ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტი

სალომე გოგისვანიძე

ღრმა სწავლება

სამაგისტრო პროგრამა ინფორმაციული ტექნოლოგიები

ნაშრომი შესრულებულია ინფორმაციული ტექნოლოგიების მაგისტრის აკადემიური ხარისხის მოსაპოვებლად

ხელმძღვანელი: ასისტენტ-პროფესორი გელა ბესიაშვილი

თბილისი

2019

რეზიუმე

"ყველაზე სევდიანი ის არის, რომ მეცნიერება უფრო სწრაფად იძენს ცოდნას, ვიდრე საზოგადოება სიბრძნეს .“

-ისაკ აზიმოვი

ღრმა სწავლება არის მანქანური სწავლების მეთოდი, რომელიც ასწავლის კომპიუტერებს ადამიანებისთვის ბუნებრივ ქმედებებს. სიღრმისეული შესწავლა არის წამყვანი ტექნოლოგია, რომელიც უკავშირდება მანქანებს მძღოლის გარეშე, საშუალებას აძლევს მათ გაარჩიონ გაჩერების ნიშანი, ფეხით სავალი გზა და ა.შ. ღრმა სწავლება არის ხმის კონტროლის გასაღები ისეთ მოწყობილობებში როგორიცაა ტელეფონები, ტაბლეტები, ტელევიზორები და ა.შ. სიღრმისეულ კვლევას დიდ ყურადღება ეთმობა ბოლო პერიოდში. მისი დახმარებით ისეთი შედეგის მიღწევაა შესაძლებელი, რაც ადრე არ იყო.

ღრმა სწავლებაში კომპიუტერული მოდელი სწავლობს შეასრულოს კლასიფიკაციის ამოცანები უშუალოდ გამოსახულებების, ტექსტისა და ხმისგან.მოდელები იწვრთნებიან დიდი რაოდენობის ეტიკეტირებული მონაცემებისა და ნეირონული ქსელის არქიტექტურის გამოყენებით, რომელიც შეიცავს ბევრ ფენებს.

Abstract

“THE SADDEST ASPECT OF LIFE RIGHT NOW IS THAT SCIENCE GATHERS KNOWLEDGE FASTER THAN SOCIETY GATHERS WISDOM.”

— ISAAC ASIMOV

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It’s achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

სარჩევი:

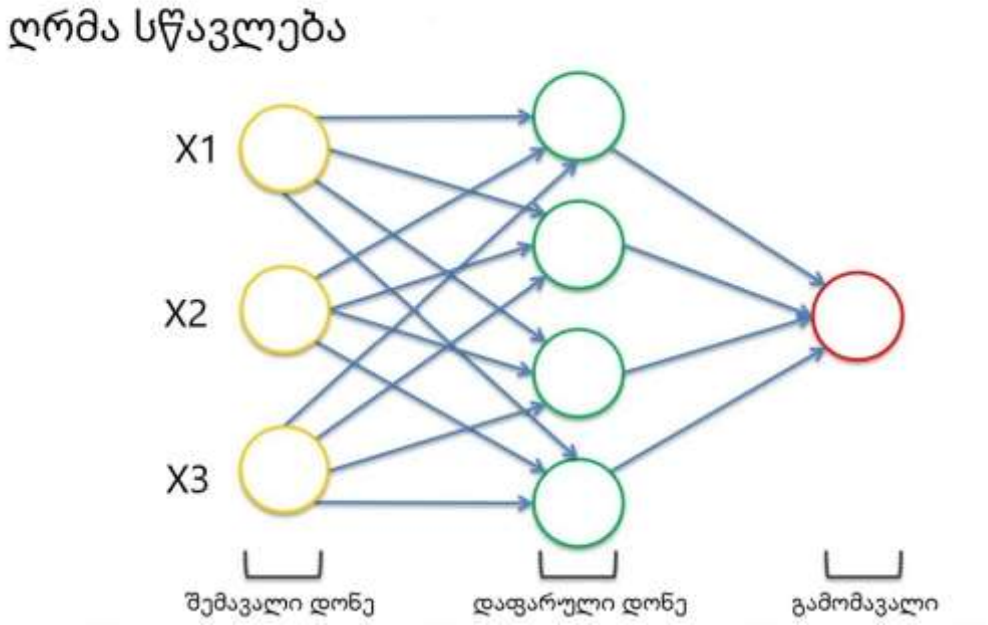
შესავალი

ტრადიციული მანქანური სწავლების მოდელები ძალიან ეფექტურია სტრუქტურული მონაცემების დასამუშავებლად და ფართოდ გამოიყენება საწარმოების მიერ საკრედიტო ქულის მინიჭების, გადინების პროგნოზირების, სამომხმარებლო გამოთვლებისა და სხვა მრავალი საქმიანობისთვის.

ამ მოდელების წარმატება დიდწილად ეფუძნება ფუნქციის განვითარების ფაზის შესრულებას: რაც უფრო დიდხანს ვიმუშაობთ ბიზნესში, მით უფრო დიდ ცოდნას მივიღებთ სტრუქტურირებული მონაცემებისგან და უფრო ეფექტიანი მოდელი გვექნება.

რაც შეეხება უნიკალურ მონაცემებს (გამოსახულებები, ტექსტი, ხმა, ვიდეო), ხელით შექმნილი ფუნქციები დიდ დროს მოითხოვს, მყიფეა და პრაქტიკაში არ არის მასშტაბური. ამიტომაა, რომ ნეირონული ქსელები სულ უფრო და უფრო პოპულარული ხდება რათა ავტომატურად აღმოაჩინონ ისეთი თვისებები, რომლებიც საჭიროა თვისებების აღმოსაჩენად ან დაუმუშავებელი მონაცემების კლასიფიკაციისთვის. ეს ცვლის ხელით შემუშავებულ ფუნქციებს და საშუალებას აძლევს მანქანას, ისწავლოს ფუნქციები და გამოიყენოს ისინი კონკრეტული ამოცანის შესასრულებლად.

გაუმჯობესებული ტექნიკა (გრაფიკული პროცესორები) და პროგრამული უზრუნველყოფა (მოწინავე მოდელები / კვლევები, რომლებიც დაკავშირებულია ხელოვნური ინტელექტისთვის) ასევე ხელს უწყობს ნეირონული ქსელების გამოყენებით მონაცემების ღრმა შესწავლას.



თავი 1

რატომ არის ღრმა სწავლება მნიშვნელოვანი?

ერთი სიტყვით, სიზუსტე. ღრმა სწავლებამ მიაღწია სიზუსტის უფრო მაღალ დონეს, ვიდრე ოდესმე. ეს ხელს უწყობს სამომხმარებლო ელექტრონიკებს გაამართლოს მომხმარებელთა მოლოდინი და მნიშვნელოვანია უსაფრთხოების კრიტიკული აპლიკაციებისთვის, როგორიცაა მანქანა მძღოლის გარეშე. ღრმა სწავლების უახლესი მიღწევები გაუმჯობესდა იმ დონემდე, რომ ადამიანებს გარკვეულ ამოცანებში ასუსტებს, მაგალითად სურათების კლასიფიკაციის ამოცანა.

მიუხედავად იმისა, რომ ღრმა სწავლება 1980 წელს იქნა გაჟღერებული, ის ახლახანს გახდა სასარგებლო, ამის ორი ძირითადი მიზეზი არსებობს:

1. ღრმა სწავლება მოითხოვს დიდი რაოდენობით მონაცემებს. მაგალითად, „მძღოლის მანქანის“ განვითარება მოითხოვს მილიონობით გამოსახულებას და ათასობით საათის ვიდეოს.

ღრმა სწავლება საჭიროებს მნიშვნელოვან კომპიუტერულ ძალას. მაღალი ხარისხის GPU-ებს აქვთ პარალელური არქიტექტურა, რომელიც ეფექტურია ღრმა სწავლისთვის. კომბინირებული კლასტერები და Cloud Computing საშუალებას აძლევს დეველოპერთა გუნდებს შეამცირონ მონაცემების დამუშავება ღრმა სასწავლო ქსელისთვის კვირადან საათამდე ან უფრო ნაკლებ დრომდე.

ღრმა სწავლების მაგალითები

ღრმა სწავლის პროგრამები გამოიყენება საწარმოებში სამედიცინო მოწყობილობების ავტომატურად სამართავად.

ავტომატური მართვის მოწმობა: ავტომობილების მკვლევარები იყენებენ ღრმა სწავლებას, ობიექტების ავტომატურად აღმოსაჩენად, როგორიცაა გაჩერების ნიშნები და შუქნიშნები. გარდა ამისა, სიღრმისეული კვლევა გამოიყენება ფეხით მოსიარულეთა გამოვლენის მიზნით, რაც ხელს უწყობს ავარიების შემცირებას.

სივრცე და დაცვა: ღრმა შესწავლა გამოიყენება სატელიტების ობიექტების იდენტიფიცირებისთვის, რომლებიც მდებარეობენ ინტერესთა სფეროში და განსაზღვრავენ ჯარების უსაფრთხო ან სახიფათო ტერიტორიებს.

სამედიცინო კვლევა: კიბოს მკვლევარები იყენებენ ღრმა სწავლებას კიბოს უჯრედების ავტომატურად აღმოსაჩენად. ლოს-ანჯელესში კალიფორნიის უნივერსიტეტის გუნდებმა თანამედროვე მიკროსკოპი შექმნეს, რომელსაც მივყავართ მაღალი განზომილებიანი მონაცემების გამოყენებისკენ, რათა ზუსტად განსაზღვრონ კიბოს უჯრედები.

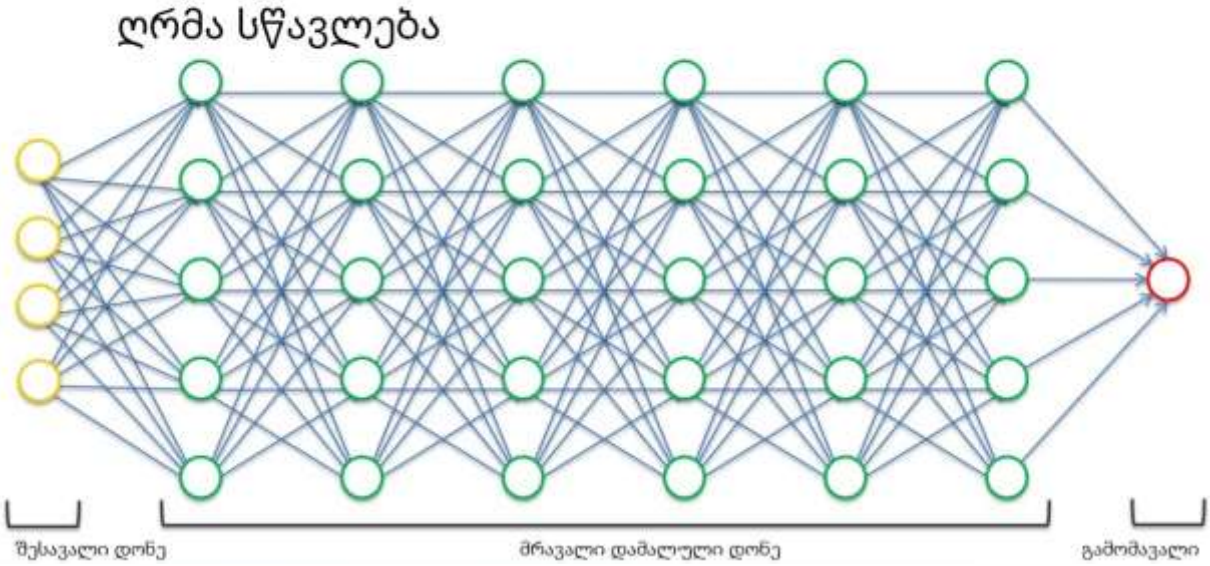
საწარმოო ავტომატიკა: ღრმა სწავლება ხელს უწყობს მუშაობის უსაფრთხოების გაუმჯობესებას მძიმე ტექნიკით აღჭურვილ გარემოში,ავტორმატურად ავლენს ადამიანებს ან ობიექტებს - იმყოფებიან თუ არა ისინი სახიფათო მანძილზე.

ელექტრონიკა: სიღრმისეული კვლევა გამოიყენება ავტომატური მოსმენისა და სიტყვის თარგმანში. მაგალითად, სახლის დახმარების ხელსაწყოები, რომლებიც პასუხობენ თქვენს ხმას და იციან თქვენი შეღავათები, იკვებება ღრმა სწავლის პროგრამებით.

როგორ მუშაობს ღრმა სწავლება?

ყველაზე ღრმა სწავლის მეთოდები იყენებენ ნეირონული ქსელის არქიტექტურას, რის გამოც ღრმა სწავლის მოდელები ხშირად განიხილება როგორც ღრმა ნეირონული ქსელები.

ტერმინი "ღრმა", როგორც წესი, ეხება ფარული ფენების რიცხვს ნეირონულ ქსელში. ტრადიციული ნეირონული ქსელები მხოლოდ 2-3 ფენას შეიცავენ, ხოლო ღრმა ქსელებს შეიძლება ჰქონდეს 150-ზე მეტი.



სიღრმისეული სწავლის მოდელები იწვრთნებიან დიდი მონაცემებისა და ნეირონული ქსელის არქიტექტურის გამოყენებით, რომელიც სწავლობს თვისებებს პირდაპირ მონაცემებიდან ხელით ჩარევის გარეშე.

რა განსხვავებაა მანქანურ და ღრმა სწავლებას შორის?

ღრმა სწავლება არის მანქანური სწავლის სპეციალიზირებული ფორმა. მანქანური სწავლების ტექნოლოგიური პროცესი იწყება შესაბამისი მახასიათებლებით, ხელით მოპოვებული გამოსახულებისგან. ფუნქციები გამოიყენება მაშინ, რომ შექმნას მოდელი, რომელიც გამოსახავს ობიექტს. ღრმა სწავლის პროცესში, შესაბამისად თვისებები ავტომატურად მოპოვებულია სურათებიდან. გარდა ამისა, სიღრმისეული კვლევა “end-to-end

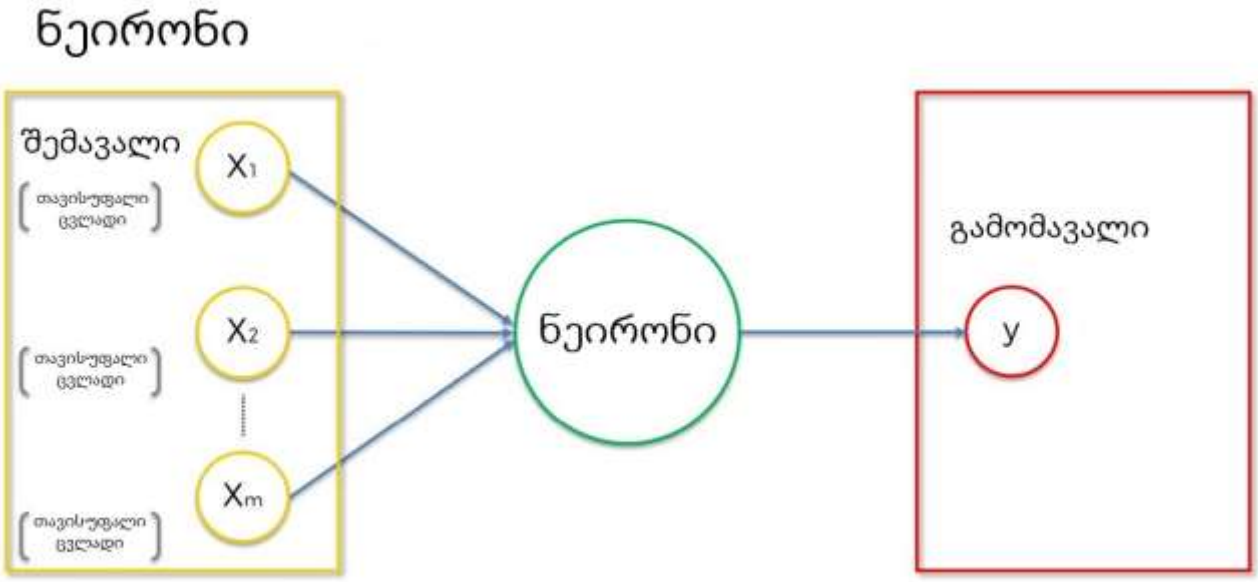
learning” ასრულებს - სადაც ქსელებს ეძლევათ მონაცემები და ამოცანა, როგორიცაა კლასიფიკაციაში და ის სწავლობს თუ როგორ გააკეთოს ავტომატურად.

ღრმა სწავლის ქსელების უპირატესობა იზრდება მონაცემების ზრდასთან ერთად.

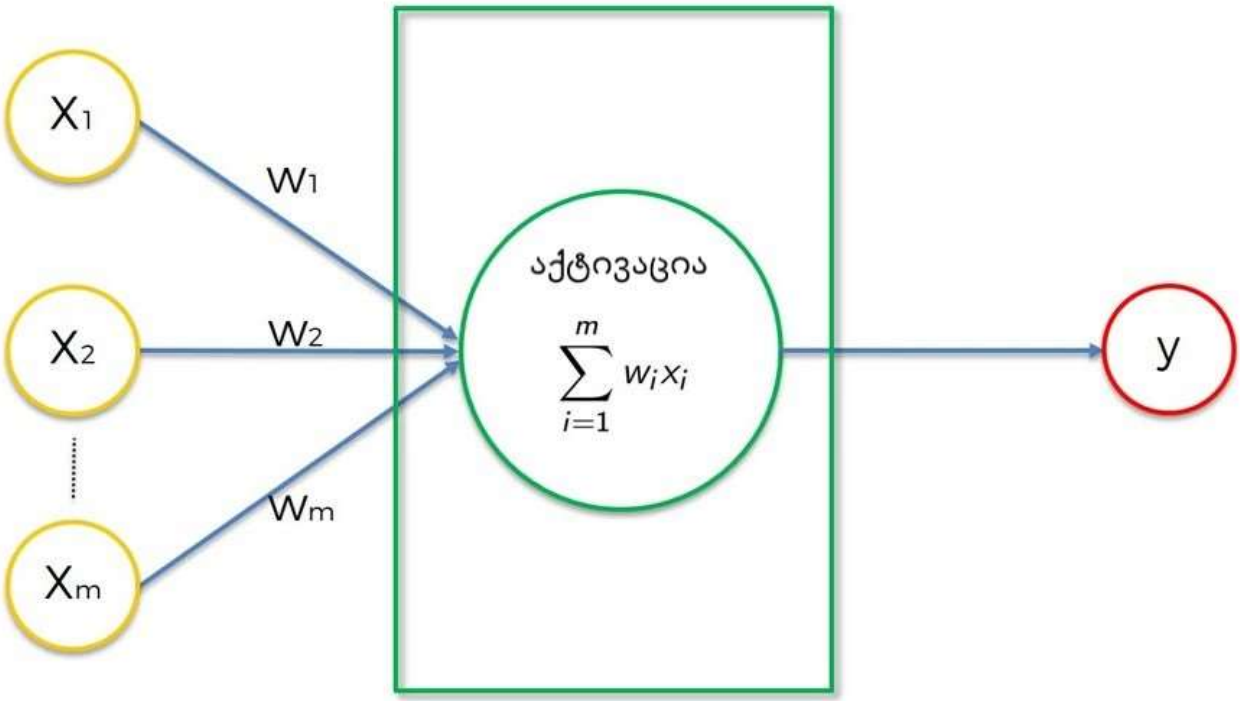
ხელოვნური ნეირონული ქსელი

ხელოვნური ნეირონებისა და ამ ნეირონების დამაკავშირებელი წახნაგებისაგან შემდგარი ქსელი. ზოგიერთი წახნაგი მოცემული ნეირონისთვის განიხილება შემომავლად, ხოლო ზოგიერთი გამავლად. ყოველი წახნაგი ხასიათდება წონით, რომელიც ზოგ მოდელებში შეიძლება შეიცვალოს დროთა განმავლობაში. საერთო ჯამური შემოსავალი ნეირონისთვის, რომელიც აღინიშნება net , განისაზღვრება, როგორც ყველა შემოსავალი სიგნალის შეწონილი ჯამი. მათემატიკური აღნიშვნებით $net = \sum w_i x_i$, სადაც w_i არის i-ური შემომავალი წახნაგის წონა , ხოლო x_i ამ წახნაგზე არსებული სიგნალის სიდიდე. ნეირონის გამომავალი სიგნალის სიდიდე არის net სიდიდის ფუნქცია, ამ ფუნქციას აგრეთვე *აქტივაციის* ფუნქცია ეწოდება.

ნეირონული ქსელის არქიტექტურა ეწოდება ნეირონების განთავსების, მათი დაკავშირების და აქტივაციის ფუნქციის ერთობლიობას.



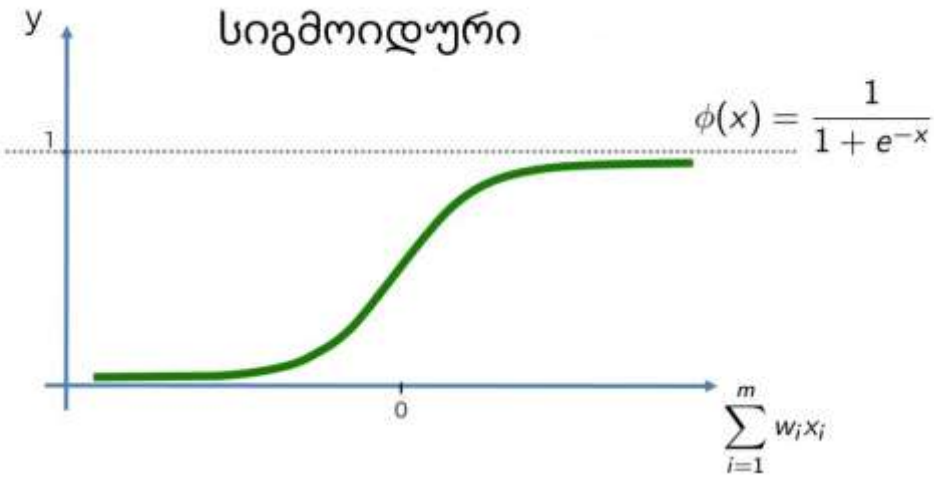
აქტივაციის ფუნქციის სახეები



აქტივაციის ფუნქციის შერჩევა ხდება კონკრეტული გამოყენების დარგის მიხედვით. მისი ძირითადი დანიშნულებაა შემომავალი სიგნალის ლიმიტირება, რადგან, როგორც წესი გამოსავალზე საჭიროა სიგნალი კონკრეტულ ფარგლებში.

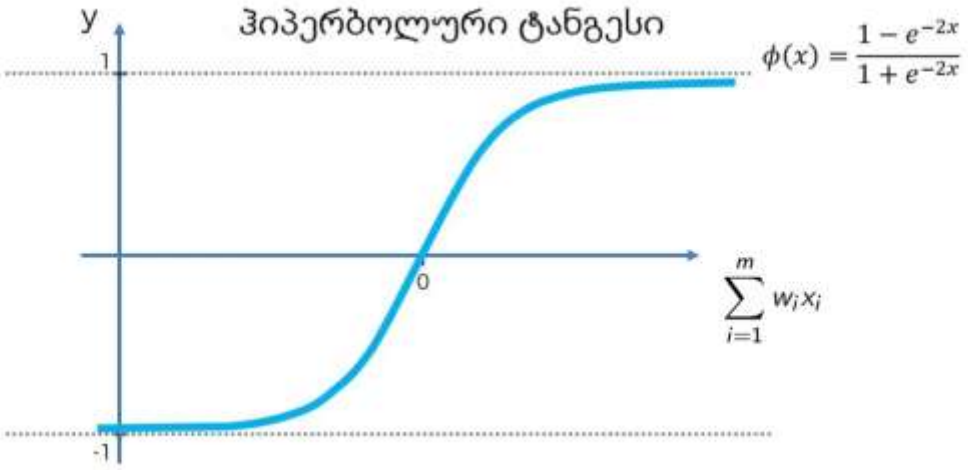
აქტივაციის ფუნქციად ყველაზე ხშირად გამოიყენება სიგმოიდი,

$$f(net) = \frac{1}{1 + exp\{-(net + \theta)\}}$$



ან ჰიპერბოლური ტანგენსი,

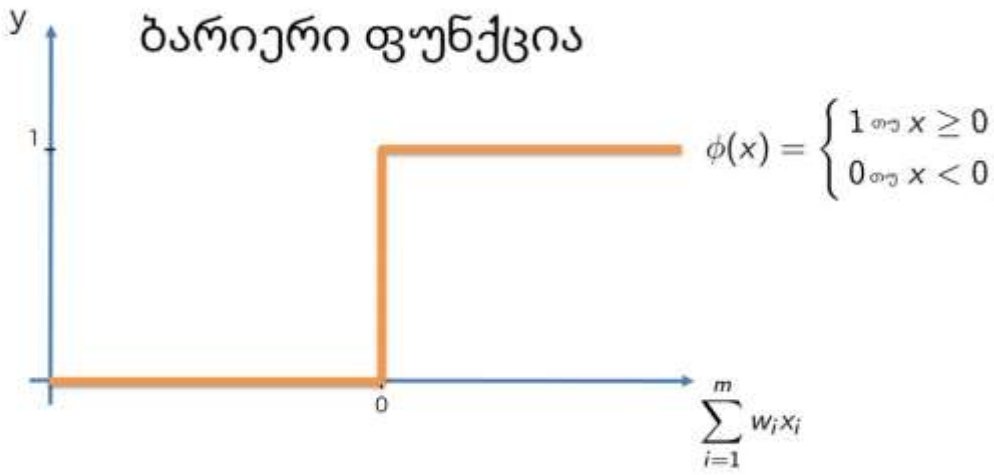
$$f(net) = \frac{exp\{-(net + \theta)\} - exp\{-(net + \theta)\}}{exp\{-(net + \theta)\} + exp\{-(net + \theta)\}}$$

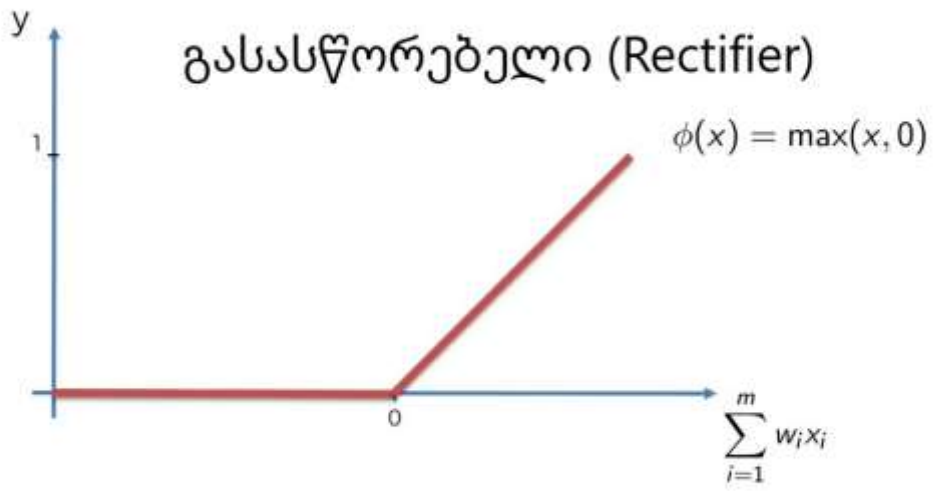


ორივე შემთხვევაში θ პარამეტრი წარმოადგენს მარცხნივ წანაცვლებას.

აქტივაციის ფუნქციებად აგრეთვე შეიძლება იყოს გამოყენებული სხვა ფუნქცია, რომელიც იძლევა შემოსაზღვრულ მნიშვნელობებს net პარამეტრის ნებისმიერი მნიშვნელობისთვის.

ქვემოთ მოცემულია სხვა აქტივაციის ფუნქციები:





გრადიენტური დაშვება

გრადიენტური დაშვება პირველი რიგის ოპტიმიზაციის ალგორითმია რომელიც პოულობს ფუნქციის მინუმუმს.

სტოქასტური გრადიენტის დაშვება

სტოქასტური გრადიენტი დაშვება (ხშირად შემოკლებით SGD) არის ობიექტური ფუნქციის ოპტიმიზაციის მეთოდი შესაფერისი გლუვი თვისებების მქონე (მაგალითად, დიფერენცირებული). მას უწოდებენ სტოქასტურს, რადგან მეთოდი იყენებს შემთხვევით შერჩეულ (ან შერეულ) ნიმუშებს გრადიენტის შეფასებისათვის, ამიტომ SGD შეიძლება ჩაითვალოს სტოქასტურ დაახლოებად გრადიენტური წარმოშობის ოპტიმიზაციისათვის.

უკუპროპაგაციული მოდელი

უკუპროპაგაციული მოდელი არის ხელოვნურ ნეირონულ ქსელებში ერთ-ერთი ყველაზე გამოყენებადი. მას გააჩნია შემომავალი და გამავალი შრეები, აგრეთვე ერთი ან მეტი შიდა შრე. მეზობელ შრეებს შორის ყველა ნეირონი ერთმანეთშია დაკავშირებული. სიგნალის მოძრაობა ხდება შემოსავლიდან გამოსავალზე შიდა შრეების სათითოდ გავლით. ნეირონების დამაკავშირებელი წახნაგების წონების შერჩევის გზით შესაძლებელია ასეთ ნეირონულ ქსელს "ვასწავლოთ" შემოსავლის შესაბამის გასავალში გარდაქმნა, მაგალითად, ხელნაწერი ასოების ამოსაცნობად.

ნეირონული ქსელის უკუპროპაგაციული მოდელი, ან უბრალოდ უკუპროპაგაციული მოდელი — ხელოვნური ნეირონული ქსელის ერთ-ერთი ყველაზე გავრცელებული მაგალითია. ის გამოიყენება ხელნაწერი ასოების, ბირჟაზე ფასების ცვლილების და სხვა რთულად გამოთვლადი სტრუქტურის ამოსაცნობად.

უმარტივესი უკუპროპაგაციული მოდელი შედგება ერთი შემომავალი შრისგან, ერთი გასავალი შრისგან და მათ შორის მდებარე ერთი შიდა (დამალული) შრისგან. შესაძლებელია ორი ან მეტი შიდა შრის გამოყენება, თუმცა ეს ნაკლებადაა გავრცელებული, რადგან დაკავშირებულია მეტ გამოთვლასთან. მეზობელი შრეების ნეირონები ერთმანეთთან სრულად არიან დაკავშირებულები (ანუ ერთი შრის ნეირონი წახნაგებით არიან დაკავშირებული მეორე შრის ყველა ნეირონთან).

შემომავალ და გამავალ შრეზე ნეირონების რაოდენობა განისაზღვრება კონკრეტული ამოცანიდან გამომდინარე. ასე მაგალითად თუ გვინდა 100x100 პიქსელის ზომის გამოსახულების ამოცნობა, დაგვჭირდება 10 000 შემომავალი ნეირონი: თითო ყოველ პიქსელზე. შიდა შრის ნეირონების რაოდენობა დგინდება ექსპერიმენტალურად: თუ რომელი რაოდენობისთვის არის ქსელი ყველაზე სწრაფად კრებადი.

უკუპროპაგაციული მოდელი მიეკუთვნება ცალმხრივად გავლად მოდელებს. მასში სიგნალი გადის მხოლოდ ერთი მიმართულებით: შემომავალი შრიდან გამავალ შრეზე, ყველა

შიდა შრის გავლით. შრეებს შორის სიგნალის გავლისთვის ვიყენებთ ზოგად პრინციპს, როგორც ეს ხელოვნურ ნეირონულ ქსელშია აღწერილი. კერძოდ კი, შემომავალ ნეირონებზე მოდებული საწყისი სიგნალი $\mathbf{x} = (x_1, x_2, \ldots, x_{n_i})$ აისახება ჯამურ შეწონილად პირველი შიდა შრის ნეირონებზე. თუ ავღნიშნავთ w_{ij} წახნაგის წონას, რომლითაც i-ური შემომავალი ნეირონი უკავშირდება j-ურ შიდა შრის ნეირონს, მაშინ საშუალო შეწონილი j-ური შიდა შრის ნეირონზე იქნება $net_j = \sum_{i=0}^{n_i} w_{ij} x_i$. გამოსავალი j-ურ შიდა შრის ნეირონზე იქნება ტოლი $o_j = f(net)j)$, სადაც f არის აქტივაციის ფუნქცია, როგორც წესი აქტივაციის ფუნქცია არის სიგმოიდი წანაცვლებით, $f(net_j) = \frac{1}{1 + exp\{-(net_j + \theta_j)\}}$. ანალოგიურად ხდება სიგნალის გატარება პირველი შიდა შრიდან შემდეგ შრეზე და ა. შ. გამოსავლამდე.

გამოსავალზე მიღებული შედეგი წარმოადგენს "პასუხს", რომლსაც ვადარებთ მოსალოდნელ პასუხს. თუ მიღებული პასუხი არ ემთხვევა მოსალოდნელს, მაშინ ჩვენ უნდა მოვახდინოთ ქსელის წახნაგების წონების კორექტირება და კიდევ ერთხელ ვცადოთ სიგნალის გატარება, და ასე სანამ ქსელი არ მოგვცემს სასურველ შედეგს. ანალოგიური ციკლი უნდა ჩატარდეს რაც შეიძლება მეტი საწყისი სტრუქტურისათვის. ასე მაგალითად, თუ გვინდა ნეირონული ქსელის გამოყენება ხელნაწერი ასოების ამოსაცნობად, უნდა "ვასწავლოთ" ქსელს რაც შეიძლება მეტი ხელნაწერი ასოების ამოცნობა, შესაბამისი შემავალი და გამავალი სტრუქტურის მიწოდებით.

როდესაც ნეირონული ქსელი საკმარისადაა "ნასწავლი" ის ავლენს უნარს ამოიცნოს არა მარტო სწავლის პერიოდში შესწავლილი სტრუქტურები, არამედ სხვა ანალოგიური სტრუქტურებიც.

რადგან უკუპროპაგაციული მოდელი მოითხოვს საწყისი სწავლის კურსს ადამიანის მონაწილეობით, ამიტომ ამ მოდელს მიაკუთვნებენ დამოძღვრად ჯგუფს.

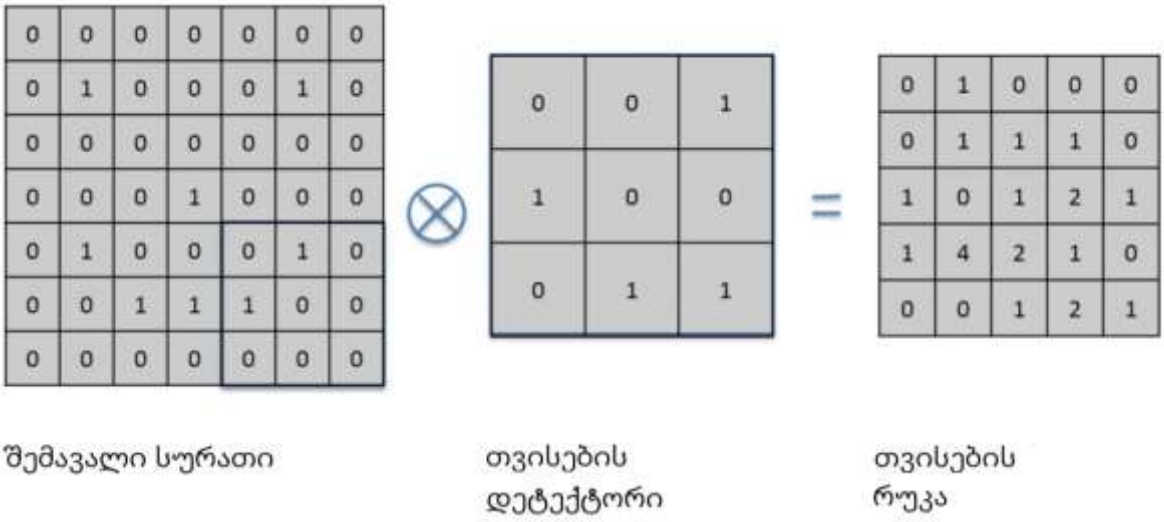
პერცეპტრონი

პერცეპტრონი არის უმარტივესი ნეირონული ქსელის არქიტექტურა, რომელიც გამოიყენება წრფივად გაყოფადი არეების ამოსაცნობად. პერცეპტრონს გააჩნია მხოლოდ ორი სრულად დაკავშირებული შრე – შემომავალი და გამავალი შრეები.

თუმცა პერცეპტრონის გამოსავალი შესაძლებელია გამოთვლილი იყოს ანალიტიკური გზით და მას შეუძლია მხოლოდ შეზღუდული სახეობის სტრუქტურების ამოცნობა (წრფივად გაყოფადი არე), ის მაინც წარმოადგენს ერთ-ერთ ყველაზე სასარგებლო მოდელს თეორიული კვლევისთვის, ვინაიდან უფრო რთული მოდელების (მაგ. უკუპროპაგაციული მოდელი ერთი შიდა შრით) თეორიული განხილვა მეტად რთულია.

კონვოლუციური ნეირონული ქსელი

პრაქტიკული ამოცანების წარმატებით გადაწყვეტა შესაძლებელი გახდა ღრმა ნეირონული ქსელების საშუალებით. ღრმა ნეირონული ქსელები გამოირჩევა ფარული შრეების სიმრავლით და სწავლების ოპტიმიზირებული ალგორითმებით. ღრმა ქსელების საშუალებით განსაკუთრებით ეფექტურად ხდება გრაფიკული გამოსახულებების ამოცნობა. გრაფიკული გამოსახულებების ამოცნობისთვის იყენებენ Convolutional ქსელებს. გრაფიკული გამოსახულების ამოცნობის მიზნით შემუშავებულია სპეციფიკური ქსელი, რომელსაც Convolutional ქსელს უწოდებენ [convolutional networks].



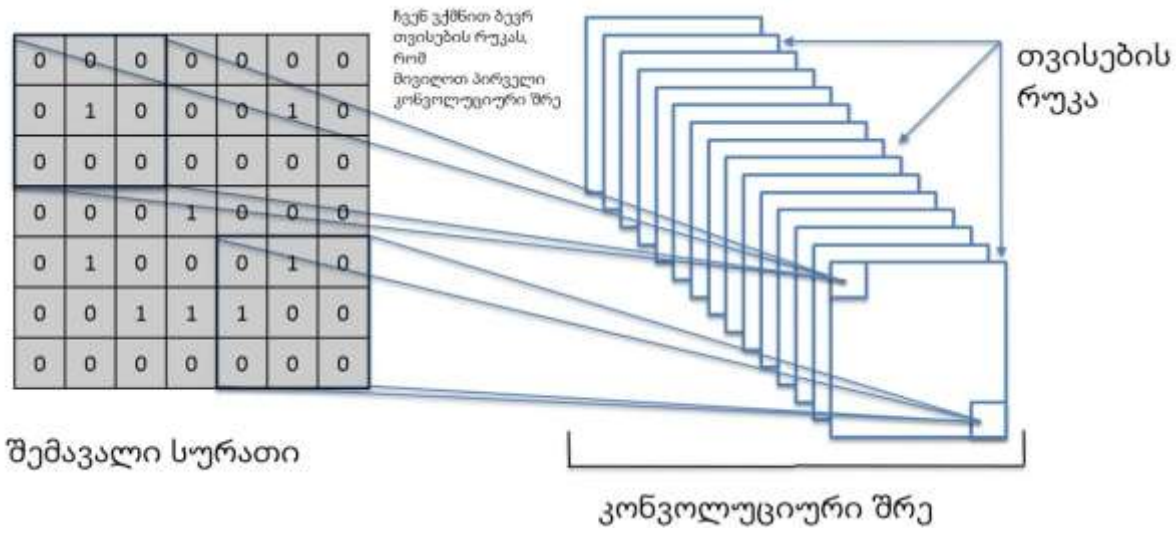
კონვოლუციური ოპერაცია

კონვოლუციური ოპერაცია - შემავალ მატრიცაში გამოვყოფთ თვისებების დეტექტორის ზომის მატრიცას და მის თითოეულ ელემენტს ვამრავლებთ თვისებების მატრიცის ელემენტებზე და ვკრებთ მათ, ასე მიიღება რუკა. რუკა ამცირებს სურათის ზომას და თავიდან გვაცილებს არასაჭირო ინფორმაციას, რაც ამცირებს მუშაობის დროს.

Convolutional ქსელი მრავალი შრისგან შედგება. შრეებს სხვადასხვა დანიშნულება და მოქმედების პრინციპი აქვს.

კონვოლუციური შრე

კონვოლუციური შრე (**Convolutional Layer**) Convolutional შრე შედგება იგივე ტიპის ნეირონებისგან, რაც სრულად ბმული ქსელის შესწავლისას განვიხილეთ. განსხვავება მდგომარეობს მხოლოდ იმაში, რომ საწყისი გრაფიკული გამოსახულების ყველა წერტილი არ არის დაკავშირებული Convolutional შრის თითოეულ ნეირონთან. ნეირონები, გრაფიკული გამოსახულების მხოლოდ მცირე უბანთან (როგორც წესი მართკუთხა ფორმის მატრიცასთან) არის შეერთებული. ნეირონების მართკუთხა უბნები მიმდევრობით არის განლაგებული საწყის გრაფიკულ გამოსახულებაზე და მთლიანად მოიცავს მას. ასეთნაირად განლაგებული ნეირონების გამოსასვლელებიდან მიღებული სიგნალები გეომეტრიულად თვალსაჩინო გამოსახულებებს იძლევა, ამიტომ მათ ისევ მატრიცის სახით გამოსახავენ და არა ისე, როგორც სრულად ბმულ ნეირონულ ქსელშია მოყვანილი.

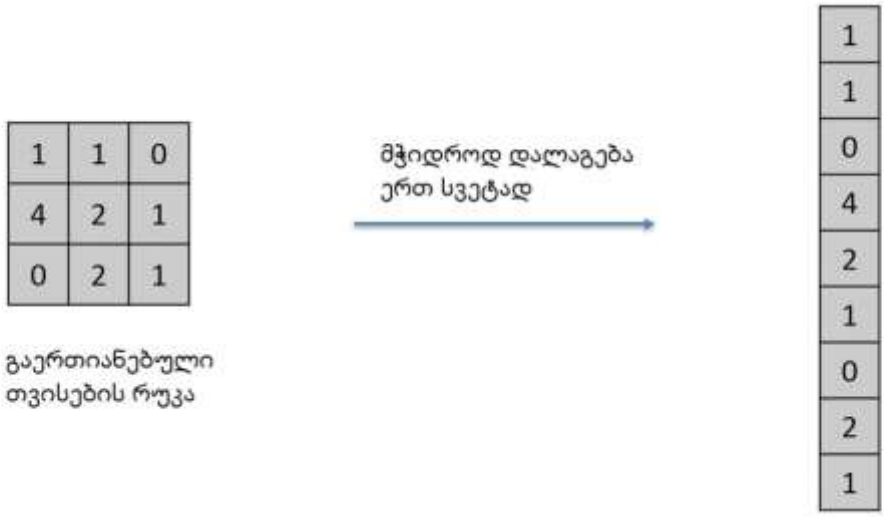


Pooling შრე

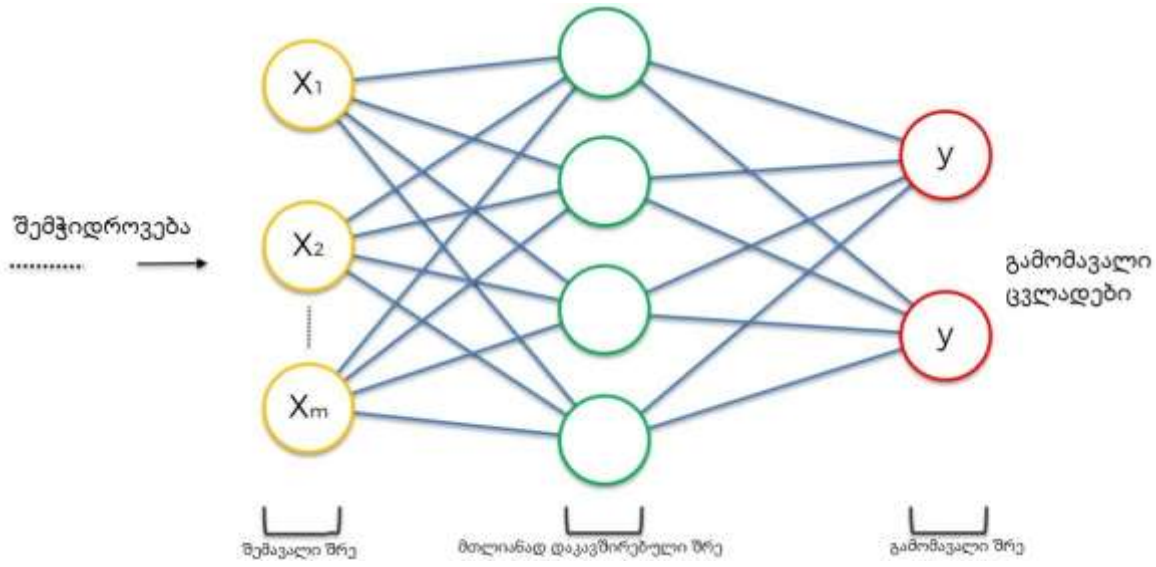
Pooling შრე (Pooling Layer) Pooling შრე უშუალოდ არის შეერთებული Convolutional შრის გამოსასვლელ სიგნალებთან. ყველა Convolutional შრეს შეერთებული აქვს ცალკე Pooling შრე. Pooling შრის დანიშნულებაა Convolutional შრიდან გამოსული სიგნალების ოდენობის შემცირება. Pooling შრის ნეირონების მიერთება არის Convolutional შრის ნეირონების ანალოგიური, თუმცა, განსხვავება ისაა, რომ Pooling შრის მეზობლად მდგომი ნეირონების მიერთების არეები არ თანაიკვეთება.

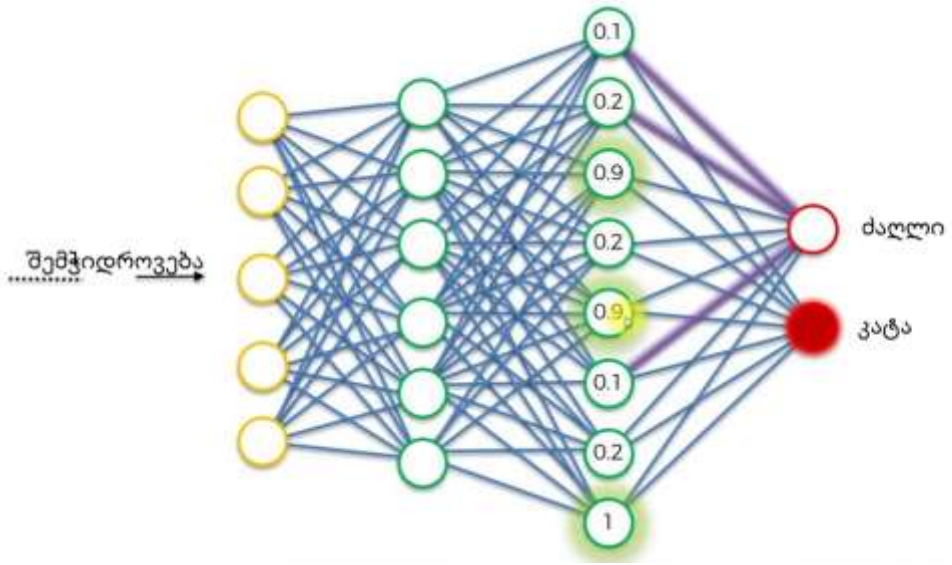
Pooling გვეხმარება ამოვიცნოთ სხვადასხვა პოზიციაში მყოფი ერთიდაიგივე ობიექტი, ასევე გაერთიანებით ვამცირებთ ცვლადების რაოდენობას.სურათზე წარმოდგენილია max pooling, ასევე არსებობს min და average.

FLATTENING - მჭიდროდ დალაგება



მთლიანი კავშირი





თავი 2

პრაქტიკული

ხელოვნური ნეირონული ქსელები ბიზნეს პრობლემისთვის

ხელოვნური ნეირონული ქსელების მოდელები ხშირად გამოიყენება ბიზნეს პრობლემის გადასაჭრელად. ჩვენ შეგვიძლია ავიღოთ მონაცემთა ნაკრები და ამ მონაცემების მიხედვით შევქმნათ მოდელი და შევასწავლოთ ნეირონული ქსელი. შემდეგ შევძლებთ ამ მოდელის მიხედვით პროგნოზირება გავუკეთოთ ახალ მონაცემთა ნაკრებს.

განვიხილოთ ბანკის კლიენტთა სია :

N	კლიენტისნომერი	გვარი	კრედიტისქულა	ადგილმდებარეობა	სქესი	ასაკი	ვადა	ბალანსი	პროდუქტისრაოდენობა	საკრედიტობარათი კი/არა	აქტიურიწევრი	ხელფასი	დატოვა ბანკი კი/არა
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0
20	15568982	Hao	726	France	Female	24	6	0	2	1	1	54724.03	0
21	15577657	McDonald	732	France	Male	41	8	0	2	1	1	170886.17	0
22	15597945	Dellucci	636	Spain	Female	32	8	0	2	1	0	138565.46	0
23	15699309	Gerasimov	510	Spain	Female	38	4	0	1	1	0	118913.53	1
24	15725737	Mosman	669	France	Male	46	3	0	2	0	1	8487.75	0
25	15625047	Yen	846	France	Female	38	5	0	1	1	1	187616.16	0
26	15738191	Maclean	577	France	Male	25	3	0	2	0	1	124508.29	0
27	15736816	Young	756	Germany	Male	36	2	136815.6	1	1	1	170041.95	0
28	15700772	Nebechi	571	France	Male	44	9	0	2	0	0	38433.35	0
29	15728693	McWilliams	574	Germany	Female	43	3	141349.4	1	1	1	100187.43	0
30	15656300	Lucciano	411	France	Male	29	0	59697.17	2	1	1	53483.21	0
31	15589475	Azikiwe	591	Spain	Female	39	3	0	3	1	0	140469.38	1
32	15706552	Odinakachuk	533	France	Male	36	7	85311.7	1	0	1	156731.91	0
33	15750183	Sanderson	553	Germany	Male	41	9	110112.5	2	0	0	81898.81	0
34	15659428	Maggard	520	Spain	Female	42	6	0	2	1	1	34410.55	0
35	15732963	Clements	722	Spain	Female	29	9	0	2	1	1	142033.07	0
36	15794171	Lombardo	475	France	Female	45	0	134264	1	1	0	27822.99	1
37	15788448	Watson	490	Spain	Male	31	3	145260.2	1	0	1	114066.77	0
38	15729599	Lorenzo	804	Spain	Male	33	7	76548.6	1	0	1	98453.45	0
39	15717426	Armstrong	850	France	Male	36	7	0	1	1	1	40812.9	0
40	15585768	Cameron	582	Germany	Male	41	6	70349.48	2	0	1	178074.04	0
41	15619360	Hsiao	472	Spain	Male	40	4	0	1	1	0	70154.22	0
42	15738148	Clarke	465	France	Female	51	8	122522.3	1	0	0	181297.65	1
43	15687946	Osborne	556	France	Female	61	2	117419.4	1	1	1	94153.83	0
44	15755196	Levine	834	France	Female	49	2	131394.6	1	0	0	194365.76	1
45	15684171	Bianchi	660	Spain	Female	61	5	153001.1	1	1	1	158338.39	0
46	15758849	Tyler	776	Germany	Female	32	4	109421.1	2	1	1	120517.46	0
47	15602280	Martin	829	Germany	Female	27	9	112045.7	1	1	1	119708.21	1
48	15771573	Okagbue	637	Germany	Female	39	9	137843.8	1	1	1	117622.8	1
49	15766205	Yin	550	Germany	Male	38	2	103391.4	1	0	1	90878.13	0
50	15771873	Baccho	776	Germany	Female	37	2	103769.2	2	1	0	194009.12	0

ქსელის ფაილში მოცემულია კლიენტთა მონაცემები : ნომერი, გვარი, კრედიტის ქულა , ადგილმდებარეობა , სქესი , ასაკი, ვადა , ბალანსი , პროდუქტის რაოდენობა , საკრედიტო ბარათი კი / არა, აქტიური წევრი არის თუ არა,ხელფასი, დატოვა ბანკი კი/არა.

ჩვენი ამოცანაა ეს მონაცემები შევისწავლოთ და ამ მონაცემებით მიღებული ცოდნის მიხედვით ახალი მონაცემების პროგნოზი გავაკეთოთ.

ამისთვის უნდა ავაგოთ ნეირონული ქსელი: პროგრამირების ენა - python.

1. ბიბლიოთეკების ინსტალაცია

```
# Artificial Neural Network - ხელოვნური ნეირონული ქსელი

# Theano -ს ინსტალაცია
# pip install --upgrade --no-deps git+git://github.com/Theano/Theano.git

# Tensorflow -ს ინსტალაცია
# Install Tensorflow from the website: https://www.tensorflow.org/versions/r0.12/get_started/os_setup.html

# Keras -ს ინსტალაცია
# pip install --upgrade keras
```

2. მონაცემთა დამუშავების პროცესი

```
# ნაწილი 1 - მონაცემთა დამუშავების პროცესი

# ბიბლიოთეკების იმპორტირება
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# მონაცემთა ნაკრების იმპორტი
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values

# კატეგორიული მონაცემების კოდირება
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]

# მონაცემთა ნაკრების სასწავლო და საკონტრო კომპლექტად დაყოფა
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

# სკალირების დონეზე
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

dataset = pd.read_csv('Churn_Modelling.csv')

X = dataset.iloc[:, 3:13].values

y = dataset.iloc[:, 13].values

X დამოუკიდებელი ცვლადების მატრიცას წარმოადგენს :

619	France	Female	42	2	0	1	1	1	101349
608	Spain	Female	41	1	83807.9	1	0	1	112543
502	France	Female	42	8	159661	3	1	0	113932
699	France	Female	39	1	0	2	0	0	93826.6
850	Spain	Female	43	2	125511	1	1	1	79084.1
645	Spain	Male	44	8	113756	2	1	0	149757
822	France	Male	50	7	0	2	1	1	10062.8
376	Germany	Female	29	4	115047	4	1	0	119347
501	France	Male	44	4	142051	2	0	1	74940.5
684	France	Male	27	2	134604	1	1	1	71705.7
528	France	Male	31	6	102017	2	0	0	80181.1
497	Spain	Male	24	3	0	2	1	0	76390
476	France	Female	34	10	0	2	1	0	26261
549	France	Female	25	5	0	2	0	0	190858
635	Spain	Female	35	7	0	2	1	1	65951.6
616	Germany	Male	45	3	143129	2	0	1	64327.3
653	Germany	Male	58	1	132603	1	1	0	5897.67
549	Spain	Female	24	9	0	2	1	1	14406.4
587	Spain	Male	45	6	0	1	0	0	158685
726	France	Female	24	6	0	2	1	1	54724
732	France	Male	41	8	0	2	1	1	170886
636	Spain	Female	32	8	0	2	1	0	138555
510	Spain	Female	38	4	0	1	1	0	118914
669	France	Male	46	3	0	2	0	1	8487.75
846	France	Female	38	5	0	1	1	1	187818
577	France	Male	25	3	0	2	0	1	124508
756	Germany	Male	36	2	136816	1	1	1	170042
571	France	Male	44	9	0	2	0	0	38433.3
574	Germany	Female	43	3	141349	1	1	1	100187
411	France	Male	29	0	59697.2	2	1	1	53483.2
591	Spain	Female	39	3	0	3	1	0	140469
533	France	Male	36	7	85311.7	1	0	1	156732
553	Germany	Male	41	9	110113	2	0	0	81898.8
520	Spain	Female	42	6	0	2	1	1	34410.6
722	Spain	Female	29	9	0	2	1	1	142033
475	France	Female	45	0	134264	1	1	0	27823
490	Spain	Male	31	3	145260	1	0	1	114067
804	Spain	Male	33	7	76548.6	1	0	1	98453.4
850	France	Male	36	7	0	1	1	1	40812.9
582	Germany	Male	41	6	70349.5	2	0	1	178674
472	Spain	Male	40	4	0	1	1	0	79154.2
465	France	Female	51	8	122522	1	0	0	181298
556	France	Female	61	2	117419	1	1	1	94153.8
834	France	Female	49	2	131395	1	0	0	194366

y დამოკიდებული ცვლადის (დატოვა ბანკი კი/არა) ვექტორს :

0	1
1	0
2	1
3	0
4	0
5	1
6	0
7	1
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	1
17	0
18	0
19	0
20	0
21	0
22	1
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	1
31	0
32	0
33	0
34	0
35	1
36	0
37	0
38	0
39	0
40	0
41	1

სასწავლო მონაცემების მიხედვით დავასწავლით მოდელს. ადგილმდებარეობის და სქესის სკალირებას ვახდენთ - ტექსტს რომ არ შეიცავდეს რიცხვებად გარდავქმნით :

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] =labelencoder_X_2.fit_transform(X[:, 2])
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]
```

ამ მონაცემებს ვყოფთ სატესტო და სასწავლო მონაცემებად . სატესტო მონაცემები 20 % -ს შეადგენს :

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

დიდი რიცხვების სკალირებას მოვახდენთ :

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

1. გავაკეთოთ ხელოვნური ნეირონული ქსელი!

```
#  ნაწილი 2 - მოდით გავაკეთოთ ANN(ხელოვნური ნეირონული ქსელი)!

# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

#  ANN -ს ინიციალიზაცია
classifier = Sequential()

# შემავალი ფენის და პირველი დაფარული ფენის დამატება
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 11))

# მეორე დაფარული ფენის დამატება
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))

# გამომავალი ფენის დამატება
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))

# ANN -ს კომპილაცია
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# მოვარგოთ ANN სატესტო ნაკრებს
classifier.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
```

ANN -ს ინიციალიზაცია :

classifier = Sequential()

პირველი დაფარული ფენა :

classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 11))

ამ ფენაში მონაწილეობას 6 ნეირონი მიიღებს, აქტივაციის ფუნქციაა გასასწორებელი (Rectifier) და გვაქვს 11 შემავალი ცვლადი.

მეორე დაფარული ფენა :

classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu')

გამომავალი ფენა:

classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
1 გამომავალი ცვლადი, აქტივაციის ფუნქცია - სიგმოიდური.

ANN -ს კომპილაცია :

```
Epoch 1/100
8000/8000 [==============================] - 3s 376us/step - loss: 0.4943 - acc: 0.7949
Epoch 2/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4279 - acc: 0.7960
Epoch 3/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4229 - acc: 0.7960
Epoch 4/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4191 - acc: 0.8162
Epoch 5/100
8000/8000 [==============================] - 1s 67us/step - loss: 0.4163 - acc: 0.8251
Epoch 6/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4142 - acc: 0.8295
Epoch 7/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4130 - acc: 0.8314
Epoch 8/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4113 - acc: 0.8325
Epoch 9/100
8000/8000 [==============================] - 0s 61us/step - loss: 0.4100 - acc: 0.8315
Epoch 10/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4091 - acc: 0.8335
Epoch 11/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4083 - acc: 0.8339
Epoch 12/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4078 - acc: 0.8337
Epoch 13/100
8000/8000 [==============================] - 1s 65us/step - loss: 0.4069 - acc: 0.8346
Epoch 14/100
8000/8000 [==============================] - 1s 66us/step - loss: 0.4060 - acc: 0.8354
Epoch 15/100
8000/8000 [==============================] - 1s 65us/step - loss: 0.4055 - acc: 0.8345
Epoch 16/100
8000/8000 [==============================] - 1s 66us/step - loss: 0.4056 - acc: 0.8349
Epoch 17/100
8000/8000 [==============================] - 1s 65us/step - loss: 0.4051 - acc: 0.8365
Epoch 18/100
8000/8000 [==============================] - 1s 66us/step - loss: 0.4047 - acc: 0.8345
Epoch 19/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4044 - acc: 0.8355
Epoch 20/100
8000/8000 [==============================] - 1s 71us/step - loss: 0.4039 - acc: 0.8356
Epoch 21/100
8000/8000 [==============================] - 1s 76us/step - loss: 0.4036 - acc: 0.8355
Epoch 22/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4031 - acc: 0.8340
Epoch 23/100
8000/8000 [==============================] - 0s 61us/step - loss: 0.4023 - acc: 0.8350
Epoch 24/100
8000/8000 [==============================] - 0s 61us/step - loss: 0.4011 - acc: 0.8344
Epoch 25/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.4004 - acc: 0.8359
Epoch 26/100
8000/8000 [==============================] - 0s 61us/step - loss: 0.4000 - acc: 0.8341
Epoch 27/100
8000/8000 [==============================] - 1s 63us/step - loss: 0.3992 - acc: 0.8354
Epoch 28/100
8000/8000 [==============================] - 1s 65us/step - loss: 0.3982 - acc: 0.8365
Epoch 29/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.3982 - acc: 0.8364
Epoch 30/100
8000/8000 [==============================] - 1s 71us/step - loss: 0.3977 - acc: 0.8357
Epoch 31/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.3974 - acc: 0.8354
Epoch 32/100
8000/8000 [==============================] - 1s 67us/step - loss: 0.3966 - acc: 0.8367
Epoch 33/100
8000/8000 [==============================] - 0s 62us/step - loss: 0.3967 - acc: 0.8360
Epoch 34/100
```

4. პროგნოზირება და მოდელის შეფასება

```
# ნაწილი 3 - პროგნოზირება და მოდელის შეფასება

# სატესტო შედეგების პროგნოზირება
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

# შევქმნათ გაურკვევლობის მატრიცა
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

y_pred არის ვექტორი, იმ კლიენტების გასწვრივ, ვისი ბანკიდან წასვლის ალბათობა 50 % ზე მეტია, წერია True.

0	False
1	False
2	False
3	False
4	False
5	True
6	False
7	False
8	False
9	True
10	False
11	False
12	False
13	False
14	True
15	False
16	False
17	False
18	False
19	False
20	False
21	False
22	False
23	False
24	False
25	False
26	False
27	False
28	False
29	False
30	False
31	False
32	False
33	False
34	False
35	False
36	False
37	False
38	False
39	False

სურათების კლასიფიკაცია კონვოლუციური ნეირონული ქსელის გამოყენებით

ჩვენი ამოცანაა გავარჩიოთ სურათზე რომელი ცხოველია გამოსახული :

1. ავაგოთ CNN

```
# ნაწილი 1 - ავაგოთ CNN

# ბიბლიოთეკების და პაკეტების იმპორტი
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

# CNN -ს ინიციალიზაცია
classifier = Sequential()

# ნაწილი 1 - კონვოლუცია
classifier.add(Convolution2D(32, 3, 3, input_shape = (64, 64, 3), activation = 'relu'))

# ნაწილი 2 - გაერთიანება
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# დავამატოთ მეორე კონვოლუციური შრე
classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# ნაწილი 3 - შეშლილობა
classifier.add(Flatten())

# ნაწილი 4 - მთლიანი კავშირი
classifier.add(Dense(output_dim = 128, activation = 'relu'))
classifier.add(Dense(output_dim = 1, activation = 'sigmoid'))

#  CNN -ს კომპილაცია
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

2. CNN -ს მორგება სურათებზე

```
# ნაწილი 2 - CNN -ს მორგება სურათებზე

from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory('dataset/training_set',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')

test_set = test_datagen.flow_from_directory('dataset/test_set',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')

classifier.fit_generator(training_set,
                         samples_per_epoch = 8000,
                         nb_epoch = 25,
                         validation_data = test_set,
                         nb_val_samples = 2000)
```

დასკვნა

დროა წინ წავიდეთ, დავბრუნდეთ მომავალში! ღრმა სწავლება საუკეთესო საშუალებაა დიდი მონაცემების დასამუშავებლად, რომელიც დაგვეხმარება დიდ აღმოჩენებში და მომავალში გაგვიმარტივებს ცხოვრებას. ზემოთ განვიხილე და ავაგე ხელოვნური და კონვოლუციური ნეირონული ქსელი, რომლიც მივუსადაგე კლასიფიკაციის ამოცანებს.

ჩემს მიერ მოყვანილი მაგალითები შეეხება ბანკის კლიენტთა და ცხოველების კლასიფიკაციას. მცირე ჩარევით ჩვენ შეგვიძლია ნებისმიერ კლასიფიკაციის ამოცანას მოვარგოთ ზემოთ ხსენებული ქსელები. ცვლილება შეეხება მონაცემთა ნაკრებს და შესაბამისად მათ გამუშავებას. კონვოლუციური ნეირონული ქსელის შემთხვევაში, უბრალოდ მონაცემების შეცვლით შეგვიძლია დავსვათ სხვადასხვა ამოცანა და მარტივად ამოვხსნათ ის, მაგალითად სიმსივნური უჯრედების აღმოჩენა და ადამიანის იდენთიფიკაცია.

მომავალში ვაპირებ აგებული ხელოვნური ნეირონული ქსელით შევიმუშავო ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტში სწავლის წარმატებული ფორმულა: ავიღებ კურსდამთავრებული სტუდენტების მონაცემებს, რომელიც განსაზღვრავს მათ მდგომარეობას, დავასწავლით მოდელს, რომელიც შემდგომ გაარჩევს და პროგნოზირებს ამჟამინდელი სტუდენტების წარმატებას.

გამოყენებული ლიტერატურა:

https://www.mathworks.com/discovery/deep-learning.html
https://www.kdnuggets.com/2018/09/introduction-deep-learning.html
https://www.udemy.com/deeplearning/
https://www.udemy.com/machinelearning/
https://www.superdatascience.com/machine-learning/
https://www.superdatascience.com/pages/deep-learning