ATLANTBH
INTERNSHIP, DATA SCIENCE PROJECT

Mašinsko učenje

Autor: Mehmed Kadrić
Mentor: Belma Ibrahimović

Sažetak rada: U ovom radu obrađena je tema mašinskog učenja. U prvom dijelu rada dat je osvrt na primjenu mašinskog učenja u praksi, a potom je izvršena osnovna podjela. Analizirani su algoritmi regresije i klasifikacije kod nadgledanog učenja i algoritam klasterizacije kod nenadgledanog učenja. Kroz primjere je na jednostavan i razumljiv način približen svaki od algoritama. Svaki algoritam je na odgovarajući nalčin ilustrovan određenim graficima i matematičkim relacijama.

Ključne riječi: učenje, nadgledano, nenadgledano, algoritam, funkcija, podaci, regresija, klasifikacija, klasterizacija

Sarajevo, februar 2018.

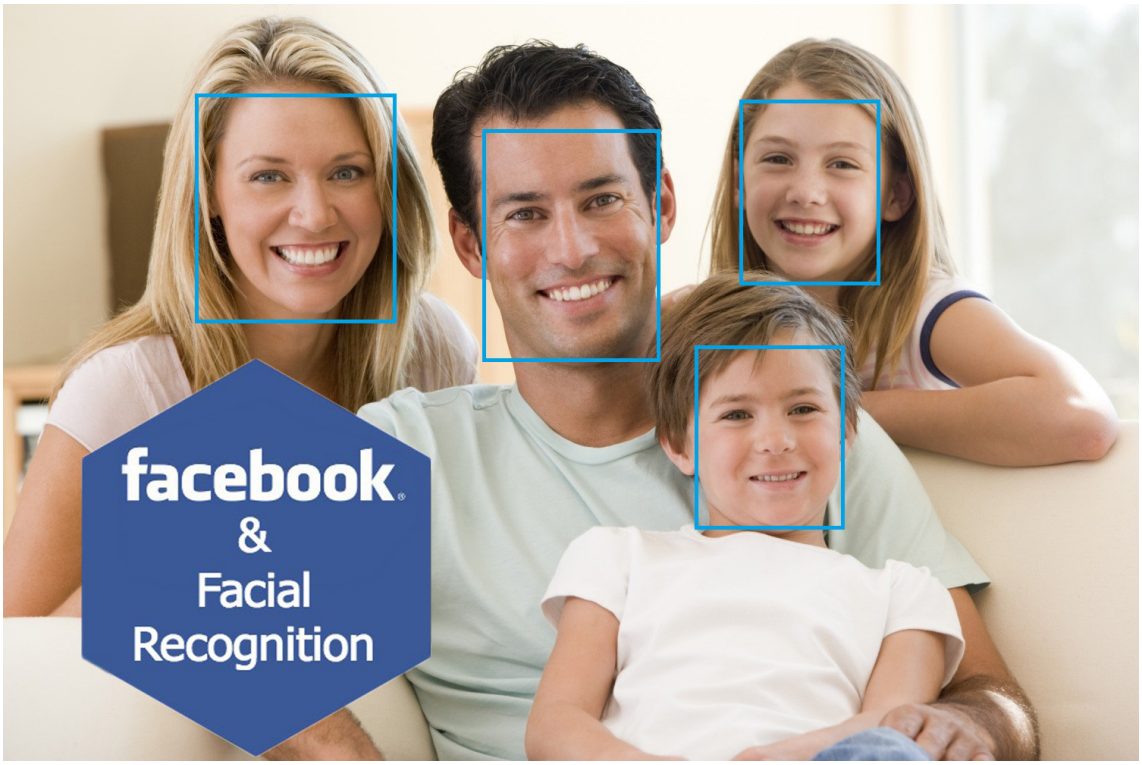
Sadržaj

1 Uvod

Mašinsko učenje predstavlja podoblast vještačke inteligencije čiji je cilj konstruisanje algoritama i računarskih sistema koji su sposobni da se adaptiraju na nove situacije i uče na bazi iskustva.

Ideja je da postoji neki algoritam koji smišlja sopstvenu logiku na osnovu podataka. Prije svega potrebno je definisati pojam Data Science (nauka o podacima). Data Science je široka oblast i podrazumijeva gradnju sistema vještačke inteligencije, odnosno softverskih sistema koji se temelje na različitim podacima, metodama mašinskog učenja i za cilj imaju autonomno izučavanje određenih zadataka pri čemu te zadatke treba obavljati brže i bolje od ljudske inteligencije.

Danas se uticaj mašinskog učenja vidi svuda oko nas. Na primjer, ukoliko se vozimo autoputem i naiđemo na naplatne kućice, na displeju će se ispisati registarske oznake našeg automobila za nevjerovatno brzo vrijeme. To je zapravo posljedica algoritma mašinskog učenja za prepoznavanje tablica i cifara na automobilu. Drugi primjer mašinskog učenja može biti predikcija cijene stana na nekoj lokaciji u gradu na osnovu atributa tog stana (na primjer, to može biti broj soba, kvadratura, adresa i slično). Dalje, na društvenim mrežama se u zadnje vrijeme pojavila mogućnost da označimo prijatelja na slici (slika 1) tako što nam automatski bude ponuđeno ime prijatelja koji je na slici. Kako je to moguće? Odgovor je, naravno, mašinsko učenje! Također, u današnje vrijeme se jako često koristi e-mail u svakodnevnoj komunikaciji. Problem koji se tu javlja je problem spam poruka koje želimo svrstati u neki poseban folder napravljen samo za spam mailove. Također, to možemo uraditi metodama mašinskog učenja.

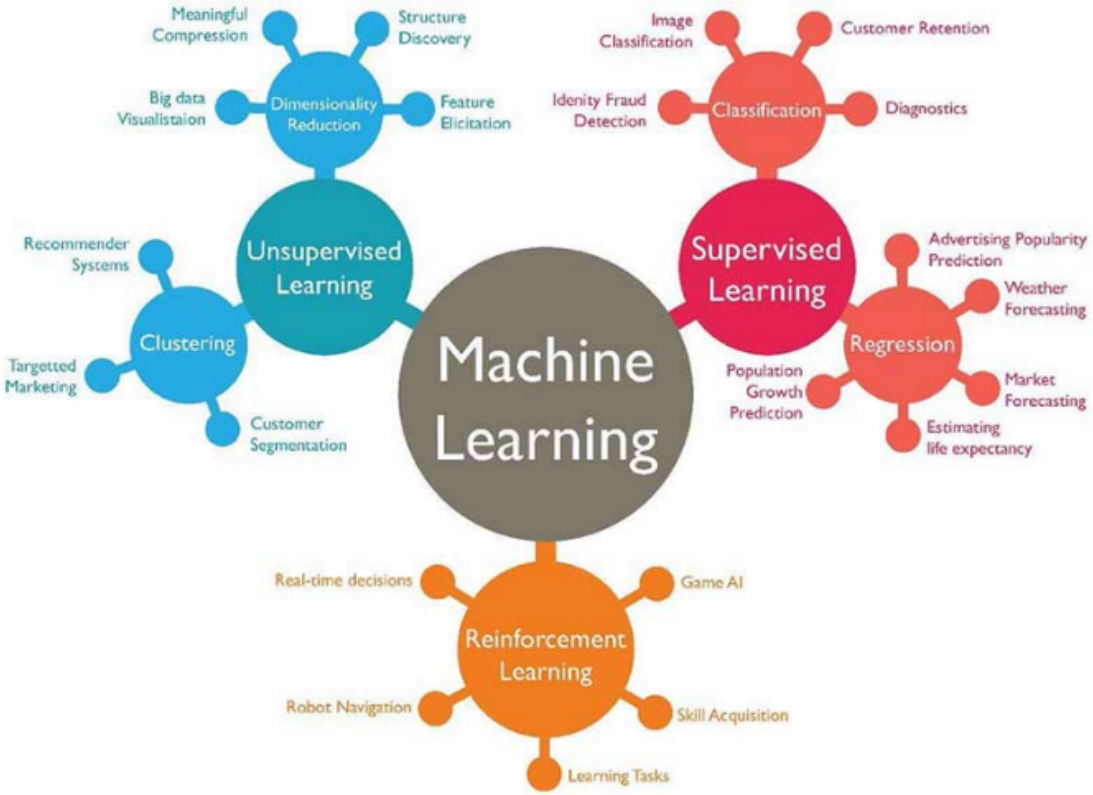


Slika 1 – Facial Recognition

Statističko učenje je pojam koji se često veže za mašinsko učenje i odnosi se na veliki broj različitih alata potrebnih za razumijevanje podataka [1]. Još 1959. godine Arthur Samuel je definisao pojam mašinskog učenja na način da je to oblast koje daje mogućnost računarima da uče bez da su eksplicitno programirani. Najpoznatija definicija mašinskog učenja danas je ona od Tom Mitchell-a koju je formulisao 1998. godine: kaže se da program uči iz iskustva E s obzirom na klasu zadataka T i mjerenja performansi P, ako se njegove performanse u zadacima T, mjerene kroz P, poboljšavaju sa svakim novim E. Da bi se bliže objasnila ova definicija, razmotrit će se sljedeći primjer: pretpostavimo da imamo program koji posmatra da li ste vi označili neki mail da je spam ili nije spam i na osnovu toga uči da bolje filtrira mailove. Za ovakvu postavku zadatka možemo reći da je zadatak T klasificiranje mailova kao spam ili nije spam. Iskustvo E je zapravo posmatranje programa kako korisnik označava mail da jeste ili nije spam, a performansa P je broj (ili odnos) mailova koji su ispravno označeni kao spam/nije spam.

1.1 Podjela mašinskog učenja

Osnovna podjela mašinsko učenja je na nadgledano (supervised) i nenadgledano (unsupervised)[2]. Na slici 2 prikazana je detaljnija podjela mašinskog učenja.



Slika 2 – Tipovi učenja

Generalno, nadgledano (supervised) učenje uključuje kreiranje statističkog modela za predikciju, ili estimaciju (procjenu), izlaza na osnovu jednog ili više ulaza. Problemi

ovog tipa se javljaju u različitim oblastima biznisa, medicine, astrofizike i javne politike. Kod nenadgledanog (unsupervised) učenja postoje ulazi, ali nemamo nadzor nad izlazom, tj. ne znamo kakav je izlaz. Bez obzira na to, moguće je pronaći odgovarajući šablon u ulaznim podacima i njihovoj strukturi. U nastavku će biti detaljnije objašnjena oba tipa mašinskog učenja.

Za razumijevanje mašinskog učenja neophodno je poznavati linearnu algebru i osnove vjerovatnoće i statistike.

2 Nadgledano učenje (supervised learning)

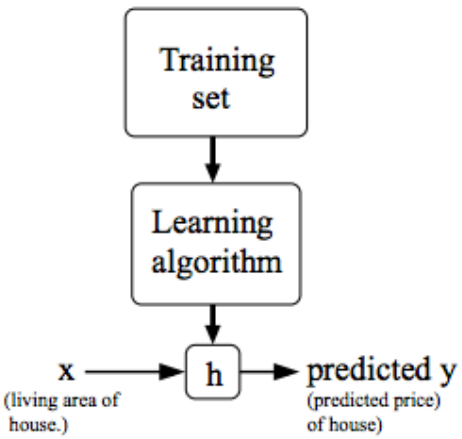
Problemi nadgledanog učenja (supervised learning) su svrstani u dvije kategorije: problem **regresije** i problem **klasifikacije**.

Nadgledano (supervised) učenje bazira se na tome da mi algoritmu damo skup podataka u kojem je dat "tačan odgovor". Na algoritmu je da proizvede "tačne odgovore" za druge primjere. Ovo se također zove i *regresija*, pokušavamo procijeniti kontinualnu vrijednost izlaza. Drugim riječima, želimo mapirati ulazne varijable u neku kontinualnu funkciju. Kod problema klasifikacije, pokušavamo izvršiti predikciju diskretnog (ne kontinualnog!) izlaza. Drugim riječima, mapiramo ulazne varijable u diskretne vrijednosti izlaza.

- Regresija - data je slika osobe, potrebno je odrediti broj godine osobe na osnovu date slike
- Klasifikacija - Dat je pacijent sa tumorom, potrebno je odrediti da li je tumor maligni ili benigni

U nastavku rada koristit će se oznaka $x^{(i)}$ za ulaznu varijablu (input feature) i $y^{(i)}$ za izlaznu varijablu (target variable). Par $(x^{(i)}, y^{(i)})$ se zove trening primjer, a skup podataka (dataset) koji ćemo koristiti da bismo "naučili" naš algoritam se sastoji od m-trening primjera $(x^{(i)}, y^{(i)}); i = 1, ..., m$.

Da bismo nešto formalnije opisali problem nadgledanog učenja, cilje je na osnovu datog skupa trening podataka naučiti funkciju $h : X \rightarrow Y$ tako da je $h_\theta(x)$ "dobar" prediktor za odgovarajuću vrijednost y. Iz historijskih razloga, funkcija h se naziva hipoteza (hypothesis). Grafički to izgleda kao na slici 3.



Slika 3 – Proces nadgledanog učenja

Varijable mogu biti karakterizirane kao *kvantitativne* ili *kvalitativne*. Kvantitativne varijable su vezane za numeričke vrijednosti. Na primjer, nečije godine, visina ili prihod, vrijednost kuće, i sl. Za razliku od toga, kvalitativne varijable su vezane za jednu od K različitih kategorija. Na primjer, spol osobe (muško ili žensko), tip vozila (malo

auto, kombi, kamion ili autobus), itd. Težimo da probleme sa kvantitativnim odzivom vežemo za problem regresije, a probleme kvalitativnim izlazom vežemo za problem klasifikacije.

Na nešto drugačiji način opisano, kada je izlaz koji želimo predvidjeti kontinualan, kao što je na primjer cijena nekretnine, pozivamo se na problem regresije. Kada izlaz može popriniti samo određeni (diskretni) broj vrijednosti (na primjer, ako na osnovu mjesta stanovanja želimo predvidjeti da li se radi o kući ili stanu), ovaj problem nazivamo problem klasifikacije.

2.1 Linearna regresija

Linearna regresija je koristan alat za predviđanje kvantitativnih odziva. Linearna regresija je od davnina poznat pojam i nezaobilazna je tema u mnogim knjigama. Bez obzira što je "star" alat, idalje je itekako koristan i široko rasprostranjen u praksi i jako je bitno razumijeti koncept linearne regresije da bi se shvatili kompleksnije metode mašinskog učenja.

Funkcija troška

Da bismo izmjerili tačnost predviđanja funkcije hipoteze, koristimo funkciju troška (cost function). Funkcija troška je najčešće srednjekvadratna greška, tj. uzima razliku između stvarnog rezultata i rezultata dobijenog iz funkcije hipoteze.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2 \tag{1}$$

U nekoj literaturi se može pronaći da nema $\frac{1}{2}$ ispred sume jer nećemo ništa izgubiti na opštosti, ali po konvenciji stoji $\frac{1}{2}$.

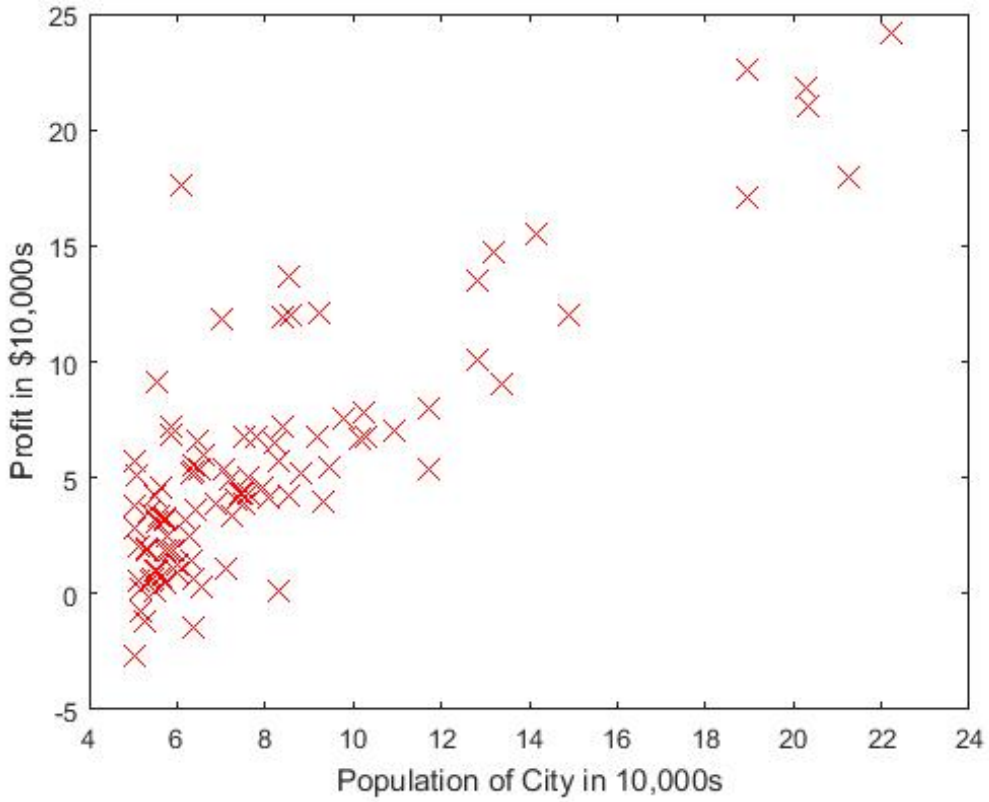
Problem ***linearne regresije*** posmatrat ćemo kroz jedan primjer. Pretpostavite da ste vlasnik nekog lanca restorana i razmatrate mogućnost da proširite svoje tržište. Vaš lanac već posjeduje kamione u različitim gradovima i imate podatke koje predstaljaju profit i populaciju tih gradova. Na slici 4 prikazan je grafik sa podacima.

Algoritam linearne regresije treba nam dati jednačinu prave koja na optimalan način fituje date podatke. Kao kriterij odabrat ćemo srednjekvadratnu grešku (opisano u prethodnom poglavlju). Cilj je, dakle, minimizirati funkciju troška kako bismo dobili optimalne parametre u relaciji 2.

$$h_\theta(x) = \theta_0 + \theta_1 x \tag{2}$$

Relacija 2 predstavlja funkciju hipoteze. Da bismo dobili parametre θ_0 i θ_1 potrebno je minimizirati relaciju 1.

Pošto imamo funkciju hipoteze i imamo način kako da mjerimo da li ta funkcija dovoljno dobro aproksimira (fituje) podatke, potrebeno je sada pokazati način dobivanja parametara funkcije hipoteze. Bit će objašnjena dva načina određivanja parametara: gradient descent (spuštanje gradijenta) i normal equation (jednačina normale)[3].



Slika 4 – Poznati podaci

Gradient descent algoritam

Kako je rečeno u prethodnom potpoglavlju, ovaj algoritam služi za minimizaciju funkcije troška. Za bolje objašnjenje ovog algoritma potrebno je nacrtati 3D grafik na kojem su dvije ose parametri θ_0 i θ_1, a treća osa predstavlja funkciju troška $J(\theta_0, \theta_1)$. Ovaj grafik prikazan je na slici 5.

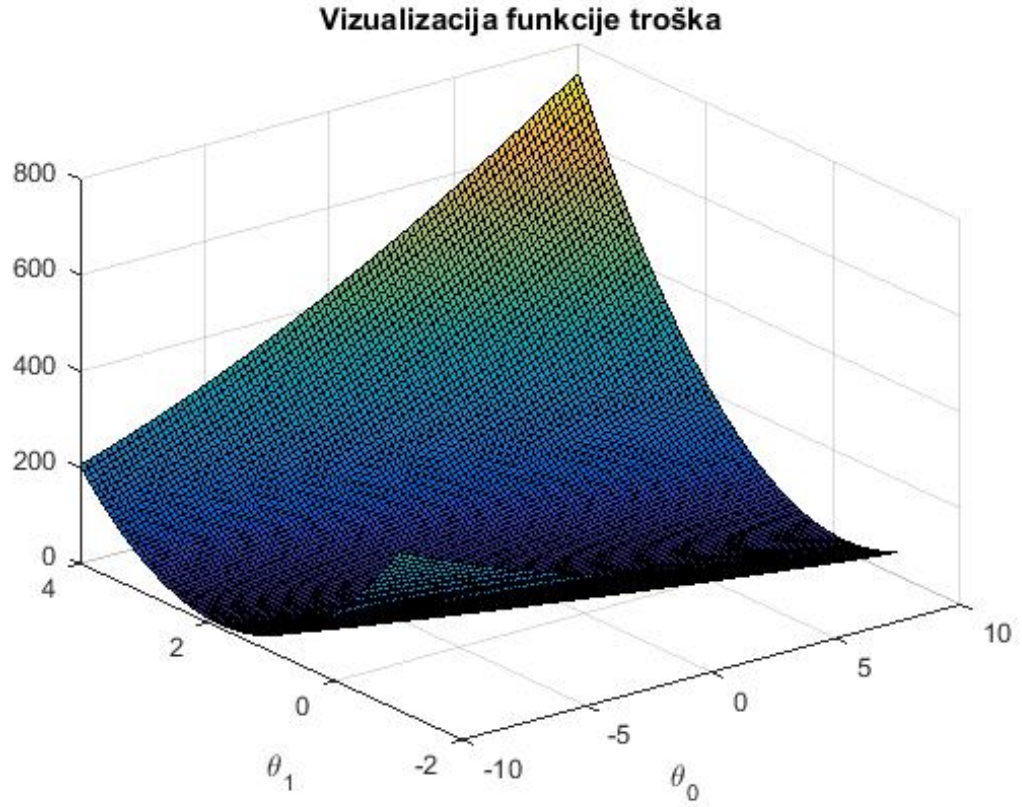
Dokaz da smo minimizirali funkciju troška je kada nađemo koordinate tačke koja predstavlja minimum funkcije prikazane na slici 5. Funkcija troška je ustvari kvadratna funkcija sa dvije varijable.

Da bismo došli do minimuma, prvo uzmemo neku slučajnu tačku na ovoj funkciji. Posmatramo nagib tangente na funkciju u toj tački i ta vrijednost treba da bude svakim korakom, po apsolutnoj vrijednosti, sve manja kako se približavamo minimumu funkcije. Veličina koraka kojim se približavamo minimumu je definisana parametrom a koji se naziva *stopa učenja*. Pseudo kod gradient descent algoritma je dat u nastavku:

repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

gdje je $j = 0, 1$ i predstavlja indeks atributa (feature index number). Svakom it-



Slika 5 – Funkcija troška[5]

eracijom j, istovremeno se mijenjaju parametri $\theta_1, \theta_2, ..., \theta_n$.

Parametar a ne smije biti ni prevelik ne premali. Ukoliko je preveliki, može doći do toga da nikada neće algoritam dospjeti u globalni minimum funkcije, tj. može se desiti da ne konvergira, nego divergira [3]. Ukoliko je parametar a premali, onda će gradient descent algoritam biti spor. Problem može biti ukoliko algoritam upadne u lokalni minimum, ali razmatrana tema je preširoka da bi se analizirali i drugi algoritmi.

Normal equation algoritam

Gradient descent algoritam daje jedan način minimizacije funkcije troška J. Drugi način je korištenjem "normalne jednačine" (normal equation). Kod ove metode uzimamo eksplicitno izvode J po θ_j i izjednačavamo ih sa nulom. Ovo nam omogućava da pronađemo optimalni vektor parametara θ bez iteracija. Jednačine normal equation algoritma:

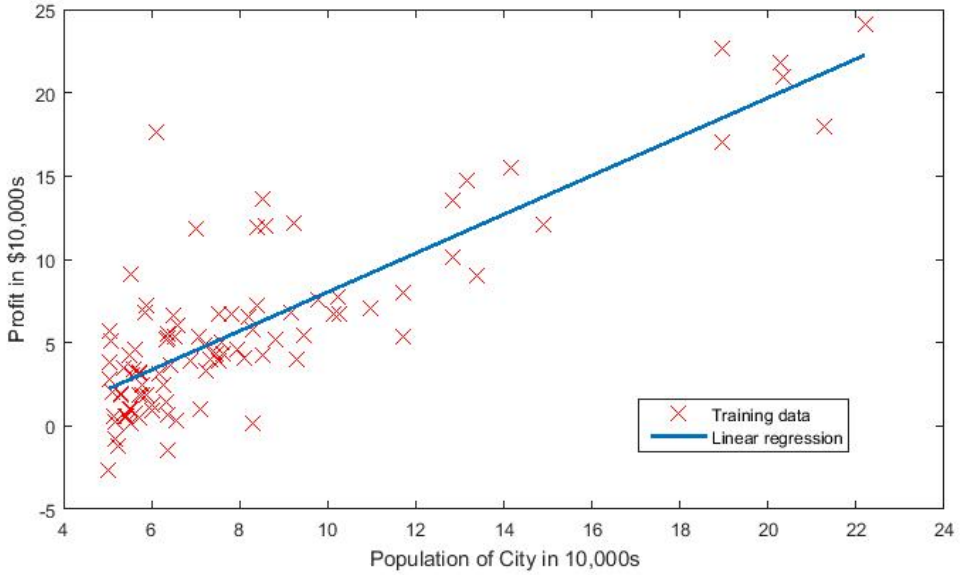
$$\Theta = (X^T X)^{-1} X^T Y$$

Usporedba ova dva algoritma minimizacije funkcije troška data je u tabeli 1.

Konačno, za razmatrani problem linearne regresije, kada primijenimo jedan od algoritama minimizacije funkcije troška dobit ćemo koeficijente θ_0 i θ_1 iz jednačine 2. To je zapravo jednačina prave i prikazana je na slici 6. Sa grafika na slici 6 možemo izvući zaključak da ukoliko želimo otvoriti radnju u gradu od 160000 stanovnika, očekivani

Tabela 1 – Usporedba Gradient descent i normal equation algoritma

Gradient descent algoritam	**Normal equation algoritam**
Potrebno izabrati parametar a	Ne treba izabrati parametar a
Zahtjeva mnogo iteracija	Nema iteracija
Kompleksnost: O(kn^2)	Kompleksnost: O(n^3)
Radi dobro i pri velikom n (broj atributa)	Spor algoritam za veliko n (broj atributa)



Slika 6 – Linearna regresija

profit je oko 150000$.

Do sada smo razmatrali model sa jednim ulazom i jednim izlazom. Međutim, intuitivno je da što više imamo atributa (features-a) to nam je bolji model. Linearna regresija sa više varijabli je poznata i kao multivarijabilna linearna regresija. Notacija za relacije u kojima možemo imati proizvoljan broj ulaznih varijabli je:
$x_j^{(i)}$-vrijednost j-tog atributa i i-tog trening primjera
$x^{(i)}$-ulaz (atributi) i-tog trening primjera
m - broj trening primjera
n - broj atributa.
Oblik funkcije hipoteze za multivarijabilnu linearnu regresiju je prikazan relacijom 3.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n \tag{3}$$

Vektorizirana implementacije ove funkcije hipoteze je:

$$h_\Theta(x) = \Theta^T X$$

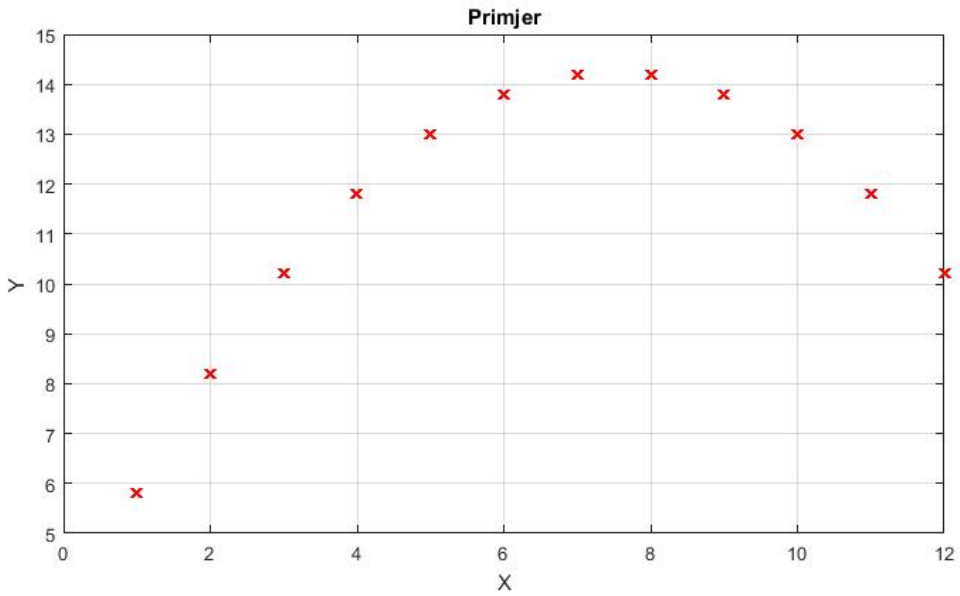
Po konvenciji je: $x_0^{(i)} = 1$ za svako i od 1 do m.

Gradient descent algoritam, generalno, ima istu formu kao i za prethodni slučaj.

2.2 Polinomijalna regresija

Funkcija hipoteze ne mora biti linearna ukoliko ne daje dovoljno dobre rezultate. Na primjeru prikazanom na slici 7 očigledno je da funkcija aproksimacije nije linearna već polinomijalna. Tada uzimamo neku drugu funkciju hipoteze, konkretno, za ovaj slučaj to je:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

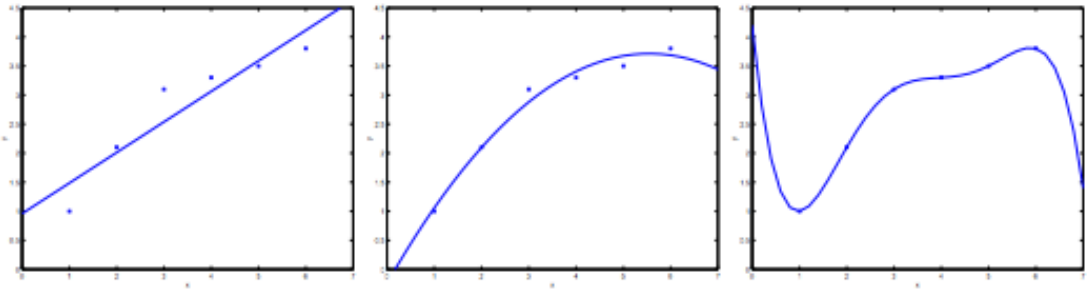


Slika 7 – Primjer podataka pogodnih za aproksimaciju polinomijalnom funkcijom drugog reda

Česta pojava je i veza između pojedinih atributa, tj. kod multivarijabilne regresije često imamo slučaj veze među pojedinim varijablama. Jako bitna stvar postaje skaliranje tih atributa zbog lakšeg pronalaženja globalnog minimuma funkcije troška. Na primjer, ukoliko se varijabla x kreće u opsegu od $1 - 100$, a u funkciji hipoteze se nalazi i četvrti stepen varijable x, to znači da će se vrijednost x^4 kretati od $1 - 100000000$.

Ukoliko dodamo neki dodatni atribut u funkciju hipoteze (na primjer x^2), recimo da imamo nešto bolje fitovanje podataka. Naivno se može zaključiti da dodavanjim drugih atributa imamo bolji model. Naravno, postoji opasnost od dodavanja prevelikog broja atributa. Na slici 8 prikazana su tri različita karakteristična slučaja. Ukoliko, na primjer, vršimo prodaju stanova na osnovu površine stana, desni grafik na slici 8 nam neće biti dobar prediktor jer ukoliko se pojavi stan sa dovoljno velikom površinom prediktor će kazati da je njegova cijena čak i manja nego stan od na primjer 15 kvadratnih metara. To je problem koji se naziva **overfitting**. Ukoliko imamo slučaj kao na lijevom grafiku na slici 8, to znači da imamo problem **underfitting**-a, tj. prediktor pravi preveliku grešku. Ako imamo previše atributa, hipoteza može fitovati trening skup podataka jako dobro, ali da nije prilagođena za nove primjere. Funkcija hipoteze koja daje najbolje rezultate je prikazana na srednjem grafiku na slici 8. Postoje dva načina rješavanja problema overfittinga:

- Smanjiti broj atributa
- Regularizacija - zadržati sve atribute, ali smanjiti vrijednost parametara Θ_j.



Slika 8 – Underfitting i overfitting

2.3 Klasifikacija

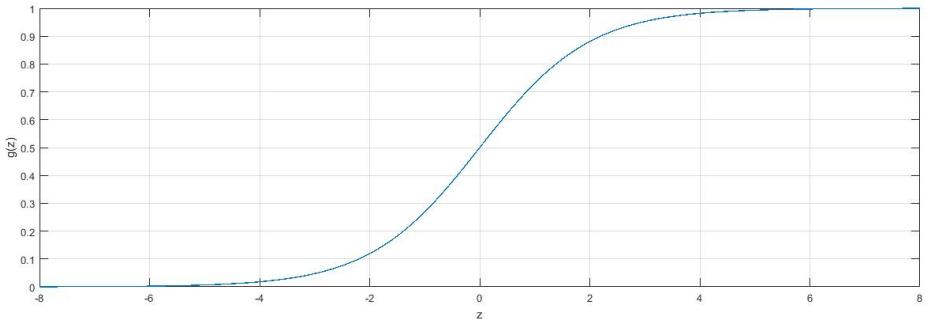
Za problem linearne regresije smo podrazumijevali da je izlazna varijabla Y kvnatitativnog karaktera. Međutim, u mnogim situacijama to nije slučaj nego je izlazna varijabla kvalitativnog karaktera. Na primjer, boja očiju je kvalitativna (plave, smeđe ili zelene). Predviđanje kvalitativnog odziva znači pripajanje nekog uzorka (ulazne varijable) nekoj kategoriji ili klasi. Obično metod klasifikacije prvo izračuna vjerovatnoću da će ulaz pripasti u svaku od kategorija kvalitativne varijable, a potom možemo tvrditi određenom sigurnošću koliko ulazna varijabla pripada određenoj klasi ili kategoriji. Postoje razne tehnike klasifikacije, ili klasifikatora, koji se mogu koristiti za predviđanje kvalitativnih odziva. Na primjer, logistic regression, linear discriminant analysis i K-nearest neighbors.

2.4 Logistička regresija

Korištenje linearne regresije kao klasifikatora ne daje zadovoljavajuće rezultate jer klasifikacije nije ustvari linearna funkcija. Nakratko ćemo ignorisati činjenicu da je riječ o diskretnim vrijednostima y. U nastavku ćemo razmotriti slučaj binarne klasifikacije, tj. vrijednost izlaza može biti 0 ili 1 (npr. spam/nije spam). Nema smisla da vrijednost funkcije hipoteze bude veća od 1 ili manja od 0. Stoga mijenjamo oblik funkcije hipoteze $h_\theta(x)$ tako da zadovoljava uslov $0 \leq h_\theta(x) \leq 1$. Novi oblik funkcije hipoteze koristi *sigmoid funkciju*, još poznatu kao logistička funkcija (Logistic Function):

$$\begin{aligned} h_\theta(x) &= g(\Theta^T x) \\ z &= \Theta^T x \\ g(z) &= \frac{1}{1+e^{-z}} \end{aligned} \tag{4}$$

Grafik funkcije 4 prikazan je na slici 9.



Slika 9 – Sigmoid funkcija

Funkcija $g(z)$ mapira bilo koji realan broj u interval između nula i jedan pretvarajući tako proizvoljnu funkciju u funkciju koja bolje odgovara klasifikaciji. Funkcija $h_\theta(x)$ će nam dati vjerovatnoću da je naš izlaz 1. Poznato je da vrijedi:

$$h_\theta(x) = P(y = 1|x; \theta) = 1 - P(y = 0|x; \theta)$$

Razmatramo primjer da li će student biti primljen na fakultet u odnosu na rezultat dva testa. Zadatak je napraviti model klasifikacije koji procjenjuje da li će kandidat biti primljen na fakultet na osnovu urađena dva testa. Poznati su sljedeći podaci od ranijih generacija prikazani na slici 10.

Da bi izlaz imao vrijednost 0 ili 1, razumijevamo vrijednost funkcije hipoteze na način:

$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$
$$h_\theta(x) < 0.5 \rightarrow y = 0$$

Od interese je pronaći seperatrisu, odnosno liniju koja razdvaja područje u kojem je $y = 0$ i područje u kojem je $y = 1$. Ona je kreirana funkcijom hipoteze.

Funkcija troška

Više ne možemo koristiti istu funkciju troška kao kod linearne regresije jer logistička regresija ima mnogo više lokalnih minimuma. Drugim riječima, neće biti konveksna funkcija. Zbog toga, funkcija troška za logističku regresiju ima oblik:

$$(\Theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

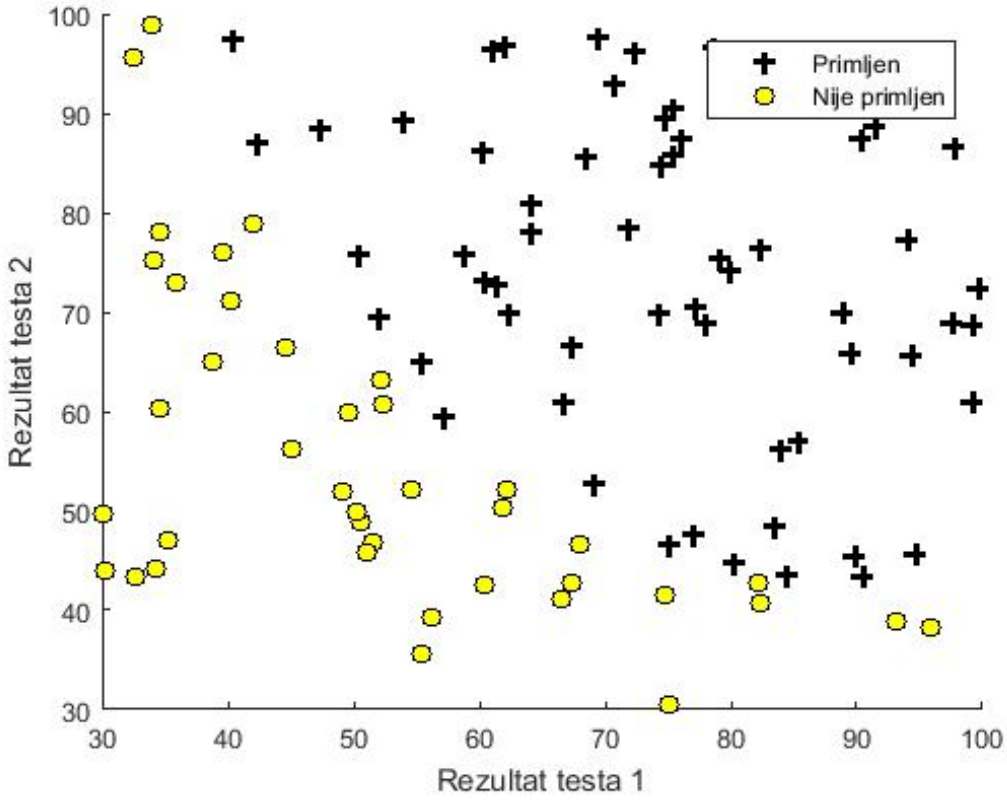
Ukoliko je $y = 1$:

$$Cost(h_\theta(x), y) = -log(h_\theta(x))$$

Ukoliko je $y = 0$:

$$Cost(h_\theta(x), y) = -log(1 - h_\theta(x))$$

Ovo je odgovarajući konveksni oblik funkcije troška za logističku regresiju. Potreban je konveksni oblik zbog pronalaska globalnog minimuma, tj. optimalnih parametara za



Slika 10 – Trening skup podataka

funkciju hipoteze $h_\theta(x)$. Prethodne dvije relacije možemo staviti u jednu (relacija 5).

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)})) + (1-y^{(i)})log(1-h_\theta(x^{(i)}))] \tag{5}$$

Gradient descent algoritam

Ovaj algoritam je praktično identičan kao kod linearne regresije.

repeat until convergence:

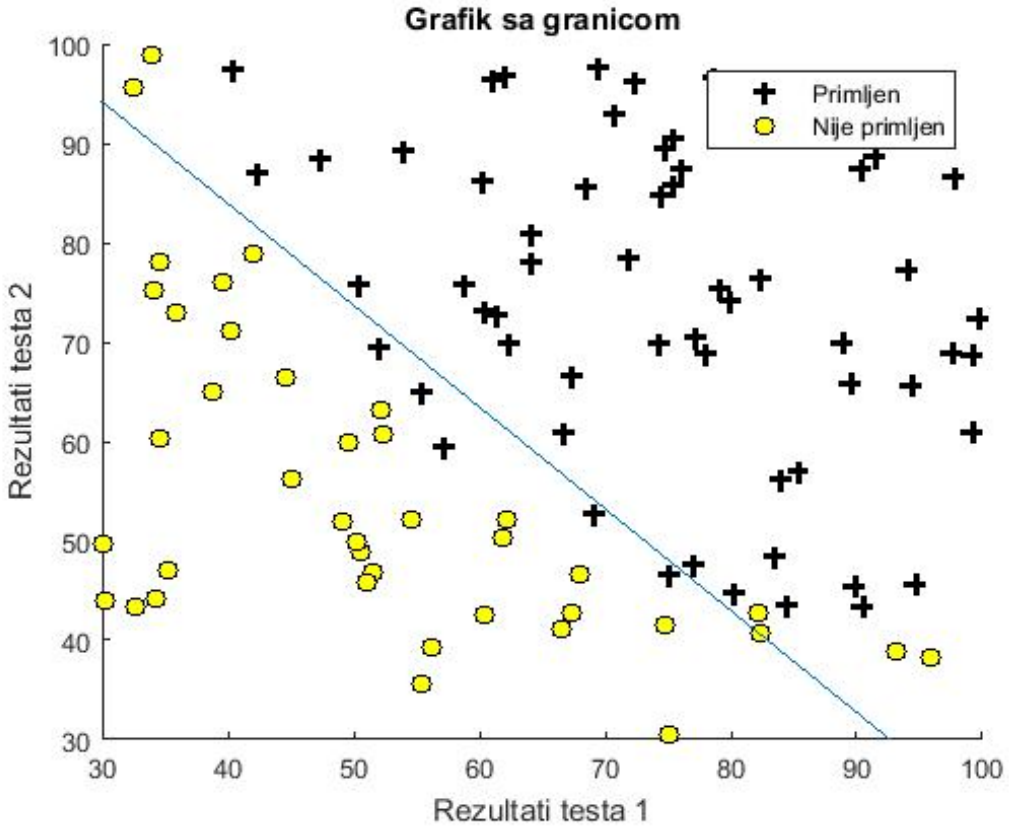
$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)$$

Opet moramo istovremeno mijenjati sve vrijednosti θ_j. Drugi sofisticirani algoritmi za optimizaciju θ:

- Konjugovani gradijent
- BFGS
- L-BFGS.

Za ove algoritme koristimo gotove bibilioteke.

Primjenjujući ovaj algoritam dolazimo do seperatrise i sa odgovarajućom vjerovatnoćom tvrdimo da će neki student biti primljen na fakultet na osnovu dva urađena testa (slika 11).



Slika 11 – Trening skup podataka sa granicom između dvije oblasti

2.5 Klasifikacija sa više kategorija

U prethodnom potpoglavlju smo razmatrali binarnu klasifikaciju gdje je izlaz mogao biti 0 ili 1. Sada imamo da je izlaz $y = \{0, 1, ..., n\}$. Sada dijelimo problem na $n + 1$ problem binarne klasifikacije gdje za svaki predviđamo vjerovatnoću da je y član jedne od naših klasa.

$$y \in \{0, 1, ..., n\}$$

$$h_\theta^{(n)}(x) = P(y = n|x; \theta)$$

$$predikcija = max(h_\theta^{(i)}(x))$$

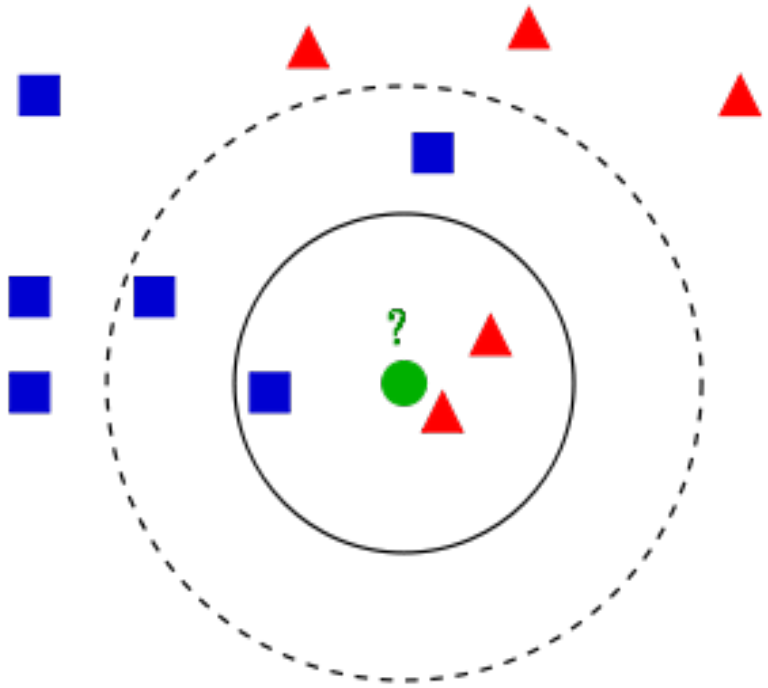
Drugim riječima, odaberemo jednu klasu, a sve ostale stavimo u drugu. To radimo nekoliko puta primjenjujući binarnu klasifikaciju za svaki slučaj i na kraju odaberemo funkciju hipoteze sa najvećom vjerovatnoćom.

2.6 K-nearest neighbors algoritam

K-nearest neighbors algoritam (KNN) ili algoritam K-najbližih susjeda je metoda koja se koristi i za klasifikacija i za regresiju. Princip rada je da bazirano na K najbližih uzoraka iz trening skupa podataka algoritam donosi odluku. Izlaz ovisi od toga da li je riječ o problemu klasifikacije ili regresije.

U slučaju klasifikacije, izlaz pokazuje da li je ulazni uzorak član neke klase. Na osnovu datog broja K i nekog primjerka x_0, KNN klasifikator prvo identificira K tačaka u trening skupu podataka koji su najbliži x_0. Objekat je klasificiran na osnovu "većinskih glasova" K susjeda. Ako je $K = 1$, onda je ulazni primjerak iste klase kao najbliži susjed.

U slučaju regresije, izlaz je kvantitativnog karaktera i predstavlja srednju vrijednost K najbližih susjeda.



Slika 12 – Primjer KNN klasifikacije

Na slici 12 prikazan je primjer KNN algoritma kao klasifikatora. Zeleni, testni, primjerak treba biti označen kao član plavih kvadratića ili crvenih trouglova. Ukoliko je $K = 3$ (puna kružnica), primjerak će biti dodjeljen crvenim trouglovima, a ukoliko je $K = 5$ (isprekidana kružnica) bit će dodjeljen plavim kvadratićima.

Odabir broja susjeda K ima ogromne posljedice na performanse alogirtma. Ukoliko je broj K mali, seperatrisa (decision boundary) "predobro" fituje trening podatke pa se javlja problem *overfitting*-a. Kako K raste, granica između dvije oblasti (klase) postaje sve fleksibilnija i teži ka tome da granica postane linearna funkcija. Ovo vodi ka problemu *underfitting*-a.

Matematički (relacija 6, za dati pozitivan broj K i primjerak x_0, KNN klasifikator prvo identificira K tačaka u trening skupu koji su najbliži primjerku x_0, predstavljeni sa N_0. Algoritam dalje računa vjerovatnoću da je izlaz $j - ta$ klasa kao odnos tačaka u

N_0 čiji je izlaz j.

$$P(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j). \tag{6}$$

gdje je I indikator varijable koji je ravan jedinici ako je $y_i = j$, a nuli u suprotnom slučaju.

Za računanje udaljenosti između kontinualnih varijabli koristi se Euklidska udaljenost (relacija 7), a za diskretne varijable Hammingova udaljenost (relacija 8).

$$d(x, y) = \sqrt{(\sum_{i=1}^{k}(x_i - y_i)^2)} \tag{7}$$

$$D_H = \sum_{i=1}^{k} |x_i - y_i| \tag{8}$$

Kod Hammingove udaljenosti vrijedi:

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

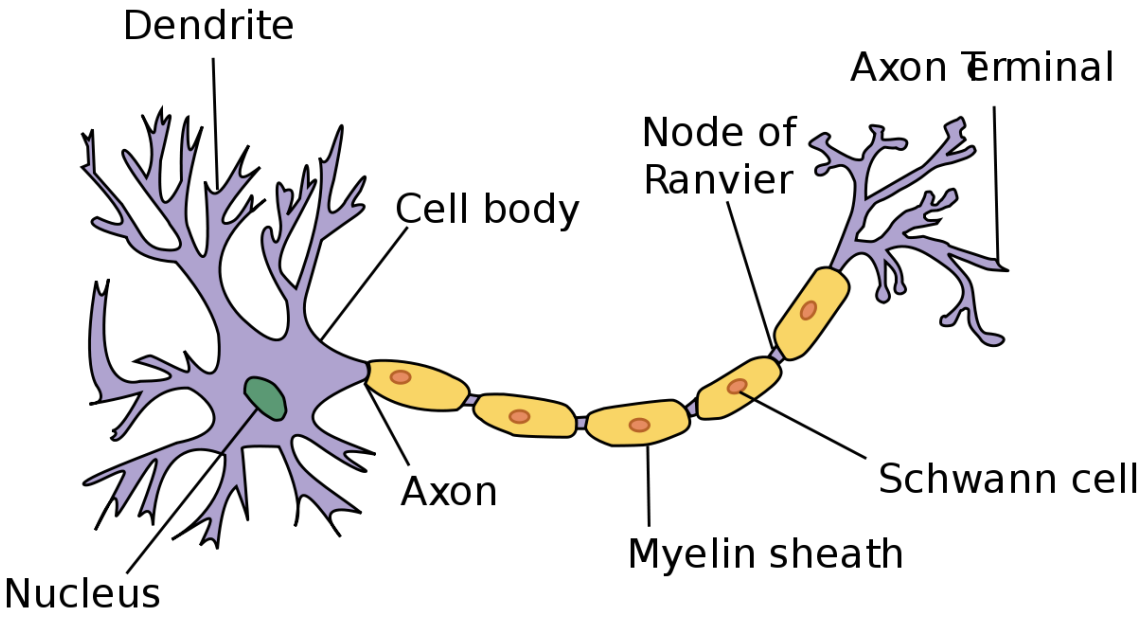
KNN algoritam se koristi još od 1970-ih godina i najpoznatija je neparametarska tehnika za klasifikaciju (nema nikakvih koeficijenata koji se trebaju računati kao kod drugih obrađenih metoda).

2.7 Neuronske mreže

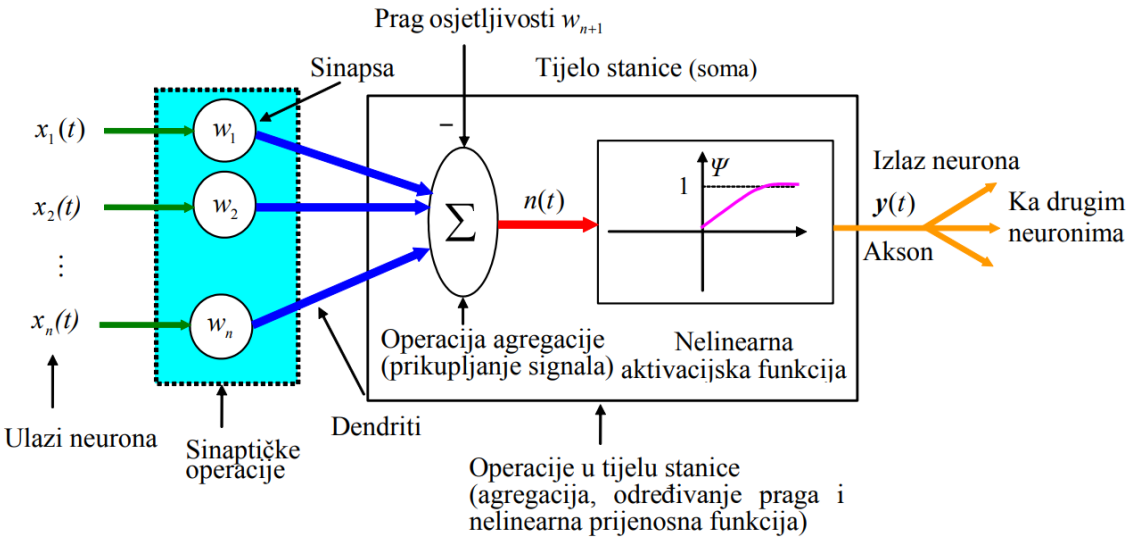
Jedan od najboljih algoritama učenja jesu neruonske mreže. Ideja je poprilično stara, ali danas je state-of-the-art tehnika. Neuronske mreže su fantastična metoda mašinskog učenja kada imamo nelinearnu funkciju hipoteze i kada imamo ogroman broj atributa. Činjenica je da za većinu problema u mašinskom učenju, n (broj atributa) je velik.

Ovaj algoritam pokušava oponašati rad mozga. Izgled neurona prikazan je na slici 13. Neuronske mreže su razvijene kroz simulaciju neurona ili mrežu neurona u mozgu. Posmatrajmo jedan neruon: neuron je ćelija u mozgu. Neuron posjeduje tijelo, određeni broj ulaza (dendrita) i izlaza (axon) preko kojih jedan neuron šalje poruku drugim neuronima. Na pojednostavljenom nivou, neuron je računarska jedinica koja prima neke podatake, nešto proračuna i onda to pošalje dalje drugim neuronima. Način na koji neuroni u mozgu komuniciraju je preko malih pulseva elektriciteta. Ovo je način kako naš organizam funkcioniše: mozak pošalje signal mišiću da se zgrči, oku da trepne, itd. Na slici 14 prikazan je matematički model neurona. Općenita struktura vještačke neuronske mreže prikazana je na slici 15. Struktura se sastoji od ulaznog sloja, skrivenog sloja i izlaznog sloja. Moguće je da postoji više od jednog skrivenog sloja. Neka od područja primjene neuronskih mreža su:

- prepoznavanje uzoraka i klasifikacija
- obrada slika
- identifikacija i upravljanje sistemima automatskog upravljanja



Slika 13 – Neuron

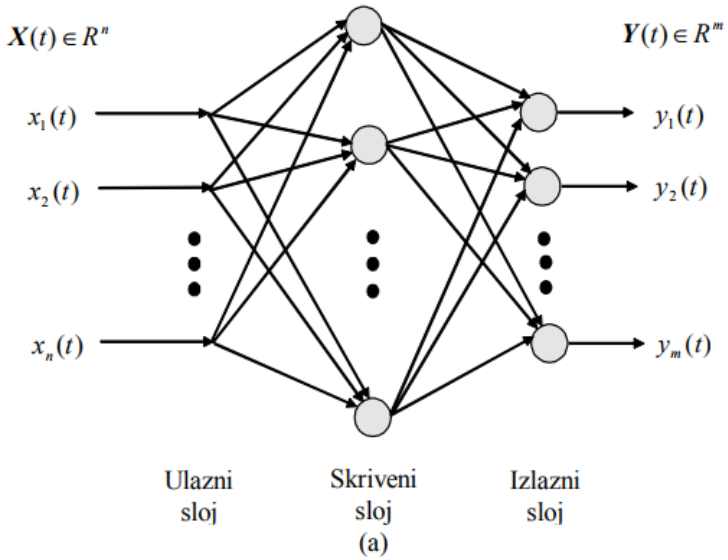


Slika 14 – Matematički model neurona[6]

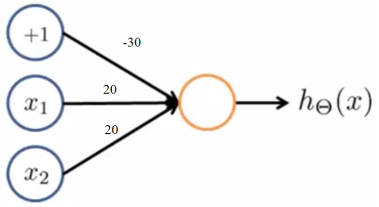
- obrada signala

Primjer

U ovom primjeru implementirat ćemo logičku funkciju AND (I) korištenjem neuronskih mreža. Neka su ulazi x_1 i x_2 i da mogu poprimiti vrijednost 0 ili 1. Naš cilj je implementirati funkciju $y = x_1$ AND x_2 - logičko I. Na slici 16 je prikazana šema neuronske mreže. Ova mreža nema skrivenih slojeva. Brojevi koji su napisani iznad linija na dijagramu predstavljaju koeficijente gdje je koeficijent -30 u vezi sa x_0, 20 u vezi



Slika 15 – Struktura vještačke neuronske mreže[6]



Slika 16 – Primjer

sa x_1 i 20 u vezi sa varijablom x_2. To je zapravo isto kao da smo napisali:

$$h_\Theta(x) = g(-30 + 20x_1 + 20x_2).$$

Aktivacijska funkcija je zapravo sigmoidalna funkcija prikazana na slici 9. Tablica istine prikazana je u tabeli 2.

Tabela 2 – Tablica istine

x_1	x_2	$h_\Theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

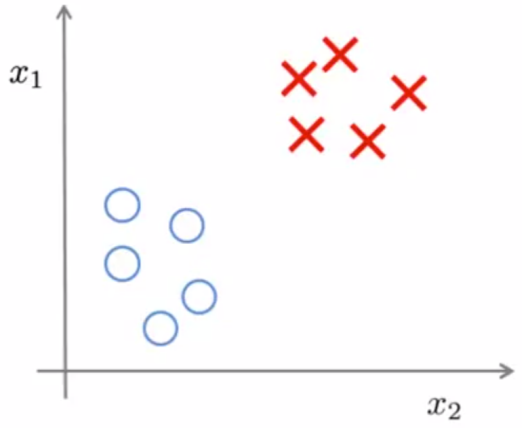
Ovaj jednostavan primjer pokazuje realizaciju logičke I funkcije. Spajanjem više ovakvih realizacija sličnih funkcija može se kreirati jako nelinearan model sa veoma dobrom performansom jer sve ćelije rade paralelno.

Vrijedi napomenuti da postoji SVM (Support Vector Machine) klasifikator koji jako

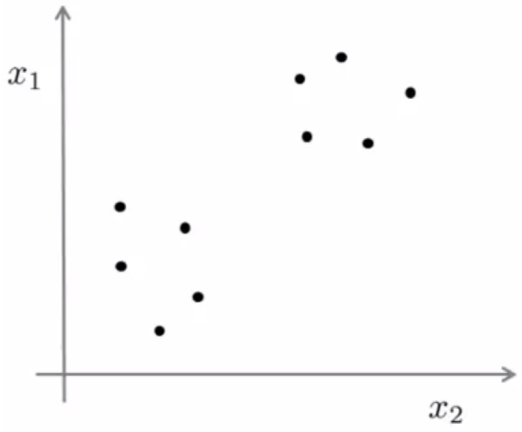
sličan logističkoj regresiji, s tim da koristi obrnutu logiku. Ukoliko imamo mali broj atributa, a ni mali ni veliki trening skup podataka, koristimo SVM klasifikator. Ukoliko je n veliko, koristimo ili logističku regresiju ili SVM. Ukoliko je n malo, lahko je dodati dodatne atribute. Neuronske mreže odlično rade za većinu slučajeva, ali je potrebno dosta vremena da se one istreniraju (tj. da se pronađu optimalne vrijednosti koeficijenata).

3 Nenadgledano učenje (unsupervised learning)

Nenadgledano učenje nam omogućuje pristup rješavanju problema kod kojih ne znamo kvakvi bi rezultati trebali biti. Ono što radi algoritam nenadgledanog učenja je da kada mu se proslijede neki podaci, on pronađe neku strukturu/vezu među tim podacima. Na primjer, na slici 17 vidimo primjer nadgledanog učenja gdje nam je bilo od interesa pronaći liniju koja razdvaja površine sa različitim izlaznim vrijednostima. Kod nenadgledanog učenja (slika 18) imamo skup podataka kod kojih ne znamo kakav je izlaz. Intuitivno je podijeliti ove podatke u dvije grupe. Algoritam koji pronalazi ove grupe naziva se *Clustering* (eng. cluster - grupa).



Slika 17 – Primjer nadgledanog učenja



Slika 18 – Primjer nenadgledanog učenja

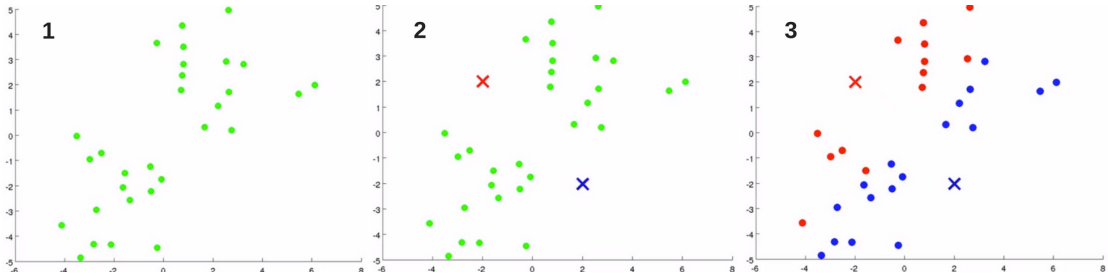
Primjena klasteringa je široka: segmentacija tržišta, razne analize društvenih mreža, analiza podataka iz astronomije, itd. **K-means** algoritam je najpoznatiji algoritam grupisanja (pripada algoritmu klasterizacije).

3.1 K-means algoritam

K-means je iterativni algoritam i radi dvije stvari[4].

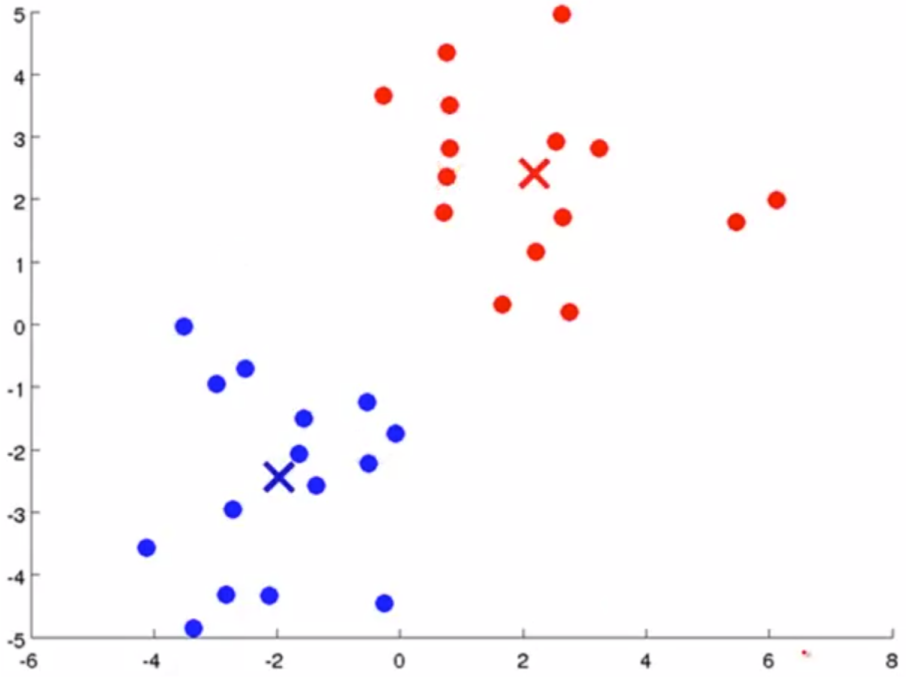
- Pridruživanje grupi
- Pomjeranje centroida

Prvo se slučajno odaberu početne tačke centroida (slučajno)[1]. Nakon toga se svakom centroidu pridruže najbliže tačke (slika 19) što predstavlja prvi korak. Drugi korak



Slika 19 – Koraci K-means algoritma[4]

je pomjeranje centroida na mjesto koje predstavlja srednju vrijednost udaljenosti od svih elemenata koji pridaju datom centroidu. Nakon toga slijedi ponovo prvi korak, tj. pridruživanje grupi, potom pomjeranje centroida i tako sve dok se ne prestanu mijenjati elementi svake od grupa. Nakon ovih iteracija dobiju je dvije različite grupe prikazane na slici 20. U nastavku je prikazan pseudo-kod za K-means algoritam.



Slika 20 – Dvije različite grupe dobivene K-means algoritmom[4]

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.

2. Iterate until the cluster assignments stop changing:

a) For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
b) Assign each observation to the cluster whose centroid is closest (where closest is defined using Eucliedan distance).

Grupisanje (klasterizacija) može biti vrlo koristan alat za analizu podataka kod nenadgledanog učenja. Ipak, postoje neki problemi koji se pojavljuju kod metoda grupisanja:

- Male odluke sa velikim posljedicama - U koliko grupa da podijelim podatke? Trebaju li podaci biti standardizovani, skalirani? Na ova pitanja nema tačnog odgovora. Svako rješenje koje daje neku zanimljivu strukutru podataka treba biti razmotreno;
- Validacija dobivenih grupa - potrebno je izvršiti validaciju dobivenih grupa nakon izvršenja algoritma;
- Ponekad će neki podaci biti dodijeljeni nekoj grupi iako to ne bi trebalo, naročito ukoliko se pojavi neki outlier (podatak koji je drastično različit od drugih)[1].

4 Zaključak

Može se zaključiti da su algoritmi mašinskog učenja primjenjivi u svakodnevnom životu i da predstavljaju temelj inteligentnih sistema. Inteligentnost sistema koje napravi čovjek ovise u mnogo čemu od kreativnosti koju ima čovjek - projektant tog sistema. Za očekivati je novu tehnološku revoluciju u smislu pojave autonomnih vozila i drugih sistema koji pripadaju oblasti umjetne inteligencije. Većina algoritama mašinskog učenja je i matematski dokazana što daje dodatnu sigurnost pri projektovanju ovakvih sistema.

5 Reference

1 Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, An Introduction to Statistical Learning - with Applications in R

2 David Barber - Bayesian Reasoning and Machine Learning

3 Joel Grus - Data Science from Scratch

4 Andrew Ng, https://www.coursera.org/

5 MATLAB documenatiation - https://www.mathworks.com/

6 Jasmin Velagić - http://people.etf.unsa.ba/ jvelagic/laras/index.php