**Algoritmi za prepoznavanje rukopisa pomoću mašinskog učenja (machine learning)**

Mašinsko učenje ima tri osnovna tipa:

1.Nadzirano učenje (Supervised learning)

Potrebno je da osoba koja nabavlja podatke odradi i klasifikaciju podataka (za probleme klasifikacije), tako da su mašini dostupni i ulazni i izlazni podaci. Učenje se provodi sa mnogo podataka koji su podjeljeni u dvije grupe (podaci za učenje i podaci za testiranje). Ukoliko je klasifikacija dobra, sa malim procentom grešaka, ovakav program se može koristiti i za klasifikaciju novih podataka iz istog tipa. Ovaj tip učenja se koristi u prepoznavanju slova, brojeva i rukopisa.

2.Nenadzirano učenje (Unsupervised learning)

Ovaj tip mašinskog učenja se zove i grupisanje (clustering), a primjenjuje se kod podataka za koje se ne pravi nikakva prepostavka (izbori, glasanja itd.)

3. Ojačano učenje (Reinforcement learning)

Ovde u algoritmu postoji agent koji vrši procjenu trenutne situacije i odluke donosi na osnovu krajnjeg rezultata. Ovaj tip mašinskog učenja koriste programi za šah. Izbor poteza se procjenjuje preko krajnjeg uspjeha igrača koji je u istoj situaciji povukao takav potez (IBM- ov Deep Blue je pobijedio svjetskog prvaka Garija Kasparova, koristeći ovaj tip učenja).

Mašinsko učenje je alat za pretvaranje mnoštva podataka u znanje (način na koji su podaci povezani). Problem prepoznavanja rukopisa je prvi put riješen sa prepoznavanjem brojeva na primjeru MNIST 60000 brojeva za učenje i 10000 za testiranje. 250 radnika američkog instituta za standardizaciju i tehnologiju i 250 studenata su napisali rukom ove brojeve i odredili koji broj slika predstavlja. Ovaj problem je riješen MLP neuronskom mrežom (Multi layer perceptron neural net), uz tačnu kategorizaciju za više od 97% brojeva. U seminarskom radu je dat i python kod ovoga rješenja. McCulloh Pitts perceptron je model koji su dala ova dva naučnika, koji su matematičkim modelom opisali način rada čovječijeg mozga. Nekoliko godina poslije Frank Rosenblatt je objavio radove koji su predstavili pravila učenja bazirana na MCP modelu u radu Perceptron, automatizacija svjesnog zaključivanja i prepoznavanja (The Perceptron,Perceiving and Recognizing Automaton 1957.) Sa pravilima koji važe za perceptron, algoritam koji je dao Rosenblatt može automatski učiti optimalne težinske koeficijente koji se množe sa ulaznim podacima, da bi se donijela odluka da li će neuron provesti signal ili ne (Aktivacija neurona). U kontekstu nadziranog učenja i klasifikacije ovakav algoritam se može upotrabljavati za predviđanja – pripada li uzorak jednoj ili drugoj klasi.

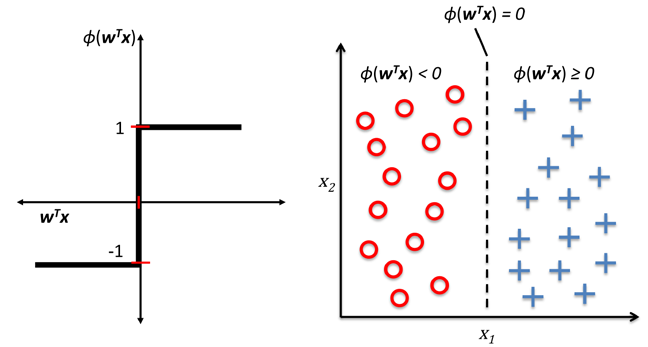
Najjednostavniji problem je binarna klasifikacija u kojoj imamo dvije klase – pozitivnu (1) i negativnu klasu (-1). Možemo definisati i funkciju aktivacije **Φ(z)** koja kao argument uzima linearnu kombinaciju ulaznih veličina **x** i vektor težinskih faktora **w,** dok **z** zovemo neto ulaz (z=w1x1+ w2x2+....+ wmxm).

Aktivacija pojedinačnog primjera vektora ulaznih veličina **x(i),** to je izlaz **Φ(z),** ukoliko je **z** veći od granične vrijednosti **θ,** tada **x** pripada klasi 1 (**Φ(z)=1**), ili klasi -1 (**Φ(z)=-1**), ako je **z< θ.** To nam govori da je funkcija aktivacije u perceptron algoritmu Hevisajdova (Haeviside) jedinačna step funkcija.

Zbog jednostavnosti možemo graničnu vrijednost **θ** predstaviti kao **w0=- θ** i **xo=1** , tako da dobijamo

**z= w0x0+ w1x1+ w2x2+....+ wmxm = wT x ,** a **Φ(z)=1** za **z> θ,** ili **Φ(z)=1** za **z< θ.**

Prikaz kako se neto ulaz **z = wT x** preslikava preko funkcije aktivacije perceptrona na binarne izlaze 1 ili -1 dat je na lijevoj slici, a na desnoj su prikazane dvije linearno djeljive klase podataka.



Osnovna ideja na kojoj je zasnovan MCP neuron i Rosenblattova granična vrijednost peceptronovog modela je predstavljena preko modela jednog perceptrona u mozgu, koji prenosi ili ne prenosi informaciju. Rosenblattovo inicijalno pravilo peceptrona je jednostavno i može se prikazati u sljedećim koracima:

1.Inicijalizuju se težinski faktori kao 0 ili mali random brojevi.

2. Za svaki ulazni vektor **x(i)** , koji se koristi za učenje provode se sljedeći koraci:

1. Računaju se izlazne vrijednosti *y*ˆ.

2. Vrši se korigovanje težinskih faktora.

Ovde je izlazna vrijednost oznaka klase podatka koju daje kao izlaz step funkcija koju smo definisali ranije. Simultana korekcija svakog težinskog koeficijenta **wj** se može prikazati kao:

**wj:= wj + Δ wj**

vrijednost korekcije težinskog faktora **Δ wj** je

**Δ wj =η(y(i) - *y*ˆ(i))**

**η-** korak učenja (learning rate), mala vrijednost između 0.0 i 1.0

**y(i) –** je stvarna klasa kojoj pripada vektor ulaza – u nadziranom učenju imamo i ulazne i izlazne podatke

***y*ˆ(i) –** ovo je prepostavljena vrijednost pripadnosti klasi za vrijeme učenja.

Važno je pomenuti da se svi težinski faktori koriguju simultano tako da se vijednost ***y*ˆ(i)** ne mijenja dok se svi **Δ wj** ne koriguju. Konkretno za 2D ulazne vektore imamo:

**Δ w0 = η(y(i) - izlaz(i))**

**Δ w1 = η(y(i) - izlaz(i))x(i) 1**

**Δ w2 = η(y(i) - izlaz(i))x(i) 2**

Prije implementacije algoritma u pythonu potrebno je prikazati koliko je jednostavno pravilo učenja u perceptronovom algoritmu. Imamo dva scenarija za gdje je perceptron ispravno predvidio težinski faktor :

**Δ wj = η(1(i) - 1))x(i) j=0 (ispravno odredio klasu 1)**

**Δ wj = η(-1(i) -- 1))x(i) j=0 (ispravno odredio klasu -1)**

Znači da za dobro klasiranje nema korekcije.

Kod progrešno klasiranja imamo takođe dva slučaja:

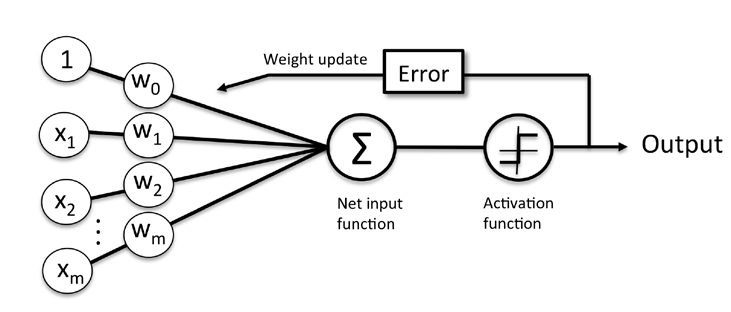
**Δ wj = η(1(i) -- 1))x(i) j= η(2)x(i) j** (za pozitivnu klasu predvidio negativnu)

**Δ wj = η(-1(i) - 1))x(i) j= η(-2)x(i) j** (za negativnu klasu predvidio pozitivnu).

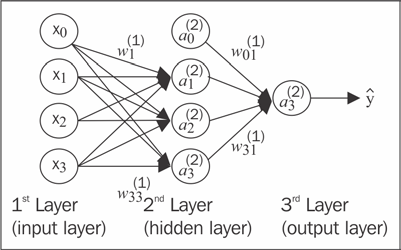
Ovim se dobija da se u slučaju pogrešnog određivanja klase korekcija vrši u pravom smijeru (krivo određena pozitivna klasa dobija korekciju u pozitivnom smjeru, dok krivo određena negativna klasa doboja korekciju u negativnom smjeru).

Važno je napomenuti da perceptron ispravno funkcioniše samo ako su klase linearno djeljive, tj. ako se između podataka može povući prava (za slučaj 2D) ili ravan (za slučaj 3D).

Grafički prikaz perceptron algoritma



Poboljšanje osobina perceptron algoritma je povezivanje više pojedinačnih neurona u **višeslojnu prema naprijed usmjerenu neuronsku mrežu (multi-layer feedforward neural network.** Konkretan tip neuronske mreže koja se objašnjava je MLP više slojni perceptron (**multi-layer perceptron).** Slika objašnjava koncept MLP koji se sastoji od tri sloja: ulazni sloj, **skriveni sloj** i izlazni sloj. Elementi skrivenog sloja su povezani na ulazni sloj, a izlazni spoj je povezan na skriveni sloj. Ako skriveni sloj zamijenimo sa više slojeva, takvu mrežu zovemo **dubokom umjetnom neuronskom mrežom** (**deep artificial neural network).**



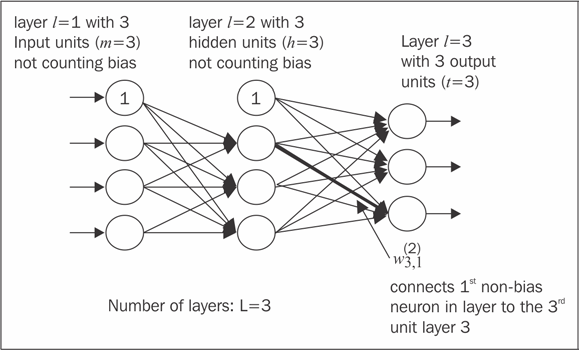
Možemo uzeti više slojeva da bi stvorili „dublju“ mrežnu arhitekturu. Ovim se pojavljuje više hiperparametara preko kojih možemo za rješenje problema razmatrati unakrsno vrednovanje.

U tom slučaju dublje neuronske mreže imaju problem nestajanja gradijenta, jer se traži gradijent funkcije greške u više slojeva, a metoda ima cilj nalaženja lokalnog minimuma (smanjivanje gradijenta) funkcije greške na engleskom **gradient descent.** Zbog ovog nestajanja gradijenta je za „duboke“ mreže razvijen poseban algoritam koji ima novo skaliranje veličina u svakom koraku. Ove metode se zovu dukoko učenje (**deep learning).**

Kao što je prikazano prethodno, i-tu aktivacionu jedinicu l-tog sloja označavamo sa **ai(l),** a **a0(1) i a0(2)** predstavljaju anomaliju, koja ima veze sa skaliranjem engl. **Bias units,** koje mi zadajemo kao 1. Aktivacija jedinica ulaznog sloja anomalija (x0=1) i ulazne veličine . Svaka jedinica sloja **l** ulaznihje povezana sa svakom jedinicom sloja **l+1** preko težinskih koeficijenata. Npr. **k**-ta jedinica u sloju **l**  prema **j-**toj jedinici u sloju **l+1** se označava **wj,k(l).** Treba obraiti pažnju daeksponent i kod vektora **x(i)** označava i- ti uzorak ulaznih podataka, a ne i-ti sloj.

U slučaju klasifikacija sa više klasa izlazna veličina koristi tehniku **Jedan od svih (engl. OvA= One vs All)**

Imamo izlazni vektor sa jedinicom na mjestu klase i sve ostale 0 . npr. [0 1 0 0 0 0 0 0 0 0]T  za klasu brojeva 1. Zbunjujuća je notacija indeksa **wj,k(l), a ne wk,j(l).** Ovo postaje jasnije kada se vektorizuje prikaz neuronske mreže. Npr. Mi skupimo sve koeficijente težinskih koeficijenata ulaznog i skrivenog sloja u matricu **W(1)** ∈ **Rhx[m+1]**, gdje su h broj skrivenih neurona su sloju , a m +1 broj skrivenih neurona + 1 (bias). U prilogu je data ilustracija:



Potrebno je objasniti kako se **kretanjem napred (eng. Forward propagation)** računa izlaz MLP modela. Da bi razumjeli učenje MLP modela, povezaćemo proceduru učenja u tri jednostavna koraka:

1. Počevši od ulaznog sloja sa ulaznim podacima za učenje računamo mrežu da bi dobili izlaz.

2. Na osnovu izlaznih podataka računamo grešku, koju želimo minimizovati koristeći funkciju troška (ili greške) koju ćemo objasniti poslije.

3. Unazad vraćamo grešku, nalazimo njene derivacije prema svakom težinskom koeficijentu u mreži i korigujemo model.

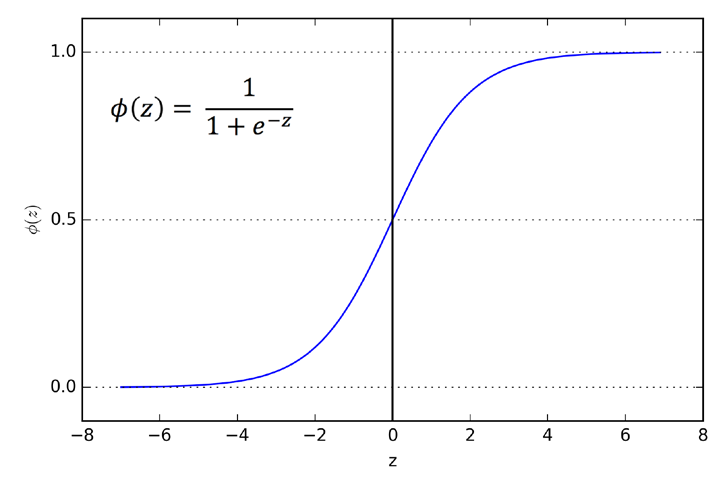
Konačno nakon mnogih epoha (epoha =prolazak kroz komplet podataka za učenje) i učenja težinskih faktora u MLP, idemo sa računanjem izlaznih vektora, koji su objašnjeni ranije.

Sada ćemo napraviti jedan korak prema napred da bi smo generisali izlaz od uzorka podataka za učenje. Svaka jedinica skrivenog sloja je povezana sa svakom ulaznom jedinicom, pa prvo računamo **a1(2)**

**z1(2)= a0(1) w1,0(1)+ a1(1) w1,1(1)+...+ am(1) w1,m(1)**

**a1(2)=Φ(z1(2))**

Ovde je **z1(2)** neto ulaz, a **Φ(\*)** funkcija aktivacije, koja mora biti diferencijabilna da bi mogla omogućiti učenje težinskih koeficijenata koji povezuju neurone koristeći gradijente. Za kompleksne probleme ova funkcija aktivacije mora biti nelinearna. Uzeta je **sigmoidna (logistička)** funkcija, koja ima potrebne osobine. Oblik sigmoidne funkcije je **Φ(z)=1/(1+e-z).** Ova funkcija ima veoma korisne osobine: neparna je, beskonačno derivabilna, preslikava domen na kodomen (0,1), a derivacija ove funkcije je **Φ'(z)= Φ(z)(1- Φ(z)** što je korisno u slučaju višestrukog deriviranja. Prikaz funkcije sigmoid je dat na slici:



MLP je primjer naprijed usmjerena neuronska mreža (feedforward neural network). Naprijed usmjerena neuronska mreža objašnjava da izlaz svakog sloja postaje ulaz sljedećeg sloja bez petlji (povratne sprege), što je osobina rekurentnih neuronskih mreža. Termini više-slojna perceptronska mreža je zbunjujući, jer kod ove mreže neurone predtavljaju sigmoidne jedinice, a ne perceptroni (aktivacijska funkcija je sigmoid, a ne Hevisajdova step funkcija). Intuitivno možemo zamisliti MLP neurone kao jedinice logističke regresije koje kontinualno vraća vrijednosti između 0 i 1 i može se zamisliti kao vjerovatnoća.

Zbog efikasnosti koda i čitljivosti sada ćemo opisati aktivaciju u kompaktnijoj formi koristeći bazne koncepte linearne algebre, što će nam omogućiti da vektorizujemo implementaciju koda preko NumPy, i tako izbjegnemo pisanje višestruko ugniježđene python for petlje.

**z(2)=W(1) a(1)**

**a(2)=Φ(z(2))**

Ovde je **a(1)** naš [m+1]x1 dimenzionalni vektor koji obuhvata ulazni vektor **x(i)**  i bias jedinicu . **W(1)** je hx[m+1] dimenzionalna matrica težinskih koeficijenata, gdje je h broj jedinica u skrivenom sloju. Nakon množenja matrice i vektora imamo hx1 neto ulazni vektor **z(2)** preko koga računamo aktivaciju **a(2)** (a(2) ∈ Rh+1). Šta više, možemo generalizovati ovaj proračun za kompletan skup svih n ulaznih podataka u skupu za učenje.

**Z(2)=W(1) [A(1)]T**

Ovde je **A(1)** jedna nx[m+1] matrica, a množenje ove dvije matrice daju hxn dimenzionalnu neto ulaznu matricu **Z(2).** Konačno primjenjujemo funkciju aktivacije **Φ(\*)** na svaku vijednost neto ulazne matrice i tako dobijamo hxn aktivacionu matricu **A(2)** za sljedeći sloj (ovde izlazni sloj).

**A(2)=Φ(Z(2))**

Slično možemo napisati i matričnu jednačinu aktivacije izlaznog sloja u vektorizovanoj formi:

**Z(3)=W(2) A(2)**

Ovde množimo txh matricu **W(2)** (t je broj izlaznih jedinica) sa hxn dimenzionalnu matricu **A(2)** da bi dobili txn dimenzionalnu matricu, čije kolone predstavljaju izlaze za svaki ulazni uzorak. Konačno primjenjujemo sigmoidnu funkciju aktivacije da bi dobili kontinualne vrijednosti izlaza naše mreže.

**A(3)=Φ(Z(3))** (A(3) ∈ Rtxn).

Sada smo došli do učenja umjetne neuronske mreže. Ovo se rješava preko koncepata logističke funkcije troška i kretanja unazad (backpropagation) koji se implementiraju pri učenju težinskih koeficijenata. Računanje logističke funkcije troška je implementirana u programskom kodu kao \_get\_cost metoda

FORMULA

**a(i)** je ovde sigmoid aktivacija i-te jedinice u jednom od slojeva u kojem računamo korak naprijed u mreži:

**a(i)=Φ(z(i))**

Sada se dodaje član regularizacije, koji nam omogućava da redukujemo suvišnu prekomjernu preciznost (overfiting). Na raspolaganju su L2 i L1 regularizacije koje su definisane sljedećim izrazima:

FORMULE

Iako naš MLP algoritam dozvoljava L2 i L1 regularizaciju, kod ćemo zbog jednostavnosti raditi samo sa L2 regularizacijom. Kada se doda član za regularizaciju naša formula logističke funkcije troška izgleda ovako;

FORMULA

Mi implementiramo MLP za više klasnu kategorizaciju, koja vraća vektor od t elemenata, koji se poredi sa tx1 dimenzionalnim vektorom :

PRIKAZ OBA VEKTORA

Sada trebamo generalizovati logističku funkciju troška na sve aktivacione jedinice j u našoj mreži bez regularizacionog elementa izgleda kao:

FORMULA

Ovde je eksponent i indeks pojedinačnog primjerka ulaza u našem skupu za učenje. Sljedeći uopšteni član za regularizaciju, koji računa sumu svih težinskih faktora u sloju l (bez biasa)dodajemo na prvu kolonu:

FORMULA

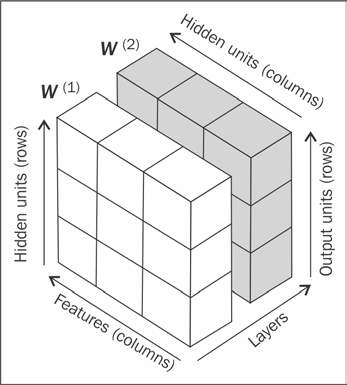
Sljedeći član u formuli predstavlja L2 penalisanje:

FORMULA

Sjetimo se da treba da minimiziramo funkciju troška J(w), tako da treba da računamo parcijalne derivacije matrice **W** prema svakom težinskom koeficijentu u svakom sloju u mreži:

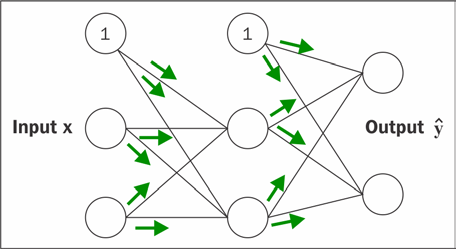
FORMULA

U sljedećem dijelu govorimo o backpropagation algoritmu , koji nam omogućava da računamo ove parcijalne derivacije da bi smo mogli minimizovati funkciju troška. Primjetićemo da imamo više **W** matrica. Ulazni sloj je povezan sa skrivenim slojem preko **W(1)** matrice, a **W(2)** matrica povezuje skriveni i izlazni sloj. Intuitivno predstavljanje matrice **W** je dato na slici:



Ovo je pojednostavljena slika jer **W(1)** i **W(2)** matrica ne moraju imati isti broj redova i kolona.

Sada smo došli do učenja neuronske mreže preko backpropagation algoritma. Ovaj algoritam je otkriven prije više od 30 godina, ali je i danas najefikasniji za učenje neuronskih mreža. U svojoj suštini, ovaj algoritam je računski vrlo efikasan način za računanje derivacija vrlo kompleksnih funkcija. Naš cilj je da koristimo te derivacije za učenje težinskih koeficijenata za parametrizaciju višeslojne umjetne neuronske mreže. Izazov u parametrizaciji neuronske mreže je da imamo vrlo veliki broj težinskih koeficijenata u višedimenzionalom prostoru. Za razliku od ostalih funkcija troška oblik funkcije greške neuronske mreže nije ni konveksan ni glatak. Imamo mnogo skokova na ovoj funkciji (lokalnih minimuma). Kad bi derivirali u smjeru od početka prema kraju imali bi potrebu da računamo Jacobian matrice.



Možemo se sjetiti lančanog pravila za deriviranje složenih funkcija , a mnogo je jednostavnije računati derivacije idući prema nazad. Počinjemo računati grešku od razlike aktivacione funkcije i stvarne oznake klase  **δ(3)=a(3)-y**

Zbog toga se ova oblast naziva umjetna inteligencija (artificial inteligence) ili iskustveni intelekt. Danas se ovi problemi rješavaju uspješnije sa CNN (konvolucionim neuronskim mrežama) i RNN/LSTM (rekurentim neuronskim mrežama sa dugom i kratkom memorijom).

Prepoznavanje rukopisa je veoma važno sa nizom primjena u: zdravstvu i farmaciji,osiguranju, bankarstvu i finansijama itd. Ovde imamo dvije podklase **OCR (Optical character recognition )**optičko prepoznavanje slova i **HWR (Handwriting recognition)** prepoznavanje rukopisa pojedinog čovjeka. Obe potklase su usko povezane sa prepoznavanjem mašinskog teksta sa slike.

Problemi u koji se pojavljuju kod prepoznavanja rukopisa su:

1. Velike varijacije u rukopisu svakog čovjeka.
2. Stilovi rukopisa istih ljudi su često različiti. Pojedinci imaju više stilova rukopisa.
3. Slike rukopisa su često lošeg kvaliteta, pa je prepoznavanje otežano.
4. Problem je loše poravnanje pisanog teksta.
5. Kitnjasti rukopis otežava prepoznavanje jer povećava varijacije rukopisa.
6. Rukopis može biti nagnut u desno, biti ravan ili nagnut u lijevo.
7. Pribavljanje podataka za mašinsko učenje je problematično.