlarida esa aylanma naqshlar ko‘rinishida bo‘lishi mumkin. Keyingi jarayonlarni har bir konvolyutsiya qatlamida filtrlar to‘plami bilan ish olib boriladi, ular esa o‘z navbatida alohida 2-o‘lchovli faollashtirish xaritasini shakllantiradi. Ushbu faollashtirish xaritasini chuqurlik bo‘yicha yig‘ib borib chiquvchi hajmni shakllantirish mumkin.

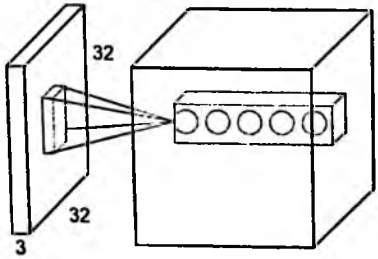
1.8.4. Lokal bog‘lanish

Katta hajmdagi ma’lumotlarni kiritish holati borasida so‘z borganda neyronlar va boshlang‘ich hajmga ega neyronlar o‘rtasida bog‘lanishni o‘rnatish maqsadga muvofiq bo‘lmaydi. Uning o‘rniga har

bir neyronni kiruvchi hajmning ma'lum bir lokal sohasiga bog'lash qulay hisoblanadi. Ushbu fazoviy bog'lanish giperparametr hisoblanib, retseptiv (sezuvchanlik) maydoni deb nomlanadi. Sezuvchanlik maydoni filtrlash maydoniga teng bo'lib qoladi. Chuqurlik o'qi bo'yicha davomiylik ko'rsatkichi boshlang'ich chuqurlik ko'rsatkichiga doim ekvivalent hisoblanadi. Fazoviy o'lchamlar (eni va bo'yi)ni ko'rib chiqishda asimmetriya mavjudligi, chuqurlik ko'rsatkichi esa: eni va bo'yicha lokal bo'lib, biroq to'liq chuqurlik ko'rsatkichi bo'yicha o'tishini yana bir marta izohlab o'tamiz.

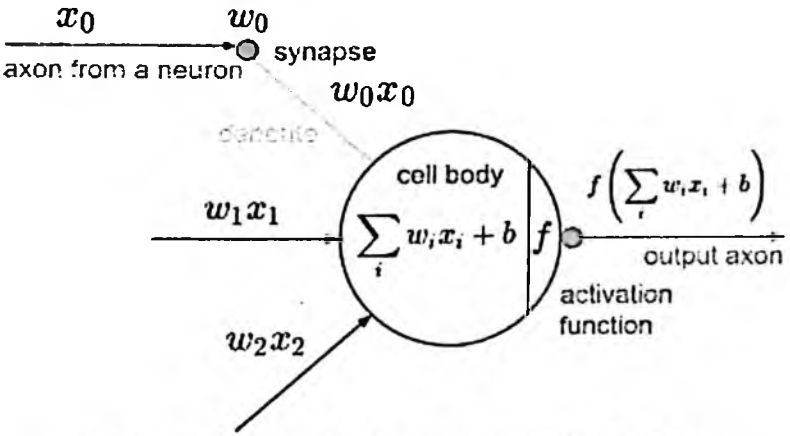
1-misol. Faraz qilaylik, kirish qismiga rasm [32×32×3] o'lchamga teng bo'lsin (masalan, RGB-tasvir CIFAR-10). Agar sezuvchanlik maydoni (filtr o'lchami) 5×5 ga teng bo'lsa, konvolyutsiya qatlamidagi har bir neyron kiruvchi hajmda (5x5x3) oraliqda og'irlikka teng bo'ladi, bu natijada 5*5*3=75 (+1 siljish parametri)ga teng bo'ladi. Shuni ta'kidlash kerakki, chuqurlik o'qi bo'yicha davomiylik 3 ga teng bo'lishi kerak, shu holdagina matematik bog'lanish to'g'ri belgilangan hisoblanadi.

2-misol. Faraz qilaylik, kirish qismiga rasm (16×16×20) o'lchamga teng bo'lsin. U holda sezuvchanlik maydoni 3×3 ga teng deb qabul qilinib, konvolyutsiya qatlamidagi har bir neyron umumiy holda kirish qismida 180 (3*3*20) bog'lanishga teng bo'ladi. Demak, bunda lokal bog'lanish eni va bo'yicha (bu yerda— 3×3) kenglikda bo'lib, biroq to'liq chuqurlik (20) bo'yicha o'tadi.



1.15-rasm. Konvolyutsiya qatlami neyronlarining lokal sathli kirish qismi (kiruvchi tasvir qismi) bilan bog'lanishi

1.15-rasmda: misol kiruvchi hajmli tasvir (qizil to‘rtburchak, parametrlari (32×32×3), CIFAR-10 tasviri) va konvolyutsion neyron tarmoqning birinchi qatlamidagi neyronlarning taxminiy hajmi. Konvolyutsion qatlamidagi har bir neyron faqat kirish lokal qismi bilan bog‘langan bo‘lib, fazoviy davomiylik eni va bo‘yi ko‘rsatkichi bilan belgilanadi, biroq neyron to‘liq chuqurlik bo‘yicha, ya’ni barcha ranglar kanalini qamrab oladi. E’tibor berish lozimki, bunda bir necha neyronlar (aynan shu misolda 5 ta), bitta sohani qamrab oluvchi to‘liq chuqurlikdan o‘tishadi (ustunlar chuqurligini quyida ko‘rib o‘tiladi).



1.16-rasm. Neyronlarning matematik ko‘rinishi.

1.16-rasmda: oddiy neyrotarmoqlar sinfiga taalluqli neyronlar o‘zgarmasligicha qoladi: ular odatdagiday og‘irlik ko‘rsatkichi va chiziqli bo‘lmagan ma’lumotlar o‘rtasida skalyar ko‘paytmasini hisoblaydi, biroq ularning bog‘lanishi lokal sathlar bilan cheklab qo‘yiladi.

1.8.5. NumPy misollar

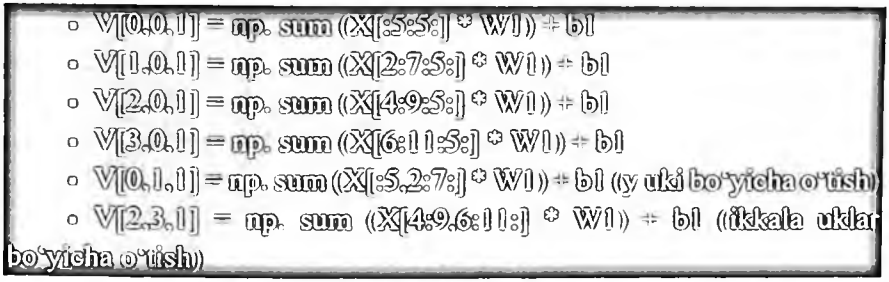
Yuqorida keltirilgan ma'lumotlarni aniqlashtirish uchun aniq misolni kod ko'rinishda ko'rib chiqamiz. Faraz qilaylik, barcha kiruvchi ma'lumotlar— bu NumPy-massiv X. Bunda:

- Chuqurlik ustun ko'rsatkichi (x, y) nuqtasida faollashuvi esa X(x,y:).
- D chuqurlikdagi faollashuv kartasiga ekvivalent bo'lgan chuqurlik qirqimi faollashuvi X(d) bo'ladi.

Faraz qilsak, kiruvchi hajm X (bu NumPy-massiv) shakli X.shape: (11,11,4) ko'rinishda bo'lsin. Keyingi qadam, faraz qilsak, nollar bilan to'ldirish qo'llanilmasin, (P=0), filtr kattaligi F=5 ga, qadam esa S=2 ga teng bo'lsin. Demak, chiquvchi hajmning fazoviy o'lchami (11−5)/2+1 = 4 ga teng bo'ladi, ya'ni eni va bo'yini to'rtga baravar deb o'rnatiladi. Chiquvchi hajmda faollashuv xaritasi (V deb nomlasak) quyidagi ko'rinishda bo'ladi (ushbu misolda bir necha elementlar hisoblanadi):

```
V[0,0,0] = np.sum(X[:5,:5,:] * W0) + b0
V[1,0,0] = np.sum(X[2:7,:5,:] * W0) + b0
V[2,0,0] = np.sum(X[4:9,:5,:] * W0) + b0
V[3,0,0] = np.sum(X[6:11,:5,:] * W0) + b0
```

Eslatib o'tamizki, NumPy da * amali massivlarning elementlar bo'yicha ko'paytmasini bildiradi. Bunda, og'irlik vektori W0 ushbu neyron og'irlik vektori hisoblanadi, b0 esa — bu siljish ko'rsatkichi bo'lib, W0 qiymati W0.shape: (5,5,4) shakl ko'rsatkichidan kelib chiqadi, chunonchi, filtr o'lchami 5 ga teng, kiruvchi hajm chuqurlik ko'rsatkichi 4 ga teng. Har bir nuqtada oddiy neyron turlardagidek skalyar ko'paytmalar olib boriladi. Shu bilan birga (parametrlarni hamkorlikda qo'llash hisobidan) bitta og'irlik va siljish qo'llaniladi, eni bo'yicha o'lchash qadami 2 ga teng bo'ladi. Ikkinchi faollashtiruvchi xaritani chiqish hajmida ko'rish uchun quyidagi amal bajariladi:



Ikkinchi faollashuv xaritasini hisoblashda boshqa parametrlar (W1) qo‘llanilishini inobatga olgan holda bu yerda ikkinchi o‘lchamlar V chuqurlik asosida indeksatsiya olib boriladi. Yuqorida keltirilgan misolda qisqartirilgan holda V massivning qolgan qismlarini to‘ldirish uchun konvolyutsiya qatlami bajaruvchi ba’zi amallariga to‘xtab o‘tmadik. Eslatib o‘tamizki, faollashuv xaritasi faollashishi funksiyasi yordamida elementlar bo‘yicha birma-bir harakatlanishadi, biroq bu funksiya bu yerda keltirilmagan.

1.8.6. Konvolyutsiya qatlami bo‘yicha amallar ketma-ketligi

Konvolyutsiya qatlami bo‘yicha xulosa keltiramiz:

- **W1xH1xD1** o‘lchamdagi ma’lumotlarni qabul qiladi
- 4 giperparametrni talab etadi:

1. **K** filtrlar soni,
2. **F** fazoviy tarqalish,
3. **S** qadami,
4. nollar bilan to‘ldirish sonli ifodasi **P**.

- O‘lcham **W2xH2xD2**, hajmini ko‘radi, bunda:

1. **W2=(W1−F+2P)/S+1**
2. **H2=(H1−F+2P)/S+1** (eni va bo‘yi bir xil hisoblanadi)
3. **D2=K**

▪ Parametrlarni hamkorlikda qo'llash usuli yordamida **F·F·D1** og'irliklarni filtrlarga uzatish bajarilib va umumiy holda og'irliklarni **(F·F·D1)·K** sonda va **K** siljishda beradi.

▪ Chiqish hajmida **d**-chuqurlik kesimi (**W2xH2 o'lchamda**) bu natija bo'lib, bu kirish hajmida **S qadamga ega d**-siljish asosida konvolyutsiya **d**- filtrlanish natijasidir.

Giperparametrlarning keng tarqalgan qiymatlari: **F=3, S=1, P=1** ga teng. Biroq umumiy tartib va empirik qoidalar mavjud bo'lib, ular asosida giperparametrlarning aniq qiymatlari belgilanadi.

1.9. Demo-konvolyutsiya tarmog'i

Quyida ishlaydigan konvolyusion – qatlam demoversiyasi keltiriladi. Uch o'lchovli ma'lumotlarni vizuallashtirish mushkul bo'lganligi sababli, barcha hajmlar (kiruvchi hajm (kuk), hajmlar og'irlik ko'rsatkichi (qizil), chiquvchi hajm (yashil)) har biri satrda joylashgan chuqurlik qirqimi asosida vizuallashtiriladi. Kiruvchi hajm o'lchami **W1=5, H1=5, D1=3** teng bo'lsin, konvolyutsiya qatlamining parametrlari— **K=2, F=3, S=2, P=1** ga teng. Ya'ni 3×3 o'lchamga teng bo'lgan 2 filtrga ega bo'lib, u 2 qiymatga teng qadam bilan ishlaydi. Chunonchi, chiquvchi hajm fazoviy o'lchami (5−3+2)/2+1 = 3 ga teng bo'ladi. Bundan tashqari, nollar bilan to'ldirish ko'rsatkichi **P=1** ga teng bo'lgan holda kiruvchi hajmga qo'llab, tashqi chegaralarini nolga o'zgartiradi. Quyida keltirilgan vizualizatsiya chiquvchi faollashuv qiymatlarini birma-bir tekshirib chiqadi (yashil) va har bir element aynan chiquvchi ma'lumotlar (kuk) elementlarini o'zaro ko'paytirish va filtrlash (qizil) orqali hisoblanishini, hamda qiymatlarni ketma-ket summalash va natijalarni siljish bo'yicha korrektirovka qilishini ifodalaydi.

1.9.1. Matritsalarni ko‘paytirish ko‘rinishida qo‘llash

Konvolyutsiya amali asosan kiruvchi ma’lumotlar lokal sohalari bilan filtr ko‘rsatkichlari o‘rtasida skalyar ko‘paytmani bajaradi. Konvolyutsiya qatlamini qo‘llashning umumiy prinsipi yuqorida keltirilgan xususiyatni bajargan holda konvolyusiya qatlamining o‘tishi natijasida yagona katta matritsa ko‘rinishga keltirish hisoblanadi:

1. Kiruvchi tasvirning lokal qismlari jarayon davomida ustunlar ko‘rinishida cho‘ziladi va ular odatda **im2col deb nomlanadi**. Masalan: kirish qismiga [227×227×3] o‘lchamdagi tasvir qabul qilinsa, 4-qadam asosida filtrlar yordamida [11×11×3] o‘lchamgacha qisqartirilishi mumkin, so‘ngra [11×11×3] o‘lchamdagi piksellar blokini vektor ko‘rinishda alohida vektor-ustun ko‘rinishda 11*11*3 = 363 o‘lchamda cho‘zish mumkin bo‘ladi. Ushbu jarayonning takrorlanishi eni va bo‘yiga nisbatan (227−11)/4+1 = 55 ta o‘rin berib, X_col matritsani [363×3025] o‘lchamda beradi, bunda har bir ustun cho‘zilgan maydonga ega bo‘ladi (umumiy holda 55*55 = 3 025 maydonga ega bo‘ladi).

2. Konvolyutsiya qatlamining og‘irlik ko‘rsatkichi mos ravishda satr bo‘ylab cho‘ziladi. Masalan: agar [11×11×3] o‘lchamga ega 96 filtr mavjud bo‘lsa, [96×363] o‘lchamli W_row matritsasini beradi.

3. Konvolyutsiya jarayonining natijasi bitta katta ko‘paytirish matritsasining np.dot (W_row, X_col) amalga oshirilishi bilan ekvivalent bo‘lib kelmoqda, bunda har bir filtr va maydon o‘rtasida skalyar ko‘paytma hisoblanadi. Ko‘rilayotgan misolda ushbu amal natijasi [96×3025] ko‘rinishda bo‘ladi, chiqish qismida har bir lokal qism uchun filtr ko‘paytmasi hosil qilinadi.

4. Natija esa [55×55×96] o‘lchamga nisbatan boshlang‘ich ko‘rinishga keltirilishi lozim.

Bunday yondashuv, sezilarli kamchilikka ega bo‘lib, X_col matritsasi ba’zi qiymatlarining ko‘p marotaba takrorlanishi sababli

xotiradan samarali foydalanish imkonini bermaydi. Boshqa tomondan, asosiy ustunligi matritsalarning samarali ko'paytmalari to'plamidan maqsadga muvofiq ravishda qo'llash mumkin. im2col ish prinsipi g'oyasini pulingni ishga tushirishda qo'llash mumkin.

1.9.2. Teskari bog'lanish orqali xatolikni tekshirish usuli

Konvolyutsiya amali uchun teskari qaytish jarayoni (berilganlar uchun ham, og'irliklar uchun ham) bu o'sha oddiy konvolyutsiya amali bo'lib, fazoviy to'ntarilgan filtrlarga ega bo'ladi.

1x1 o'lchamda konvolyutsiya

Eng avval shuni aytish lozimki, ko'pgina tajribalarda (masalan, **Network in Network**) 1×1 formatdagi konvolyutsiya usuli qo'llanilgan. Bu ko'rinishdagi konvolyutsiya usulini xususan ikkilamchi signallarda qo'llash umuman noqulay hisoblanadi, chunki, odatda signallar 2 o'lchamli bo'lib, 1×1 o'lchamdagi konvolyutsiya ahamiyatsiz bo'lib qoladi (bu oddiy nuqtali masshtablash bo'lib qoladi). Biroq CNNda bu o'zgacha bajariladi, chunki 3 o'lchovli ma'lumotlar bilan ishlab, filtr to'liq fazoviy chuqurlik bo'yicha o'tib filtrlanadi. Masalan, kirish qismida [32×32×3] o'lchamdagi tasvir qabul qilinsa, 1×1 o'lchamdagi konvolyutsiya amali bajarilsa, u holda samarali ravishda 3 o'lchovli skalyar ko'paytma bajariladi (chunki kirish qismi 3 kanaldan iborat).

1.9.3. Kengaytirilgan konvolyutsiya

Yangi tadqiqotlar **(Fisher Yu. va Vladlen Koltun maqolasida)** natijasida konvolyutsiya qatlamining yana bitta giperparametri—dilatatsiya (kengaytirish) aniqlandi. Yuqorida konvolyutsiya qatlamining ketma-ket joylashgan filtrlari ishini ko'rib chiqdik, biroq shunday filtrlar mavjudki, har bir yacheyka oralab (sakrab) ishlaydi. Bu oraliqlar kengaytmalar deb nomlanadi. Misol: 3 o'lchamga teng w

filtrning bitta o'lchamiga nisbatan **x** kirish qismida w[0]*x[0] + w[1]*x[1] + w[2]*x[2] amali bajarilsin. Bu 0 dan dilatatsiya ekanligini bildiradi. 1 dilatatsiyasi uchun quyidagi hisoblashlar olib borilishi lozim: w[0]*x[0] + w[1]*x[2] + w[2]*x[4]. Boshqacha qilib aytganda, 1 ga teng oraliq mavjud bo'lib qoladi. 0-kengaytmali filtrlar bilan muvofiqlashgan qo'llanilishi ba'zi hollar uchun samarali hisoblanadi, chunki kiruvchi fazoviy berilganlarni kam qatlamlar asosida birlashtirish imkonini beradi. Masalan: har bir filtr ustki qismida 2 ta 3x3 o'lchamli konvolyutsiya qatlami o'rnatilsa, 2-qatlam neyronlari 5x5 o'lchamdagi kirish qismi funksiyasi (ya'ni 5x5 neyronlar samaralik maydoni) bo'lib qoladi. Agar kengaytirilgan konvolyutsiya qatlami qo'llanilsa, samaralik maydoni tezroq kengayib boradi.

1.10. Puling qatlam

CNN arxitekturasida odatda konvolyutsiya qatlamlari ketma-ketligida oralariga puling (quyi tanlash) qatlamni o'rnatish oddiy holat bo'lib hisoblanadi. Uning vazifasi tasvir fazoviy o'lchamini bosqichma-bosqich kichraytirish bo'lib, tarmoqda parametrlar va hisoblashlar sonini kamaytirishi hamda qayta o'rganish jarayonini nazorat qilish imkonini berish hisoblanadi. Puling qatlam kiruvchi berilganlarning fazoviy chuqurligi ko'rsatkichidan bog'liqsiz holda maksimum funksiyasini qo'llagan holda tasvirning fazoviy masshtablash amalini bajaradi. Ko'pincha 2x2 o'lchamga ega va qadami 2 ga teng filtrli qatlam qo'llaniladi; bunday qatlam kiruvchi berilganlarning fazoviy chuqurligini har bir qism uchun ham eniga, ham bo'yiga diskretizatsiya bo'yicha 2 barobar qisqartiradi, bunda 75% ga xalos etiladi. Har bir MAX amali 4 ta qiymatdan eng maksimalini tanlaydi. Bunda fazoviy chuqurlik o'lchami o'zgarmas qoladi. Umuman olganda, puling qatlami quyidagicha ishlaydi:

- **W1xH1xD1** o'lchamga ega berilganlarni kirish qismida qabul qiladi.

- 2 ta giperparametrni talab etadi:

1. **F** uning davomiyligini,
2. **S** qadamini.

- Boshlang'ich parametrlarni kiritadi, chunki aniq belgilangan boshlang'ich funksiyani hisoblaydi.

Quyi tanlashlarda nollar bilan to'ldirish qo'llanilmaydi. Amaliyotda puling qatlamining faqat 2 ta variatsiyasini **F=3, S=2** giperparametrlari bilan (umumlashtiruvchi quyi tanlash deb nomlanadi) va eng keng tarqalgan qatlam **F=2, S=2** giperparametrlarga ega ko'rinishlari. Katta hajmdagi maydonga ega pulinglar destruktiv hisoblanadi.

1.10.1. Umumiy puling

Maksimal quyi tanlash holati uchun qo'shimcha sifatida shuni aytish lozimki, puling qatlamlar boshqa funksiyalarni ham bajarishi mumkin, masalan, quyi tanlashlarning o'rtacha ko'rsatkichliligi yoki L2-normallashgan tanlovni hosil qilish. Ehtimollik jihatdan o'rtalashgan tanlash ko'pincha qo'llanilib kelingan, biroq oxirgi vaqtlarda qo'llanilishi qulay bo'lgan maksimizatsiyalangan tanlovga nisbatan ikkilamchi bo'lib qoldi.

Teskari bog'lanishda xatolikni aniqlash usulida.

Konvolyutsiya qatlamini o'rganish davomida max (x, y) amali uchun teskari bog'lanishda xatolikni aniqlash jarayoni kirish qismi uchun gradiyent sifatida qabul qilinadi, qaysiki to'g'ri bog'lanishda kengroq qiymatlarga ega bo'lgan. Binobarin, to'g'ri bog'lanishda puling qatlamida maksimal faollashish indeksini aniqlash jarayoni bajarilishi lozim, chunki teskari bog'lanishda gradiyentlash yo'li samarali belgilanishi kerak bo'ladi.

1.10.2. Pulingdan xolis bo‘lish

Ba’zi mutaxassislar puling amalini ortiqcha hisoblab, qo‘llamaydi. Masalan: D. T. Springenberg, A. Dosoviskiy, T. Broks va M. Ridmiller o‘zlarining ishlarida puling qatlamidan xolis bo‘lishni taklif etib, uning o‘rniga arxitekturada konvolyutsiya qatlamlarni takroran qo‘llash usulini taklif etgankar. Tasvir ifodalanishi o‘lchamini qisqartirish uchun katta hajmdagi konvolyutsiya qatlamini qo‘llashni tavsiya qildi. Subdiskretizatsiyani rad etish generativ modellarda o‘rganish jarayoni uchun variatsion avtosotsiator (VAE) yoki raqobatdoshli generativ tarmoq (GAN) ko‘rinishda asosiy rolni o‘ynaydi. Yaqin orada konvolyutsion neyron tarmoq arxitekturasi puling qatlamidan hosil bo‘ladi yoki juda kam sonda qo‘llanila boshlanadi.

1.11. To‘liq bog‘langan qatlam

Oddiy neyrotarmoqlar kabi to‘liq bog‘langan qatlamda neyronlar oldingi qatlam barcha faol nuqtalari bilan to‘liq bog‘lanishga ega. Ularning faollashuv ko‘rsatkichlari matritsalararo siljish jarayonida matritsalarni ko‘paytirish orqali aniqlanadi.

1.11.1. To‘liq bog‘langan qatlamlarni konvolyusion qatlamga o‘zgartirish

To‘liq bog‘langan qatlamlar va konvolyutsion qatlam o‘rtasidagi farq shundan iboratki, konvolyutsiya qatlami neyronlari kirish qismi lokal sohalari bilan bog‘langan va ular hamkorlikda parametrlardan foydalanishi mumkin. Biroq ikki qatlamda ham neyronlar o‘z xususiyatlari bilan farqlansa-da, skalyar ko‘paytmani hisoblaydi va shuning uchun ularning funksional ko‘rinishi aynan bir xil. Bundan tashqari, bu ikki ko‘rinishdagi qatlamlar negizida konvertatsiyalash mumkin:

- Barcha ko‘rinishda konvolyutsion qatlam uchun uning vazifasini bajarishi mumkin bo‘lgan to‘liq bog‘langan qatlam mavjud. Og‘irliklar matritsasi katta hajmdagi matritsa ko‘rinishida bo‘lib, ba’zi bloklardan tashqari (lokal bog‘langanlik sababli) u asosan 0 lar bilan to‘ldirilgan, ko‘pgina og‘irlik ko‘rsatkichlari (parametrlarni hamkorlikda qo‘llanilishi uchun) teng bo‘ladi.
- Va aksincha, barcha ko‘rinishda to‘liq bog‘langan qatlam konvolyutsion qatlamga o‘zgartirilishi mumkin. Masalan, to‘liq bog‘langan qatlam chuqurligi **K=4 096** teng bo‘lib, [7x7x512] o‘lchamga teng bo‘lgan kirish berilganlariga yo‘naltirilgan bo‘lib, uni quyidagi parametrlarga ega konvolyutsion qatlamga o‘girish mumkin: filtr o‘lchami **F=7**, nollar bilan to‘ldirish mavjud emas (**P=0**), qadami **S=1** va filtr chuqurligi **K=4 096** ga teng. Boshqacha qilib aytganda, filtr o‘lchami kiruvchi berilganlar o‘lchami bilan bir xil o‘rnatiladi. Chiqish qismida [1x1x4096] o‘lchamni olamiz, chunki chuqurlikning binar ustuni barcha kirish berilganlari bo‘yicha «to‘g‘ri» tartibda o‘tadi, shuning uchun, boshlang‘ich to‘liq bog‘langan qatlam beradigan natijani beradi.

1.11.2. Qayta o‘zgartirish roli

To‘liq bog‘langan qatlamlarni konvolyutsion qatlamga o‘zgartirish imkoni amalda keng qo‘llanilmoqda. Kirish qismida [224×224×3] o‘lchamga ega tasvirni qabul qilib, uning hajmini faollashish hajmiga [7x7x512]gacha qisqartirish uchun bir qator konvolyutsiya va puling qatlamlarni qo‘llovchi konvolyutsion neyron tarmoqni ko‘rib chiqamiz. AlexNet arxitekturasida bu amal 5 ta puling qatlamini qo‘llash orqali amalga oshirilib, unda kiruvchi fazoviy tasvir diskretizatsiyasi qisqartiriladi va natijada tasvir hajmi 224/2/2/2/2/2 = 7 ga teng bo‘lib qoladi. AlexNet arxitekturasi 2 ta 4 096 hajmga ega to‘liq bog‘langan qatlamni va klasslarni hisoblash uchun 1000 neyronga ega 1 ta to‘liq

bog‘langan qatlamni qo‘llaydi. Izohlangan algoritm bo‘yicha 3 ta qatlamdan barchasini yig‘uvchi qatlamga o‘zgartirishimiz mumkin:

1. [7x7x512] o‘lchamdagi kirish qismiga yo‘naltirilgan to‘liq bog‘langan birinchi qatlamni yig‘uvchi qatlam bilan o‘zgartirishimiz mumkin, bunda uning filtr o‘lchami **F=7** ga teng bo‘lib, chiquvchi berilganlar hajmi [1x1x4096] ko‘rinishda bo‘ladi.

2. Ikkinchi to‘liq bog‘langan qatlamning filtr o‘lchami **F=1** ga teng bo‘lgan yig‘uvchi qatlam bilan o‘zgartirish mumkin, bunda chiquvchi berilganlar hajmi [1x1x4096] ko‘rinishda bo‘ladi.

3. Uchinchi to‘liq bog‘langan qatlamning filtr o‘lchami **F=1** ga teng bo‘lgan yig‘uvchi qatlam bilan o‘zgartirish mumkin, bunda chiquvchi berilganlar hajmi [1x1x1000] ko‘rinishda bo‘ladi.

Ushbu har bir qayta o‘zgartirishlar natijasida konvolyutsiya qatlami filtrida og‘irlik ko‘rsatkichlari matritsasini **W** o‘zgartirishi (masalan, ko‘rinishini o‘zgartirish) mumkin. Demak, bu o‘zgartirishlar asosida bir martaga “o‘tish”da CNNni katta hajmdagi tasvir ko‘pgina sohalari bo‘yicha o‘tkazish mumkin.

Masalan: tasvir o‘lchami 224×224 bo‘lib, uning hajmi [7x7x512], ya’ni 32 ga qisqartirildi. U holda 384×384 o‘lchamdagi tasvirni ushbu o‘zgartirilgan arxitektura bo‘yicha o‘tkazilsa, unga ekvivalent bo‘lgan [12×12×512] hajmdagi tasvirni beradi, chunki 384/32 = 12. Natijada bitta [1x1x1000] hajmga ega baholash vektori o‘rniga, to‘liq 6×6 baholash massivi sinfiga ega bo‘lib, 384×384 to‘liq hajmi bo‘yicha o‘tkaziladi.

Konvolyutsiya neyrotarmog‘i 224x224 o‘lchamda 384×384 tasvir bo‘yicha 32 piksel qadam bilan o‘tishining natijasi va birlamchi konvertatsiyalangan CNN o‘tish bilan bir xil natija beradi.

Tabiiyki, qayta ishlangan neyron tarmoqning bir martalik o‘tishi, original CNNning 36 pozitsiya bo‘yicha bir necha marotaba iteratsiyalanishiga nisbatan samarali hisoblanadi. Bu usul odatda unumdorlikni ko‘tarish uchun amalda qo‘llaniladi. Masalan, tasvir o‘lchamini kattalashtirish lozim bo‘lganda konvertatsiyalangan

konvolyutsion neyrotarmoq qo‘llaniladi va barcha tasvir fazoviy qismlari baholanadi va u o‘rtacha ko‘rsatgichga keltiriladi.

CNN qadamini 32 pikseldan past holatda boshlang‘ich ko‘rinishda tasvirga joriy etish uchun to‘g‘ri o‘tish qiymatlarini qayta ko‘paytirish orqali amalga oshirish mumkin. Masalan: agar 16 pikselli qadamni qo‘llash lozim bo‘lsa, o‘zgartirilgan konvolyutsiya neyrotarmog‘i bo‘yicha olingan natijalarni ikki marotaba birlashtirish lozim bo‘ladi: eng avval boshlang‘ich tasvir o‘tkaziladi, so‘ngra tasvir eni va bo‘yi bo‘yicha 16 pikselga surilgan holatda ikkinchi marotaba o‘tkaziladi.

1.12.CNN arxitekturasi

Ko‘rib o‘tganimizdek, konvolyutsion neyrotarmoqlar odatda 3 xil asosiy qatlamdan iborat, bularga: konvolyutsiya, puling va to‘liq bog‘langan qatlamlar kiradi. Faollashtirish funksiyasi ham qatlam ko‘rinishda joriy etilib, ular chiziqsiz funksiya ko‘rinishda har bir element asosida olib boriladi. Quyida ushbu qatlamlar o‘zaro hamkorlikda to‘liq CNN hosil qilishini ko‘rib chiqamiz.

1.12.1. Qatlamlar ko‘rinishi

Ko‘pincha CNN arxitektuasini qurishda bir necha qatlamlar ketma-ketligi qo‘llanilib, konvolyutsiya, rektifikatsiya qatlamlari bir necha marta qo‘yilib, ulardan so‘ng puling qatlami qo‘yiladi; ushbu shablon boshlang‘ich tasvir juda kichik o‘lchamga yetkazilguncha qo‘llaniladi. Ma’lum bir vaqtdan so‘ng to‘liq bog‘langan qatlamlarga o‘tish amalga oshirilib, so‘nggi to‘liq bog‘langan qatlam chiquvchi ma’lumotga, masalan, sinfini aniqlash ma’lumotiga ega bo‘ladi. Boshqacha qilib aytganda, eng keng tarqalgan CNN arxitekturasi quyidagi sxema ko‘rinishida bo‘ladi:

INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC

Bunda «*» takrorlanishini bildiradi, POOL? esa qo‘shimcha puling qatlamini ko‘rsatadi. Bundan tashqari, N >= 0, (odatda, N <= 3), M >= 0, K >= 0 (odatda, K < 3) teng bo‘ladi. Quyida keltirilgan namuna bo‘yicha CNN arxitekturasi keltirilgan:

- INPUT -> FC, chiziqli klassifikatorni joriy etadi. Bunda N = M = K = 0.
- INPUT -> CONV -> RELU -> FC
- INPUT -> [CONV -> RELU -> POOL]*2 -> FC -> RELU -> FC. Bunda har bir puling qatlami orasida konvolyutsiya qatlami joylashgan.
- INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL]*3 -> [FC -> RELU]*2 -> FC. Bunda 2 konvolyutsiya qatlami har bir puling qatlami orasida yotadi.

Yirik va chuqur tarmoqlarda bu arxitekturani qo‘llash samarali natija beradi, chunki bir nechta konvolyutsion qatlamlarning bir necha marta qo‘llanilishi kirish qismidagi ma’lumotlarning to‘liq xususiyatlarini ochib berishga va so‘ngra ularni puling qatlamida qo‘llash imkonini beradi.

Odatda, yirik konvolyutsiya maydonini qo‘llashdan ko‘ra, bir nechta katta bo‘lmagan filtrlash qatlamlaridan foydalanish samarali hisoblanadi. Faraz qiling, qatlamlarda chiziqsiz o‘tishni inobatga olib, 3×3 o‘lchamdagi konvolyutsiya qatlamini birma-bir qo‘yamiz. Bunda har bir neyron birinchi konvolyutsiya qatlami kirish qismida 3×3 o‘lchamga ega bo‘ladi. Ikkinchi qatlamda neyronlar birinchi qatlam bo‘yicha 3×3 o‘lchamiga va kirish qismining 5×5 o‘lchamiga ega bo‘ladi. Xuddi shunday uchinchi qatlamda neyronlar 3×3 o‘lcham ikkinchi qatlam bo‘yicha va kirish qismining 7×7 o‘lchamiga ega

bo‘ladi. Faraz qiling, ushbu uchta konvolyutsiya qatlami o‘rniga bitta 7×7 o‘lchamli retseptiv maydonga ega bitta CONV qatlam qo‘llaniladi. Bu qatlam neyronlari kirish qismi (7×7) kenglikda retseptiv maydoniga ega bo‘ladi, biroq bir necha kamchiliklar mavjud.

Birinchidan, 3 ta konvolyutsiya qatlamidan iborat stek qatlam xususiyatlarini ifodalovchi chiziqsiz funksiyaga ega bo‘la turib, kirish qismining chiziqli funksiyasini hisoblaydi.

Ikkinchidan, barcha kirish qismi berilganlari to‘laligicha S ta kanallarga ega. Unda bitta konvolyutsiya qatlami o‘lchami 7×7 bo‘lib, **Cx esa (7x7xC)=49C²** parametrlariga ega bo‘lib, 3 qatlamdan iborat stek **3x** faqat **(Cx (3x3xC))=27C²** parametrga ega bo‘ladi. Ma’lumki, kichik filtrga ega CONV qatlamlarning bir nechtasini «qo‘llash» usuli kam parametrlar yordamida kiruvchi ma’lumotlar xususiyatlarining kuchli tomonlarini ifodalashga imkon beradi. Bu yerda bitta kamchilik bu oraliq davrda olinadigan natijalarning barchasini saqlash uchun katta hajmdagi xotiraning talab etilishi hisoblanadi.

Amaliyotda esa ImageNet ma’lumotlar bazasi bilan ishlovchi usul qo‘llaniladi. Mutaxassislar kerakli arxitekturani tanlashda 90% saralanib chetlashtiriladi, chunki aynan ImageNet bilan ishlovchi arxitekturani tanlash, o‘rganish jarayonidan o‘tgan modelni yuklash va qo‘yilgan masalani moslashtirib, sozlashning o‘zi kifoya. Konvolyutsiya neyrotarmog‘ini noldan boshlab o‘rganish mutaxassislardan talab etilmaydi.

1.12.2. Aniq misollar

Konvolyutsiya neyrotarmoqlarning eng keng tarqalgan arxitekturasini ko‘rib chiqamiz:

- **LeNet**. eng birinchi bo‘lib, Yan Lekun 1990-yillarda konvolyutsiya neyrotarmog‘ini samarali qo‘llab, arxitektura ishlab chiqishga erishdi. **LeNet arxitekturasi** pochta indekslari, raqamlarni aniqlash va o‘qib olishda qo‘llanilgan.

▪ **AlexNet**. Bu arxitektura Aleks Krijevskiy, Ilya Suskever va Djeff Xinton ishlari natijasida erishilgan bo'lib, CNNning kompyuter videokuzatuv sohasida mashhur bo'lishiga olib keldi. **AlexNet** arxitekturasi 2012-yil **ImageNet ILSVRC Challenge** da taqdim etilib, barcha raqobatlarda yutib chiqdi (ikkinchi o'rinni egallagan arxitekturada 26% xatolikka nisbatan 16% xatolik bilan yutdi).

▪ **ZF Net**. 2013-yil ILSVRC yutgan Metyu Zeller va Rob Fergyuslaming konvolyutsiya neyron tarmog'i bo'lib, **ZF Net** (abbreviaturasi Zeiler va Fergus) nom bilan mashhur. Ushbu arxitektura AlexNet versiyasining takomillashtirilgan holati bo'lib, konvolyutsiya tarmog'i o'rtacha o'lchami oshirilib, birini qatlam filtrining qadami kamaytirildi.

▪ **GoogLeNet**. 2014-yili yuqorida ko'rsatilgan tanlovda yutib chiqqan **Shegedi** Google korporatsiyasining boshqa xodimlari tomonidan ishlab chiqilgan CNNdir. Bu arxitekturaning yutug'i shundan iboratki, kirish qismi modulini (Inception Module) ishlab chiqish va joriy etish bo'lib, parametrlar sonini 60 mln. dan 4 mln. gacha qisqartirish imkonini berdi. Parametrlarning qisqartirilishi tarmoq yuqori qismidagi to'liq bog'langan qatlamlarni o'rta puling qatlamlari bilan almashtirish hisobidan amalga oshirildi.

▪ **VGGNet**. GoogLeNet dan keyin tez orada ILSVRC 2014 tanlovida **Karen Simonyan va Endryu Sisserman**_tarmog'i e'lon qilindi. Mutaxassislar uning imkoniyatlarini namoyish etib, asosiy unumdorlik omili bu tarmoqning chuqurlik ko'rsatkichidir. Bu tarmoq 16 ta konvolyutsiya va to'liq bog'langan qatlamga ega bo'lib, bir jinsli arxitektura 3×3 o'lchamda konvolyutsiya va 2×2 puling amalini boshidan oxirigacha bajaradi. Boshlang'ich model Caffe freymvorkda chuqur o'rganish uchun Plug and Play rejimida ishlaydi.

▪ **ResNet. Qoldiqli tarmoq (Residual Network)** Kayming Xe tomonidan ishlab chiqilgan bo'lib, ILSVRC tanlovida g'olib chiqdi. Asosiy xususiyatlari — paketli normallashtirishni intensiv va maxsus skip-bog'lanishlar qo'llaniladi. Arxitektura so'ngida to'liq bog'langan

qatlamlar mavjud emas. Hozirgi kunda ResNet konvolyutsion neyrotarmoqlar sohasida keng qo'llaniladigan va eng yuqori yutuq hisoblanadi.

1.12.3. VGGNet izohi

VGGNet to'liq izohini ko'rib o'tamiz. Tarmoq to'laligicha konvolyutsiya qatlamlaridan iborat bo'lib, 3×3 o'lchamda 1 qadam bilan konvolyutsiya amalini bajaradi, 1 bilan to'ldiradi. Shu bilan birga puling qatlami 2×2 o'lchamda maksimal tanlovni 2 qadam bilan amalga oshiradi va nollar bilan to'ldirmaydi. Har bir qadamda o'lcham o'zgarishini ifodalash va og'irlik ko'rsatkichi soni o'zgarishini kuzatish mumkin:

INPUT: [224×224×3] xotira: 224*224*3=150K og'irlik ko'rsatkichi: 0

CONV3-64: [224×224×64] xotira: 224*224*64=3.2M og'irlik ko'rsatkichi: (3*3*3)*64 = 1,728

CONV3-64: [224×224×64] xotira: 224*224*64=3.2M og'irlik ko'rsatkichi: (3*3*64)*64 = 36,864

POOL2: [112×112×64] xotira: 112*112*64=800K og'irlik ko'rsatkichi: 0

CONV3-128: [112×112×128] xotira: 112*112*128=1.6M og'irlik ko'rsatkichi: (3*3*64)*128 = 73,728

CONV3-128: [112×112×128] xotira: 112*112*128=1.6M og'irlik ko'rsatkichi: (3*3*128)*128 = 147,456

POOL2: [56×56×128] xotira: 56*56*128=400K og'irlik ko'rsatkichi: 0

CONV3-256: [56×56×256] xotira: 56*56*256=800K og'irlik ko'rsatkichi: (3*3*128)*256 = 294,912

CONV3-256: [56×56×256] xotira: 56*56*256=800K og'irlik ko'rsatkichi: (3*3*256)*256 = 589,824

CONV3-256: [56×56×256] xotira: 56*56*256=800K og'irlik ko'rsatkichi: (3*3*256)*256 = 589,824

```
POOL2: [28×28×256] xotira: 28*28*256=200K og'irlik ko'rsatkichi: 0
CONV3-512: [28×28×512] xotira: 28*28*512=400K og'irlik ko'rsatkichi: (3*3*256)*512 = 1,179,648
CONV3-512: [28×28×512] xotira: 28*28*512=400K og'irlik ko'rsatkichi: (3*3*512)*512 = 2,359,296
CONV3-512: [28×28×512] xotira: 28*28*512=400K og'irlik ko'rsatkichi: (3*3*512)*512 = 2,359,296
POOL2: [14×14×512] xotira: 14*14*512=100K og'irlik ko'rsatkichi: 0

CONV3-512: [14×14×512] xotira: 14*14*512=100K og'irlik ko'rsatkichi: (3*3*512)*512 = 2,359,296
CONV3-512: [14×14×512] xotira: 14*14*512=100K og'irlik ko'rsatkichi: (3*3*512)*512 = 2,359,296
CONV3-512: [14×14×512] xotira: 14*14*512=100K og'irlik ko'rsatkichi: (3*3*512)*512 = 2,359,296
POOL2: [7×7×512] xotira: 7*7*512=25K og'irlik ko'rsatkichi: 0
FC: [1×1×4096] xotira: 4096 og'irlik ko'rsatkichi: 7*7*512*4096 = 102,760,448
FC: [1×1×4096] xotira: 4096 og'irlik ko'rsatkichi: 4096*4096 = 16,777,216
FC: [1×1×1000] xotira: 1000 og'irlik ko'rsatkichi: 4096*1000 = 4,096,000
TOTAL xotira: 24M * 4 bytes ≅ 93MB / image (only forward! ~*2 for bwd)
TOTAL parametrlar: 138M parametrlar
```

Shunga e'tibor berish lozimki, xotiraning ko'p qismi eng birinchi konvolyutsiya qatlamlarida qo'llaniladi, parametrlarning ko'pchiligi esa to'liq bog'langan qatlamlarda oxirgilarida joylashgan. Ushbu aniq

misolda to‘liq bog‘langan qatlamda 100 ta og‘irlik ko‘rsatkichi mavjud, chunonchi, uning umumiy soni 140 mln. dan iborat.

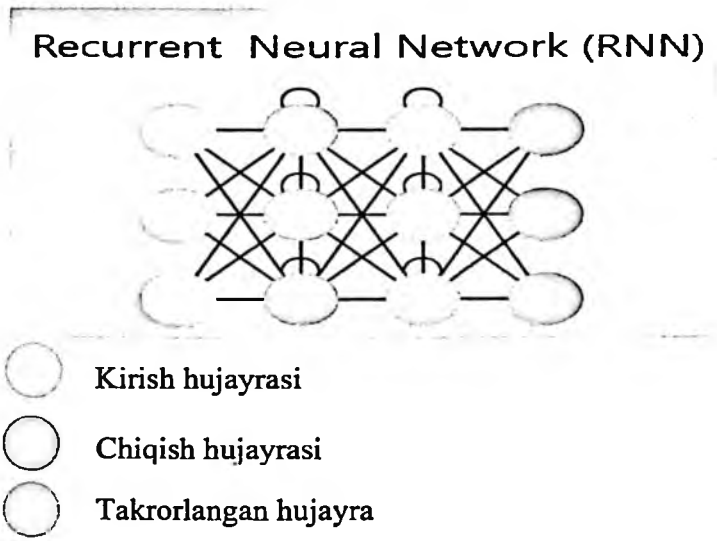
1.12.4. Hisoblashlar bo‘yicha tavsiyalar

CNN qurishda asosiy to‘siq bu xotira. Zamonaviy grafik protsessorlar xotira limiti 3, 4 yoki 6 Gb, biroq eng zamonaviy GPU 12 Gb ga ega. Uchta asosiy xotira talab etiladi:

- Ma’lum bir oraliqda berilganlar hajmi. Bu har bir CNN qatlamida qayta ishlanmagan faollashish signallari va ularning gradiyentlari. Odatda, faollashish signallari konvolyutsiya neyrotarmoqning birinchi qatlamlarida joylashgan bo‘ladi. Shuni aytish joizki, faollashish signallari teskari bog‘lanishda xatolikni aniqlashda qo‘llaniladi. Biroq unumli foydalanishda CNN faqatgina tadqiqot jarayonida qo‘llaniladi va natijada faollashuvlar soni faqat joriy holatni saqlash hisobidan qisqarishi mumkin. Bunda oldingi holatlar pastki qatlamlarga birin-ketin suriladi.
- Parametrlar o‘lchami. Bu tarmoq parametrlari soni, ularning teskari bog‘lanishda xatolikni aniqlashdagi gradiyenti va kesh qadamidan iborat. Demak, parametrlar vektorini saqlash uchun talab etiladigan xotira taxminan 3 ga teng bo‘lgan koeffitsientga ko‘paytirilishi lozim.
- Barcha konvolyutsiya neyrotarmog‘ining qo‘llanilishida **ko‘p jinsli** xotiradan foydalanish lozim, ya’ni tasvir berilganlari paketi ko‘rinishda bo‘lib, yanada takomillashish imkoni bilan qo‘llash.

Qiymatlar (faollashuv, gradiyent va h.k.)ning umumiy soni aniq bo‘lgandan so‘ng uni Gb larga o‘tkazdirish kerak bo‘ladi. Qiymatlar soni 4 ga ko‘paytirilsa baytlar soni kelib chiqadi, so‘ngra 1 024 ga bir necha marta bo‘lish orqali xotirani kilobayt, so‘ng megabayt va gigabaytda olasiz. Agar qurilgan tarmoq “mos kelmasa”, uni paket o‘lchamini qisqartirish orqali «moslashtirish» mumkin, chunki xotiraning ko‘p qismi aynan faollashuv orqali egallanadi.

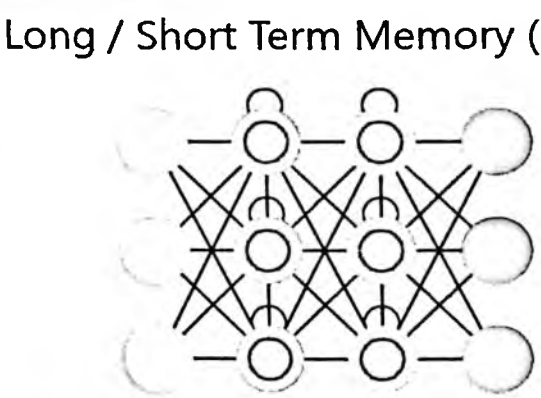
1.13. Rekurrent neyron tarmoqlar sinfi



1.17-rasm. Rekurrent neyron

Rekurrent neyron tarmoq (recurrent neural network, RNN) FFNN tarmoq turkumidagi tarmoq bo‘lib, biroq RNN neyronlar ma’lumotlarni faqat oldingi qatlamdan olish bilan cheklanmaydi, balki to‘liq rekurrent tarmoqdan qabul qilib oladi. Ya’ni, kelayotgan barcha ma’lumotlar ketma-ketligini xotirada saqlashni o‘rganadi. RNN murakkabligi — gradiyentning pasayib borib yo‘qolishi bo‘lib, vaqt o‘tishi bilan tarmoq saqlayotgan ma’lumotlarini yo‘qota boshlaydi. Garchi neyron holatiga ta’sir qilmasdan, faqat og‘irlik ko‘rsatkichini aniqlashda ta’sir qilsa-da, ketma-ketliklar haqidagi ma’lumotlar aynan ularda saqlanadi. Rekurrent neyron tarmoqlar asosan ma’lumotlarni avtomatik to‘ldirishda qo‘llaniladi.

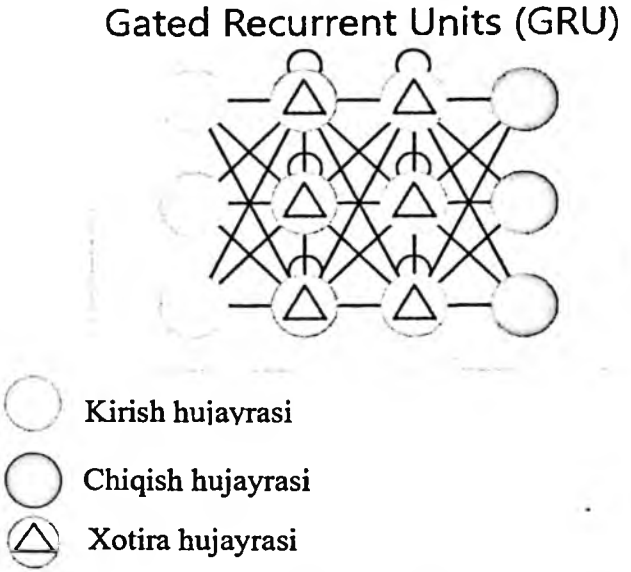
Uzoq va qisqa muddatli xotira



1.18-rasm. Uzoq va qisqa muddatli xotira.

Uzoq va qisqa muddatli xotira (long short term memory, LSTM) tarmog'i rekurrent neyron tarmoqlardagi ma'lumot yo'qotish muammosini hal etadi, bunda filtrlarni va mavjud xotira yacheykasini qo'llaydi. Har bir neyron xotira yacheykasiga va uchta filtrga ega: kiruvchi, chiquvchi va unutuvchi. Filtr maqsadi—ma'lumotlarni himoyalash. Kiruvchi filtr oldingi qatlamdan saqlash lozim bo'lgan ma'lumotlar hajmini belgilaydi. Chiquvchi filtr esa, keyingi qatlam qabul qiluvchi ma'lumotlar hajmini belgilaydi. Unutuvchi filtr xotirada qiymatlarni saqlash darajasini nazorat etadi, masalan, agar tarmoq biron-bir kitobni o'rganayotgan jarayonida yangi bo'limga o'tsa, oldingi bo'limdagi ba'zi bir matnlarni unutadi. LSTM tarmog'i murakkab strukturalarni yaratishga qodir, biroq katta hajmdagi resurslarni talab etadi.

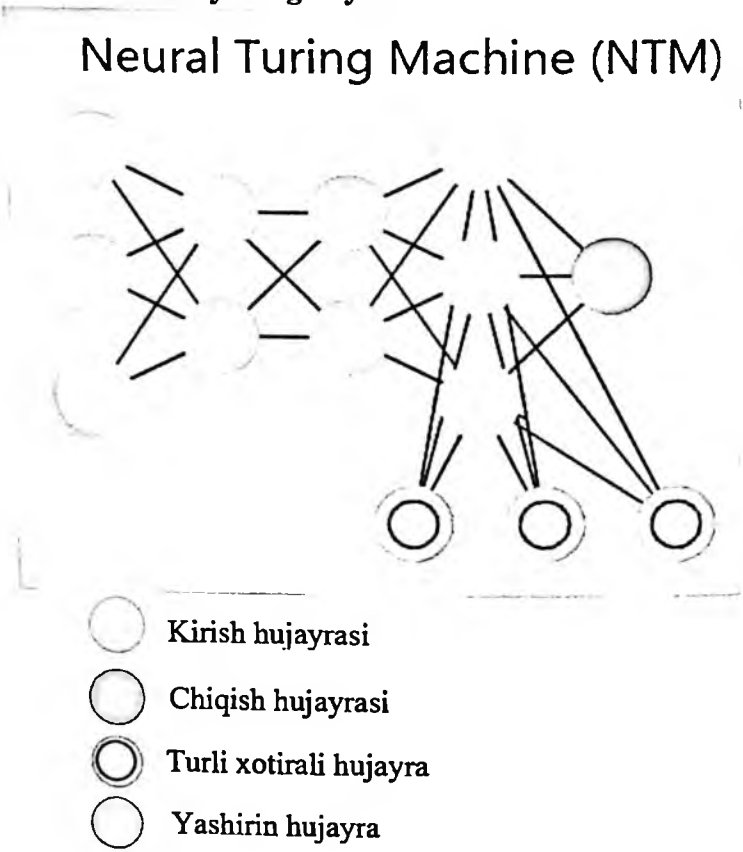
Boshqariladigan rekurrent bloki



1.19-rasm. Boshqariladigan rekurrent blok

Boshqariladigan rekurrent blok (gated recurrent unit, GRU) – bu rekurrent neyron tarmoqlarning mantiqiy mexanizmi bo'lib, uzoq va qisqa muddatli xotira tarmog'ining bir ko'rinishi hisoblanadi. GRU bir filtrga kam bo'lib, bog'lanish ko'rinishi ham o'zgacha: kirish, chiqish va unutish filtri o'rniga bitta yangilash filtri qo'llaniladi. Bu filtr orqali oldingi holatdan olinadigan ma'lumotni va oldingi qatlamdan olinadigan ma'lumotlarni tahlil qiladi. Holatni tashlash filtri unutish filtri kabi ishlaydi, biroq joylashuvi boshqacha. Keyingi qatlamga holat haqida to'liq ma'lumot uzatiladi, ya'ni chiqish filtri bu yerda mavjud emas. GRU va LSTM o'rtasidagi farq shundaki, GRU tezroq va sodda ishlaydi, biroq uning imkoniyatlari cheklangan.

1.14. Tyuring neyron mashinasi



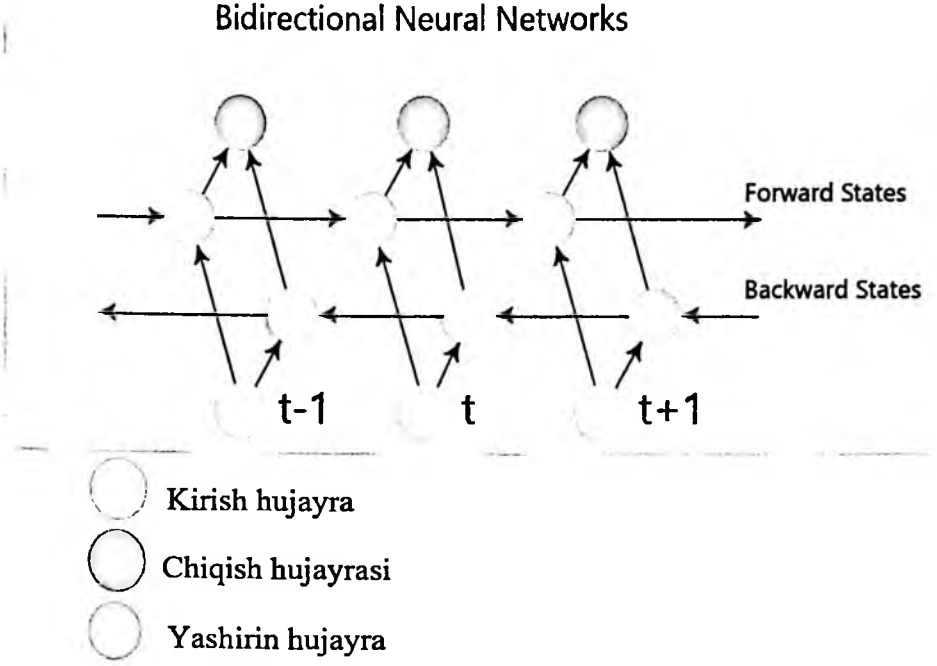
1.20-rasm. Tyuring neyron

Tyuring neyron mashinasi (neural Turing machine, NTM) uzoq va qisqa muddatli xotiraga ega abstrakt tarmoq sifatida qaraladi. Shu bilan birga, neyron tarmoq ichida kechadigan jarayonni izohlaydigan tarmoq deb qabul qilinadi. Xotira yacheykasi neyrondan alohida joylashgan. Ya'ni Tyuring mashinasidagidek hisoblash yacheykasi xotira yacheykasidan ajratilgan (Tyuring mashinasi), biroq tizimda hisoblashlar belgilangan tartibda bajariladi. Demak, NTM tarmog'ini aniq dasturlash orqali emas, balki teskari bog'lanishda

xatolikni aniqlash usulida o‘rganish mumkin. Bu o‘z navbatida ma’lumotlarni saqlash tizimini va neyron tarmog‘i imkoniyatlarini birlashtirishga olib keladi, aynan shu sababli bu tarmoq Tyuring mashinasi deb nomlanadi: ya’ni ma’lumotlarni o‘qish, yozib olish va o‘qib olgan ma’lumot turiga qarab o‘z holatini o‘zgartirishi mumkin va uni to‘liq Tyuring deb nomlanadi.

1.15. Ikki yo‘nalishli rekurrent neyrotarmoq

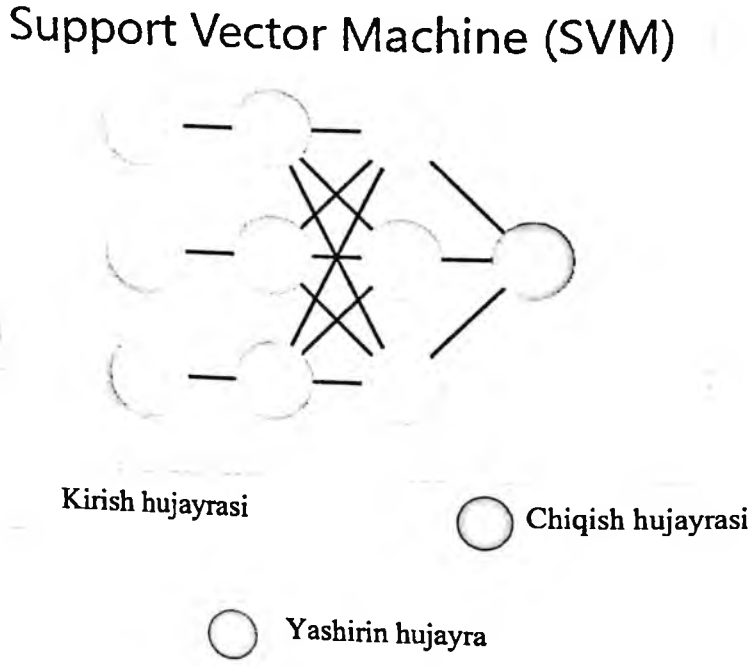
Ikki yo‘nalishli rekurrent neyrotarmoq, uzoq va qisqa muddatli xotiraga va ikki yo‘nalishli boshqaruvchi rekurrent blokka ega ikki yo‘nalishli tarmoq



1.12-rasm. Ikki yo‘nalishli rekurrent neyrotarmoq (BRNN).

Ikki yo‘nalishli rekurrent neyrotarmoq (BRNN) uzoq va qisqa muddatli xotiraga (BLSTM) va ikki yo‘nalishli boshqaruvchi rekurrent blokka (BGRU) ega ikki yo‘nalishli tarmoq sxemada ko‘rsatilmagan. Ular o‘zining bir yo‘nalishli tarmoqda nusxasini takrorlaydi, biroq ular qo‘llaydigan ma’lumotlar faqat oldingi qatlamdan emas, balki keyingi qatlamdan ham olinadi. Masalan: LSTM kabi bir yo‘nalishi tarmoq «mushuk» so‘zini aniqlashda harflar bir yo‘nalishda berib bajarilsa, ikki yo‘nalishda keyingi harfni ham berib aniqlashadi. Shu kabi tarmoqlar tasvirlarni kengaytirish va bo‘shliqlarni to‘ldirish holatlarida qo‘llash uchun mo‘ljallangan.

1.16. Tayanch vektorlar mashinasi



1.22-rasm. Tayanch vektorlar mashinasi.

Tayanch vektorlar mashinasi (support vector machine, SVM) o'qituvchi bilan o'rganish asosida ishlovchi algoritmlari tasnifi oilasi hisoblanadi. SVM optimallashtirish bilan bog'liq masalalarning optimal yechimini topadi. Tayanch vektorlar mashinasining klassik versiyasi chiziqli taqsimlanadigan ma'lumotlarni kategoriya bo'yicha ajratish imkoni mavjud. Masalan, Leopold va Matroskin obrazlarida mushuklar o'rtasidagi farqni aniqlab berishi mumkin. O'rganish jarayonida tarmoq ma'lumotlarni ikki o'lchovli muhitga o'tkazadi va ularni maksimal aniqlikda to'g'ri chiziq bilan ajratib sinflarga bo'linadi, bunda chiziqning har bir tomoniga bir sinfga mansub ma'lumotlar joylashishi va shu ikki tomonda joylashgan eng yaqin nuqtalar o'rtasida masofa maksimal bo'lishiga erishiladi. Ushbu masofa bo'shliq deb, nuqtalar esa tayanch vektorlar deb nomlanadi. To'g'ri chiziqni belgilashda bo'shliq maksimallashtiriladi, bu esa sinflarga optimal bo'lishni ta'minlaydi. Shu bilan birga SVM n-o'lchovli ma'lumotlarni ajratish imkonini beradi. Shuni ta'kidlash lozimki, tayanch vektorlar mashinasini har doim ham neyron tarmoq sifatida qaralmaydi.

1.17. Tayanch vektorlar mashinasi asosida ma'lumotlarni tasniflash

Support Vector Machine, SVM. Bunday tasnif keng ko'lamda qo'llaniladi. Tasnifning eng boshlang'ich masalasi kamida ikkita sinfdan bittasiga mansubligini aniqlash hisoblanadi. Odatda, bu obyekt R bo'shliqda n o'lchovli vektor hisoblanadi. Vektor koordinatalari obyektning alohida atributlarini izohlaydi. Masalan, berilgan RGB modelida, ***c rangi***, uch o'lchovli muhitda vektor hisoblanadi: ***c***=(*red, green, blue*).

Agar sinflar faqat ikkita bo'lsa binar tasnif deb aytiladi. Agar sinflar bir nechta bo'lsa— ko'p sinflilik (multisinfli) tasnif deb nomlanadi. Shu bilan birga har bir sinf obrazi – obyektlar mavjud bo'lishi mumkin bo'lib, ularning qaysi sinfga mansubligi oldindan

ma'lum bo'ladi. Bunday ko'rinishdagi masalalarni o'qituvchi yordamida o'rganish deb, oldindan aniq bo'lgan ma'lumotlar esa o'rganiladigan to'plam deb yuritiladi. (Eslatma: agar sinflar avvalidan berilmagan bo'lsa, klasterlash masalasi hisoblanadi.)

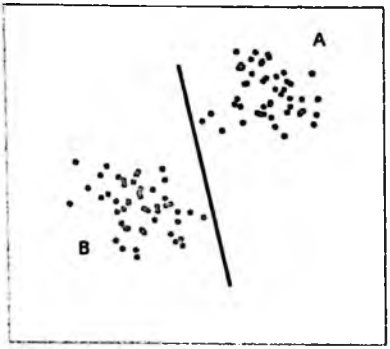
Demak, o'qituvchi yordamida o'rganishga misol ko'ramiz, tasnif masalasining matematik izohi quyidagicha: masalan, $\mathbf{X}$ —obyektlar fazosi bo'lsin (masalan, $\mathbb{R}^n$), Y — esa sinflar (masalan, $Y = \{-1,1\}$). O'rganiladigan to'plam esa: $(x_1, y_1), \ldots, (x_m, y_m)$. $F : X \to Y$ (klassifikator) funksiyasini tuzish masalasi qo'yilgan bo'lib, x tasodifiy obyektni y sinf bilan taqqoslash kerak bo'lsin.

1.17.1. Tayanch vektorlar mashinasi

Multitasnif masalasi sifatida ishlashi mumkin bo'lsa-da, ushbu usul binar klassifikator usuliga mansub.

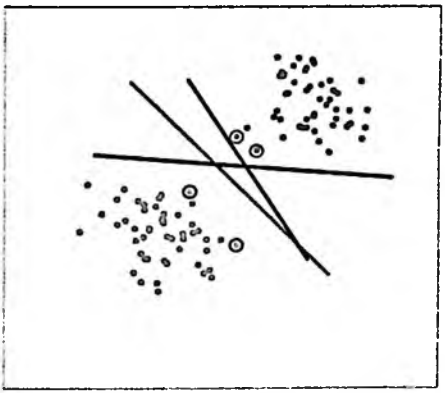
Usul g'oyasini quyidagi misol sifatida ko'rsatish mumkin: tekislikda nuqtalar berilgan bo'lib, ular ikki sinfga bo'lingan, rasmda ushbu ikki sinfni ajratadigan chiziq o'tkazilgan (qizil chiziq). Keyinchalik barcha yangi nuqtalar avtomatik ravishda quyidagi usulda tasniflanadi:

- chiziqdan yuqori nuqta A sinfga mansub;
- chiziqdan past nuqta B sinfga mansub.

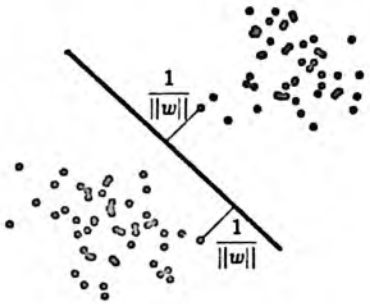


1.23-rasm. Tayanch vektorlar usuliga misol.

Ushbu to‘g‘ri chiziq ajratuvchi chiziq deb nomlanadi. Biroq katta hajmdagi tekislikda to‘g‘ri chiziq asosida sinflarga ajratib bo‘lmaydi, chunki «chiziqdan past» yoki «chiziqdan yuqori» tushunchalari o‘z mohiyatini yo‘qotadi. Shuning uchun, to‘g‘ri o‘rniga fazoviy gipertekislik qo‘llanilib, uning o‘lchov ko‘rsatkichi boshlang‘ich fazoga nisbatan bir kam o‘lchamda bo‘ladi. Masalan: R^3 da gipertekislik sifatida oddiy ikki o‘lchovli tekislik hisoblanadi. Ko‘rib o‘tilayotgan misolda sinfga ajratish uchun bir necha chiziqlar mavjud.



1.24-rasm. Tasniflash uchun sinf nuqtalarining joylashuvi

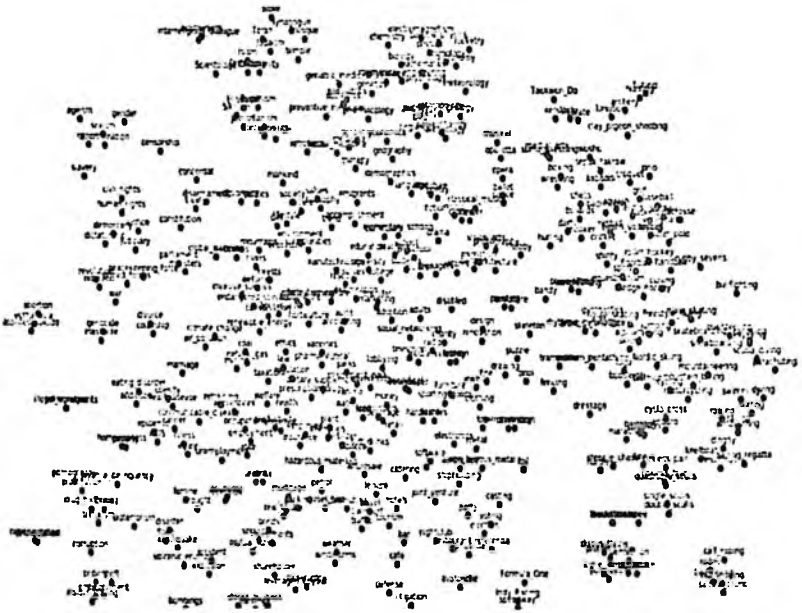


1.25-rasm. Vektorlar joylashuvi.

Tasnif nuqtayi nazaridan qaralganda to‘g‘ri chiziq shunday tanlanishi kerakki, chiziqdan har bir sinf nuqtalarigacha bo‘lgan masofa maksimal qiymatga ega bo‘lishi lozim. Boshqacha qilib aytganda, sinflarga aniq ajratuvchi chiziqni belgilash lozim. Bunday chiziq umuman olganda – optimal ajratuvchi gipertekislik deb aytiladi.

Ajratuvchi gipertekislikka yaqin joylashgan vektorlar esa tayanch vektorlar deb nomlanadi.

1.18. Word2Vec



1.26-rasm. So‘zlarning vektorli ko‘rinishi: barcha yechim kontekstda

So‘zlar vektori (word vectors) — so‘zlarning sonli ifodasi bo‘lib, ular o‘rtasida memantik bog‘lanishni saqlaydi. Masalan, cat (mushuk) vektori uchun eng yaqin so‘z dog (it). Biroq pencil (qalam) so‘zining vektorli ko‘rinishi cat vektoridan ancha farqlanadi. Bu asosan ikkita

so‘zning kontekstda birgalikda uchrashish ehtimolligi bilan aniqlanadi (ya’ni, [cat, dog] yoki [cat, pencil]). Quyidagi gap tarkibini ko‘rib o‘tamiz:

53-rasm. Word2Sec ga misol.

Bunda qaysi so‘zlar to‘g‘ri kelishi aniq (pencil to‘g‘ri kelmaydi). Aynan nima uchun to‘g‘ri kelmasligini aniqlash lozim bo‘ladi. Grammatika, talaffuz to‘g‘ri kelsa (til miqyosida), nima uchun to‘g‘ri kelmaydi? Barchasi aynan kontekstga borib taqaladi, pencil ma’no jihatdan to‘g‘ri kelmaydi. Bu misol bo‘yicha kontekst ahamiyatli ekanligini bildiradi. Word2vec algoritmi kontekstni qo‘llaydi va so‘zlarning sonli ifodasini shakllantiradi. Shuning uchun, bitta kontekstda birgalikda kelgan so‘zlar o‘xshash vektorlarga ega bo‘ladi.

1.18.1. Word2vec ni qo‘llash

Real loyihalarda Word2vec qo‘llanilishini tushunish uchun google scholar tizimiga kirib, NLP bilan bog‘liq ma’lumotlarni izlaymiz (masalan, savol-javob tizimlari, chat-botlar, mashinali tarjima va h.k.). 2013-yildan keyingi yillar, ya’ni word2vec paydo bo‘lgan muddatdan boshlab qo‘shilgan hujjatlar bo‘yicha filtrlaymiz va so‘zlarning vektorli ifodalanishi bo‘yicha juda ko‘p maqolalarni ko‘rish mumkin.

So‘zlarning vektorli ifodalanishi ko‘pgina sohalarda qo‘llaniladi:

- ma’lum bir tilni modellashtirishda;
- chat-botlar;
- mashinali tarjima;
- savol-javob tizimlari ...va boshqalar.

Barcha zamonaviy NLP ilovalari asosan word2vec algoritmlariga asoslangan. So‘zlar ifodalanishining mavjud

modellarini takomillashtirishni ko‘rib o‘tamiz. Ular yordamida semantik o‘xshash so‘zlarni bir-biriga yaqin vektorlarda ifodalash imkonini berib, ma’no jihatdan uzoq bo‘lgan so‘zlarni boshqacha ifodalaydi. Bu model xususiyati samarali natijaga olib keladi.

So‘zlar vektorini yaratish ketma-ketligi

Boshlang‘ich ko‘rsatkichga ega bo‘lmagan so‘zlar ketma-ketligini o‘rganish uchun bir necha masalalarni yechish lozim:

Berilganlarni bo‘laklarga ajratilgan shakllarni hosil qilish (kirish so‘zi, chiqish so‘zi), bunda har bir so‘z n uzunlikdagi ikkilamchi vektor ko‘rinishga ega bo‘ladi, bunda i- qiymat i- o‘rinda 1 bilan belgilansa, qolgan hollar uchun 0 bilan belgilanadi (one-hot kodlash);

kirish va chiqish qismlariga one-hot vektorlarni qabul qiluvchi modelni yaratish;

to‘g‘ri so‘zni bashorat qiluvchi yo‘qotishlar funksiyasini aniqlash, va modelni optimallashtirish;

o‘xshash so‘zlar o‘xshash vektorlarga ega ekanligiga amin bo‘lib, model sifatini aniqlash.

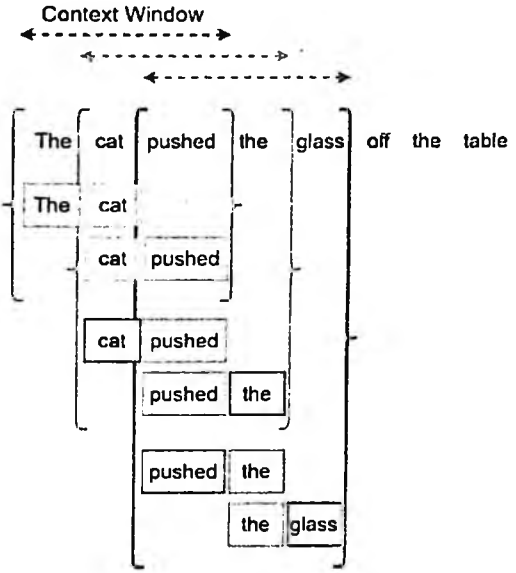
Boshlang‘ich matndan tartiblangan berilganlarni yaratish

Quyidagi misolni ko‘rib chiqamiz:

The cat pushed the glass off the table (mushuk stakanni stoldan tushirdi).

Kerakli berilganlar quyidagicha shakllantiriladi: rasmdagi har bir qavs birlamchi kontekst oynasi. Ko‘k rangli maydon kiruvchi one-hot vektor (maqsadli so‘z)ni ifodalaydi, qizil maydon-chiquvchi one-hot vektorni (maqsadli so‘zdan tashqari kontekst oynasidagi barcha so‘zlar, ya’ni kontekstli so‘z). Bitta kontekst oynasidan ikkita berilganlar elementi hosil qilinadi (ya’ni birta maqsadli so‘zga ikkita qo‘shni so‘z mos keladi). Oyna o‘lchami odatda foydalanuvchi tomonidan aniqlanadi. Kontekst oynasi qanchalik katta bo‘lsa, model ishi shuncha samaralidir, biroq algoritm bajarilish vaqtiga ta’sir

qiladi. Maqsadli so‘z bilan maqsadli berilganlarni adashtirmaslik lozim, ular umuman boshqa boshqa.



1.27-rasm. Embedding layer neyrotarmoqni aniqlash.

Bizning neyrotarmoq yuqorida keltirilgan kiruvchi berilganlar asosida o‘rganish jarayonini o‘tadi. Bizga quyidagilar lozim:

kiruvchi one-hot vektorlar to‘plami;

chiquvchi one-hot vektorlar to‘plami (o‘rganish jarayonidan so‘ng);

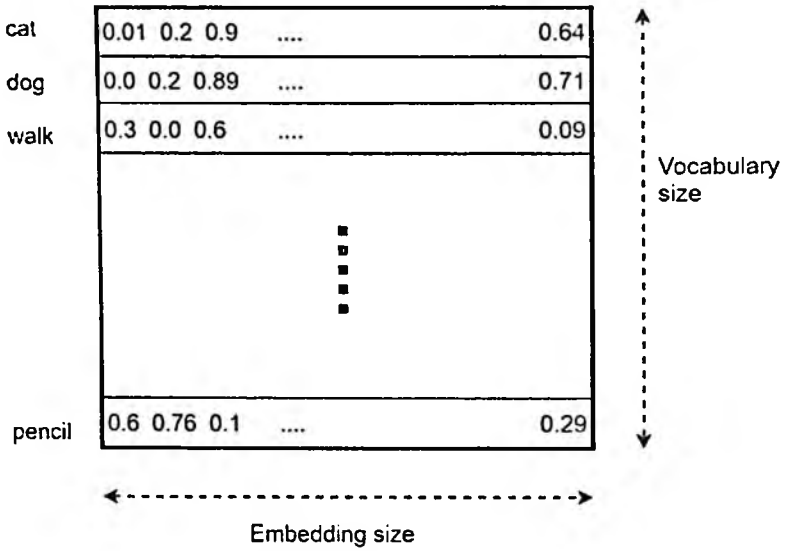
embedding layer;

neyrotarmoq.

So‘nggi ikkita qismni batafsil ko‘rib chiqamiz.

Embedding layer dan boshlaymiz. Lug‘atda mavjud barcha so‘zlar vektorlarini saqlaydi. Faraz qiling katta hajmdagi matritsa (lug‘atdagi so‘zlar soni x so‘zlarning ixchamlashgan vektorli ifodasining o‘lchamini ifodalaydi). Ushbu o‘lcham (Embedding size)

sozlanadigan parametr hisoblanadi. U qanchalik katta bo‘lsa, model shunchalik samarali bo‘ladi (biroq embedding size hajmi ma’lum bir chegaraga yetganda unumdorlik oshishi to‘xtaydi). Ushbu yirik matritsa (neyrotarmoq kabi) tasodifiy ravishda initsiallashtiriladi va optimallashtirish jarayonida bitlar bo‘yicha sozlanib boriladi. Quyidagi ko‘rinishga ega:



1.28-rasm. Embedding layer da yaratilgan so‘zlar matritsasi.

Neyron tarmoq

Modelning so‘nggi elementi bu neyron tarmoq. O‘rganish jarayonida neyrotarmoq kirish vektorini qabul qilib, barcha so‘zlar to‘plami bo‘yicha kontekstga mos tushadigan so‘zlar ketma-ketligini bashorat qilishga urinadi (shu bilan birga, buni quyidagicha ham ifodalash mumkin, ya’ni so‘zlar one-hot kodlanishining chiziqli kombinatsiyasi deb ham tushunish mumkin). So‘ngra yo‘qotish funksiyasi yordamida noto‘g‘ri tasnif uchun modelni “jazolash”

to‘g‘risi uchun “taqdirlash” mumkin. Hozirgi misolda bitta kirish va bitta chiqish berilganlarni bir martaga qayta ishlash bilan cheklanib qolamiz. Haqiqiy loyihalarda berilganlar batch (ya’ni, masalan, 64 ta elementdan iborat guruhlar) ko‘rinishda qayta ishlanadi. O‘rganish jarayonini umumiy ko‘rinishda izohlaymiz.

Kiritilgan so‘z (maqsadli so‘z)ga mos keluvchi vektorni embedding layer dan aniqlaymiz.

Ushbu vektorni neyrotarmoqqa “iste’molga” beramiz, so‘ngra to‘g‘ri keluvchi (kontekstga mos) so‘zni taklif etishga harakat qilamiz.

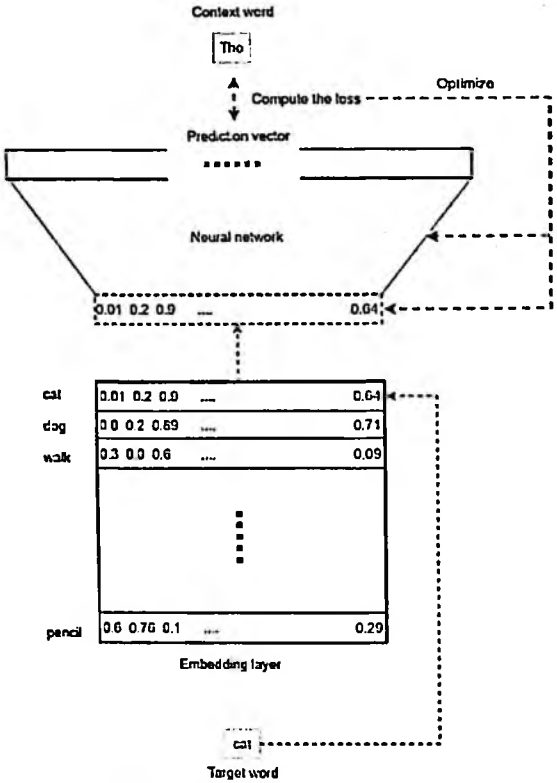
Taklif qilingan so‘z va kontekst oynasida mavjud so‘zni taqqoslab, yo‘qotish funksiyasini aniqlaymiz.

Yo‘qotish funksiyasini gradiyentli stoxastik pasayish bilan birga qo‘llab, neyrotarmoqni va embedding layer ni optimallashtiramiz.

Bashorat qilishda softmax funksiyasini qo‘llab, ehtimolliklar taqsimoti bo‘yicha bashoratni normallashtirishga olib kelinadi.

Barchasini birlashtiramiz.

Word2vec algoritmning barcha tashkil etuvchilarni bila turib, ularni birlashtirish imkoni paydo bo‘ladi. Model o‘rganish jarayonidan o‘tgandan so‘ng embedding layer ni diskka saqlash va semantikali vektorlardan to‘laqonli foydalanish mumkin bo‘ladi. Umumiy ko‘rinishi quyidagicha:



1.29-rasm. Skip-gram algoritmi.

Bu model skip-gram algoritmi deb nomlanib, word2vec ning bitta algoritmi hisoblanadi va aynan unga e'tibor ajratamiz. Boshqa bir algoritm "so'zlarning cheksiz to'plami "(continuous bag-of-words model, CBOW).

Yo'qotishlar funksiyasi modelini optimallashtirish

Kalit qismdan biri bu— yo'qotishlar funksiyasi bo'lib, standart kesishmali entropiya funksiyasi (softmax cross entropy loss) tasnif masalalarining samarali yechimidir. Biroq word2vec modeli uchun bu funksiya yetarlicha qulay emas, bu asosan sodda tahlil masalalarida, masalan, ikki ko'rinishda chiqish holatlari: musbat va manfiy bo'lgan

masalalarida samarali hisoblanadi. So‘zlar bilan ishlash masalalarida qaysiki milliardgacha so‘zlar bilan ishlashga to‘g‘ri kelsa, lug‘at matritsa hajmi 100000 gacha yoki undan ham oshib borishi mumkin, natijada softmax-normallashtirish murakkablashadi. Bunga sabab, softmax to‘liq hisoblashi uchun barcha chiqish tugunlari bo‘yicha yo‘qotish funksiyasini hisoblashiga to‘g‘ri keladi.

Shuning uchun, qo‘llashga ixcham bo‘lgan alternativa, ya’ni sampled softmax loss funksiyasini qo‘llaymiz. Standart kesishmali entropiyadan ancha farqlanadi.

Eng avval kesishmali entropiya funksiyasini hisoblaymiz, ya’ni maqsadli so‘zning mavjud kontekstli so‘zi bilan bashorat qilingan kontekstli so‘z va haqiqiy so‘zga mosligi bilan aniqlanadi. So‘ngra kross-entropiya yo‘qotish k ko‘rsatkichini, ya’ni mos kelmaydigan namunalar (maqsadli so‘z + kontekst oynadan tashqari bo‘lgan so‘z)ni aniqlash lozim bo‘ladi, bunda ma’lumotlardagi shovqinlik taqsimoti qo‘llaniladi.

Yo‘qotish funksiyasi quyidagicha aniqlanadi:

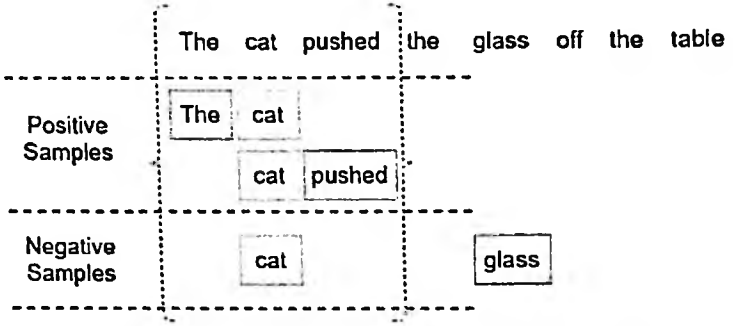
$$Loss = SigmoidCrossEntropy(Prediction, Correct\ Word) + \sum_{1}^{K} E_{Noise\ ID} SigmoidCrossEntropy(Prediction, Noise\ ID)$$

SigmoidCrossEntropy bu xatolik ko‘rsatkichi bo‘lib, boshqalarga bog‘liqsiz holda bitta chiqish tuguni bo‘yicha aniqlash mumkin. Bu yechim lug‘at hajmi oshgan hollar uchun juda samarali hisoblanadi. Bu funksiya qanday amalga oshirilishini bilish shart emas, chunki TensorFlow da bu funksiya sozlangan, biroq k qanday parametr ekanligini anglash lozim. Eng muhimi —sampled softmax loss ikki xil obyekt bilan ishlaganda xatolik sifatida nimani aniqlashini anglash hisoblanadi:

bashorat qilingan vektorda to‘g‘ri tanlangan kontekstli so‘z indeksi (kontekst oynasidagi so‘z indeksi);

k shovqinli so‘zlar indeksi.

Buni misolda ko‘rib o‘tamiz. Bunda k = 1 (cat + glass):



1.30-rasm. So‘zlar kontekstini bashorat qilish sxemasi.

TensorFlow da: skip-gram algoritmining amalga oshirilishi

Bu qismda barcha qismlarni birlashtirib, algoritmning amalga oshirilishini ko‘rib o‘tamiz. Quyidagilarni ko‘rib chiqamiz:

Berilganlar generatori;

(TensorFlow)da skip-gram modeli;

Skip-gram algoritmni ishga tushirish.

Berilganlarni generatsiyalash

Kodlarni chuqur o‘rganib chiqmaymiz, chunki berilganlarni generatsiyalash ichki mexanizmini ko‘rib chiqdik. Amallarni Python da ko‘rib chiqamiz:

```
def generate_batch (batch_size, window_size):
  global data_index

  # two numpy arras to hold target words (batch)
  # and context words (labels)
  batch = np.ndarray (shape=(batch_size), dtype=np.int32)
  labels = np.ndarray (shape=(batch_size, 1), dtype=np.int32)

  # span defines the total window size
  span = 2 * window_size + 1
```

```
        # The buffer holds the data contained within the span
        queue = collections.deque(maxlen=span)

        # Fill the buffer and update the data_index
        for _ in range(span):
          queue.append(data[data_index])
          data_index = (data_index + 1) % len(data)

        for i in range(batch_size // (2*window_size)):
          k=0
          # Avoid the target word itself as a prediction
          for j in
list(range(window_size))+list(range(window_size+1,2*window_size+
1)):
            batch[i * (2*window_size) + k] = queue[window_size]
            labels[i * (2*window_size) + k, 0] = queue[j]
            k += 1

          # Everytime we read num_samples data points, update the queue
          queue.append(data[data_index])

          # If end is reached, circle back to the beginning
          data_index = (data_index + np.random.randint(window_size))
% len(data)

        return batch, labels
```

skip-gram modelni aniqlaymiz.

Eng avval ba'zi bir gipermatnlarni aniqlaymiz:

```
batch_size = 128
embedding_size = 64
window_size = 4
num_sampled = 32 # Number of negative examples to sample.
```

batch_size joriy vaqtda biz qayta ishlaydigan berilganlar elementlar sonini belgilaydi. embedding_size bu vektor uzunligi. Window_size giperparametri kontekst oyna o'lchamini belgilaydi. Va oxirgisi, num_sampled — yo'qotish funksiyasidagi mos kelmaydigan namunalar soni (k). So'ngra kiruvchi va chiquvchi berilganlarni aniqlaymiz:

```
tf.reset_default_graph()
# Training input data (target word IDs).
train_dataset = tf.placeholder(tf.int32, shape=[batch_size])
# Training input label data (context word IDs)
train_labels = tf.placeholder(tf.int32, shape=[batch_size, 1])
```

train_dataset kirish qismiga so'zlar identifikatorlari ro'yxatini batch_size qabul qilib, tanlangan maqsadli so'zlar to'plamini ifodalaydi. Tanlangan maqsadli so'zlarga mos keluvchi kontekstli so'zlar batch_size ro'yxatini Train_labels ifodalaydi.

So'ngra neyron tarmoq parametrlarini aniqlaymiz:

```
##################################################
#          Model variables                       #
##################################################

# Embedding layer
embeddings = tf.Variable(tf.random_uniform([vocabulary_size,
embedding_size], -1.0, 1.0))

# Neural network weights and biases
softmax_weights = tf.Variable(
  tf.truncated_normal([vocabulary_size, embedding_size],
                      stddev=0.1 / math.sqrt(embedding_size))
)
softmax_biases                                          =
tf.Variable(tf.random_uniform([vocabulary_size],-0.01,0.01))
```

Embedding layer -----TensorFlow da embeddings o'zgaruvchi orqali, softmax_weights – o'zgaruvchi og'irlik ko'rsatkichi, siljiy parametri—softmax_biases orqali belgilanadi.

Embedding layer va neyrotarmoqni birlashtirib, natijani optimallashtiramiz:

Look up embeddings for a batch of inputs.

embed = tf.nn.embedding_lookup(embeddings, train_dataset)

tf.nn.embedding_lookup funksiyasi kirish qismiga embedding layer va so'zlar identifikatorlari to'plamini qabul qiladi (train_dataset), chiqish qismiga mos keluvchi vektorni uzatadi.

Sampled softmax loss funksiyasi:

```
##################################################
#          Computes loss                         #
##################################################
loss = tf.reduce_mean(tf.nn.sampled_softmax_loss(
weights=softmax_weights, biases=softmax_biases, inputs=embed,
labels=train_labels, num_sampled=num_sampled,
num_classes=vocabulary_size) )
```

Bunda tf.nn.sampled_softmax_loss kirish qismiga og'irlik ko'rsatkichi (softmax_weights), oldingi funksiyalar asosida olingan siljishlar (softmax_biases), embed to'plamini, to'g'ri keluvchi kontekstli so'zlar identifikatorlarini (train_labels), shovqinli namunalar sonini (num_sampled), lug'at hajmini (vocabulary_size) qabul qiladi. embedding layer parametrlari va neyrotarmoq bo'yicha yo'qotish funksiyasini optimallashtiramiz:

```
##################################################
#          Optimization                          #
##################################################
optimizer = tf.train.AdamOptimizer(0.001).minimize(loss)
```

Normiruem embedding layer:

```
##################################################
```

```
# For evaluation #
#####################################################
norm = tf.sqrt(tf.reduce_sum(tf.square(embeddings), 1, keepdims=True))
normalized_embeddings = embeddings / norm
```

Kodni ishga tushiramiz.

TensorFlow modelni qanday ishga tushirish lozim? Eng avval session ni aniqlaymiz va o‘zgaruvchilarni tasodifiy ravishda initsiallashtiramiz:

```
num_steps = 250001
session = tf.InteractiveSession()
# Initialize the variables in the graph
tf.global_variables_initializer().run()
print('Initialized')
average_loss = 0
```

So‘ngra oldindan belgilangan qadamlar asosida berilganlar guruhini shakllantiramiz: maqsadli so‘zlar (batch_data) va kontekst so‘zlar (batch_labels):

```
for step in range(num_steps):
  # Generate a single batch of data
  batch_data, batch_labels = generate_batch( batch_size, window_size)
```

So‘ngra shakllantirilgan har bir guruh uchun embedding layer ni va neyron tarmoqni session.run([optimize, loss],...) yordamida optimallashtiramiz. Shu bilan birga xatolikni aniqlaymiz, bu uning kamayishiga ishonch hosil qilish uchun lozim bo‘ladi.

```
# Optimize the embedding layer and neural network
  # compute loss
  feed_dict = {train_dataset : batch_data, train_labels : batch_labels}
  _, l = session.run([optimizer, loss], feed_dict=feed_dict)
```

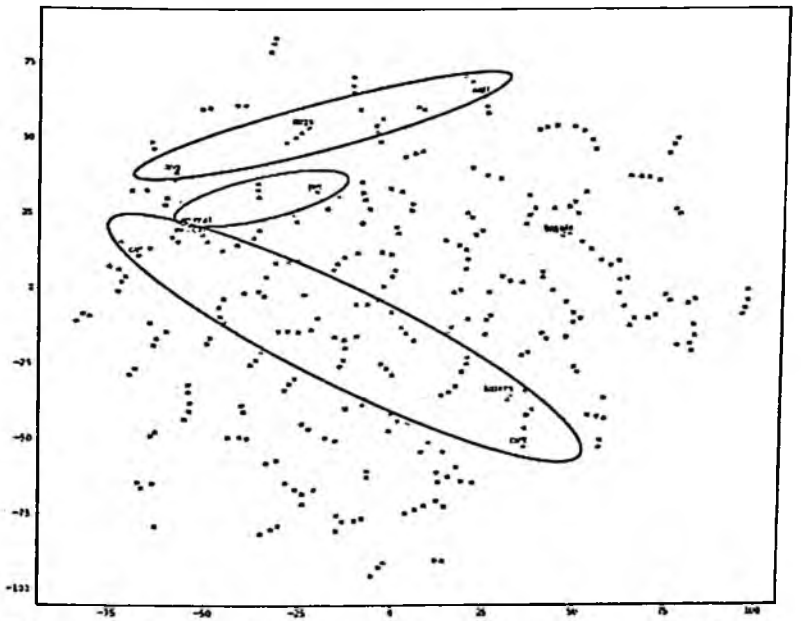
Har bir besh ming qadamda ekranga o‘rtacha xatolikni chiqaramiz:

```
if (step+1) % 5000 == 0:
    if step > 0:
      average_loss = average_loss / 5000
    print('Average loss at step %d: %f' % (step+1, average_loss))
    average_loss = 0
```

natijada vektorlar olinib, u keyinchalik ma'lum bir so'zlarni vizuallashtirish uchun qo'llanilishi mumkin:

```
sg_embeddings = normalized_embeddings.eval()
session.close()
```

Agar natijani t-SNE ga o'xshash algoritm yordamida vizuallashtirsak, quyidagi ko'rinishni olamiz:



1.31-rasm. So'zlarning semantik nuqtayi nazardan joylashuvi.

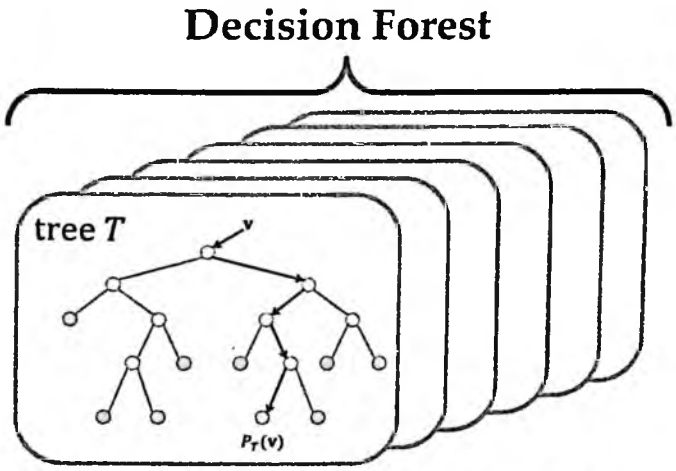
Mushuk so'ziga yaqin so'zlarni ko'rish mumkin, ular ma'lum bir maydonda joylashgan (cat, kitten, cats, wildcat), it so'ziga yaqin bo'lgan so'zlar (dog, dogs, wolf) boshqa maydonda joylashgan. Bu so'zlar

o‘rtasida joylashgan so‘zlar (masalan: animal yoki pet) ma’no jihatdan ham mushuk, ham it so‘zlariga yaqin ekanligini ko‘rish mumkin.

Xulosa qilib aytganda, so‘zlarning vektorli ifodasi juda aniq ishlovchi instrument bo‘lib, zamonaviy mashina asosida o‘rganishni yaxshilashga ulkan hissa qo‘shadi. Word2vec ning asosiy ishlash prinsipini ko‘rib o‘tdik. So‘ngra skip-gram algoritmi izohi uning TensorFlowda amalga oshirilishini ko‘rib chiqdik. Natija ko‘rinishda so‘zlarning vektorli ko‘rinishini vizuallashtirib, uning semantikasi saqlanganligiga ishonch hosil qildik.

II bob. SUN'IY NEYRON TARMOQLARI ALGORITMLARI VA FOYDALANISH USLUBLARI

2.1. Yechimlar daraxti algoritmlari



2.1-rasm. Daraxtsimon yechimlar algoritmi.

Iterativ dixotomizator 3 (ID3) – bu yuqoridan pastga qarab daraxtsimon strukturani hosil qiladi. Bunda ildizdan boshlanib, har bir tugunda tekshirilishi lozim bo'lgan atribut tanlanadi. Har bir atribut ma'lum bir statistik vosita yordamida baholanadi, bu atribut asosida berilganlarni samarali taqsimlash amalga oshiriladi. Tanlangan atribut ildiz hisoblanib, uning qiymatlari tarmoqlanadi, so'ngra jarayon boshqa atributlar asosida davom etadi. Atribut tanlangandan so'ng orqaga qaytish imkoni bo'lmaydi.

C4.5 va C5.0 (yondashuvning ikki ko'rinishi). C4.5 keyingi iteratsiya Quinlan – bu ID3ning eng yangi versiyasi. Yangi funksiyalar (ID3ga nisbatan): (I) ham uzluksiz, ham diskret funksiyalarni qabul qiladi; (II) to'liq bo'lmagan berilganlar tugunini qayta ishlaydi; (III) yaqinlashtirish aniqligi tushishi bilan bog'liq muammolarni yuqoridan

past usuli asosida, odatda "qirqim" nomiga ega usul hal etadi va (IV) turli og'irlik ko'rsatkichlari xususiyati ko'rinishda o'rganish jarayoni qiymatlari qo'llanilishi mumkin. C5.0, Quinlan eng so'nggi iteratsiyasi hisoblanadi.

Tasnif va regressiya daraxti (CART) – CART odatda "yechimlar daraxti" iborasining abbreviaturasi sifatida qo'llaniladi. Umumiy holda CARTning realizatsiyasi yuqorida keltirilgan C4.5 realizatsiyasiga o'xshash. Bitta farqli tomoni bu CART qiymatli taqsimlashga asoslangan daraxtni yaratadi, bunda berilganlarga rekursiv holatda qo'llaniladi, o'z navbatida C4.5 qoidalar to'plamini yaratish bosqichiga ega.

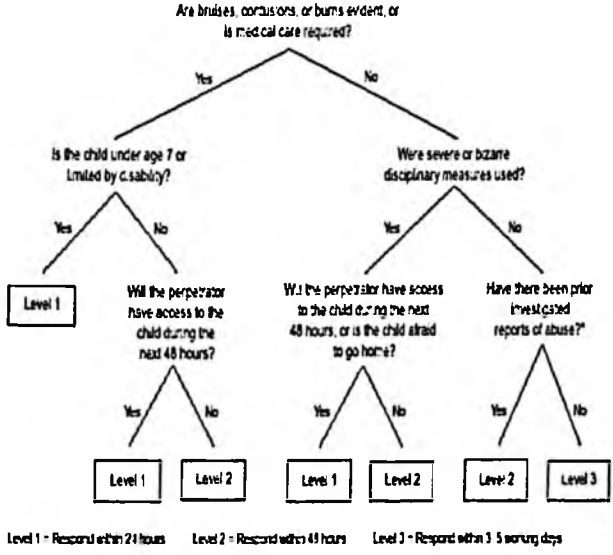
Chi-kvadrat ·o'zaro bog'lanishni avtomatik aniqlash (CHAID) – natijalarni va bashorat qilingan qiymatlarni kategoriya bo'yicha o'zaro bog'lanishini aniqlash algoritmi. Optimal yechimga yetmagunga qadar, barcha bashorat qilinuvchi qiymatlar o'rtasida kategoriyalar bo'yicha kesishmali jadvallarni yaratadi. Optimal yechimga yetgandan so'ng qayta taqsimlash amali to'xtatiladi. CHAID bashorat modelini yoki daraxtini yaratadi, uning yordamida berilganlarni umumlashtirish, ular o'rtasidagi bog'lanishlarni izohlash imkonini beradi. CHAID ni tahlil qilganda nominal, tartiblangan va uzluksiz berilganlarni qo'llash mumkin, bunda cheksiz bashorat qilinuvchi qiymatlar kuzatuvchilar soniga mos ravishda kategoriyalarga bo'linadi. Kategoriyaga ega katta hajmdagi berilganlar to'plamida shablonlarni aniqlashda samarali bo'lib, berilganlarni umumlashtirishning qulay usuli hisoblanadi, chunki berilganlar o'rtasidagi munosabatlar oson vizuallashtiriladi.

Decision Stump – ML modeli bir darajali yechim daraxtidan iborat bo'lib, bitta ichki tugun (ildiz)ga ega va u so'nggi tugunlar (yaproqlar) bilan bog'langan. Bu model bitta kirish funksiyasi qiymatlari asosida bashorat o'tkazadi.

M5–M5 yechimlar daraxti bo'lib, tugunlarda chiziqli regressiya funksiyasi imkoniyati mavjud. Aniqlik ko'rsatkichi bilan birga katta hajmdagi yuzga yaqin atributlar asosida masalalarni yechish imkoniga

ega. M5 daraxt modeli – regressiya masalalarini o‘rganish uchun yechimlar daraxti bo‘lib, Y o‘zgaruvchi qiymatlarini bashorat etish uchun qo‘llaniladi. M5 daraxti CART daraxtida qo‘llaniladigan yondashuvni qo‘llasa-da, o‘rta kvadratik xatolikni funksiya ko‘rinishida tanlaganda, so‘nggi tugun uchun konstantani o‘zlashtirmaydi, balki chiziqli regressiya ko‘p o‘lchovli modeliga to‘g‘ri keladi.

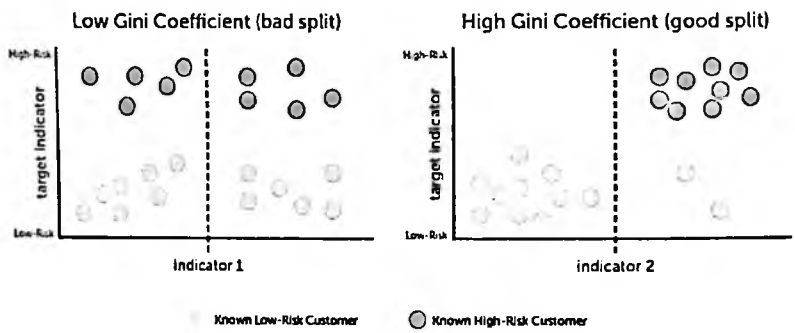
Demak yechimlar daraxti – oddiy deterministik strukturasi bo‘lib, aniq tasniflash masala bo‘yicha qaror qabul qilish qoidasini modellashtirishda qo‘llaniladi. Har bir tugunda bitta xususiyat tanlanib, aynan o‘sha bo‘yicha qaror qabul qilinadi. Taqsimlanish tugashi uchun eng so‘nggi tugunda nuqtalar siyraklanishi lozim. Bunday yaproqli tugun natijani ifodalaydi (turli sinflar bo‘yicha tasnif ehtimolligidir). Quyidagi chizma to‘liq izoh beradi:



2.2-rasm. Turli sinflar bo‘yicha tasnif ehtimolligi.

Quyidagicha amalga oshiriladi

Yechimlar daraxti samaradorligida hal qiluvchi omil bu samarali tarmoqlash jarayonidir. Har bir tugunni taqsimlash maksimal aniq bo‘lishi lozim. Aniqlik deganda sinflarga to‘g‘ri bo‘lish va u asosda bilimlarni kengaytirish unumdorligini bildiradi. Tasvirda 1000 ta so‘z taqsimot ifodasi keltirilgan va unda ma’lum bir intuitsiya keltirilgan:



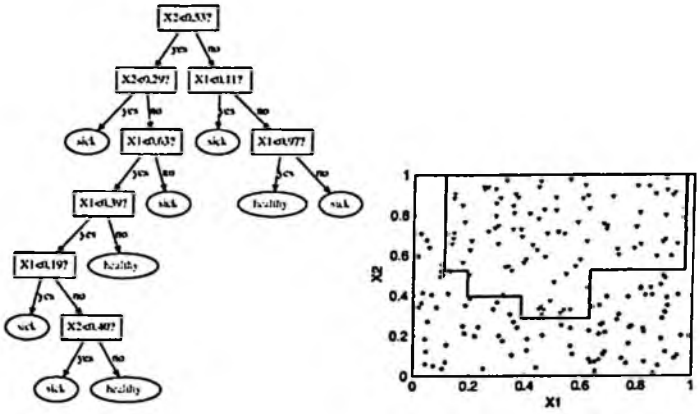
2.3-rasm. 1000 ta so‘z taqsimot ifodasi.

Ikkita keng tarqalgan taqsimot usulida keltirilgan:

1. Djin aralashmasi.
2. Ma’lumotning olinishi.

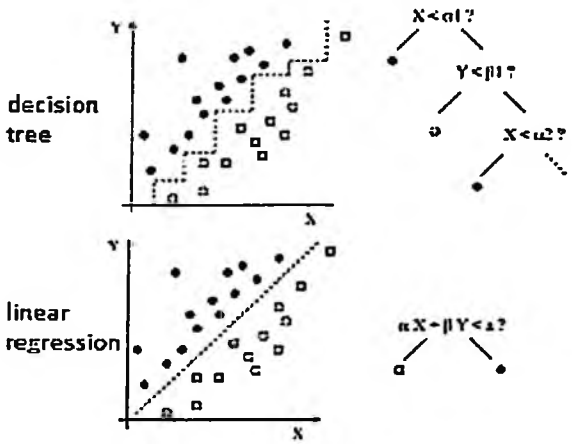
Vizuallashtirish:

Har bir taqsimot to‘g‘ri chiziqqa olib keladi va berilganlar to‘plamini ikki qismga tasniflaydi. Demak, yakuniy qaror chegarasi to‘g‘ri chiziq (to‘rtburchak)dan iborat.



2.4-rasm. Yechimlar daraxti asosida berilganlar to'plamini ikki qismga tasniflash

• Regressiyaga nisbatan yechimlar daraxti daraxtlar tasnifi chegaralarini birlashtirganda zinama-zina ko'rinishga olib keldi.



2.5-rasm. Daraxtlar tasnifi chegaralarini birlashtirganda zinama-zina ko'rinish

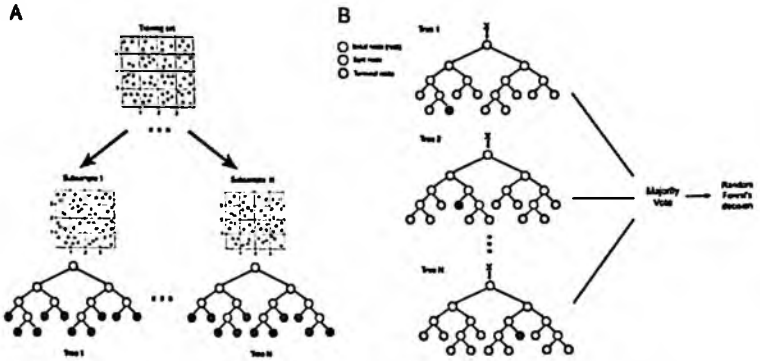
2.2. Tasodifiy o‘rmon

Tasodifiy o‘rmon bu yechimlar daraxti algoritmining takomillashtirilgan ko‘rinishidir. Tasodifiy o‘rmonning asosiy g‘oyasi, bir nechta yechimlar daraxti to‘plamidan iborat (“tasodifiy o‘rmon” tushunchasi shu yerdan kelib chiqqan). Har bir yechimlar daraxti aralash klassifikator (berilganlarning quyi to‘plami)ni beradi. Ularning har biri berilganlarning turli xususiyatlarini fiksirlaydi. Bu daraxtlar ansambli ekspertlar guruhi kabi ishlab, bu yerda har bir ekspert faqat o‘zining fan sohasi bo‘yicha tekshiradi.

So‘ngra tasnif jarayonida “eng ko‘p ovoz olgan” sinfni shakllantiradi. Ekspertlar misolida izohlaydigan bo‘lsak, bu bitta savolni barcha ekspertlarga berib, javoblar varianti ham beriladi va ko‘pchilik tanlagan javob variantiga qarab ovozlar hisoblanadi. Regressiya ko‘rinishida esa barcha daraxtlar bo‘yicha o‘rtachasini olib natijani bashorat qilish mumkin. Shu bilan birga boshqa daraxtlarga nisbatan qaror qabul qiluvchi daraxtlarni validatsiya jarayoni orqali aniqlashtirishimiz mumkin.

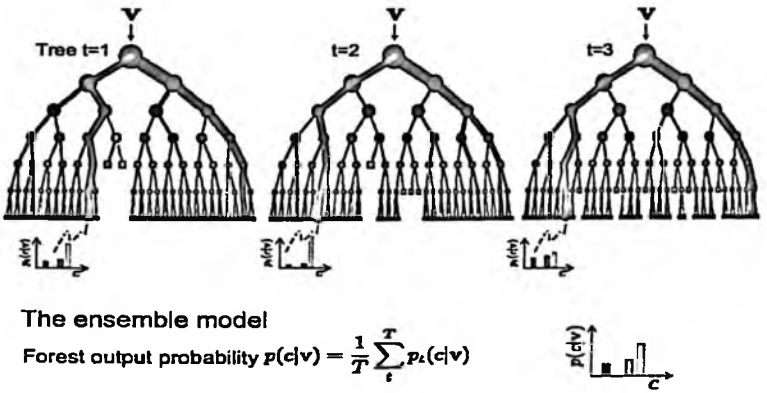
Vizuallashtirish:

- Tasnif uchun ko‘pgina “ovozlar” ekspert (daraxt)lar tomonidan beriladi.



2.6-rasm. Tasodifiy o‘rmon – yechimlar daraxti algoritmi.

• Shu bilan birga ehtimolliklardan foydalanib, tasniflash uchun cheklanmani o‘rnatishimiz mumkin.



2.7-rasm. Tasodifiy o‘rmonda asosiy giperparametrlar.

Tasodifiy o‘rmonda asosiy giperparametrlar

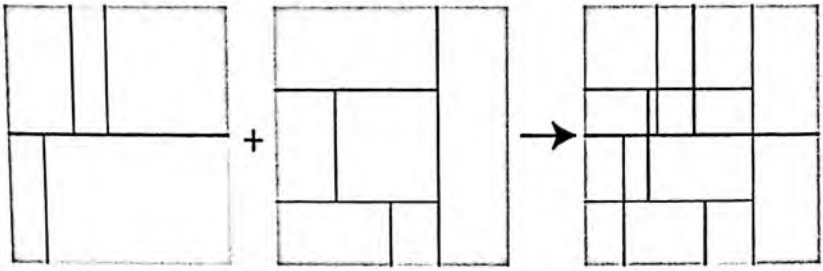
1. **N-tree**: o‘rmon daraxtlari soni. Odatda, 100 tagacha. Daraxtlarning ko‘pligi ba’zan ortiqchalik qiladi.

2. **Mtry**: aniq daraxt bo‘yicha taqsimotda qo‘llanilishi mumkin bo‘lgan tasodifiy tanlanadigan o‘zgaruvchilar soni.

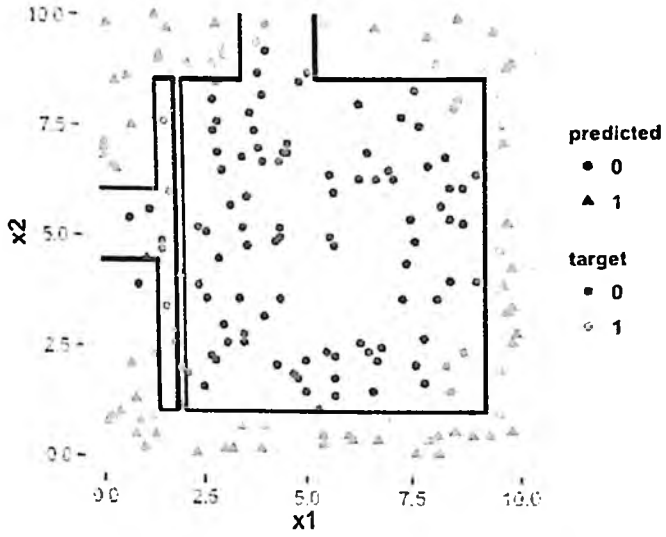
3. **O‘zgartirmoq**: namuna tanlovi o‘zgartirish yoki o‘zgarmas holda amalga oshirilishi lozim.

Tasodifiy o‘rmon yechimlar chegarasi.

Random Forest ko‘pgina daraxtlar to‘plamini qo‘llashi sababli, qaror qabul qilish chegaralarini murakkab darajada belgilashi mumkin. Quyida tasodifiy o‘rmon yaratishi mumkin bo‘lgan qaror chegaralari ko‘rinishi keltirilgan:



2.8-rasm. Tasodifiy o‘rmon yaratishi mumkin bo‘lgan qaror chegaralari ko‘rinishi

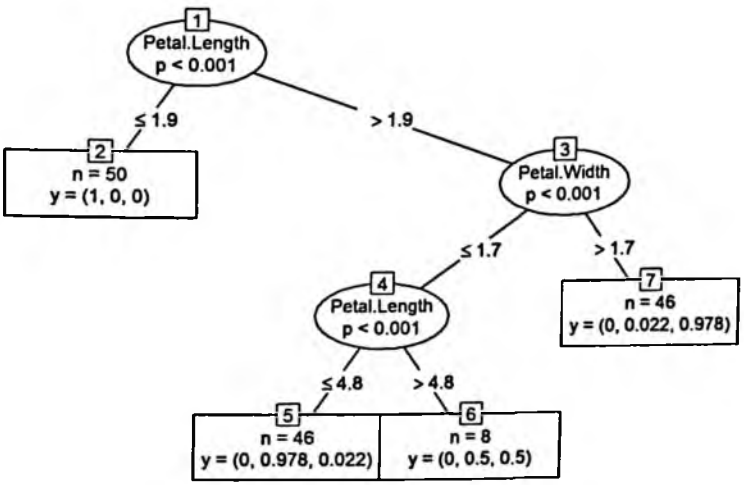


2.9-rasm. Random Forest ko‘pgina daraxtlar to‘plami

2.2.1. R da tasodifiy o‘rmon.

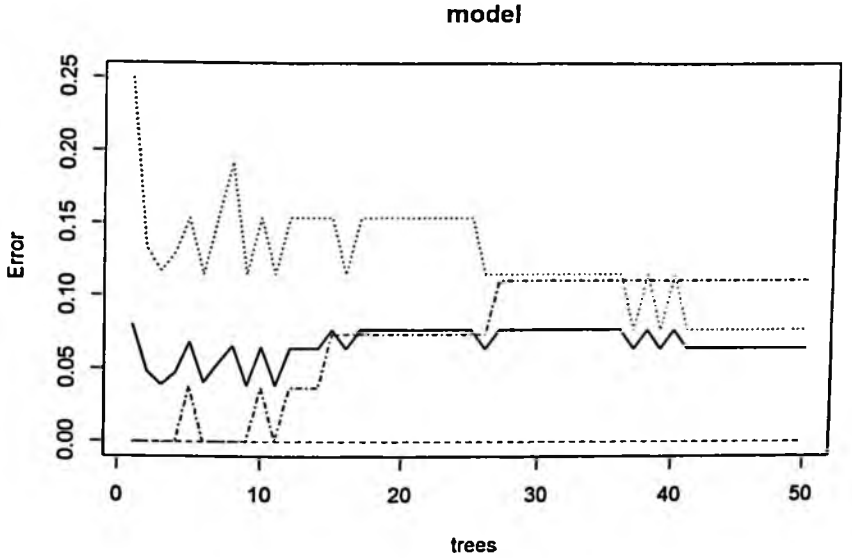
```
#Random Forest in R using IRIS data
#Split iris data to Training data and testing data
 ind &lt;- sample(2,nrow(iris),replace=TRUE)
```

```
train &lt;- iris[ind==1,]
test &lt;- iris[ind==2,]
head(train, 3)

##  Sepal.Length Sepal.Width Petal.Length Petal.Width
Species
## 3          4.7         3.2          1.3         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 7          4.6         3.4          1.4         0.3  setosa

#Try plotting how a decision tree for IRIS will look like
x &lt;- ctree(Species ~ ., data = iris)
plot(x, type = "simple")
```



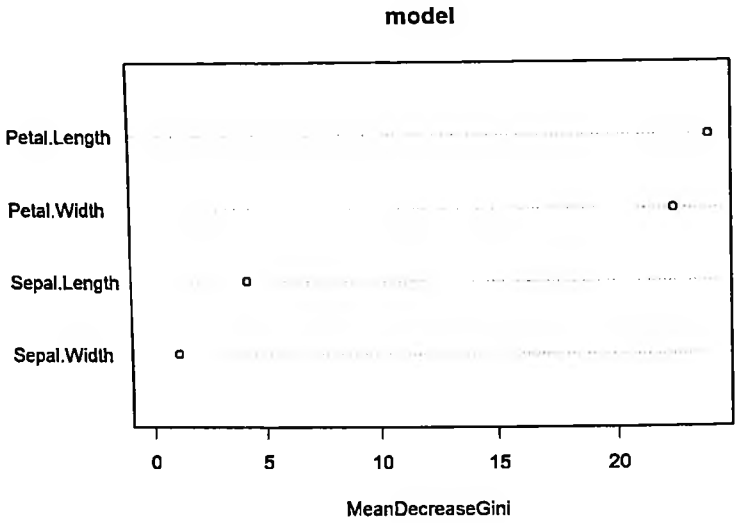
2.10-rasm. Dasturning grafik ko'rinishi.

```
#Train a RF model
model &lt;- randomForest(Species~., data=train , ntree=50,
mtry = 2,proximity=TRUE)
#Print RF model details
print(model)

##
## Call:
##  randomForest(formula = Species ~ ., data = train, ntree
= 50,     mtry = 2, proximity = TRUE)
##               Type of random forest: classification
##                     Number of trees: 50
## No. of variables tried at each split: 2
##
##        OOB estimate of  error rate: 6.41%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         25          0         0  0.00000000
## versicolor      0         24         2  0.07692308
## virginica       0          3        24  0.11111111

#Plot error vs ntree for classes
plot(model)
```



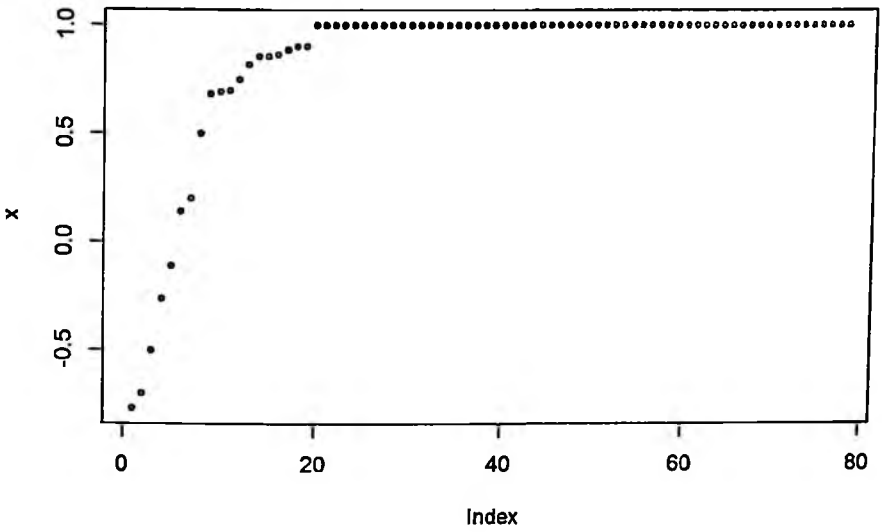
2.11-rasm. Dastur modelining grafik ko'rinishi.

```
#Use the value of ntree where error becomes constant

#Plot variable importance matrix
varImpPlot(model)
```



2.12-rasm. Dastur modelining grafik ko'rinishi.

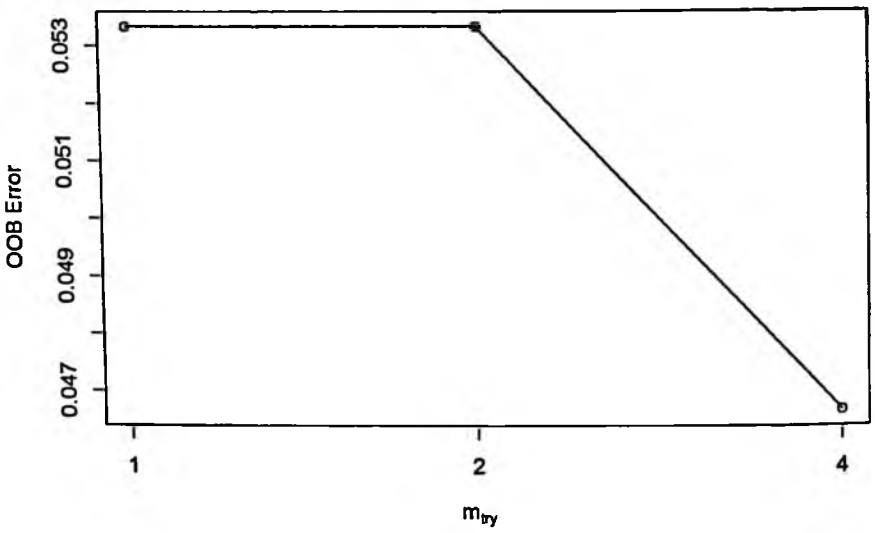
```
#Test the model on testdata
pred &lt;-predict(model , newdata=test)
table(pred, test$Species)

##
## pred       setosa versicolor virginica
##  setosa       25         0         0
##  versicolor    0        23         1
##  virginica     0         1        22

#Plot the margin (positive -&gt;  correct classification)

plot(margin(model,test$Species))
```



2.13-rasm. Dastur modelining grafik ko'rinishi.

```
1  #Search for optimal value of mtry for random forest
2
3  tune &lt;- tuneRF(iris[,-5],iris[,5])
```

```
1  ## mtry = 2  OOB error = 5.33%
2  ## Searching left ...
3  ## mtry = 1     OOB error = 5.33%
4  ## 0 0.05
5  ## Searching right ...
6  ## mtry = 4     OOB error = 4.67%
7  ## 0.125 0.05
```



2.14-rasm. Dastur modelining grafik ko'rinishi.

Yutuq va kamchiliklari

Yutuqlari:

- Qaror qabul qilishning aniq modellaridan biri.
- Katta hajmdagi berilganlar bilan ishlashda samaralidir.
- O'zgaruvchilar ahamiyatlilarini ajratib olish uchun qo'llanilishi mumkin.
- Funksiyani ishlab chiqishni talab etmaydi (masshtablash va normallashtirish).

Kamchiliklari:

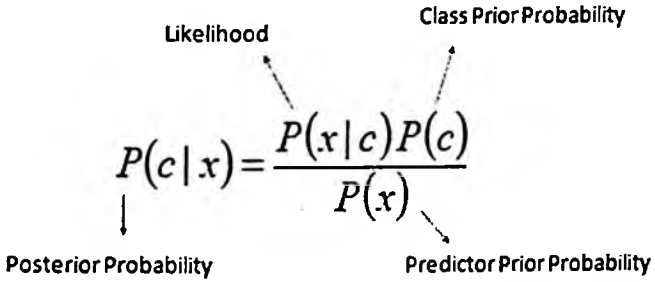
- Shovqinli berilganlar bilan ishlashda orttirmalilik paydo bo'lishi.
- Yechimlar daraxtlaridan farqli ravishda natijalar murakkab talqin qilinadi.
- Giperparametrlar yuqori aniqlikda sozlanishini talab etadi.

Ilovalar

Tasodifiy o‘rmon turli sohalarga tatbiq etilgan bo‘lib, ba’zi ilovalar quyidagi imkoniyatlarni o‘z ichiga oladi:

- Obyektlarni aniqlash.
- Molekular biologiya (aminokislota ketma-ketligini tahlil etish).
- Masofali zondirovanie (obrazlarni aniqlash).
- Astronomiya (yulduzlar galaktikasi tasnifi va h.k.).

2.3. Bayes algoritmi



$$P(c|X) = P(x_1|c) \times \ldots \times P(x_n|c) \times P(c) \qquad (42)$$

Sodda bayes usuli – bu sinfda muayyan funksiyaning mavjudligi boshqa bir funksiyaning mavjudligini belgilamasligini, ya’ni bog‘liqlik mavjud emasligini bildiradi. Amalda qo‘llash asosida olinadigan qiymatlar ehtimollikni aniqlash usulini belgilaydi. Masalan: P(c|x) qiymati P (c)dan P(x) esa P(x|c) aniqlash, bu usul katta hajmdagi berilganlar to‘plami bilan ishlashda samarali hisoblanadi.

Sodda bayes gausslangan usuli – ehtimolliklar taqsimotini gausslangan (normal) deb qabul qiladi. Uzluksiz taqsimot uchun sodda Bayes Gauss algoritmi tanlash algoritmi hisoblanadi.

Multinominal bayes – bu sodda bayes usulining aniq amaliy ko‘rinishi bo‘lib, bunda P (Featurei | Class) multinominal taqsimlash (qatlamlar soni, ehtimolligi va h.k.). Bu asosan hujjatlarni tasniflash masalalarini yechishda qo‘llaniladi (hujjat qaysi kategoriyaga sport,

siyosat, texnologiya va boshqalar. kirishini aniqlash). Klassifikator qo'llaydigan funksiyalar/bashorat qilinuvchi qiymatlar, hujjatlarda takrorlanayotgan so'zlar mavjudligi.

Bitta bog'lanish bo'yicha o'rtacha baholash – sodda bayes klassifikatorlarda atributlar o'rtasida bog'lanishni aniqlash masalalarini yechish uchun ishlab chiqilgan. Ko'pincha AODE, hisoblash hajmi oshirilgan sodda bayes tarmog'iga nisbatan aniqroq klassifikatorlarni ishlab chiqadi.

Bayes ishonch tarmog'i (BBN) – ehtimollik grafik modeli (statistik model ko'rinishi) bo'lib, o'zgaruvchilar va ular shartli bog'lanishlarni asiklik yo'naltirilgan grafi orqali ifodalaydi (DAG). Masalan: bayes tarmog'i kasallik va simptomlar o'rtasida ehtimolli bog'lanishlarni ifodalashi mumkin. Simptomlarni inobatga olgan holda, mavjud kasallik ehtimolligini hisoblash uchun tarmoqdan foydalanish mumkin. BBN – bu diagrammaning maxsus ko'rinishi (yo'naltirilgan graf deb nomlangan) bo'lib, o'zaro bog'langan ehtimolliklar jadvaliga ega.

Bayes tarmog'i (BN) – bayes tarmog'ining maqsadi shartli bog'lanishlarni modellashtirish bo'lib, ya'ni mo'ljallar grafida berilganlar o'rtasida shartli bog'lanish mavjudligi sababini ifodalashni anglatadi. Ulardan foydalangan holda grafikda ixtiyoriy o'lcham borasidagi xulosani aynan omillar asosida chiqarish imkonini beradi.

Yashirin Markov modellari (HMM) – grafik modellar ehtimolliklari sinfi bo'lib, kuzatilayotgan o'zgaruvchilar to'plamida mavjud noma'lum (yashirin) o'zgaruvchilar ketma-ketliklarini bashorat qilish imkonini beradi. Masalan: uning yordamida ob-havo (yashirin o'zgaruvchi)ni bashorat qilishimiz mumkin, bunda biron kishi kiygan kiyimi borasidagi ma'lumot asos (kuzatuv) bo'lib xizmat qiladi. HMMni Bayes tarmog'i sifatida qarash mumkin, ya'ni vaqt taqsimoti bo'yicha kuzatuvlar yordamida ma'lum vaqt oraliqlarida olinadigan natijalar asosida, bashorat qilishda aniqlik ko'rsatkichi yuqori bo'lgan yashirin holatlar ketma-ketligini olishga yo'naltirilgan.

Shartli tasodifiy maydonlar (CRF) – mashinali o‘rganishning klassik modeli bo‘lib, ketma-ket modellar o‘rganishi uchun qo‘llaniladi. Bu ko‘rinishdagi klassifikator turli sinflar bo‘yicha yechim chegarasini modellashtiradi. Bu klassifikator va modellarni yaratuvchilar o‘rtasidagi farq shundaki, birinchi ko‘rinish, bu ehtimolliklar shartli taqsimoti bo‘yicha modellashtirsa, ya’ni P (y | x), modellarni yaratish usulida ehtimolliklar hamkorli taqsimotlarni modellashtirishga, ya’ni P (x, y) harakat qiladi. Ularning asosiy prinsipi bu kirish ketma-ketligiga logistik regressiyani qo‘llashdir. Yashirin Markov modellar CRF bilan ba’zi umumiy xususiyatga ega bo‘lib, ulardan biri bu kirish ketma-ketligida qo‘llanilishidir. Ko‘pincha CRFs NLP (нейро-лингвистическое программирование)da qo‘llaniladi.

Misol ko‘rib o‘tamiz

CRFs – daraxtsimon yoki noaniq graflar ko‘rinishidagi ketma-ketliklarni, ya’ni strukturalanmagan ketma-ketlikni umumlashtirish uchun qo‘llaniladi. Ko‘rilayotgan misolimizda chiziqli ketma-ketlikka ega CRF strukturasiga asoslanamiz. Misol so‘ngida esa umumiy CRF va chiziqli ketma-ketlikka ega CRF bilan umumlashtirgan holda izoh beriladi.

2.3.1. Asosiy nazariya

Mashinali o‘rganish tizimlarining ko‘pchiligi CRFni qo‘llaydi, bunda biologik ketma-ketlikdan boshlab, kompyuterli video kuzatuv, tabiiy til ma’lumotlarini qayta ishlash masalalarida qo‘llanilib kelingan, CRF bo‘yicha 20000 dan ortiq maqolalar chiqarilgan.

So‘nggi yillarda CRF modellari LSTM bilan birlashtirilgan va eng samarali natijalar olingan. NLP sohasida CRF ketma-ketliklar qoidasi deb hisoblanadi: agar aniqlik darajasini oshirmoqchi bo‘lsangiz ma’lumotlar ketma-ketligiga LSTM qatlamiga o‘rnatishning o‘zi yetarli bo‘lib hisoblanadi.

Ketma-ketliklar tasnifi masalasida asosiy maqsad ketma-ketliklar ehtimolligi (y)ni kiritilayotgan ketma-ketliklar vektori ($\mathbf{X}$)ga nisbatan aniqlanadi. Uning ko'rinishi $P(y|X)$.

Eng avval, ifodalar izohini ko'rib chiqamiz:

- O'rganish to'plami: kirish va maqsadli natija ketma-ketliklar juftligi $\{(X_i, y_i)\}$
- I – kiruvchi ketma-ketliklar vektorlari: $X_\ell = [x_1, \ldots, x_l]$
- I – nchi maqsadli ketma-ketliklar to'plami: $Y_\ell = [y_1, \ldots, y_l]$
- ℓ – ketma-ketliklar uzunligi.

(X, y) tanlov uchun doimiy tasnif masalasida $P(y|X)$ni hisoblanadi, bunda ketma-ketlikdagi k – pozitsiyadagi har bir element uchun ehtimolliklar ko'paytmasi bajariladi, bunda $1 \leq k \leq \ell$:

$$P(y|X) = \prod_{k=1}^{l} P(y_k|x_k) = \prod_{k=1}^{l} \frac{\exp(U(x_k, y_k))}{Z(x_k)} = \frac{\exp(\sum_{k=1}^{l} U(x_k, y_k))}{\prod_{k=1}^{l} Z(x_k)} \quad (43)$$

$P(y_k|x_k)$ ifodani normallashtirilgan eksponentani qo'llab modellashtiramiz. Bu *softmax* amali bilan bir xil bo'lib, neyron tarmoqlarda keng qo'llaniladi. *Exp* qo'llash lozim bo'lgan hollarni ko'rib o'tamiz:

1. **Miqdorning yetarlicha bo'lmaganligi:** juda kichik sonlarni ko'paytirishda kichik sonlar olinadi, bu o'z navbatida *miqdor yetmasligiga* olib keladi.

2. **Manfiy bo'lmagan natijalar:** barcha qiymatlar 0 va musbat sonlar oralig'ida ifodalaniladi.

3. **Monoton holda o'sib boruvchi:** bunda yuqori darajali qiymatlarni oshiradi, past darajali qiymatlarni kamaytiradi. Bu jarayon *argmax* amali bilan bir xildir.

Bir to'plamda ikki o'zgaruvchi: ikkita simvolni to'plamdan tanlaymiz: *U va Z.* Ular izohini ko'rib o'tamiz.

$U(x, y)$ tanlovlar yoki birlamchi ko'rsatkichlar kabi bo'lishi mumkin. Ya'ni y ko'rsatkichga nisbatan x vektor k - vaqt qadam oralig'ida ifodalaniladi. Buni LSTM ning k- chiqish qismi sifatida qabul

qilishingiz mumkin. Aslida nazariy jihatdan **x** vektor turli ko'rinishda bo'lishi mumkin. Amalda esa **x** vektor atrofdagi elementlarning ketma-ketliklar zanjiri hisoblanadi, ya'ni kiritish oynasidagi so'zlardan iborat. Har bir *unar* omil modelda og'irlik ko'rsatkichi asosida belgilangan. LSTM natijasi deb qabul qilinsa, anglash ancha oson bo'ladi.

$Z(x)$ ni odatda **taqsimlash funksiyasi** deb nomlanadi. Bularning barchasini oddiy hol omili deb qabul qilamiz, chunki ehtimollik darajalari: har bir ko'rsatkich uchun qiymati 1ga teng deb baholanishi lozim. *Softmax* funksiyasining maxraji kabi qabul qilish lozim.

Yuqorida softmax funksiyasini qo'llagan holda ehtimollikni aniqlaydigan doimiy tasniflovchi modelni ko'rib o'tdik. Keyingi qadamda yangi og'irlik ko'rsatkichini qo'shgan holda y_k so'ng y_{k+1} kelish ehtimollikni modellashtirish lozim bo'ladi. Bu model asosida ketma-ketliklar o'rtasida bog'lanishni aniqlaymiz, demak, *chiziqli-zanjirli CRF* tushunchasi paydo bo'ladi. Buning uchun, oldingi ehtimollikni $P(y_k + 1|y_k)$ga ko'paytirib, eksponensial xususiyat sifatida foydalanishimiz mumkin, $U(x, y)$ unar ko'rinishi va o'rganilayotgan yangi $T(y, y)$ baholash yig'indisi ko'rinishiga olib keladi:

$$P(y|X) = \frac{\exp\left(\sum_{k=1}^{l} U(x_k, y_k) + \sum_{k=1}^{l-1} T(y_k, y_{k+1})\right)}{Z(X)} \quad (44)$$

$T(y, y)$ – **bu** matritsa *(nb_labels , nb_labels)* kattalikda bo'lib, har bir qiymat o'rganiladigan parametr deb qabul qilinadi va *i*- metkadan *j*-*y* metkaga o'tishni bildiradi. Yangi o'zgaruvchilarni ko'rib o'tamiz:

- **Chiqarib tashlash yoki unar ballar** *(U):* bu ballar $\mathbf{x}_k$ kirish berilganlarini inobatga olgan holda y_k bo'lish ehtimolligini belgilaydi.
- **O'tish ko'rsatkichi** *(T):* bu ko'rsatkich y_k dan keyin y_{k+1} ketma-ketlikda kelish ehtimolligini belgilaydi.

• **Taqsimot funksiyasi** *(Z):* normallashtirish koeffitsiyenti bo'lib, oxirgi bosqichda ehtimollikni aniqlash uchun qo'llaniladi.

Demak, taqsimot funksiyasi Z ni aniqlash qoldi:

$$Z(X) = \sum_{y'_1} \sum_{y'_2} \dots \sum_{y'_k} \dots \sum_{y'_l} \exp\left(\sum_{k=1}^{l} U(x_k, y'_k) + \sum_{k=1}^{l-1} T(y'_k, y'_{k+1})\right) \quad (45)$$

Z (X) hisoblash murakkabligi ko'rinib turibdi, unda juda ko'p sikllar ichma-ich joylashgan Bu summa har bir vaqt kesimida bo'lishi mumkin bo'lgan barcha kombinatsiyalar yig'indisi. Aniqrog'i, metkalar to'plami ustida $\ell!$ (faktorial) hisoblash lozim bo'ladi. Bu vaqt borasida murakkablikka $O(\ell! |Y|^2)$ olib keladi.

Bu borada samarali natija olinishi uchun rekurrent bog'lanishni va dinamik dasturlashni qo'llash mumkin. Ushbu algoritm bajarilishi ketma-ketligiga qarab to'g'ri algoritm yoki teskari algoritm deb nomlanadi.

Kod

Kodni yaratishda eng avval nn.Module v *pytorch* dan olingan CRF sinfni yaratish bilan boshlaymiz, bunda gradiyentni avtomatik kuzatish imkoni beriladi.

Maxsus cheklanishlar qo'yib to'lish chegarasi bo'yicha harakatni aniqlash imkonini beradi.

```
import torch
from torch import nn
class CRF(nn.Module):
    """
    Linear-chain Conditional Random Field (CRF).
    Args:
        nb_labels (int): number of labels in your tagset, including
special symbols.
        bos_tag_id (int): integer representing the beginning of
sentence symbol in
```

```
            your tagset.
        eos_tag_id (int): integer representing the end of sentence
symbol in your tagset.
        batch_first (bool): Whether the first dimension represents the
batch dimension.
        """
        def __init__(
            self, nb_labels, bos_tag_id, eos_tag_id, batch_first=True
        ):
            super().__init__()
            self.nb_labels = nb_labels
            self.BOS_TAG_ID = bos_tag_id
            self.EOS_TAG_ID = eos_tag_id
            self.batch_first = batch_first
            self.transitions = nn.Parameter(torch.empty(self.nb_labels,
self.nb_labels))
            self.init_weights()
        def init_weights(self):
            # initialize transitions from a random uniform distribution
between -0.1 and 0.1
            nn.init.uniform_(self.transitions, -0.1, 0.1)
            # enforce contraints (rows=from, columns=to) with a big
negative number
            # so exp(-10000) will tend to zero
            # no transitions allowed to the beginning of sentence
            self.transitions.data[:, self.BOS_TAG_ID] = -10000.0
            # no transition alloed from the end of sentence
            self.transitions.data[self.EOS_TAG_ID, :] = -10000.0
```

2.3.2. Yo‘qotish funksiyasi mohiyati

Tasnif masalasining asosiy mohiyati o‘rganish davomidagi xatolikni minimallashtirishdir. *L* yo‘qotish funksiyasini aniqlab amalga

oshirishimiz mumkin, bunda kirish ma'lumotlari sifatida bashorat natijalari va aniq ko'rsatmalarni qabul qiladi, so'ngra ular teng bo'lsa nolni, agar teng bo'lmasa xatolikni bildiradi.

E'tibor bering, $P(\mathbf{y} \mid X)$ni hisoblash, ya'ni aynan uni maksimallashtirish kerak bo'ladi. Bu masalani minimallashuv ko'rinishiga keltirish uchun ehtimollikning teskari logarifmini olamiz.

Ehtimollikning logarifmik teskari yo'qotish funksiyasi kabi ma'lum *(NLL-Loss)*. Bizning misolda bu quyidagi ko'rinishda bo'ladi: $L = -log(P(\mathbf{y} \mid X))$. Log funksiyasini qo'llagan holda $log(a/b) = log(a) - log(b)$, quyidagi ifodani olamiz:

$$-\log\big(P(\mathbf{y} \mid X)\big) = -\log\left(\frac{\exp\left(\sum_{k=1}^{l} U(x_k, y_k) + \sum_{k=1}^{l-1} T(y_k, y_{k+1})\right)}{Z(X)}\right) =$$

$$\log\big(\mathrm{Z(X)}\big) - \log\big(\exp\left(\sum_{k=1}^{l} U(x_k, y_k) + \sum_{k=1}^{l-1} T(y_k, y_{k+1})\right)\big) = \log\big(\mathrm{Z(X)}\big) -$$

$$\left(\sum_{k=1}^{l} U(x_k, y_k) + \sum_{k=1}^{l-1} T(y_k, y_{k+1})\right) = \mathrm{Z}_{log}(\mathrm{X}) - \left(\sum_{k=1}^{l} U(x_k, y_k) + \sum_{k=1}^{l-1} T(y_k, y_{k+1})\right) \qquad (46)$$

Bunda Z_{log} hisoblashlar *log* funksiyasidan kelib chiqqan holda aniqlanadi. Bu algoritmni keyinchalik qo'llashda bizning masalani yechimini osonlashtiradi. Buning uchun kod ko'rinishi qanday bo'lishini ko'rib chiqamiz:

```
def forward(self, emissions, tags, mask=None):
    nll = -self.log_likelihood(emissions, tags, mask=mask)
    return nll
def log_likelihood(self, emissions, tags, mask=None):
    """Compute the probability of a sequence of tags given a
sequence of
    emissions scores.
    Args:
```

```
            emissions (torch.Tensor): Sequence of emissions for each
label.
                Shape of (batch_size, seq_len, nb_labels) if batch_first is
True,
                (seq_len, batch_size, nb_labels) otherwise.
            tags (torch.LongTensor): Sequence of labels.
                Shape of (batch_size, seq_len) if batch_first is True,
                (seq_len, batch_size) otherwise.
            mask (torch.FloatTensor, optional): Tensor representing
valid positions.
                If None, all positions are considered valid.
                Shape of (batch_size, seq_len) if batch_first is True,
                (seq_len, batch_size) otherwise.
        Returns:
            torch.Tensor: the log-likelihoods for each sequence in the
batch.
                Shape of (batch_size,)
        """
        # fix tensors order by setting batch as the first dimension
        if not self.batch_first:
            emissions = emissions.transpose(0, 1)
            tags = tags.transpose(0, 1)
        if mask is None:
            mask = torch.ones(emissions.shape[:2], dtype=torch.float)
        scores = self._compute_scores(emissions, tags, mask=mask)
        partition = self._compute_log_partition(emissions,
mask=mask)
        return torch.sum(scores - partition)
```

To‘g‘ri yechim – bu *NLL yo‘qotish* funksiyasi bo‘lib, oddiy *log_ehtimollik* funksiyasidan oldin minus belgisini

qo‘yamiz. *Log_ehtimollik* funksiyasining o‘zi boshlang‘ich natijalarni baholash, *log* ketma-ketligi hisoblash usulini aniqlaydi:

```
input = [['lorem', 'ipsum 'dolor', 'sit', 'amet'],
 ['another', 'sentence', 'here', '<pad>', '<pad>']]
mask = [[1, 1, 1, 1, 1],
[1, 1, 1, 0, 0]]
```

Ifoda surat qismini hisoblash: natijalar

Biz *log* ni *exp* ga nisbatan qo‘llaganimiz sababli, bu surat ko‘rinish hisoblanadi. ya’ni har bir vaqt oralig‘ida ko‘rsatkichlar va o‘zgarishlar yig‘indisi hisoblanadi.

```
def _compute_scores(self, emissions, tags, mask):
    Args:
        emissions (torch.Tensor): (batch_size, seq_len, nb_labels)
        tags (Torch.LongTensor): (batch_size, seq_len)
        mask (Torch.FloatTensor): (batch_size, seq_len)
    Returns:
        torch.Tensor: Scores for each batch.
            Shape of (batch_size,)
    """
    batch_size, seq_length = tags.shape
    scores = torch.zeros(batch_size)
    # save first and last tags to be used later
    first_tags = tags[:, 0]
    last_valid_idx = mask.int().sum(1) - 1
    last_tags = tags.gather(1,
last_valid_idx.unsqueeze(1)).squeeze()
    # add the transition from BOS to the first tags for each batch
    t_scores = self.transitions[self.BOS_TAG_ID, first_tags]
    # add the [unary] emission scores for the first tags for each
batch
```

```
        # for all batches, the first word, see the correspondent
emissions
        # for the first tags (which is a list of ids):
        # emissions[:, 0, [tag_1, tag_2, ..., tag_nblabels]]
        e_scores = emissions[:, 0].gather(1,
first_tags.unsqueeze(1)).squeeze()
        # the scores for a word is just the sum of both scores
        scores += e_scores + t_scores
        # now lets do this for each remaining word
        for i in range(1, seq_length):
            # we could: iterate over batches, check if we reached a mask
symbol
            # and stop the iteration, but vecotrizing is faster due to gpu,
            # so instead we perform an element-wise multiplication
            is_valid = mask[:, i]
            previous_tags = tags[:, i - 1]
            current_tags = tags[:, i]
            # calculate emission and transition scores as we did before
            e_scores = emissions[:, i].gather(1,
current_tags.unsqueeze(1)).squeeze()
            t_scores = self.transitions[previous_tags, current_tags]
            # apply the mask
            e_scores = e_scores * is_valid
            t_scores = t_scores * is_valid
            scores += e_scores + t_scores
        # add the transition from the end tag to the EOS tag for each
batch
        scores += self.transitions[last_tags, self.EOS_TAG_ID]
        return scores
```

Ushbu kodni tushunish uchun to‘plam ichidagi barcha holatlar uchun amallar bir xil deb qabul qilish kerak. Demak, har bir to‘plam

uchun birinchi so‘z tegi tags[:, 0] yuklatiladi. Xuddi shunday vaqt qadamlari bo‘yicha yig‘indini hisoblab, uzunlikni aniqlaymiz, bu esa oldingi misol bo‘yicha [5, 3] qiymatda bo‘lishi mumkin. Amalda faktorial ko‘rsatkichli ko‘rinishda formoy (batch_size,) bo‘lishi mumkin. Masalan, 28 satrga e’tibor qaratsak:

emissions[:, 0].gather(1, irst_tags.unsqueeze(1)).squeeze()

1.Eng avval boshlang‘ich vaqt oralig‘idan barcha ketma-ketliklarni emissions[:, 0] olamiz, bunda tenzor (batch_size, nb_labels) ko‘rinishda qaytariladi.

2.So‘ngra LongTensor first_tags da joylashgan barcha ustun (dim=1) qiymatlarini olamiz, ular (batch_size,) ko‘rinishda bo‘ladi. Emissions ko‘rinishi 2D-matritsa bo‘lganligi sababli, biz so‘nggi o‘lchamni first_tags shunday olishimiz kerakki, natija (batch_size, 1): first_tags.unsqueeze (1) ko‘rinishda bo‘lsin.

3.Keyingi qadam, ikki ko‘rinish ham bir xilligini inobatga olib, gather funksiyasini qo‘llagan holda first_tags tarkibidan belgilangan o‘lchamda tanlovlarni olish mumkin: emissions [:, 0].tanlash (1, first_tags.unsqueeze(1))

4.Natijada (batch_size, 1) ko‘rinishdagi matritsaga ega bo‘lamiz va uni qisqartirgan holda 1D LongTensor ga erishamiz.

Bu sodda protsedura kodlarda berilgan o‘lchamda ko‘rsatkichlarni guruhlash uchun qo‘llaniladi.

Bu kod borasida so‘nggi izoh sifatida shuni aytish lozimki, to‘ldirish simvoli bilan bog‘liq ko‘rsatkichlarni inobatga olmagan holda quyidagi amalni bajarish kerak bo‘ladi: ikki vektor elementlarini o‘zaro ko‘paytirgan holda 0 gacha yetkazamiz va vaqt oralig‘i o‘tish vazifasini bajaradi.

Taqsimlash funksiyasini hisoblash: to‘g‘ri algoritm.

Demak, yetarlicha ballar hisoblangandan so‘ng, maxraj qismiga e’tiborni qaratamiz. Taqsimlash funksiyasini samarali hisoblash uchun to‘g‘ri algoritmni qo‘llaymiz. Quyida uni *log* sohada hisoblashni qisqacha ko‘rib o‘tamiz.

To‘g‘ri algoritmning kodli ko‘rinishini ko‘rib chiqamiz:

1) har bir y'_2 qiymatlarini initsiallashtirish:

$$\propto_1 (y'_2) = \sum_{y'_1} \exp\left(U(x_i, y'_1) + T(y'_1, y'_2)\right) \quad (47)$$

2) barcha y'_{k+1} qiymatlar uchun k=2 dan ℓ-1 gacha hisoblash (log-space):

$$\log(\propto_k (y'_{k+1})) = \log \sum_{y'_k} \exp\left(U(x_k, y'_k) + T(y_k, y_{k+1}) + log(\propto_{k-1} (y'_k))\right) \quad (48)$$

3) eng oxiri:

$$Z(X)\log \sum_{y'_l} \exp\left(U(x_l, y'_l) + log(\propto_{k-1} (y'_k))\right) (49)$$

Shuni ta’kidlash lozimki, 2-qadamda exp ketma-ketliklar yig‘indisini olamiz. Joriy hisoblashda y'_k qiymati juda katta bo‘lsa, eksponenta juda yirik songacha oshishi mumkin. Eng avval ketma-ketlik oxiridan olib boramiz va to‘lib ketish muammosini oldini olamiz, bu amal barqarorligini quyidagicha amalga oshiramiz:

$$\log \sum_k \exp\left((z_k) = \max(z) + log \sum_k exp\,(z_k - max(z))\right) \quad (50)$$

Chap tomondagi ifoda o‘ng tomondagiga tengligini quyidag isbotlaydi:

_ = log sum(exp(zk))
= log sum(exp(zk-c) * exp(c))
= log exp(c) + log sum(exp(zk-c))
= c + log sum(exp(zk-c))

PyTorch algoritmi yordamidagi kod keltiriladi:

```
    def _compute_log_partition(self, emissions, mask):
        """Compute the partition function in log-space using the
forward-algorithm.
        Args:
            emissions (torch.Tensor): (batch_size, seq_len, nb_labels)
            mask (Torch.FloatTensor): (batch_size, seq_len)
        Returns:
            torch.Tensor: the partition scores for each batch.
                Shape of (batch_size,)
        """
        batch_size, seq_length, nb_labels = emissions.shape
        # in the first iteration, BOS will have all the scores
        alphas = self.transitions[self.BOS_TAG_ID, :].unsqueeze(0) +
 emissions[:, 0]
        for i in range(1, seq_length):
            alpha_t = []
            for tag in range(nb_labels):
                # get the emission for the current tag
                e_scores = emissions[:, i, tag]
                # broadcast emission to all labels
                # since it will be the same for all previous tags
                # (bs, nb_labels)
                e_scores = e_scores.unsqueeze(1)
                # transitions from something to our tag
                t_scores = self.transitions[:, tag]
                # broadcast the transition scores to all batches
                # (bs, nb_labels)
                t_scores = t_scores.unsqueeze(0)
                # combine current scores with previous alphas
                # since alphas are in log space (see logsumexp below),
                # we add them instead of multiplying
                scores = e_scores + t_scores + alphas
```

```
            # add the new alphas for the current tag
            alpha_t.append(torch.logsumexp(scores, dim=1))
        # create a torch matrix from alpha_t
        # (bs, nb_labels)
        new_alphas = torch.stack(alpha_t).t()
        # set alphas if the mask is valid, otherwise keep the current
values
        is_valid = mask[:, i].unsqueeze(-1)
        alphas = is_valid * new_alphas + (1 - is_valid) * alphas
    # add the scores for the final transition
    last_transition = self.transitions[:, self.EOS_TAG_ID]
    end_scores = alphas + last_transition.unsqueeze(0)
    # return a *log* of sums of exps
    return torch.logsumexp(end_scores, dim=1)
```

Yuqorida keltirilgan kod baholash ko‘rsatkichini aniqlash amaliga juda o‘xshash bo‘lib, aslida esa, biz ularni oldingi teratsiyasiga qarab yig‘ib boramiz. Faqat bitta satr biz uchun yangilik:

* alphas = is_valid * new_alphas + (1 — is_valid) * alphas: bu satrdagi kodda biz alpha joriy qiymatini to o‘tish qismiga yetmaguncha yangisiga o‘zgartiramiz, o‘tish pozitsiyasiga yetganda o‘z holicha saqlaymiz. Buni quyidagi misolda i=1 vaqt oralig‘ida ishlash ko‘rinishida keltirilgan:

```
>>> mask
tensor([[1., 0., 0.],
    [1., 1., 0.],
    [1., 1., 1.]])

>>> alphas
tensor([[-0.7389, -0.6433, -0.0571, -0.3587, -2.1117],
    [ 1.0372,  1.8366, -0.9350, -1.2656, -0.5815],
    [ 0.1011,  0.7373,  0.0929, -0.8695,  0.7016]])
```

```
>>> new_alphas
tensor([[11.1889, 10.6471, 11.0028, 11.0248, 11.0909],
        [10.3975, 11.0104,  8.5674, 10.2359, 13.9150],
        [10.1440,  9.9298, 11.3141, 10.1534, 10.3397]])
        >>> is_valid = mask[:, 1].unsqueeze(-1)
>>> is_valid
tensor([[0.],
        [1.],
        [1.]])
        >>> is_valid * new_alphas + (1 — is_valid) * alphas
tensor([[-0.7389, -0.6433, -0.0571, -0.3587, -2.1117],
        [10.3975, 11.0104,  8.5674, 10.2359, 13.9150],
        [10.1440,  9.9298, 11.3141, 10.1534, 10.3397]])
```

Biz 2-va 3-ketma-ketlikni yangiladik, biroq 1- ni o‘zgarmas qoldirdik, sababi timestep i=1 da biz o‘tish pozitsiyasiga erishdik.

Shuni ta’kidlash lozimki, logsumexp qo‘llanilganda log-muhitga o‘tiladi, shuning uchun baholar to‘plamiga alphani qo‘shib qo‘yishning o‘zi yetarli. Va so‘ngida yana bir logsumexp amalini olib, oxirgi qiymatlarni qaytarish imkonini olishimiz mumkin.

Eng samarali ko‘rsatkich ketma-ketligini aniqlash

Taqsimlash funksiyasini aniqlagandan so‘ng, teskari algoritmni hisoblasak, ya’ni ketma-ketlikning teskari kesishishi aniqlanganlik natijasida $P(y_k \mid X)$ ni har k qadamda maksimallashtirish ko‘rsatkichini aniqlash mumkin. Agar CRF taqsimot to‘g‘ri deb qabul qilingan holat uchun optimal yechim ifodasini quyidagicha keltirish mumkin:

$$P(y_k|X) = \frac{\exp\left(U(x_k, y_k) + \log\left(\alpha_{k-1}(y_k)\right) + \log(y_k(\beta_{k+1}))\right)}{\sum_{y'_k} \exp\left(U(x_k, y'_k) + \log\left(\alpha_{k-1}(y'_k)\right) + \log\left(\beta_{k-1}(y'_k)\right)\right.}$$

(51)

Bunda α baholash to‘g‘ri algoritmda aniqlanadi, β-baholash esa teskari algoritmdan. Shuning uchun eng samarali y* ketma-ketliklarini aniqlash uchun har bir qadamda argmax hisoblanadi:

$$y^* = \underset{y_{k1}, y_2, \cdots y_l}{argmax} P(y_k)X \qquad (52)$$

```
    def decode(self, emissions, mask=None):
        """Find the most probable sequence of labels given the
emissions using
        the Viterbi algorithm.
        Args:
            emissions (torch.Tensor): Sequence of emissions for each
label.
                Shape (batch_size, seq_len, nb_labels) if batch_first is
True,
                (seq_len, batch_size, nb_labels) otherwise.
            mask (torch.FloatTensor, optional): Tensor representing
valid positions.
                If None, all positions are considered valid.
                Shape (batch_size, seq_len) if batch_first is True,
                (seq_len, batch_size) otherwise.
        Returns:
            torch.Tensor: the viterbi score for the for each batch.
                Shape of (batch_size,)
            list of lists: the best viterbi sequence of labels for each batch.
        """
        if mask is None:
            mask = torch.ones(emissions.shape[:2], dtype=torch.float)
        scores, sequences = self._viterbi_decode(emissions, mask)
        return scores, sequences
    def _viterbi_decode(self, emissions, mask):
        """Compute the viterbi algorithm to find the most probable
sequence of labels
        given a sequence of emissions.
        Args:
```

```
            emissions (torch.Tensor): (batch_size, seq_len, nb_labels)
            mask (Torch.FloatTensor): (batch_size, seq_len)
        Returns:
            torch.Tensor: the viterbi score for the for each batch.
                Shape of (batch_size,)
            list of lists of ints: the best viterbi sequence of labels for
each batch
        """
        batch_size, seq_length, nb_labels = emissions.shape
        # in the first iteration, BOS will have all the scores and then,
the max
        alphas = self.transitions[self.BOS_TAG_ID, :].unsqueeze(0) +
emissions[:, 0]
        backpointers = []
        for i in range(1, seq_length):
            alpha_t = []
            backpointers_t = []
            for tag in range(nb_labels):
                # get the emission for the current tag and broadcast to all
 labels
                e_scores = emissions[:, i, tag]
                e_scores = e_scores.unsqueeze(1)
                # transitions from something to our tag and broadcast to
 all batches
                t_scores = self.transitions[:, tag]
                t_scores = t_scores.unsqueeze(0)
                # combine current scores with previous alphas
                scores = e_scores + t_scores + alphas
                # so far is exactly like the forward algorithm,
                # but now, instead of calculating the logsumexp,
                # we will find the highest score and the tag associated
 with it
```

```
                max_score, max_score_tag = torch.max(scores, dim=-1)
                # add the max score for the current tag
                alpha_t.append(max_score)
                # add the max_score_tag for our list of backpointers
                backpointers_t.append(max_score_tag)
            # create a torch matrix from alpha_t
            # (bs, nb_labels)
            new_alphas = torch.stack(alpha_t).t()
            # set alphas if the mask is valid, otherwise keep the current
values
            is_valid = mask[:, i].unsqueeze(-1)
            alphas = is_valid * new_alphas + (1 - is_valid) * alphas
            # append the new backpointers
            backpointers.append(backpointers_t)
        # add the scores for the final transition
        last_transition = self.transitions[:, self.EOS_TAG_ID]
        end_scores = alphas + last_transition.unsqueeze(0)
        # get the final most probable score and the final most probable
tag
        max_final_scores, max_final_tags = torch.max(end_scores,
dim=1)
```

Viterbi algoritmi

Demak, eng ehtimolli ko‘rsatkichlar ketma-ketligini aniqlash uchun teskari algoritmni hisoblash lozim bo‘lmaydi, chunki to‘g‘ri algoritm har bir vaqt qadami davomida maksimal ballni kuzatishning o‘zi yetarlidir. So‘ngra tugaganidan keyin max (argmax) amallari bo‘yicha teskari yo‘nalishda ballarni maksimallashtiruvchi ketma-ketlikni deshifrlash mumkin bo‘ladi. Quyidagi kod shuni amalga oshiradi:

```
def decode(self, emissions, mask=None):
```

```
        """Find the most probable sequence of labels given the
emissions using
        the Viterbi algorithm.
        Args:
 emissions (torch.Tensor): Sequence of emissions for each label.
Shape (batch_size, seq_len, nb_labels) if batch_first is True,
 (seq_len, batch_size, nb_labels) otherwise.
mask (torch.FloatTensor, optional): Tensor representing valid positions.
If None, all positions are considered valid.
Shape (batch_size, seq_len) if batch_first is True,
 (seq_len, batch_size) otherwise.
   Returns:
torch.Tensor: the viterbi score for the for each batch.
Shape of (batch_size,)
list of lists: the best viterbi sequence of labels for each batch.
    """
    if mask is None:
       mask = torch.ones(emissions.shape[:2], dtype=torch.float)
    scores, sequences = self._viterbi_decode(emissions, mask)
    return scores, sequences
  def _viterbi_decode(self, emissions, mask):
    """Compute the viterbi algorithm to find the most probable sequence
  of labels
    given a sequence of emissions.
    Args:
       emissions (torch.Tensor): (batch_size, seq_len, nb_labels)
       mask (Torch.FloatTensor): (batch_size, seq_len)
    Returns:
       torch.Tensor: the viterbi score for the for each batch.
                Shape of (batch_size,)
            list of lists of ints: the best viterbi sequence of labels for each
  batch
```

```
        """
        batch_size, seq_length, nb_labels = emissions.shape
        # in the first iteration, BOS will have all the scores and then, the
max
        alphas = self.transitions[self.BOS_TAG_ID, :].unsqueeze(0) +
emissions[:, 0]
        backpointers = []
        for i in range(1, seq_length):
            alpha_t = []
            backpointers_t = []
            for tag in range(nb_labels):
                # get the emission for the current tag and broadcast to all
labels
                e_scores = emissions[:, i, tag]
                e_scores = e_scores.unsqueeze(1)
                # transitions from something to our tag and broadcast to all
batches
                t_scores = self.transitions[:, tag]
                t_scores = t_scores.unsqueeze(0)
                # combine current scores with previous alphas
                scores = e_scores + t_scores + alphas
                # so far is exactly like the forward algorithm,
                # but now, instead of calculating the logsumexp,
                # we will find the highest score and the tag associated with
it
                max_score, max_score_tag = torch.max(scores, dim=-1)
                # add the max score for the current tag
                alpha_t.append(max_score)
                # add the max_score_tag for our list of backpointers
                backpointers_t.append(max_score_tag)
            # create a torch matrix from alpha_t
            # (bs, nb_labels)
```

```
            new_alphas = torch.stack(alpha_t).t()
            # set alphas if the mask is valid, otherwise keep the current
values
            is_valid = mask[:, i].unsqueeze(-1)
            alphas = is_valid * new_alphas + (1 - is_valid) * alphas
            # append the new backpointers
            backpointers.append(backpointers_t)
        # add the scores for the final transition
        last_transition = self.transitions[:, self.EOS_TAG_ID]
        end_scores = alphas + last_transition.unsqueeze(0)
        # get the final most probable score and the final most probable
tag
        max_final_scores, max_final_tags = torch.max(end_scores,
dim=1)
```

Bu algoritm Viterbi algoritmi nomi bilan ma'lum. Bu log_partition funksiyasida qo'llagan to'g'ri algoritmga juda o'xshash, biroq barcha ketma-ketliklar uchun doimiy ballar o'rniga, maksimal ball va ularni maksimallashtiruvchi ko'rsatkichlar mavjud. Boshqacha qilib aytganda, faktorial logsumexp o'rniga faktorial maksni, ya'ni maksimal va argmaxni qaytaruvchi amalni qo'lladik.

Bularning barchasi so'nggi ko'rsatkichlarni olib, teskari ketma-ketlikka o'tish orqali i "argmax"ni aniqlash hisoblanadi. Aytib o'tilganlar quyidagi ko'rinish kodiga ega bo'ladi:

```
        # find the best sequence of labels for each sample in the batch
    best_sequences = []
            emission_lengths = mask.int().sum(dim=1)
            for i in range(batch_size):
                # recover the original sentence length for the i-th sample in
    the batch
                sample_length = emission_lengths[i].item()
```

```
            # recover the max tag for the last timestep
            sample_final_tag = max_final_tags[i].item()
            # limit the backpointers until the last but one
            # since the last corresponds to the sample_final_tag
            sample_backpointers = backpointers[: sample_length - 1]
            # follow the backpointers to build the sequence of labels
            sample_path = self._find_best_path(i, sample_final_tag,
sample_backpointers)
            # add this path to the list of best sequences
            best_sequences.append(sample_path)
        return max_final_scores, best_sequences
    def _find_best_path(self, sample_id, best_tag, backpointers):
        """Auxiliary function to find the best path sequence for a
specific sample.
            Args:
                sample_id (int): sample index in the range [0, batch_size)
                best_tag (int): tag which maximizes the final score
                backpointers (list of lists of tensors): list of pointers with
                shape (seq_len_i-1, nb_labels, batch_size) where
seq_len_i
                represents the length of the ith sample in the batch
            Returns:
                list of ints: a list of tag indexes representing the bast path
        """
        # add the final best_tag to our best path
        best_path = [best_tag]
        # traverse the backpointers in backwards
        for backpointers_t in reversed(backpointers):
            # recover the best_tag at this timestep
            best_tag = backpointers_t[best_tag][sample_id].item()
            # append to the beginning of the list so we don't need to
reverse it later
```

```
        best_path.insert(0, best_tag)
    return best_path
```

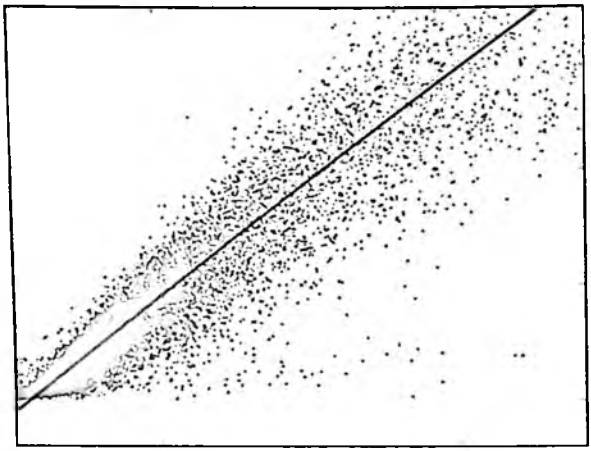
E'tibor qiling, har bir namuna uchun backtrace takrorlab, har bir vaqt qadami uchun ko'rsatkich qo'yilgan holda best_pathni boshidan maksimallashtiradi. Natijada, birinchi element birinchi ko'rsatkichga mos keluvchi ro'yxat hosil qilinadi.

Barcha namunalar uchun find_best_path amalini bajarish bilan ishni tugatamiz.

Xulosa

Agar CRF modelini ishlab chiqarishda qo'llamoqchi bo'lsangiz, uning tekshirishdan o'tkazilgan va samarali amalga oshirilgan ko'rinishini, masalan, pytorch paketini yoki allennlp kutubxonasi taklif qilgan modelini qo'llash tavsiya qilinadi.

2.4. Regression algoritmlar



2.15-rasm. Regression algoritmlar.

Eng kichik kvadratlarning doimiy regressiyasi (OLSR) – chiziqli regressiya usuli bo'lib, model yaratish asosida noma'lum parametrlarn baholanishidir, bunda kuzatilayotgan va bashorat qilingan

berilganlar o'rtasida xatolik kvadratlari summasini minimallashtiradi (kuzatilayotgan qiymatlar va baholanilayotgan qiymatlar).

Chiziqli regressiya – real qiymatlarni baholashda qo'llaniladi (uylar narxi, qo'ng'iroqlar soni, savdo umumiy hajmi va h.k.), uzluksiz o'zgaruvchi asosida amalga oshiriladi.

Logistik regressiya – diskret qiymatlarni baholash uchun qo'llaniladi (ikkilik qiymatlar, masalan, 0/1, ha /yo'q, rost/ yolg'on), erkini o'zgaruvchilar asosida amalga oshiriladi.

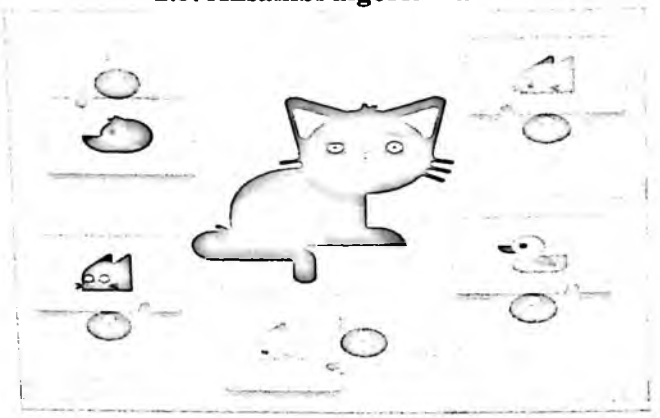
Qadamli regressiya – toki funksiyalar bo'yicha optimal ballga erishmaguncha modelga birma-bir funksiyalarni qo'shib boradi. Qadamli tanlash to'g'ri va teskari yo'nalishda birma-bir bajarilib, o'zgaruvchilarni kiritish va o'chirish amallari o'zgaruvchilarning barqaror to'plamiga erishilguncha ketma-ket bajariladi.

Adaptiv regressiyaning ko'p omilli splayni (MARS) – regressiyaning moslashuvchan usuli bo'lib, nochiziq munosabatlarni va o'zaro bog'lanishlarni izlaydi, bu esa o'z navbatida bashorat aniqlik ko'rsatkichini maksimallashtiradi. Ushbu algoritmlar nochiziq hisoblanib, modelni nochiziq shablonlarga moslashtirishni talab etmaydi.

Tarqoq diagrammani lokal baholash orqali tekislash (LOESS) – ikki o'zgaruvchi orasida tekis chiziqni tanlash yoki bashorat qilingan to'rttagacha o'zgaruvchilar bo'yicha tekis sirtni tanlash usuli. Bunda agar berilganlar chiziqli taqsimotga ega bo'lmasa, regressiya usulini qo'llashingiz mumkinligini bildiradi. Qo'llaydigan regressiya lokal-belgilangan regressiya deb nomlanadi. Bog'langan va bog'liqsiz o'zgaruvchilar o'rtasida nochiziq munosabat bo'lgan holatlarda ham LOESSni qo'llash mumkin. Hozirgi kunga kelib ko'pgina algoritmlar (masalan, to'g'ri bog'lanishga ega klassik neyron tarmoq, tayanch vektorlar mashinalari, yaqin qo'shni berilganlar algoritmlari va h.k.) global o'rganish tizimini hosil qiladi. Bunda ular global yo'qotish funksiyalari (masalan, summa kvadrati xatoligi)ni qisqartirishda qo'llaniladi. Lokal o'rganish tizimlari o'rganish global muammosini bir

nechta soddaroq muammolarga bo‘lib chiqadi. Global usulning kamchiliklaridan biri bu parametrning hech qanday qiymati yetarlicha yaqinlashish ko‘rsatkichini ta’minlamaydi. Biroq LOESS ni qo‘llash – global funksiyaga yaqinlashish ko‘rsatkichi alternativasini hosil qiladi.

2.5. Ansambl algoritmlar



2.16-rasm. Ansambl algoritmi

Ansambl usuli – o‘rganish algoritmlari bo‘lib, klassifikatorlar to‘plamini yaratadi, so‘ngra yangi berilganlarni bashorat ehtimolligi bo‘yicha tasniflaydi. Ansamblning boshlang‘ich usuli Bayes o‘rtacha usuli bo‘lib, oxiridagi algoritmlari esa chiquvchi ma’lumotlar xatoliklarini to‘g‘rilash, summalash va optimallashtirish ishlarini o‘z ichiga oladi.

Boosting algoritmlar sinfi bo‘lib, zaif o‘quvchilarni kuchli o‘quvchilar uchun stimullashtiradi (ya’ni aniq tasnif asosida to‘g‘rilovchi klassifikator, yangi berilganlar uchun taxminlar o‘rniga aniq ko‘rsatmalarni belgilaydi). Ansamblning bu usuli, joriy o‘rganish algoritmining barcha modelini bashorat qilishda aniqlikni oshirish imkonini beradi. Bu usul daraxt ketma-ketligi (tasodifiy tanlov) uchun mos kelib, har bir bosqichdagi maqsad bu oldingi daraxtdagi aniq xatolik topish hisoblanadi. Eng avval oldindan taxmin qilish va o‘rganish

jarayonidagi dispersiya bartaraf etish uchun qo'llaniladi. Odatda bir nechta baholash ko'rsatkichi asosida umumiy bashorat qilish imkonini berib, bitta baholash ko'rsatkichiga nisbatan ishonchlilikni oshiradi (bir nechta zaif yoki o'rta hol prediktorlarni birlashtirib bitta kuchli prediktor yaratish imkonini beradi).

Bootstrapped Aggregation (Bagging) – odatda yechim daraxtida dispersiya ko'rsatkichini kamaytirish hollarda qo'llaniladi. Asosiy g'oya bu o'rganish tanlovidan tasodifiy tanlangan berilganlar asosida to'plamlarni yaratish bo'lib, har bir to'plam yechimlar daraxtini o'rganish uchun qo'llaniladi. Natijada turli modellar ansambliga ega bo'lamiz. Turli yechim daraxtlari bo'yicha bashoratlar o'rtacha qiymati qo'llanilib, bitta yechim daraxtiga nisbatan ishonchliroq hisoblanadi.

AdaBoost – qisqa yechim daraxti bilan birga qo'llanilib, birinchi daraxt yaratilgandan so'ng, har bir o'rganish ekzemplyarda yechim daraxti unumdorligi ko'rsatkichi aniqlab olinadi va u asosida keyingi yaratiladigan yechim daraxti har bir o'rganish ekzemplyariga ta'sir darajasi aniqlanadi. Bashorat qilinishi mushkul bo'lgan berilganlar og'irlik ko'rsakichi katta, oson bashorat qilinuvchi hollar esa kichik og'irlik ko'rsatkichiga ega bo'ladi. Modellar birin-ketin yaratilib, ularning har biri og'irlik ko'rsakichlari koeffitsiyentlarni o'rganish ekzemplyar uchun yangilaydi, bu ko'rsatkichlar ketma-ketlikdagi keyingi daraxt o'rganish jarayoniga ta'sir o'tkazadi. Barcha yechim daraxtlari yaratilgandan so'ng, yangi berilganlar uchun bashoratlar qilinib, har bir daraxning unumdorligi o'rganish jarayonidagi ko'rsatkich bo'yicha baholanadi.

Yig'ilgan, umumlashtirilgan (aralashtirish) – konvolyutsiya, aralashtirish va umumlashtirish turli nomli bir xil jarayon hisoblanadi. Bu protseduralar bashorat aniqligini oshirish uchun mo'ljallangan bo'lib, bir necha mashinali o'rganish modelini birlashtirish yoki kombinatsiyalash orqali amalga oshiriladi. Mantiqan bu yangi model o'rganish jarayonidan o'tuvchi ansamblli algoritmlar bo'lib, ikki va

undan ortiq modellar yoki berilganlar to‘plami bashoratini umumlashtiradi.

Gradient Boosting Machines (GBM) – stimullashtirish asosida kengaytirish usuli. Gradiyentni oshirish = gradiyent ko‘rsatkichi + kuchaytirish. Kuchaytirilgan algoritm bo‘lib, katta hajmdagi berilganlar bilan ishlashda qo‘llanilib, aniq bashorat qilish uchun qo‘llaniladi.

Stimullashtirilgan gradiyentli regressiyaga ega daraxtlar (GBRT) – tasnif va regressiyaning moslashuvchan parametrsiz usuli. Bashoratning sust modellari ansambli ko‘rinishdagi modelni, odatda yechim daraxtlarini yaratadi. GBRT odatda modelni bosqichma-bosqich yaratib, ularni umumlashtiradi va tasodifiy differensiallashgan yo‘qotish funksiyasini optimallashtirish imkonini beradi.

Random Forest – yechim daraxtlari va paketlarda kengaytmalar algoritmi ansambli. E’tibor bering, yechimlar daraxti to‘plami «o‘rmon» deb nomlanadi. Yana bir qadamni bajarishni talab etib, bunda berilganlarning tasodifiy to‘plami bilan birga obyektlarni ham tasodifiy tanlaydi, ya’ni daraxt yaratilishida barcha obyektlar qo‘llanilmaydi. Yangi obyektni atributlar asosida tasniflash uchun har bir daraxt tasnifni aniqlaydi va bu daraxt shu sinfga “ovoz beradi” deb aytiladi. “O‘rmon” eng ko‘p “ovoz” olgan tasnifni tanlaydi.

Yandex CatBoost texnologiyasi

Bugungi kunda Yandex Open Sourceda o‘zining CatBoost kutubxonasini yaratdi, bu kompaniyaning mashinali o‘rganish sohasidagi ko‘p yillik tajribasini hisobga olgan holda ishlab chiqilgan. Uning yordami bilan siz modellarni turli-tuman ma’lumotlarga, shu jumladan raqamlar shaklida (masalan, bulutlar turlari yoki tovarlar toifasi) tasavvur qilish qiyin bo‘lgan modellarga samarali o‘rgatishingiz mumkin. Manba kodi, hujjatlar, benchmarklar va kerakli vositalar allaqachon GitHub da Apache 2.0. litsenziyasi ostida chop etilgan.

CatBoost – bu gradiyent Boostingga asoslangan yangi mashinali o‘rganish usuli. Ranjirlash, bashorat va tavsiyalar qurish muammolarni hal qilish uchun Yandexsga joriy etiladi. Bundan tashqari, u allaqachon

Yevropa yadroviy tadqiqotlar tashkiloti va Yandex Data Factory sanoati mijozlari bilan hamkorlik doirasida qo'llanilmoqda. Xo'sh, CatBoost boshqa ochiq analoglardan qanday farq qiladi?

"Mashinali o'rganish" atamasi 50-yillarda paydo bo'ldi. Bu atama kompyuterga insonga osonlik bilan berilgan muammolarni hal qilishni o'rgatishga urinishni anglatadi, ammo ularni hal qilish yo'lini rasmiylashtirish qiyin. Mashinali o'rganish natijasida kompyuter aniq ko'rsatilmaydigan xatti-harakatlarni namoyish qilishi mumkin. Hozirgi kunda ko'pchiligimiz buni bilmagan holda ko'p marotaba mashinali o'rganish asosidagi samarali yutuqlarga duch kelamiz. Ijtimoiy tarmoqlarda kuzatuvli bog'langan ma'lumotlar to'plamini yaratish, onlayn-do'konlarda "o'xshash tovarlar" ro'yxatlari, banklarda kreditlar berish va pul aylanmasi hisobotini, inson moyillik darajasini aniqlashda ishlatiladi. Mashinali o'rganish texnologiyalari fotosuratlarda shaxslarni qidirishni amalga oshiradi. Ikkinchidan, neyron tarmoqlari hayotda ishlatiladi va izlanishlar olib borilmoqda, bu esa har qanday murakkablikdagi muammolarni hal qilish uchun noto'g'ri fikr bo'lishi mumkin.

Neyron tarmoqlari yoki gradiyent Boosting

Aslida, mashinali o'rganish ko'rinishlari juda ko'p bo'lib, turli xil usullar qo'llaniladi va neyrotarmoqlari ulardan faqat bittasidir. Buning tasdig'i Kaggle platformasida turli tanlovlarda turli usullar raqobatlashgan, biroq juda ko'p hollarda gradiyent boosting g'alaba qozongan.

Neyron tarmoqlari muayyan vazifalarni mukammal hal qiladi, masalan, bir xil ma'lumotlar bilan ishlash kerak bo'lganda, yoki tasvir, ovoz, matndan iborat ma'lumotlar bilan ishlaganda samarali hisoblanadi. Yandexda ular bizga qidiruv so'rovlarini yaxshiroq tushunishga, internetda shunga o'xshash rasmlarni qidirishga, navigatorda ovozingizni tan olishga va yana ko'p narsalarga yordam beradi. Lekin bu mashinali o'rganish uchun barcha vazifalar emas. Faqat neyron tarmoqlar tomonidan hal etilmaydigan jiddiy muammolar mavjud – ular

gradiyent boostingga muhtoj. Bu usul juda ko‘p ma’lumotlar mavjud bo‘lgan va ularning tuzilishi turli bo‘lgan hollarda samarali hisoblanadi.

Misol uchun, ko‘plab omillar (harorat, namlik, radar ma’lumotlari, foydalanuvchi kuzatuvlari va boshqalar) hisobga olinadigan aniq ob-havo bashoratiga muhtoj bo‘lsangiz yoki qidiruv natijalarini sifat jihatidan tartibga solish kerak bo‘lganligi, bu o‘z vaqtida Yandexni o‘z mashinali o‘rganish usulini ishlab chiqishga undagan.

Matriksnet

Birinchi qidiruv tizimlaridan biri bo‘lib, hozirgi tizimlar kabi emas, balki soddaroq edi. Aslida, birinchi ko‘rinishdagi izlash tizimida so‘zlar asosida saytlardan izlash tizimi juda oz bo‘lib, ular orasida hech qanday raqobat mavjud bo‘lmagan. Keyin sahifalar ko‘payib ketib, ular reytingga muhtoj bo‘lib qoldi. Turli xil asoratlar – so‘zlarning chastotasi, tf-idf hisobga olindi. Keyin har qanday mavzuda juda ko‘p sahifalar bor edi, birinchi muhim yutuq — ular havolalarni hisobga olishni boshladilar.

Ko‘p o‘tmay, Internet tijorat jihatdan muhim bo‘lib qoldi va o‘sha paytda mavjud bo‘lgan oddiy algoritmlarni aldashga urinayotgan ko‘plab yolg‘onchilar paydo bo‘ldi. Va ikkinchi muhim yutuq bor edi – qidiruv tizimlari qaysi sahifalar yaxshi ekanini tushunish uchun foydalanuvchilarning xatti-harakatlari haqida o‘z bilimlaridan foydalana boshladilar.

O‘n yil muqaddam, inson hujjatlarni tartibga solish masalasini hal etdi. Har qanday so‘rov juda ko‘p: yuz minglab, ko‘pincha millionlab natijalar berishi mumkin. Ularning aksariyati qiziq emas, foydasiz, faqat so‘rov so‘zlarini tasodifan eslatib turadi yoki odatda spam hisoblanadi. Sizning so‘rovingizga javob berish uchun siz topilgan barcha natijalardan o‘nta eng yaxshisini darhol tanlashingiz kerak bo‘ladi. Buni esa maqbul sifat bilan amalga oshiradigan dasturni yozish dasturchi uchun murakkab vazifa bo‘lib qoldi. Keyingi o‘tish jarayoni sodir bo‘ldi — qidiruv tizimlarida mashinali o‘rganishdan faol foydalana boshlandi.

Yandex 2009-yilda gradiyent Boostingga asoslangan Matriksnet usulini joriy qildi. Foydalanuvchilarning umumiy fikri va "insonlarning donoligi" bu tizimda reyting yordam berishi mumkin. Veb-saytlar va odamlarning xatti-harakati haqidagi ma'lumotlar turli xil omillarga aylanadi, ularning har biri reyting formulasini yaratish uchun matritsadan foydalaniladi. Aslida, tartiblash formulasi endi mashinali ko'rinishda yoziladi. Aytgancha, ayrim omillar sifatida biz neyron tarmoqlarning natijalari (masalan, o'tgan yili aytilgan Palek algoritmi)ni ishlatamiz.

Matritsaning muhim xususiyati shundaki, u qayta o'qishga chidamli. Bu sizga ko'plab reyting ko'rsatkichlarini hisobga olish imkonini beradi va ayni paytda mashinali mavjud bo'lmagan naqshlarni topishida kam vaqt sarflab ma'lumot olish imkonini beradi. Boshqa mashinali o'rganish usullari kamroq omillarga bog'liq holda bog'lanishlarni aniqlashga yoki ko'proq namuna, ya'ni o'rgatishga asos bo'lgan ma'lumotlarni talab qilishi mumkin bo'ladi.

Matritsaning yana bir muhim xususiyati shundaki, reyting formulasi juda tor so'rovlar sinflari uchun alohida sozlanishi mumkin. Misol uchun, faqat musiqa talablari bo'yicha qidiruv sifatini samaralashtirish. Shu bilan birga, so'rovlarning qolgan sinflari reytingni tushirmaydi.

Gradient Boostingga asoslangan deyarli har qanday zamonaviy usul raqamlar bilan ishlaydi. Kirishda musiqa janrlari yoki ranglar majmuyi ko'rinishda mavjud bo'lsa ham, bu ma'lumotlar hali ham raqamlar tilida tasvirlangan bo'lishi kerak. Bu ularning mohiyatini noto'g'ri talqin qilishga va modelning aniqligi pasayishiga olib keladi.

Buni do'konda mahsulot katalogi bilan oddiy misolda namoyish etamiz. Mahsulotlar bir-biridan ancha ko'p farqlanib, ular orasida tartiblangan bog'lanish mavjud emas, bu ularni tartibga solish va har bir mahsulotga mazmunli raqamni belgilash imkonini beradi. Shuning uchun, bu holatda, har bir mahsulotga oddiy ID beriladi. Bu raqamlarning tartibi hech narsa demaydi, lekin algoritm bu tartibni ishlatadi va undan noto'g'ri xulosalar chiqaradi.

Mashinali o'rganish bilan shug'ullanadigan tajribali mutaxassis aniq xususiyatlarni raqamga aylantirishning yanada intellektual usulini taklif

qilishi mumkin, ammo dastlabki ishlov berish axborotning bir qismini yo'qotishga va yakuniy yechim sifatining yomonlashishiga olib keladi.

Shuning uchun, mashinali o'rganish faqat raqamlar bilan emas, balki to'g'ridan to'g'ri toifalar bilan ishlashni o'rganishi muhim edi, ular tomonidan mustaqil ravishda aniqlanadi. CatBoost ham raqamli belgilar ham kategoriyalar bilan bir xil darajada yaxshi ishlashi uchun mo'ljallangan. Shu bilan birga, muqobil yechimlarga nisbatan, alternativ ma'lumotlar bilan ishlash yuqori sifat natija beradi. U bank sohasidan boshlab, ishlab chiqarishga qadar turli sohalarda qo'llanilishi mumkin. Bu texnologiya nomi Categorical Boosting so'zidan kelib chiqqan.

Benchmarklar

Kutubxonaning nazariy farqlari haqida uzoq vaqt gapirish mumkin, ammo amalda bir marta ko'rsatish yaxshiroqdir. Aniqlik uchun CatBoost kutubxonasining ishini XGBoost, LightGBM i H20 ochiq analoglari bilan ochiq axborot markazlarida solishtirildi.

CatBoost amalda

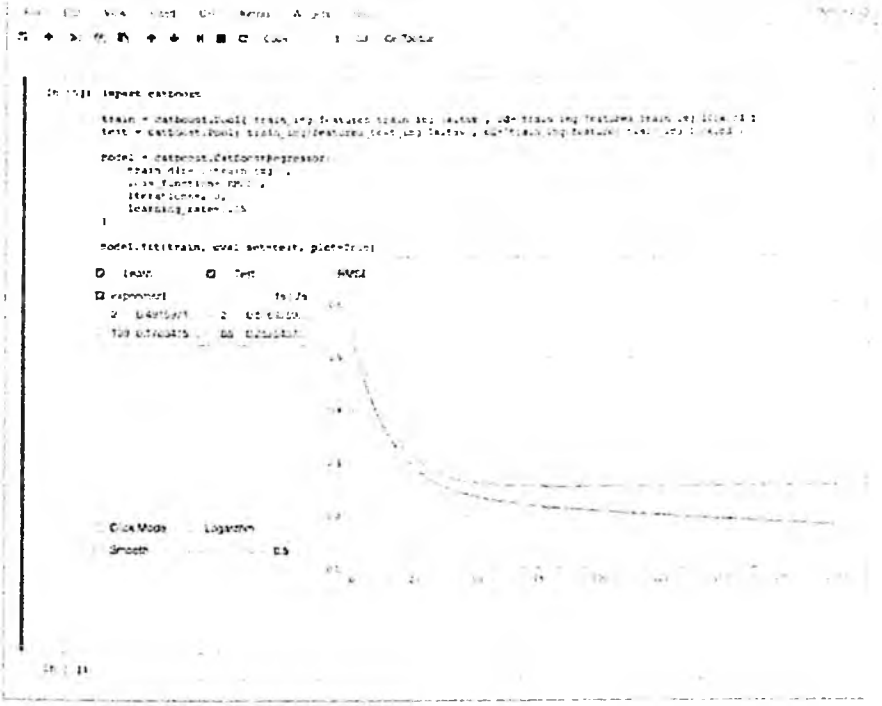
Yangi usul allaqachon Yandex xizmatlarida sinovdan o'tgan. U qidiruv natijalarini yaxshilash, Yandex Dzen tavsiyalarini ko'rsatish uchun ishlatilgan va Metium texnologiyasidagi ob-havo bashoratini hisoblash uchun qo'llanilib, barcha holatlarda Matriksnetdan yaxshiroq natija ko'rsatdi. Kelajakda CatBoost boshqa sohalarda ham tatbiq etiladi.

CatBoost Yevropa yadroviy tadqiqotlar tashkiloti bilan hamkorlik doirasida ham foydalanishga muvaffaq bo'ldi. Katta adron kollayderda LHCb detektori mavjud bo'lib, u og'ir kvarklarning o'zaro ta'sirlarida materiya va anti-materiya asimmetriyasini o'rganish uchun ishlatiladi. Eksperimentda qayd etilgan turli zarralarni aniq kuzatib borish uchun detektorda bir nechta maxsus qismlar mavjud bo'lib, ularning har bir zarrachalarning maxsus xususiyatlarini aniqlaydi. Bu holda eng qiyin vazifa detektorning turli qismlaridan zarracha borasida ma'lumotlarni eng aniq, umumiy ma'lumot ko'rinishda birlashtirishdir. Bu yerda mashinali o'rganish ayni muddao bo'lib qoladi. CatBoost ma'lumotlarini birlashtirish uchun olimlar yakuniy yechimning sifat ko'rsatkichlarini

yaxshilashga muvaffaq bo'lishdi. CatBoost natijalari boshqa usullar yordamida olingan natijalardan yaxshiroq ekanligi isbotlangan.

CatBoostdan qanday foydalanish kerak?

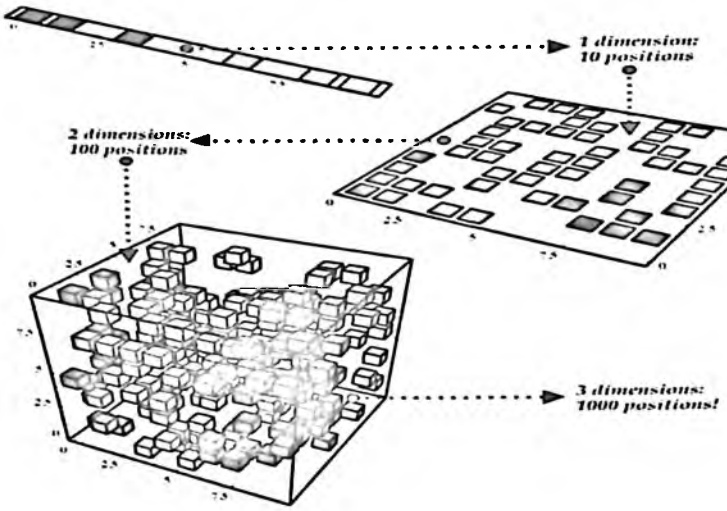
CatBoost bilan ishlash uchun uni kompyuteringizga o'rnatish kifoya. Kutubxona Linux, Windows va macOS operatsion tizimlarini qo'llab-quvvatlaydi va Python R. dasturlash tillarida mavjud. Shu bilan birga Yandex vizuallashtirish dasturi CatBoost Viewerni ishlab chiqdi, u o'z navbatida grafiklar asosida o'rganish jarayonini kuzatish imkonini beradi.



2.17-rasm. CatBoost o'rganish jarayonini grafida kuzatish

CatBoost – OPEN SOURSE ko'rinishda mavjud bo'lgan birinchi rus tilidagi mashinali o'rganish texnologiyasidir. Kutubxonani ochiq kodli ko'rinishda joylashtirishga sabab mashinali o'rganish sohasi rivojiga yo'l ochib berish hisoblanadi.

2.6. O‘lchamni kichraytirish algoritmlari



2.18-rasm. O‘lchamni kichraytirish algoritmi

·O‘lchamlarni kamaytirish – o‘lchamlarni kamaytirish algoritmi bu boshqa algoritmlar yechimlar daraxti, tasodifiy “O‘rmon”, PCA va faktorli tahlil asosidagi o‘rganilayotgan tasodifiy o‘lchamlarni kamaytirish imkonini beradi.

Asosiy komponentlar tahlili (PCA) – statistik protsedura bo‘lib, korrelyatsiyalangan kuzatuvlar to‘plamini chiziqli korrelyatsiya qilinmagan asosiy komponentlar deb nomlangan o‘zgaruvchilar to‘plamiga o‘girish. Birinchisi eng muhim komponent, so‘ng ikkinchi, uchinchi va h.k. davom etadi.

Erkin komponentli tahlil (ICA) – tasodifiy o‘lchamlar, kattaliklar yoki signallar to‘plami negizida yotgan yashirin omillarni aniqlashning statistik usuli.

Asosiy komponentlar regressiyasi (PCR) – multikollinearlik mavjud berilganlar to‘plamining regressiyasini tahlil qilish usuli. Asosiy g‘oya bu asosiy komponentlarni aniqlash, so‘ngra bu komponentlarning

ba'zilarini chiziqli regressiya modelida prediktor ko'rinishda qo'llash, eng kichik kvadratlar usulini qo'llash asosida amalga oshiriladi.

Eng kichik kvadratlarning qisman regressiyasi (PLSR) – PCR odatda prediktor o'zgaruvchilar o'zgarishini izohlovchi komponentni yaratadi, bunda natija qiymatlari inobatga olinmaydi. PLSR esa natija qiymatlarini inobatga olgan holda kam sonli komponentlar asosida model yaratishga olib keladi.

Sammon Mapping algoritm, ko'p o'lchovli muhitni kam o'lchovli muhit ko'rinishda ifodalab, ko'p o'lchovli fazo nuqtalari o'rtasidagi masofalar strukturasini saqlagan holda kam o'lchovli fazoda ifodalaydi. Bu esa ma'lum bir berilganlar to'plamini nochiziq ifodalashning optimal ko'rinishini tanlash imkonini beradi. Garchi PCA dispersiyani oddiygina maksimallashtirsa-da, ba'zan boshqa bir o'lchamni maksimallashtirish talab etilishi mumkin, bunda o'zgartirishdan keyin ham murakkab struktura saqlanib qoladi. Shunday o'lchamlar turli bo'lib, ulardan biri Sammon xaritasi. Bu asosan berilganlarni boshlang'ich tahlilida qo'llashga to'g'ri keladi.

Ko'p o'lchovli masshtablash (MDS) – berilganlar to'plamining alohida hollar uchun o'xshashlik darajasini vizuallashtirish vositasi.

Projection Pursuit – statistik usul ko'rinishi bo'lib, ko'p o'lchovli berilganlar ichida eng "ahamiyatli"larni izlash amalini o'z ichiga oladi. Ko'pincha to'g'ri taqsimotdan chetlashgan holatlar eng ahamiyatli hisoblanadi.

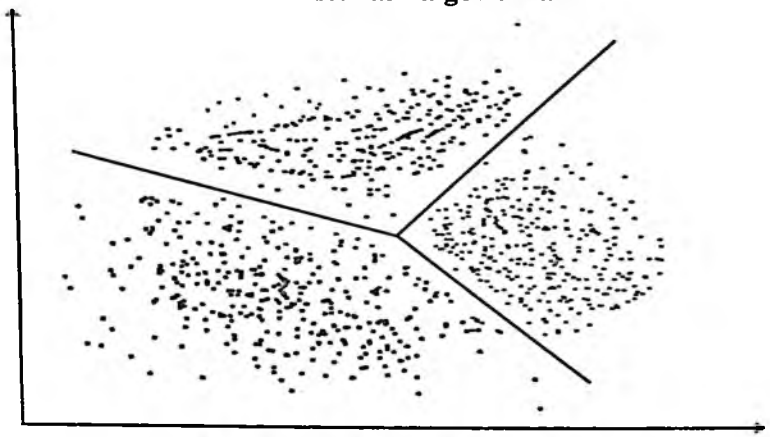
Chiziqli diskriminant tahlil (LDA) – agar tasnif algoritmi kerak bo'lsa, eng avval logistik regressiyadan boshlash lozim. Biroq, LR odatda sinflar tasnifining ikkita muammosi bilan cheklanib qoladi. Agar masala ikkitadan ortiq sinflarga ega bo'lsa, LDA qo'llash lozim. LDA o'lchamni kichraytirish algoritmi kabi ham ishlaydi, bunda o'lchamlar sonini original holatdan G-1 gacha kichraytiradi, bunda G – sinflar soni.

Aralashma diskriminant tahlili (MDA) – chiziqli diskriminant tahlilning kengaytirilgan holati. Tasnifning boshqariladigan usuli bo'lib, aralash modellarga asoslangan.

Kvadrat diskriminant tahlil (QDA). Chiziqli diskriminant tahlil faqat chiziqli chegaralarni o'rganishi mumkin. Kvadrat diskriminant tahlil esa kvadratik chegaralarni o'rganadi (shuning uchun moslashuvchan). Biroq, LDAdan farqli ravishda, QDAda har bir sinf kovariatsiyasi bir xil degan taxmin mavjud emas.

Moslashuvchan diskriminant tahlil (FDA) – klassifikatsion model bo'lib, chiziqli regressiya modellari muvofiqlashuviga asoslangan. Bunda, chiziqli taqsimot uchun berilganlar aniq ko'rsatishda va diskriminant sath uchun bir nechta regressiya splaynlari mavjudligini ta'minlashda optimal baholash qo'llaniladi.

2.7. Klasterlash algoritmlari



2.19-rasm. Klasterlash algoritmi

K-vositalar – KNNdan farqlanadigan algoritm bo'lib (ularni adashtirish kerak emas), K bu X berilganlarni K klasterga taqsimlash lozimligini bildiradi. Bunda har bir berilgan nuqtasi unga yaqin bo'lgan klasterga bog'lanadi. Asosiy g'oya bu barcha klasterlar uchun klasterlarda mavjud masofalar kvadratining yig'indisini qisqartirish hisoblanadi.

Bir kanalli klasterlash – iyerarxik klasterlash usullarining biri hisoblandi. U klasterlarni pastdan yuqoriga qarab guruhlashga asoslangan. Bitta bog'lanishli klasterlashda ikkita klasterlarning o'xshashligi – bu klasterlarda o'xshash elementlarining ko'pligini bildiradi.

K medianlar – K algoritmning bir varianti hisoblanadi. Asosiy g'oya har bir klaster uchun o'rtacha qiymatni hisoblash (uning sentroidini aniqlash) o'rniga, medianani hisoblaymiz.

Maksimal kutilish (EM) – K o'rtacha qiymatiga analog hisoblanib, klasterlar ehtimollikni ifodalaydigan og'irlik ko'rsatkichiga ega klasterlar berilganlarni o'zlashtiradi. Ustunlik tomoni shundaki, model generatsiyalanib boradi, chunki har bir model uchun ehtimolliklar taqsimoti aniqlanadi.

Iyerarxik klasterlash berilganlarni klasterlar taqsimlash bir bosqichda amalga oshirilmaydi. Uning o'rniga bir nechta bosqichlar qo'llanilib, barcha berilganlarga ega bitta klasterdan boshlab, bitta berilganga ega N klaster bilan tugaydi.

Noravshan klasterlash – klasterlash ko'rinishida har bir berilgan bittadan ortiq klasterga bog'liq.

DBSCAN (qo'shimchali zichlikka asoslangan fazoviy klasterlash) – yuqori ko'rsatkichli zichlikka ega klasterlarni past ko'rsatkichli zichlikka ega klasterlardan ajratish. DBSCAN uchun ikkita parametr talab etiladi: ikki nuqta orasidagi eng minimal masofa va tekislikni shakllantirishda eng kichik zichlik ko'rsatkichi. Bu bir-biriga yaqin (odatda evklid masofa) minimal sonli nuqtalarni guruhlash degan ma'noni anglatadi.

OPTIKA (Klaster strukturasini identifikatsiyalash uchun nuqtalarni tartiblash) – asosiy g'oya DBSCAN ga o'xshash bo'lib, biroq unda mavjud kamchilikni bartaraf etadi: bu turli zichlikka ega berilganlar ichida muhim klasterlarni aniqlash masalasidir.

Manfiy bo'lmagan matritsali faktorizatsiya (NMF) – chiziqli algebraik model bo'lib, katta o'lchamli vektorlarni kichik o'lchamli

vektorlar orqali ifodalash ko'rinishiga olib keladi. Asosiy komponentlar tahlili (PCA)ga o'xshash, ammo NMF vektorlar manfiy bo'lmasligini talab etadi. Ularni kichik o'lchamda joylashtirganda koeffitsiyentlar ham manfiy bo'lmasligini talab etadi.

Dirixle latentli taqsimoti (LDA) – ehtimollik modeli ko'rinishi bo'lib, korpusda mavjud tematikani aniqlash. Masalan, hujjatda to'plangan so'zlar kuzatuv obyekti bo'lsa, klasterlarni belgilash uchun ikkita ehtimollik ko'rsatkichi lozim bo'ladi: P (so'z| mavzu), berilgan mavzularda so'zning mavjudlik ehtimolligi. R (mavzular | hujjatlar), berilgan hujjatda mavzularning mavjudlik ehtimolligi. Bu qiymatlar boshlang'ich belgilangan tasodifiy qiymatlar asosida aniqlanadi. So'ngra ular har bir hujjatdagi har bir so'z uchun takrorlanib boriladi, natijada ular mavzusi aniqlanadi.

Gauss aralash modeli (GMM). Asosiy g'oya – bu ko'p o'lchovli gauss ehtimolliklar taqsimoti aralashmasini aniqlash hisoblanib, ular yordamida barcha ko'rinishdagi boshlang'ich berilganlarni model-lashtirib bo'lsin. Shu bilan birga k-means kabi klasterlarni aniqlashda ham qo'llash mumkin. Maqsad sodda bo'lib, berilganlarni aniq ifodalovchi gaussli parametrlarni topish hisoblanadi.

2.8. K-oraliqda qo'shni xususiyatlarni qo'llash usulining tasniflash masalalarni yechishda qo'llanilishi

K-oraliqda qo'shni xususiyatlarni qo'llash (angl.: *k-nearest neighbors method*, *k-NN*) usuli – tasniflash masalalarni yechish usullaridan biri hisoblanadi.

Faraz qilinsinki, aniq tasniflangan obyektlar soni berilgan (ya'ni har bir obyekt mansub bo'lgan sinf ma'lum). Maqsad, yangi kiritilgan obyekt mavjud bo'lgan sinflar ichidan qaysi biriga mansub ekanligini aniqlovchi qoidani ishlab chiqishi kerak (bunda sinflar oldindan aniq hisoblanadi).

k-NN negizida quyidagi tartib amalga oshiriladi: *yangi obyekt o'zining qo'shni obyektlarning ko'pchiligi mansub bo'lgan sinfga mansub deb hisoblanadi.* «Qo'shni» obyekt deb o'rganilayotgan obyektga qaysidir ma'noda yaqin bo'lgan obyektlar tushuniladi.

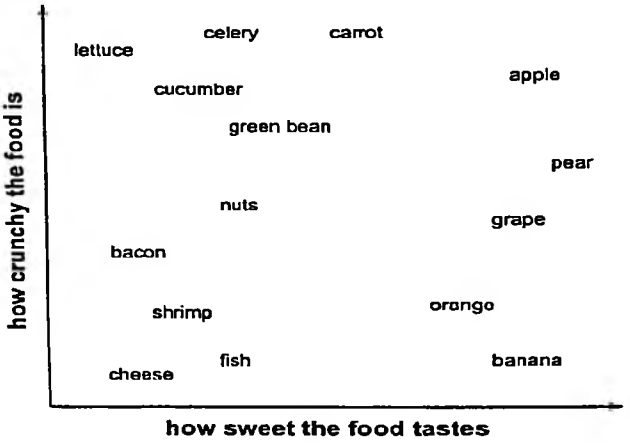
Bunda, obyektlar o'rtasidagi yaqinlik darajasini, ya'ni ular o'rtasidagi «masofani» aniqlab bilish lozim bo'ladi. Biroq bu evklid masofa bo'lishi shart emas. Bunda obyektlar o'lchov ko'rsatkichi, masalan, rangi, shakli, ta'mi, hidi, (insonlar o'rtasida aniqlash bo'lgan holat uchun) qiziqishlari, xulq-atvori va boshqalar bo'lishi mumkin. Tabiiyki, kNN usulini qo'llash uchun ma'lum bir birlik (ya'ni yaqinlik funksiyasi) berilgan bo'lishi lozim.

Ma'lum bir xususiyatlari bilan yaqin bo'lgan obyektlar boshqa xususiyatlari bilan ham yaqin bo'lishi mumkin (ya'ni bitta sinfga mansub bo'lishi mumkin).

k-NN usulini sodda misolda ko'rib chiqamiz.

Mahsulot	**Shirinlik**	**Zichlik**	**Sinfi**
olma	9	8	meva
bekon	1	4	protein
banan	10	1	meva
...	...	...	...

Mahsulot xususiyatlari 10 ballik shkalada baholanmoqda, bu qiymatlar 2 o'lchovli tekislikda koordinata nuqtalari ko'rinishida joylashadi. Ikki xususiyati ikkita koordinata o'qlari bo'yicha belgilanib, mahsulotlar joylashuvi aniqlanadi, natijada quyidagi rasm hosil bo'ladi.



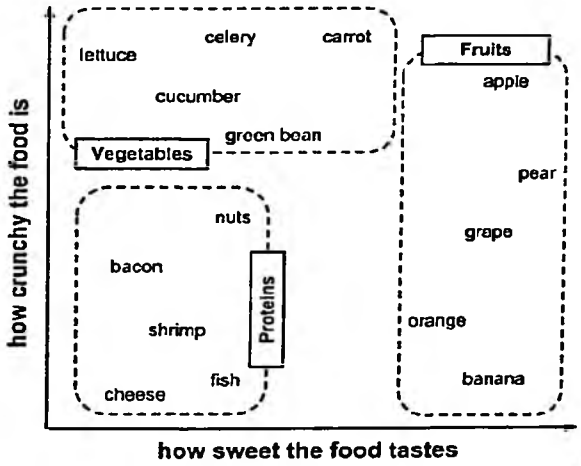
2.20-rasm. So‘zlarning 2 o‘lchovli tekislikda joylashuvi.

Ko‘rib o‘tilayotgan mahsulotlar turi (sinflanishi) ustunlar ko‘rsatkichi bo‘yicha ko‘rsatilgan, grafikda joylashuvi bo‘yicha ular vizual sinflari yaqqol ko‘rinib turadi:

– chap tomon yuqorida joylashgan mahsulotlar sabzavotlar «guruhlashgan» (bodring, sabzi, ko‘kat) – shirin bo‘lmagan, biroq karsillash xususiyatiga ega;

– chap tomon pastda – proteinga boy mahsulotlar (bekon, qisqichbaqa, pishloq, baliq, yong‘oq) – shirin bo‘lmagan, karsillash xususiyatiga ega emas;

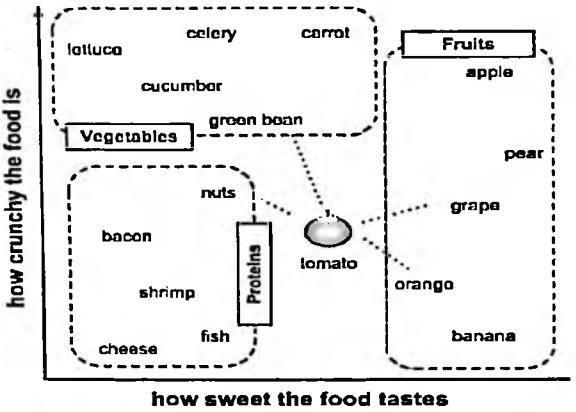
– o‘ng tomondan mevalar «joylashib olgan» (olma, nok, uzum, apelsin, banan) – boshqa sinflanishga nisbatan ular shirin, biroq karsillash ko‘rsatkichi bo‘yicha turli ko‘rinishga ega.



2.21-rasm. Mahsulotlarni guruhlarga taqsimot sxemasi.

№	Mahsulot	Sinfi	Pomidorga yaqinlik darajasi
1.	apelsin	meva	1,4
2.	uzum	meva	2,2
3.	qisqichbaqa	protein	3,5
4.	bekon	protein	3,6
5.	yongok	protein	3,6
6.	pishloq	protein	4,0
7.	dukkaklilar	protein	4,2
8.	bodring	sabzavot	5,9
9.	olma	meva	6,1
10.	sabzi	sabzavot	6,8
11.	ko'kat	sabzavot	7,0
12.	ko'kat - salat	sabzavot	7,2
13.	banan	meva	9,2

Mahsulot uchta sinfni tashkil qilmoqda, endi yangi berilgan mahsulot qaysi sinfga mansubligini aniqlash masalasi tursin.



2.22-rasm. Berilgan so'zning guruhga mansubligini aniqlash.

Pomidor – bu sabzavot yoki meva?

k-NN usuli bo'yicha uni k nchi qo'shni obyektlari sinfiga mansub deb qabul qilinadi. Obyektlar o'rtasidagi masofani *Evklid ko'rsatkichi*, ya'ni obyektlar o'rtasidagi masofani (x_1, y_1) va (x_2, y_2) koordinatalarni $\sqrt{((x_1 - y_1)^2 + (x_2 - y_2)^2}$ orqali ifodalaymiz. Pomidor shirinlik ko'rsatkichi 3, karsillash ko'rsatkichi 7 bo'lsa, olmaga nisbatan yaqinlik darajasi taxminan 6,1; 3,6, 9,2 ga teng bo'ladi. Quyida pomidorga nisbatan mahsulotlarning joylashuvini oshish tartibida keltiramiz:

So'ngra k qiymatni tanlab, qo'shni obyektlar mansub bo'lgan sinfni aniqlash lozim bo'ladi. Chunki k=1 bo'lsa, yaqin qo'shni obyekt – bitta va u apelsin – meva. k=2 bo'lganda apelsin va uzum, har ikkisi meva. k=3 bo'lganda 2 mevaga (apelsin va uzum) va qisqichbaqa (protein)ga ega bo'lamiz. Demak, k-NN usuli yana: «meva» degan javobni beradi. k=4 bo'lganda javob «bir xil ehtimollik bilan meva yoki protein». k=5, k=6, k=7, k=8 bo'lganda «g'olib» protein hisoblanadi. Ushbu jarayonni k oshirgan sayin davom ettirish mumkin. Demak, *k*-

NN usulida olinadigan natija k parametr tanlanmasiga juda ham bog'liq bo'lib qoladi.

Shunday savol tug'iladi: *k-NN usulida olinadigan xato natijalarning qisqarishiga olib keladigan k parametr qiymati qanday tanlanilishi lozim?*

k qiymati juda kichik deb tanlanilsa, sinfi noto'g'ri tanlanilgan obyekt «voz kechish»ga olib keladi, ya'ni noto'g'ri qarorga olib keladi. Agar *k* parametr qiymati oshirilsa, bunday obyektlar yo'qotilishi oldini olish mumkin. Biroq bunda boshqacha xatolik tug'iladi. Buni anglash uchun *k* qiymatini obyektlar soniga *N* teng deb olamiz. Bunda, tabiiyki, ommabop sinf «g'olib» chiqadi, va bunda obyektgacha masofa hech qanday ahamiyatga ega bo'lmaydi. K parametrning optimal qiymati tanlanganidan so'ng «*bias-variance tradeoff*» deb nomlanadi, ya'ni «voz kechish» va «dispersiya» o'rtasida kelishuvga olib keladi. Amaliyotda ko'pincha $k = \lfloor\sqrt{N}\rfloor$ qo'llaniladi, ya'ni bizning misolda *k*=3 va natijada tasnif natijasida, *pomidor – meva* bo'lib qoladi.

Tanlash «to'g'ri»ligiga ishonchingiz komil bo'lsa, *k* ni kichik qiymatda tanlash mumkin. Shu bilan birga «*weighted voting*» nomli usul ham mavjud (ya'ni «o'lchamga ega tanlanma»), o'rganilayotgan obyekt qo'shni obyektlari uzoqda joylashgan obyektlarga nisbatan katta og'irlik ko'rsatkichiga ega. k-NN usulni qo'llashning yana bir omili – bu berilganlarni boshlang'ich tayyorlash.

2.9. K-NN usulni qo'llashdan oldin ma'lumotlarni boshlang'ich tayyorlash

Eslatib o'tamizki, ko'rib o'tilayotgan misolda ikkita xususiyat (shirinlik darajasi va "karsillash" xususiyati) inobatga olinayapti, bir xil shkalada o'lchanilib, qiymatlar 0 dan 10 ko'rsatkichda belgilangan. Amalda turli xususiyatlar turli o'lchov birlikka va ko'rsatkichda o'lchanishi mumkin. Real obyektlar o'rtasida farqni aniqlashda xatoliklarga va muammolarga olib kelishi mumkin, ularni oldini olish

uchun k-NN usulni qo'llashdan oldin *normallashtirish* (*masshtablashtirish*) olib boriladi (angl.: *scaling*).

Normallashtirishning turli xil usullari mavjud. Ko'pincha qo'llaniladigan usullarni ko'rib o'tamiz:

$$x_i \equiv \frac{x_i - x_{min}}{x_{max} - x_{min}}. \quad (53)$$

(1) formula xususiyatlarning absolut qiymatidan nisbiy qiymatlarga o'tishini ko'rsatadi. Yangi o'zgaruvchilar uchun afzallik tomoni bu ularning 0 dan 1 gacha qiymatlarni (yoki foiz olinsa 0 dan 100) qabul qiladi.

Masshtablashtirishning ikkinchi usuli:

$$x_i \equiv \frac{x_i - \bar{x}}{s}, \quad (54)$$

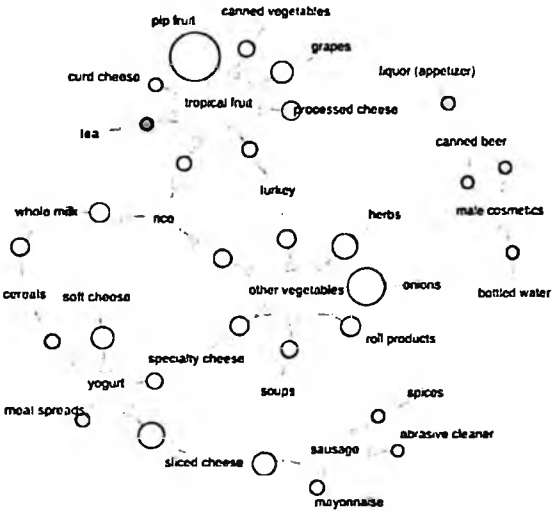
Bunda, $\bar{x}$ – o'rtacha tanlanma (ya'ni . $\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i$), s – tanlangan o'rta kvadratik siljish (ya'ni,

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad . (55)$$

Ma'lumki, agar ξ to'g'ri taqsimotga μ va σ parametrga ega bo'lsa, u holda $\eta \equiv \frac{\xi - \mu}{\sigma}$ to'g'ri taqsimotga ega, biroq uning parametrlari taqsimoti 0 va 1 ga mos ravishda teng hisoblanadi (bunday ifodalar *standart* gauss deb nomlanadi).

Barcha xususiyatlarni ham uning soni bilan ifodalab bo'lmaydi. Buning uchun *dummy coding* ga olib kelinadi. Masalan, «jinsi» nomli xususiyatni erkaklar uchun 1 ayollar uchun 0 bilan ifodalash mumkin **[13]**.

2.10. Assotsiatsiya qoidasi bo‘yicha o‘rganish algoritmlari



2.23-rasm. Assotsiatsiya qoidasi bo‘yicha o‘rganish algoritmi.

Assotsiatsiya qoidalarini o‘rganish – bu tranzaksiya to‘plamini inobatga olib, asosiy maqsad tranzaksiyadagi boshqa elementlar asosida keyingi kiruvchi elementlarni bashorat qilish qoidasini topish.

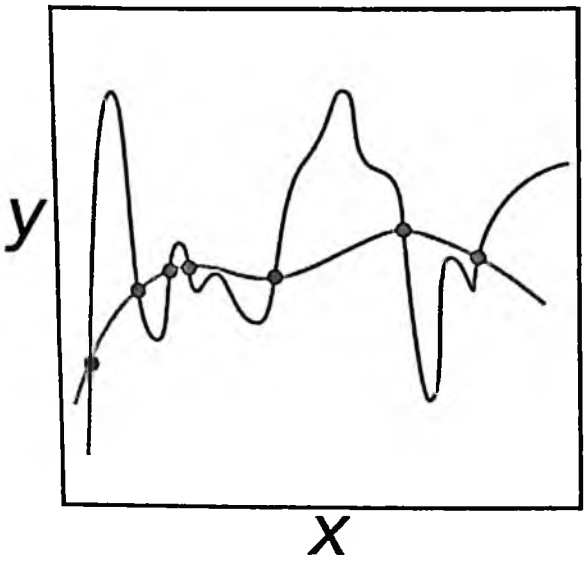
Apriori berilganlarni intellektual tahlil etishda muhim o‘rin egallaydi. Ma’lum bir jismlar (bir yoki bir nechta jismlar to‘plami)ni va ularga xos assotsiatsiya qoidalarini mayning jarayonida qo‘llash samarali hisoblanadi. Odatda, bu algoritm ko‘p sonli tranzaksiyaga ega ma’lumotlar bazasida qo‘llaniladi. Masalan: mahsulotlar mijozlar tomonidan supermarketlarda xarid qilinadi. Apriori algoritmi tizim a’zolari sonini quyidagi prinsipda qisqartiradi: agar elementlar to‘plami tez-tez uchrashsa, demak, uning quyi to‘plami ham tez-tez uchraydi.

Eclat (ekvivalentlik sinfini o‘zgartirish) – Apriori algoritmidan farqi shundaki, izlash amali kenglik bo‘yicha emas, balki chuqurlik bo‘yicha bajariladi. Apriori algoritmida mahsulotga asoslangan element

(1, 2, 3, 3 va h.k. pozitsiyalar) qo'llaniladi. 0 Eclat algoritmida esa tranzaksiya elementlar asosida uzatiladi (savat 100 200 va h.k.).

FP (tez-tez takrorlanadigan holat) – joriy tranzaksiyalar bo'yicha bozordagi xaridni tahlil qilishga yordam beradi. Ma'no jihatidan birgalikda xarid qilinishi mumkin bo'lgan mahsulotlar to'plamini aniqlashga urinadi. FP-Growth Aprioriga nisbatan afzaldir, chunki Apriori tez-tez qo'llaniladigan elementlarni aniqlash uchun tranzaksiya ma'lumotlarini qayta-qayta skanerlash uchun ko'p vaqt talab etadi.

2.11. Tartibga solish algoritmlari



2.24-rasm. Tartibga solish algoritmlari

Tizmalar regressiyasi (L2 tartibga solish) – asosiy g'oya – bu berilganlarning qayta o'rganish muammolarini yechish. Standart chiziqli polinominal regression modeli aynan o'zgaruvchilar xususiyatlarida yuqori darajadagi kolleniarlik (ozod hadlar o'rtasida chiziqli bog'lanishning mavjudligi) bo'lgan hollarda inqirozga uchraydi. Tizmalar regressiyasi o'zgaruvchilarga siljish kvadrat koeffitsiyentini

qo‘shadi. Bunday siljish kvadrat koeffitsiyenti o‘zgaruvchilar yoyilishini ta’minlab model dispersiyasini kamaytiradi. Tizmalar regressiyasida bitta kamchilik mavjud, u yakuniy modelning barcha n funksiyalarini o‘z ichiga oladi.

Absolut kichraytirish va tanlash operatori (LASSO, L1 Regularization) – tizmalar regressiyasidan farqli ravishda faqat katta koeffitsiyentlarni qisqartiradi. Giperparametr θ juda katta bo‘lgan hollar uchun Lasso natijasida ba’zi koeffitsiyentlar ko‘rsatkichi nolga teng bo‘lib qoladi. Demak, Lasso natijasida olinadigan modellar tanlangan o‘zgaruvchilar asosida tuzilib, regression tahlil yordamida olingan modelga nisbatan ularni anglab olish juda oson.

Egiluvchan to‘r Lasso va tizmalar regressiyasining ba’zi xususiyatlarini mujassamlashtirgan. Lasso ko‘pgina funksiyalarni chetlashtirsa, tizmalar regressiyasi funksiyalar ta’sirini qisqartiradi, bu odatda bashorat qilishda ahamiyatga ega bo‘lmagan *y* qiymatlar. Bu algoritm esa bir necha funksiyalar ta’sirini qisqartirsada, biroq barcha funksiyalarni chetlashtirmaydi.

Kichik burchak ostida regressiya (LARS) – to‘g‘ri qadamli regressiya bilan bir xil bo‘lib, har bir bosqichda oldingi natija asosida prediktorni aniqlaydi. Bir xil korrelyatsiyaga ega bir nechta prediktorlar mavjud bo‘lganda bitta prediktor bo‘ylab harakatlanish o‘rniga prediktorlar o‘rtasidagi burchak bo‘yicha harakatlanadi.

Ikkinchi bob bo‘yicha xulosa

Bobda yechimlar daraxti algoritmlari, tasodifiy o‘rmon, r da tasodifiy o‘rmon, bayes algoritmining asosiy nazariya, yo‘qotish funksiyasi mohiyati, regression va ansambl algoritmlari, o‘lchamni kichraytirish algoritmlari, klasterlash algoritmlari, k-oraliqda qo‘shni xususiyatlarni qo‘llash usulining tasniflash masalalarini yechishda qo‘llanilishi mohiyati va mazmuni, k-nn usulni qo‘llashdan oldin ma’lumotlarni boshlang‘ich tayyorlash, assotsiatsiya qoidasi bo‘yicha o‘rganish algoritmlari va tartibga solish algoritmlarining umumiy tasnifi va amaliyotda foydalanish bo‘yicha tavsiyalar ishlab chiqilgan.

XULOSA

Bugungi kunda sun'iy intellekt hayotimizga mustahkam kirib, ko'plab muammolarni hal qilishda yordam beradi. Ilmiy-fantastik filmlarning kelajagini yaqinlashtiradigan sun'iy intellektning eng istiqbolli yo'nalishlaridan biri neyron tarmoqlardir. Zotan, ulardan tijoratda, ayniqsa, marketing ishlarida faol foydalanilmoqda, xavfsizlik va boshqa sohalarda qo'llaniladi. Ushbu sohada olib borilgan tadqiqotlar, masalan, Microsoft va Google kabi eng ilg'or kompaniyalar bilan shug'ullanadi, bu deyarli har kuni bu sohada yangi kashfiyotlar paydo bo'lishiga yordam beradi.

Sun'iy neyron tarmoqlari biologik prinsip asosida qurilgan bo'lib, bir qator taxminlar bilan ishlab, ko'plab bog'lanishlar asosida amallarni bajaradi. Inson miyasi singari, bu tarmoqlar ham o'rganish qobiliyatiga ega. Sun'iy neyron tarmoqlari uchun ta'lim – bu vazifani samarali hal qilish uchun tarmoq arxitekturasi (neyronlar o'rtasidagi aloqalar tuzilishi)ni va sinaptik bog'lanishlar og'irlik ko'rsatkichlari (signallarga ta'sir qiluvchi koeffitsiyentlari)ni sozlash jarayoni tushuniladi. Odatda, neyron tarmoqni o'qitish ba'zi namunalar asosida amalga oshiriladi [1]. Trening davomida tarmoq belgilangan vazifalarni yaxshiroq bajarishga, belgilangan buyruqlarga javob berishga kirishadi.

Quyida eng keng tarqalgan sohalarni ko'rib o'tamiz:

Ma'lumotni topish, tasvirni aniqlash. 2016-yilning kuzida Yandex neyron tarmoqlarga asoslangan yangi Palex qidiruv algoritmini ishga tushirdi, Google analoglari «Kolibri» va RankBrain. Bu algoritmlar aniqroq qidirishga yordam beradi. Palex sahifa sarlavhalarini tahlil qiladi va ularning ma'nosini anglab oladi, tez orada barcha matn shu jarayonni o'tadi [4].

Tasvirni aniqlash – bu turli neyron tarmoqlari tomonidan aniqlangan tasvir asosida qidirish tizimida berilgan so'rov bo'yicha

o‘xshash tasvirlarni izlab topishni amalga oshiradi. Yandex va Googleni eng mashhur qidiruv tizimi deb olish [5] mumkin. Yuklab olish yoki shunga o‘xshash tasvirlar uchun qidiruv vazifasini tanlab, rasmda sichqoncha tugmasini bosish, foydalanuvchi u muvaffaqiyatli ko‘rayotganligi va analoglarini aniqlashga neyron tarmog‘i buyruq beradi, keyingi yangi yuklangan foto tasvirlangan qaysi ko‘rinish ekanligini tasniflaydi va teglar yaratadi. Ammo texnologiya yana bir qadam olg‘a tashladi: millionlab odamlarning fotosuratlari birma-bir ko‘rib o‘tilib, aniqlangan holda qonuniyat hosil qildi va shov-shuv bo‘lgan FindFaceoga o‘xshash insonlarni aniqlab berishi mumkin. 2015-yilda yuzni tanib olish bo‘yicha xalqaro tanlovda loyiha eng yaxshi deb topildi, hatto Googledan tanib olish texnologiyasini ham chetlab o‘tdi. Va 2016-yilda neyron tarmoqlar videotasvirlarda o‘z yuzini yashirgan holda tasvirga olingan ko‘rishlarni o‘rgandilar va ularni aniqlashga erishdilar. Birgina YouTube yozuvidan qayta ishlangan tasvirlar 80–90 foizni, foto muharrirlari yordamida yaxshilab o‘zgartirilgan tasvirlarni tanishish aniqligi 50–75 foizgacha bo‘lgan. Endi odamni yashirin holatda qolib ketganda yuz ko‘rinishni aniqlash jarayoniga muhtojlik qolmadi.

Nutqni aniqlash, tarjima qilish, takrorlash. Inson ovozli ma’lumotini Okey Google qabul qilish va anglash imkoniyatiga ega, ammo Google tomonidan sotib olingan DeepMind neyron tarmog‘i inson nutqini yanada aniqroq imitatsiya qilishni o‘rgandi [6]. Shuni aytish kerakki, hozirgi vaqtda xorijiy so‘zlarni tarjima qilish texnologiyasi neyron tarmoqlari tufayli doimiy ravishda takomillashtirilmoqda. Yaqinda siz chet ellik kishi bilan gaplashadigan tilni bilishingiz shart emas bo‘lib qoladi. Chunki yaqinda ikkita texnologiyani birlashtirgan holda, bemalol to‘g‘ridan to‘g‘ri tarjima qilib yuklanadigan texnologiyadan foylanishingiz mumkin bo‘ladi. Yaqinda Google o‘zining SI har qanday professionaldan ko‘ra lablar harakati bo‘yicha o‘qishni o‘rganganini e’lon qildi. Neyron tarmoq

orqali olingan fotosuratlarda bo'lgani kabi, 5 ming soatlik turli teledasturlar yozuvlari ham o'tkazib yuborilgan, natijada DeepMind so'zlarning bir qismini yutgan holda ham lablar harakati bo'yicha o'qishni o'rgandi. Hozirgi vaqtda DeepMind 30% odamlar orasida professionallardan ko'ra lablar harakati bo'yicha o'qish bilan yaxshi ishlaydi [3]. Bularning barchasi subtitr yaratish va callcentrlarda yordamchilarni ishlatish uchun katta salohiyatga ega.

San'at. Neyron tarmoqlari, masalan, belgilangan parametrlarda bo'lgani kabi, fotosuratni qayta ishlashi mumkin. Masalan, muallif tomonidan ko'rsatilgan reproduksiyaga o'xshash uslubda oddiy rasmga aylantirish yoki eskizni barcha elementlarini chizish orqali san'at mahsuliga aylantirish imkoni mavjud. Shuningdek, neyron tarmoq o'z xohishiga ko'ra yakuniy tasvir uslubini mustaqil ravishda tanlagan holda yaratishi mumkin. Neyron tarmoqlari musiqa yozish, ba'zi oddiy kuylar yaratish, yoki ular musiqa bilan birgalikda unga mos tushuvchi so'zlar qo'yib to'liq albom yozish kabi imkoniyatlar mavjud. Biroq musiqa o'zi odamlar tomonidan yaratilgan, va keyin qo'shma natijada so'zlar qo'yilib asarlar yaratilgan, lekin inson tomonidan yaratilgan asardan butunlay farq qilinishi aniqlangan. Film uchun birinchi treylerni yaratish, badiiy filmni suratga olgan ssenariyni yozish – san'at sohasida endi bularning barchasini inson emas, neyron tarmoqlar amalga oshirishi mumkin.

Ilm-fan sohasi. Neyron tarmoqlari veb-saytlar uchun noyob matnlarni yozadilar, hali professional emas, biroq juda ma'noli, ba'zi SI axborot agentliklari uchun yangiliklar, bundan tashqari, ular ilmiy maqolalar yaratadilar. Tajriba shuni ko'rsatdiki, neyron tarmoq bazasiga bir necha o'nlab yozilgan maqolalar joylashtirilganda, tarmoq ularni tahlil qilib, yangi maqolalar yozdi va ularni bir qator ilmiy jurnallarga ham yubordi, bu yerda ba'zilari hatto omma e'tiboriga havola etildi. Bu haqiqat ushbu jurnallarning

tahrirlovchilarining beparvoligi va tarmoq tomonidan yozilgan maqolalarning yuqori sifati haqida gapirish mumkin.

Agar ilm-fan haqida gapiradigan bo'lsak, unda sun'iy intellekt juda tez rivojlanmoqdi.

Tibbiyotda deyarli har kuni neyron tarmoqlardan foydalanishning yangi kashfiyotlari paydo bo'lmoqda, ko'z qobig'i ko'rinishi asosida kasallikni aniqlash kashfiyotining o'zi bu bir yangilik. Akusher robotlar, bemorlarni parvarish qilishni o'rganish uchun o'zlariga o'xshash bo'lgan ichki ma'lumot almashinish imkoniyati bo'lgan robotlar, inson tanasida yashovchi va kasallikning har qanday boshlanishini bartaraf etadigan nanorobotlar haqida deyarli haqiqat deb hisoblangan. Avtomobilsozlikda haydovchining funksiyasi kuzatuvchining funksiyasiga o'zgartirilgan holda o'z-o'zini o'rgatadigan mashinalar. Dronlar va robotlar yer bo'yicha boshqarishni o'rganish, minimal to'qnashuvlar va har qanday sirt bilan harakat qilishni o'rganishadi. Ilm-fan taraqqiyoti yuqori xavfli hududlarda insonni davolash va inson o'rniga yordam berib, minglab hayotni saqlab qolishga yordam beradi.

Xizmatlar sohasi. Yuqorida aytilganlarni tahlil qilib, inson o'rnini bosadigan sohalarni tushunish uchun yetarli bo'ladi. Zotan, xatlarni o'qish va to'g'ri javobni taklif qilish orqali elektron pochtalarga javob berishga yordam beradigan robotlar ham mavjud. Mijozlarning savollariga javob berishni o'rganadigan onlayn maslahatchilar, birinchi navbatda haqiqiy menejerlarni kuzatib, javob berishga harakat qiladilar. Agar javoblari noto'g'ri bo'lsa, menejerlar javoblarni to'g'rilab, uni hisobga olgan holda o'zgarishlarni amalga oshiradilar. Luka kompaniyasi yana ilg'or natijalarga erishdi, u odamning xatti-harakatlarini kuzatib boradigan va uning elektron nusxasini yaratish, uning xabarlarini o'rganish, boshqa odamlar bilan to'liq muloqot qilish, ma'lumot topish, o'qitish uchun neyron tarmoq yaratdi. Neyro tarmoqlari hozirda foydalanuvchilarni o'rganmoqda va

ma'lum bir iste'molchining ehtiyojiga muvofiq reklama taklif qilmoqda. Kelajakda biz mijozlarni to'liq avtomatlashtirilgan qo'llab-quvvatlash tizimini yo'lga qo'yishni kutmoqdamiz, ijtimoiy tarmoqlar, guruhlar, onlayn-do'konlarning barcha menejerlari sun'iy intellekt bilan boshqariladi, savollarga javob beradi, muammolarni pochta, telefon orqali hal qiladi, bularning barchasi tezkor va sifatli amalga oshiriladi.

Yuqorida aytib o'tilgan, sohalarda neyron tarmoqlarning qo'llanilishi yoki allaqachon ishlatilishining kichik bir qismi bo'lib, ishlab chiqilayotgan yoki rejada bo'lgan neyron tarmoqlar ancha ko'p. 2011- yildan boshlab neyron tarmoqlar asosida SI sohasiga yillik investitsiyalar hajmi 15 barobar oshdi, biroq bu sohada rivojlanayotgan Start-Uplar soniga nazar tashlasangiz, bu o'n minglab va tahlilchilarning tadqiqotlariga ko'ra, yuzlab odamlar bir necha yil ichida yuz milliardlab dollar turadigan investitsiyani ko'zlamoqda. Bugungi kunda faqat bitta yuzni aniqlash bozori 3 milliard dollarga teng va bu neyron tarmoqlarning faqat bitta yo'nalishi. Bunday tez rivojlanish inson hayotining ko'plab sohalarida takomillashib, muntazam ishlarni osonlashtiradi, biroq ayni paytda ko'plab ishlarni qisqartirish xavfi paydo bo'ladi, ba'zan esa butun ish o'rnini to'liq yo'q qilishga olib keladi. Chunki neyron tarmoq uni tezroq, yaxshiroq va arzonroq qiladi. Zamonaviy yondashuvlar vazifalarni bajarish uchun yangi yo'nalishlar, yangi ish vositalariga ega bo'ladi. Unda butun dunyo va hayot o'zgaradi.

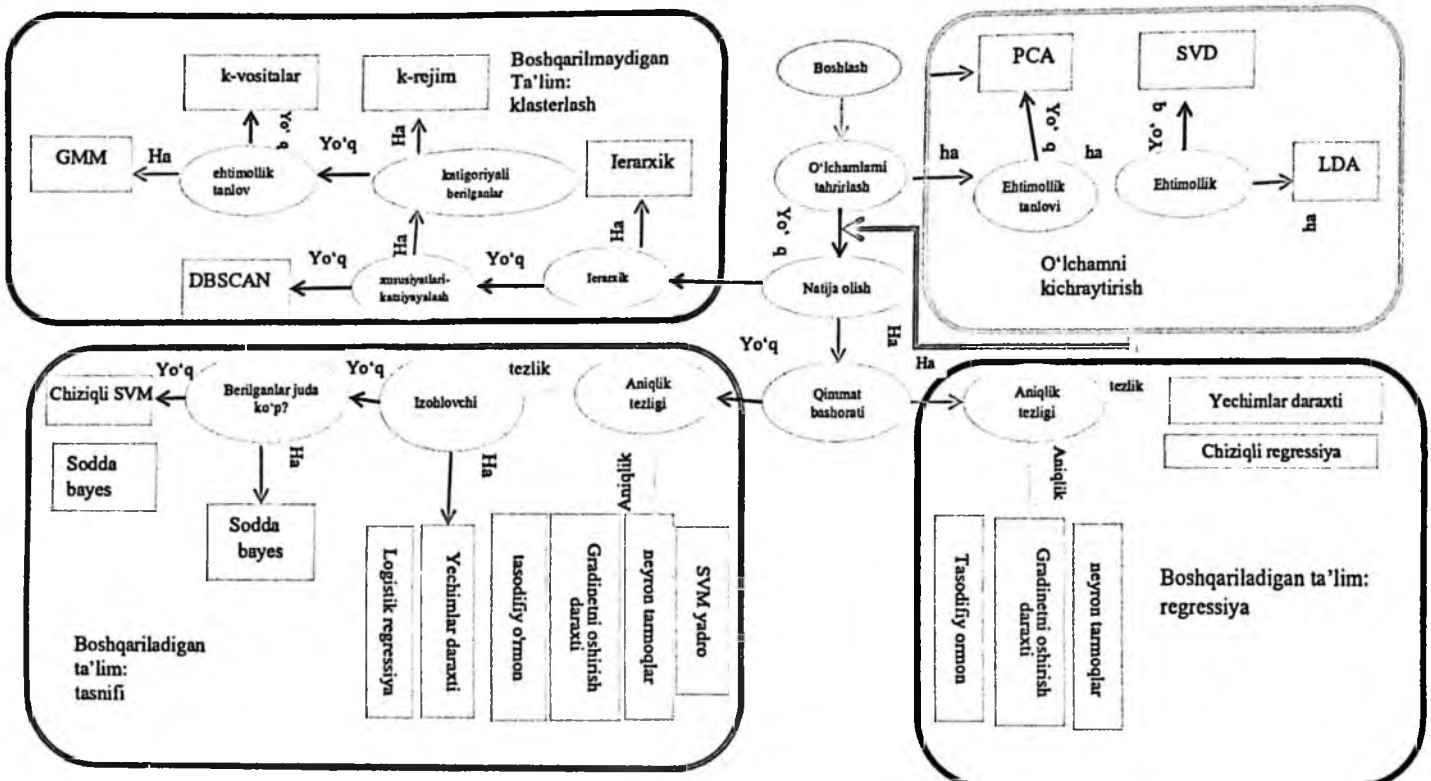
Endi, har qanday mashinali o'rganish algoritmi bilan ishlashni xohlovchi mutaxassislar oldiga "Qaysi algoritmni tanlash?" degan savol qo'yiladi. Savolga javob bir necha omillarga asoslanib olinadi.

Hattoki, yaxshi mutaxassis ham bir nechta algoritmni qo'llab, so'ngra ulardan birini tanlaydi. "Bitta algoritmni tanlash kerak" degan g'oyani olg'a surmaymiz, balki eng avval bir nechta omilga tayanadigan algoritmlarni qo'llab ko'rishni tavsiya beramiz.

Keltirilgan variantlar boshlang'ich tadqiqotchilar va analitiklar uchun mo'ljallangan soddalashtirilgan tavsiyalar berildi.

Keltirilgan algoritmlar, mashinali o'rganish bo'yicha tadqiqotchilar, ekspertlar tavsiyalari va maslahatlari negizida tuzilgan. Turli fikrlarni umumlashtirib, farqlarni ajratib tuzildi. Ba'zan bitta yo'nalish algoritm ko'rinishlari qo'yilgan masala yechimiga juda ham mos tushsa, ba'zan hech biri ham to'g'ri kelmasligi mumkin.

Aniq masalalarni yechishda mashinali o‘rganish algoritmini tanlash variantlari



FOYDALANILGAN ADABIYOTLAR

1. Bayes classifier [Электронный ресурс]. режим доступа: http://dataaspirant.com/naive-bayes-classifier-machine-learning/ - Data dostupa:12.05.2017.

2. Resources for Text, Speech and Language Processing [Elektronnыy resurs].Rejim dostupa: http://www.cs.technion.ac.il/ ~gabr/resources/pointers.html - Data dostupa: 20.01.2017.

3. Investigations on dynamic neural networks [Elektronnыy resurs]. Rejim dostupa: http://people.idsia.ch/~juergen/ SeppHochreiter1991 Thesis

4. AdvisorSchmidhuber. pdf - Data dostupa: 03.02.2017 Khachaturova K.R. Information technology as a means of development of creative abilities of primary school pupils in natural science lessons // Globalnыy nauchnыy potensial. 2015. № 9 (54). S. 111-113.

5. Brett Lantz. Machine Learning with R. Pack Publishing. Birmongham-Mumbai,2013.

6. Brown E.W. Applying Neural Networks to Character Recognition. Northeastern University internal paper.

7. Bishop C. M. Neural Networks for Pattern Recognition. Oxford University Press Inc., 2003.

8. Pinkus A. Approximation theory of the MLP model in neural networks. Acta Numerica, 1999.

9. Haykin S. Neural Networks: A Comprehensive Foundation. NY: Macmillan, 1994.

10. Nigrin A. Neural Networks for Pattern Recognition. Cambridge, MA: The MIT Press, 1993.

11. Zurada J.M. Introduction To Artificial Neural Systems. Boston: PWS Publishing Company, 1992.

12. DARPA Neural Network Study, AFCEA International Press, 1988.

13. http://www.osp.ru/text/302/179978/_p1.html.

14. https://bi.snu.ac.kr/Courses/g-ai09-2/hopfield82.pdf

15. https://www.researchgate.net/profile/Terrence_Sejnowski/publication/242509302_Learning_and_relearning_in_Boltzmann_machines/links/54a4b00f0cf256bf8bb327cc.pdf

16. https://apps.dtic.mil/dtic/tr/fulltext/u2/a620727.pdf

17. https://pdfs.semanticscholar.org/f582/1548720901c89b3b7481f7500d7cd64e99bd.pdf

18. https://papers.nips.cc/paper/3112-efficient-learning-of-sparse-representations-with-an-energy-based-model.pdf

19. https://arxiv.org/pdf/1312.6114v10.pdf

20. https://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf

21. http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

22. https://arxiv.org/pdf/1503.03167v4.pdf

23. https://arxiv.org/pdf/1406.2661v1.pdf

24. https://arxiv.org/pdf/1512.03385v1.pdf

25. https://neuronus.com/theory/nn/955-nejronnye-seti-kokhonena.html

26. https://feisky.xyz/machine-learning/appendix/algorithms.html

27. https://towardsdatascience.com/ml-algorithms-one-sd-%CF%83-74bcb28fafb6

MUNDARIJA

MO‘MINOV BAHODIR BOLTAYEVICH

SUN’IY INTELLEKT

Toshkent – «Innovatsion rivojlanish nashriyot-matbaa uyi» – 2023

Muharrir:	A.Haydarov
Texnik muharrir	M.Tursunov
Musavvir	Sh.Zoxidova
Musahhih	S.Muratova
Kompyuterda sahifalovchi:	M.Zoyirova

E-mail: nashr2019@inbox.ru. Tel.: +99899.920-90-35
№ 3226-275f-3128-7d30-5c28-4094-7907, 10.08.2020.
Bosishga ruxsat etildi 13.10. 2023.
Bichimi 60x84 1/16. «Timez Uz» garniturasi.
Ofset bosma usulida bosildi.
Shartli bosma tabog‘i: 11,5. Nashriyot bosma tabog‘i 10,75.
Tiraji: 30. Buyurtma № 130/51.

«Innovatsion rivojlanish nashriyot-matbaa uyi»
bosmaxonasida chop etildi.
100174, Toshkent sh., Olmazor tumani, Universitet ko‘chasi, 7-uy.