Univerzitet Crne Gore
Prirodno-matematički fakultet
Podgorica

Andrija Vučinić

Detekcija lica na slikama pomoću neuronskih mreža

Specijalistički rad

Podgorica, Septembar 2010. godine

Univerzitet Crne Gore
Prirodno-matematički fakultet
Podgorica

Detekcija lica na slikama pomoću neuronskih mreža

Specijalistički rad

Kriptografija
Doc. dr Vladimir Božović

Andrija Vučinič
Studijski program računarske nauke
2405986210277

Podgorica, Septembar 2010. godine

Sažetak

Detektovanje lica je najčešće prvi korak u mnogim računarskim sistemima baziranim na vizuelnoj percepciji te je time važnost detekcije ogromna. U ovom radu predstavljamo algoritam za detekciju lica zasnovan neuronskim mrežama. Drugo poglavlje sadrži generalni pregled o matematičkoj osnovi, funkcionisanju, strukturi i procesima učenja neuronskih mreža. Posebnu pažnju smo posvetili samoj formulaciji problema detekcije lica, ističući postojeće probleme i izazove u slučaju sistema koji su zasnovani na neuronskim mrežama. Algoritam, skupovi podataka, tehnike i rezultati su dati u zaključku rada.

Abstract

Face detection is usually the first step in many computer systems based on visual perception, therefore making the detection of huge importance. An alghoritm for face detection using neural networks is presented in this paper. The second chapter contains a general review about the mathematical basis, function, structure and learning processes of neural networks. We devote special attention defining the face detection problem, emphasizing existing problems and challenges related to systems based on neural networks. The alghoritm, data sets, techniques and results are given in the conclusion of the paper.

Sadržaj

1. Uvod

Istraživanje ljudskog mozga traje hiljadama godina. Međutim, tek se sa pojavom moderne elektronike nazrela mogućnost imitacije ljudskog mozga i procesa razmišljanja. U tom pogledu "vještačke neuronske mreže"[1], kao pokušaj elektronske imitacije bioloških neuronskih mreža u mozgu, igraju nezaobiloznu ulogu. Prva značajnija istraživanja na temu neuronskih meža su napravili neuro-fiziolog, Voren MekKuloh i mladi izuzetno talentovani matematičar Volter Pits u 1943. MekKuloh je proveo 20 godina života, razmišljajući o "događaju" u nervnom sistemu koji nam omogućava da mislimo, osjećamo... Tek nakon susreta sa Pitsom, uspio je formalizovati svoju teoriju, a potom i dizajnirati primitivnu vještaču neuronsku mrežu pomoću elementarnih elektronskih kola.

Sledeći veliki talas u razvoju neuronskih mreža stigao je u 1949. sa knjigom "Organizacija ponašanja" koju je napisao Donald Heb. Heb je podržao i dodatno ojačao MekKuloh-Pits teoriju. Glavni korak naprijed, ostvaren u njegovoj knjizi, je objašnjenje kako se ojačavaju neuronski putevi prilikom korišćenja.

Tokom pedesetih godina prošlog vijeka su skoro zastala istraživanja na temu neuronskih mreža. Međutim, nekoliko pojedinaca-entuzijasta je nastavilo rad u toj oblasti. Jedan od njih, Marvin Minski je 1954. napisao doktorsku tezu, "Teorija Neuronsko-analognog pojačavanja sistema i njena primjena na modelu mozga", koja se u osnovi bavila neuronskim mrežama. Isti autor je objavio naučni rad pod nazivom "Koraci ka vještačkoj inteligenciji", što je bio jedan od prvih radova sa detaljnim osvrtom na pitanje vještačke inteligencije. Taj rad sadrži i sekciju na temu onog što danas nazivamo neuronskim mrežama. 1956. godine u okviru Dartmut ljetnjeg istraživačkog projekta, počeo je prvi istraživački poduhvat u oblasti neuronskih mreža. (http://www.dacs.dtic.mil)

Nekoliko godina kasnije, Džon fon Nojman je razmišljao o imitiranju jednostavnih neuronskih funkcija pomoću telegrafskog prenosa ili vakum cijevi. To je dovelo do pronalaska fon Nojman mašina. Petnaest godina nakon objavljivanja pionirskog rada MekKuloha i Pitsa, predstavljen je novi pristup u području istraživanja neuronskih mreža. Naime, 1958. Frenk Rosenblat, neuro-biolog sa Kornel univerziteta je počeo da radi na ''Perceptronu''. Perceptron je bila prva "praktična" vještačka neuronska mreža. Izgrađen je pomoću prilično primitivnih hardverskih komponenti tog vremena. Perceptron je zasnovan na istraživanju koje je oobavljeno na oku muve. Signal koji ''govori'' muvi da bježi kad je opasnost u blizini se stvara u oku. Jedan od glavnih nedostataka Perceptrona su vrlo ograničene mogućnosti, što su dokazali Marvin Minski i Sejmur Papert u svojoj knjizi iz 1969. pod nazivom "Perceptroni".

Neuronske mreže su vremenom postale vrlo popularne u naučnoj zajednici, tako da se već 1986. godine okupilo više od 1800 delegata na konferenciji o vještačkim neuronskim mrežama. Danas, neuronske mreže prožimaju mnoge oblasti istraživanja. Glavni izazov se sastoji u pronalaženju načina da se elektronski implementiraju određeni teorijski principi funkcionisanja neuronskih mreža. Vodeće kompanije u sferi elektronike trenutno razvijaju tri tipa neuro-čipova: digitalni, analogni i optički. U slučaju da se ovakvi čipovi mogu uključiti u dizajn neuronskih mreža, budućnost ove oblasti izgleda veoma obećavajuća.

1.1 Neuronauka i računari

Neuronauka je nauka o nervnom sistemu, posebno mozga. Jedna od najvećih misterija nauke je kako i da li mozak stvara (omogućava) misao. Činjenica da jaki fizički udarci u glavu mogu dovesti do mentalnih oštećenja ide u prilog tvrdnji da misao "stanuje" u mozgu. Oko 335. godine P.H. Aristotel je napisao: "Od svih životinja, čovjek ima najveći mozak proporcionalno svojoj veličini." Ipak, tek je u osamnaestom vijeku mozak priznat kao centar svjesnosti. Prije toga, kandidati su bili srce i slezina.

1 Termin "neuronske mreže" će se dalje u radu odnositi isključivo na vještačke neuronske mreže izuzev u slučaju kad je drugačije naglašeno.

Istraživanja Pola Broke (1824-1880) o afaziji (deficit govora) kod pacijenata sa oštećenim mozgom 1861. su demonstrirala postojanje oblasti mozga odgovornih za specifične kognitivne funkcije. Specijalno, on je pokazao da je stvaranje govora lokalizovano na dijelu lijeve hemisfere koja se danas naziva Brokina oblast. Do tada se već znalo da se mozak sastoji od nervnih ćelija, **neurona**, ali je tek 1873. Kamilo Golgi (1843-1926) razvio tehniku bojanja koja je omogućavala opservaciju individualnih neurona (Slika 1.1). Ovu tehniku je koristio Santijago Ramon Kajal (1852-1934) u njegovim pionirskim studijama moždane neuronalne strukture. Oba naučnika su dobila Nobelovu nagradu 1906. godine, i smatraju se očevima današnje neuronauke. Nikola Raševski (1899-1972) je prvi koji je primijenio matematičke modele u izučavanju nervnog sistema.

Danas imamo neke podatke o funkcionalnosti određenih oblasti mozga, tj. koje djelove tijela kontrolišu ili od kojih djelova dobijaju čulne ulaze. Ove veze se mogu radikalno promijeniti u periodu od samo nekoliko nedjelja, dok neke životinje imaju redundantne[1] veze. Međutim, nemamo potpuno shvatanje o tome kako neka oblast preuzima funkcionalnost kad je određena oblast oštećena. Pored toga, ne postoji dobro utemeljena i opšte prihvaćena teorija o čuvanju individualne memorije..

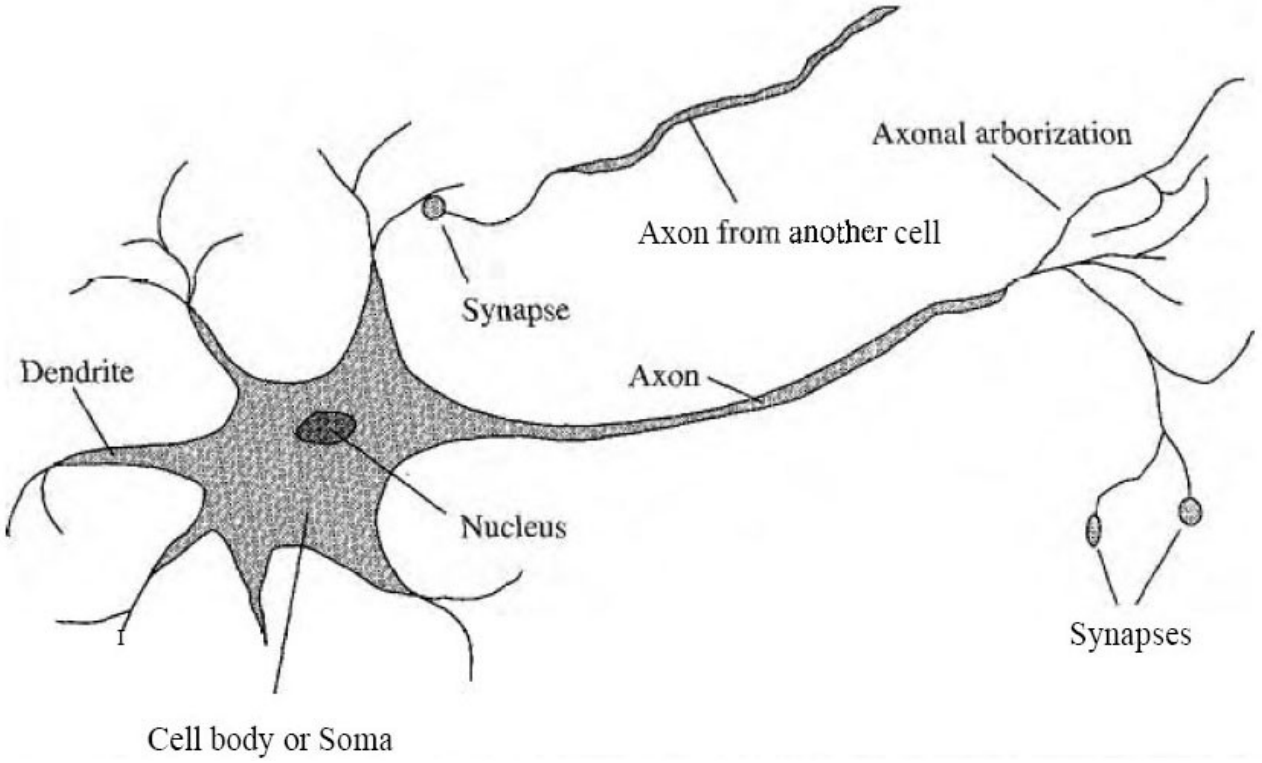
Mjerenja moždane aktivnosti su počela 1929. godine sa otkrićem Hansa Bergera, elektroencefalografa (EEG). Razvoj funkcionalnog magnetno-rezonantnog snimanja daje neuronaučnicima detaljne slike moždane aktivnosti, dozvoljavajući mjerenja koja odgovaraju kognitivnim procesima u mozgu. U međuvremenu, postignut je napredak u jednoćelijskim snimcima aktivnosti neurona. Individualni neuroni se mogu stimulisati električno, hemijski ili čak optički, dopuštajući da se neuronske ulazno-izlazne veze mapiraju. Bez obzira na sve napretke, i dalje smo daleko od razumijevanja kako kognitivni procesi stvarno funkcionišu.

Istinski nevjerovatan zaključak jeste da kolekcija jednostavnih ćelija može dovesti do misli, akcija i svjesnosti, ili riječima Džona Searla, mozak je uzrok svijesti. Jedina alternativna teorija je misticizam: da mozak funkcionise u nekom mističnom carstvu koje je izvan fizičke nauke.

Mozak i digitalni računari imaju različite osobine. Na slici 1.2 se može vidjeti da kompjuteri imaju ciklus koji je oko milion puta brže od moždanog. Mozak to nadoknađuje time što ima mnogo više prostora za podatke i veze koje čak ni najbolji personalni računari nemaju, dok današnji superkompjuteri imaju kapacitete slične mozgu. Treba znati da mozak obično ne koristi sve neurone istovremeno. Čak i sa kompjuterom virtuelno neograničenih kapaciteta, ne bismo znali kako da dostignemo nivo inteligencije mozga. Bez obzira, sama ideja da skup (mreža) jednostavnih elemenata može dovesti do veoma složenih sistema stvorila novi pristup rješavanju problema - konekcionizam.

Konekcionizam je skup pristupa rješavanju problema u oblastima vještačke inteligencije, kognitivnih nauka, neuronauka i filozofije uma, koji modelira mentalne i fizičke radnje kao aktivnost mreže sastavljene od elementarnih jedinica. Naime, veliki sistemi su obično izgrađeni od elementarnih jedinica koje pojedinačno nemaju ''ideju'' o konačnoj funkciji sistema, te ne mogu samostalno obaviti funkciju cjeline. Najčešći oblik konekcionizma su neuronske mreže.

1 Višestruke veze, koje obavljaju iste funkcije. Ako jedna otkaže, druge mogu da preuzmu njenu funkciju.



Slika 1.1 - Djelovi nervne ćelije (neurona). Svaki neuron se sastoji od tijela ćelije (soma) i jezgra (nucleus). Iz tijela ćelije se granaju vlakna koja se nazivaju dendriti i jedno dugo vlakno koje se naziva akson. Akson je veoma dug, mnogo duži nego što bi se mogli zaključiti sa slike. Obično su dugi oko 1 cm (100 puta duži od prečnika tijela ćelije), ali mogu biti dugi i do 1 metar. Neuron ostvaruje veze sa 10 do 100,000 drugih neurona preko spojnica koje se nazivaju sinapse. Signali se propagiraju od neurona do neurona komplikovanim elektrohemijskim reakcijama. Ovi signali kontrolišu aktivnost mozga kako kratkoročno tako i dugoročno u smislu povezanosti neurona. Smatra se da ovi mehanizmi formiraju osnovu učenja u mozgu. Najveći dio procesiranja informacija se obavlja u moždanoj kori, spoljnom sloju mozga. Osnovna jedinica tvorbe kore, koja je oko 4mm kod ljudi, je kolona tkiva oko 0.5mm prečnika. Kolona tkiva sadrži oko 20,000 neurona.

	Computer	Human Brain
Computational units	1 CPU, 10^8 gates	10^{11} neurons
Storage units	10^{10} bits RAM	10^{11} neurons
	10^{11} bits disk	10^{14} synapses
Cycle time	10^{-9} sec	10^{-3} sec
Bandwidth	10^{10} bits/sec	10^{14} bits/sec
Memory updates/sec	10^9	10^{14}

Slika 1.2 - Poređenje računarske snage (2003. godine) i mozga. Vrijednosti kod računara se svakom decenijom povećavaju za faktor od 10. Vrijednosti mozga su nepromijenjene zadnjih 10,000 godina.

Tradicionalno, termin **neuronska mreža** se odnosio na mrežu bioloških neurona. Taj termin se sve češće danas odnosi na vještačke neuronske mreže, koje su izgrađene od vještačkih neurona ili čvorova. Dakle, termin ima 2 različite upotrebe:

1. Biološke neuronske mreže su sastavljene od bioloških neurona koji su povezani i funckionalno vezani za periferni ili centralni nervni sistem. U neuronaukama, često ih poistovjećujemo sa grupom (mrežom) neurona koja obavlja neku određenu fiziološku funkciju.
2. Vještačke neuronske mreže se sastoje od međusobno povezanih vještačkih neurona (programske konstrukcije koje imitiraju osobine bioloških neurona). Svaki neuron ima određen broj ulaza i jedan izlaz koji je definisan matemičkom funkcijom (različiti neuroni mogu imate različite funkcije). Vještačke neuronske mreže su organizovane po slojevima (nivoima). Sloj ima određen broj neurona. Neuroni jednog sloja za ulaze imaju izlaze neurona prethodnog sloja ili, rekurentno, izlaze njih samih. Vještačke neuronske mreže obavezno imaju ulazni sloj (ulaz) i izlazni sloj (izlaz). Slojevi između ulaznog i izlaznog nivoa se nazivaju *skrivenim*. Pod treniranjem (učenjem) mreže podrazumijevamo proces podešavanja parametara, tako da mreža daje najbolji rezultat za posmatrani problem (zadatak). Vještačke neuronske mreže se mogu koristiti za identifikaciju funkcija bioloških neuronskih mreža, ili za rješavanje problema vještačke inteligencije bez obaveznog kreiranja modela stvarnog biološkog sistema.

Razmišljanje o funkcionisanju mozga je dalo svoje rezultate proširujući opseg tehničkih ideja za rješavanje određenih problema.

Prvi matematički model neuronske mreže je napravljen 1943. godine, ali zbog loših algoritama učenja korištene su samo perceptron mreže (imaju samo ulazni i izlazni sloj). Problem je bio u slučaju višeslojnih mreža. Nije postojao dovoljno dobar način kojim bi se greška iz izlaznog sloja prenijela u skrivene slojeve. Brajson i Ho su 1969. godine predložili algoritam **propagacije unazad**. Algoritam je ostao nezapažen do 1986. godine, kada su trojica naučnika objavili rad koji je pokrenuo "renesansu" na polju vještačkih neuronskih mreža[1]. Algoritam je primijenjen na mnoge probleme učenja u kompjuterskim naukama i psihologiji, i rezultati objavljeni u kolekciji "Paralelno distribuirano procesiranje" su se brzo proširili i izazvali veliko uzbuđenje.

Jedna od danas najpoznatijih primjena vještačke neuronske mreže je prepoznavanje pisanih brojeva[2], u poštama Amerike.

U poglavlju 2 dajemo uvod u neuronske mreže, njihovu strukturu kao i načine učenja mreža. Dati su primjeri uspješnih primjena neuronskih mreža. U poglavlju 2.2 je dat matematički aparat koji je potreban za poglavlja 2.3-2.5 koja se odnose na treniranje mreže.

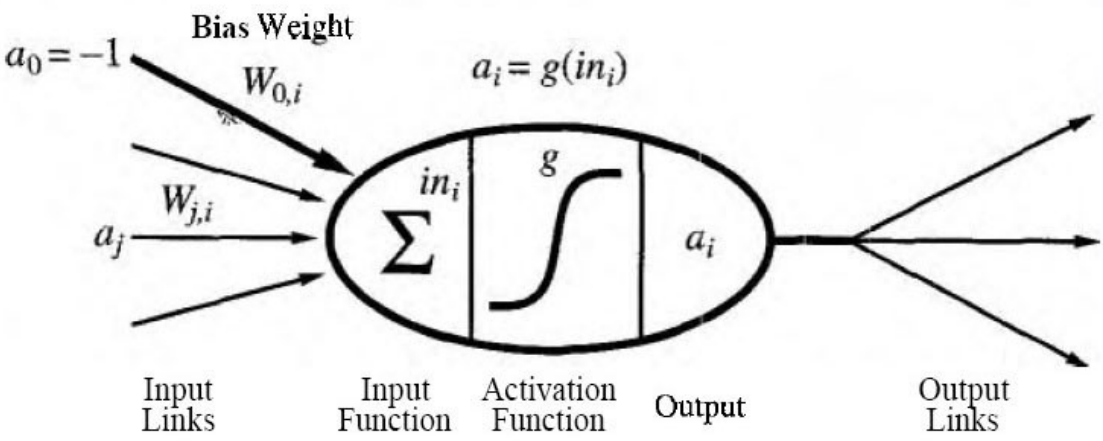
U poglavlju 3 definišemo problem detekcije lica na slikama i navodimo karakteristične probleme. Diskutujemo uticaj predparsiranja[3] i mogućnosti kreiranja robustnog[4] modela bez predparsiranja.

2. Neuronske mreže

Jedna od hipoteza je da se mentalna aktivnost zasniva primarno na elektrohemijskim aktivnostima u mrežama moždanih ćelija koje se nazivaju **neuroni**. Inspirisani ovom hipotezom, rani radovi u vještačkoj inteligenciji su bili usmjereni ka kreiranju **vještačkih neuronskih mreža**. (Konekcionizam, paralelno distribuirano procesiranje, i neuronska izračunavanja) Na slici 2.1 je prikazan jednostavan matematički

1 David E. Rumelhart, Geoffrey E. Hinton und Ronald J. Williams. Learning representations by back-propagating errors., Nature (London) 323, S. 533-536
2 OCR - Optical Character Recognition
3 Priprema podataka za neki program. Normalizacija, uklanjanje šuma, razni filteri.
4 Model koji daje dobre rezultate i u slučaju velikih varijacija podataka.

model koji su predložili MekKuloh i Pits (1943)[1]. Grubo rečeno, neuron "okida" (šalje signal) kada linearna kombinacija njegovih ulaza pređe neki prag. Osobine mreže su određene njenom strukturom (topologijom) i osobinama "neurona".



Slika 2.1 - Jednostavan matematički model neurona. Izlaz neurona je $a_j = g(\sum_{i=0}^{n} w_{i,j} a_i)$, gdje je a_i izlaz iz *i-tog* neurona, a $w_{i,j}$ težina dodijeljena vezi od *i-tog* neurona do *j-tog* neurona.

Od 1943. se razvilo puno detaljnijih i realističnijih modela, kako za neurone tako i za veće sisteme u mozgu, i dovelo do stvaranja moderne oblasti **computational neuroscience**. Istraživanja u oblasti vještačke inteligencije i statistike usmjerena su više apstraktnim osobinama ovih mreža, kao npr. obavljanje paralelnog računanja, tolerisanje ulaza sa šumom i mogućnošću učenja. Naravno, danas postoji još puno sistema koji imaju ove osobine, ali neuronske mreže ostaju jedna od najpopularnijih i efektivnih formi sistema sa mogucnošću učenja i vrijedne su proučavanja same za sebe. Neke od oblasti u kojima se uspješno primjenjuju neuronske mreže su data mining (nalaženje znanja u bazama), e-mail spam filtriranje, prepoznavanje objekata (identifikacija lica, radarski sistemi..), donošenje odluka (šah, dame).

2.1 Struktura neuronskih mreža

Neuronske mreže su izgradjene od čvorova ili jedinica (slika 2.1) koje su povezane usmjerenim vezama. Veza od čvora i do čvora j sluzi da bi se proslijedio izlaz a_i od i do j, tj. cvoru j predajemo izlaz cvora i. Svaka veza ima i dodijeljenu numeričku težinu $w_{i,j}$ koja određuje jačinu i znak konekcije. Svaki čvor ima vještački ulaz $a_0 = 1$ sa dodijeljenom težinom $w_{0,j}$. U svakom čvoru se prvo računa težinska suma ulaza:

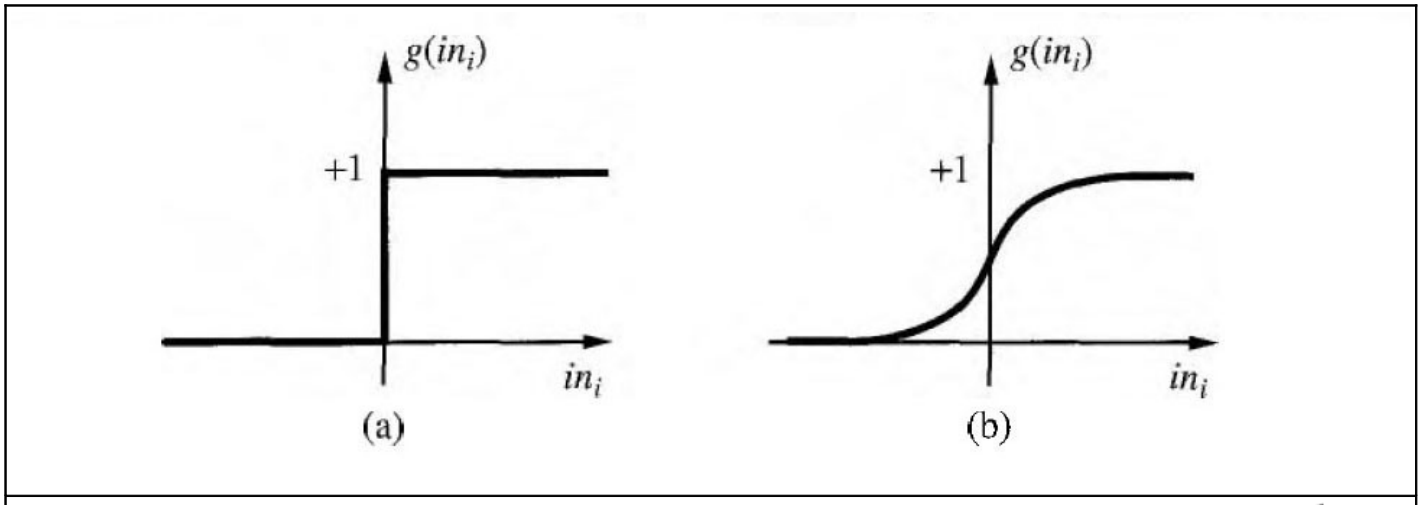
$$in_j = \sum_{i=0}^{n} w_{i,j} a_i$$

Zatim se primijeni **aktivaciona funkcija** g na ovu sumu da bi se dobio izlaz:

$$a_j = g(in_j) = g(\sum_{i=0}^{n} w_{i,j} a_i)$$

1 McCulloch, W . and Pitts, W . (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 7:115 - 133

Aktivaciona funkcija je obično ili funkcija sa jakim pragom[1], u kom slucaju se naziva **perceptron**, ili logistička funkcija[2], u kom slučaju se naziva **sigmoidni perceptron**. Izlaz mreže je u matematičkom smislu funkcija koja nastaje kompozicijom aktivacionih funkcija neurona. Tu funkciju nazivamo **izlaznom funkcijom mreže**. Najčešće je izlazna funkcija mreže nelinearna funkcija.[3]



Slika 2.2 - (a) Funkcija sa jakim pragom. (b) Logistička funkcija (naziva se često sigmoidna) $\frac{1}{1+e^{-x}}$

Postoje dva fundamentalno različita načina povezivanja neurona u mrežu. **Feed-forward mreža** ima veze samo u jednom smjeru - tj. formira usmjereni aciklični graf. Svaki čvor dobija ulazne podatke od čvorova koji se nalaze u prethodnom sloju, a izlazne podatke daje čvorovima u sledećem sloju. Ne postoje ciklusi. Feed-forward mreža je funkcija ulaznih parametara, te nema interna stanja osim težine grana.

Rekurentna mreža je mreža u kojoj neuroni mogu izlaze prosleđivati na svoje ulaze. Ovo znači da izlazi neurona formiraju dinamički sistem (dinamički sistem je matematički formalizam koji opisuje neko "pravilo" čije stanje zavisi od trenutka u kome posmatramo). Shodno tome izlazi se mijenjaju i opet vraćaju na ulaze neurona. Poslije određenog vremena, mreža može doći u stabilno stanje (nema promjena na izlazima mreže), oscilirati (naizmjenično na izlazu imati neku vrijednost iz određenog skupa) ili pokazivati haotično ponašanje (izlazi se mijenjaju bez ikakvog smisla). Štaviše, odgovor mreže za date ulaze zavisi od inicijalnog stanja, koje može zavisiti od prethodnih ulaza. Dakle, rekurentne mreže (za razliku od feed-forward) mogu imati kratkoročnu memoriju. Ovo ih čini interesantnijima za modeliranje mozga, ali takođe i teže za razumijevanje.

Feed forward mreže su najčešće uređene u nivoima, tako da svaki čvor dobija ulaz samo od čvorova koji se nalaze u prethodnom nivou.

Do sad smo razmatrali samo mreže sa jednim izlazom, ali neuronske mreže mogu imati više izlaza. Na primjer, ako želimo da obučimo mrežu da sabira dva bita, koji mogu uzeti vrijednosti 0 ili 1, trebaće nam jedan izlaz za sumu, i jedan za prenos. Takođe, kad problem učenja uključuje klasifikaciju u više od dvije klase - npr, kategorizacija ručno pisanih brojeva - obično se koristi po jedan izlaz za svaku klasu.

1 Za ulaz x i prag a funkcija vraća 1 ako je x > a, inače vraća 0
2 Za sada je dovoljno reći da za ulaz x vraća vrijednost iz intervala 0, 1. O logističkoj funkciji više riječi u poglavlju 2.3.4.
3 Stuart Russel and Peter Norvig, A modern approach to artificial intelligence (3rd edition)

2.2 Paradigme učenja

Postoje 3 glavne paradigme učenja: nadgledano učenje, nenadgledano učenje i pojačano učenje[1]. Pojam *učenja* u vještačkoj inteligenciji je podešavanje parametara izabranog modela u korist dobijanja optimalnog rješenja. Bilo koja arhitektura mreže se može iskoristiti za zadatke učenja.

2.2.1 Nadgledano učenje

Zadatak **nadgledanog učenja** jeste:

Za dati **skup za treniranje** od N ulaz-izlaz uređenim parovima

$$(x_1,y_1),(x_2,y_2),...,(x_N,y_N)$$

gdje je svako y_j (naziva se *labela*) generisano nepoznatom funkcijom $y = f(x)$, naći **hipotezu** h koja aproksimira originalnu funkciju f.

To radimo tako što minimizujemo grešku mreže na izlazu, po svim uređenim parovima. Funkcija greške, obično L_2 (kvadratna greška)[2], se naziva **funkcijom cijene**.

Zadaci koji pripadaju ovoj paradigmi su prepoznavanje šablona (klasifikacija) i aproksimacija funkcije (regresija). Prepoznavanje lica pripada ovoj paradigmi.

2.2.2 Nenadgledano učenje

Zadatak **nenadgledanog učenja** jeste:

Za dati **skup za treniranje** od N ulaza

$$x_1,x_2,...,x_N$$

izvši klasifikaciju podataka u odnosu na zadati kriterijum.

Primjer je kompetitivno učenje. Pretpostavimo da sve podatke koji nam dolaze želimo da podijelimo u 3 grupe (3 neurona u izlaznom nivou).

1. Sve ulaze usmjerimo na 3 čvora u izlaznom nivou, tako da je svaki ulaz povezan na svaki izlazni čvor. Inicijalizujemo sve težinske faktore nasumično vrijednostima između 0 i 1. Neka je izlaz svakog čvora težinska suma ulaza.
2. Kada mreža dobije ulaz, izlazni čvor sa najvećim rezultatom se proglašava pobjednikom. Ulaz se klasifikuje da pripada tome čvoru.
3. Pobjednik mijenja svoje težinske faktore, tako što povećava težine vezama koje imaju veće vrijednosti, a smanjuje onima koje imaju manje vrijednosti.

Kako sistem dobija više podataka, tako će svaki od 3 izlaza da sve bolje uzima u obzir (raspodjelom težinskih faktora) ulaze koji pripadaju određenoj grupi.

Zadaci koje rješava ova paradigma su obično vezani za organizaciju podataka, klasterizaciju ili statističku raspodjelu.

1 Reinforcement learning
2 $(y - hw(x))^2$, gdje je y očekivani izlaz, a $h_w(x)$ izlaz postavljene hipoteze

2.2.3 Pojačano učenje

U ovoj paradigmi podaci nisu dati, nego se generišu agentovom[1] interakcijom sa okolinom. Agent u svakom trenutku vremena t, uradi akciju y_t i okolina generiše posljedicu x_t, nekom nepoznatom dinamikom sistema. Ideja je da agent nauči koje akcije generišu povoljne ishode za određeni cilj.

Na primjer, neka je dat sistem koji funkcioniše po nepoznatim pravilima. Pretpostavimo da postoji robot koji je osposobljen da izvrši n različitih akcija, i kome je dat neki cilj. Zadatak ove paradigme je da robot izvršavajući akcije razluči koje od njih dovode do cilja u datom sistemu.

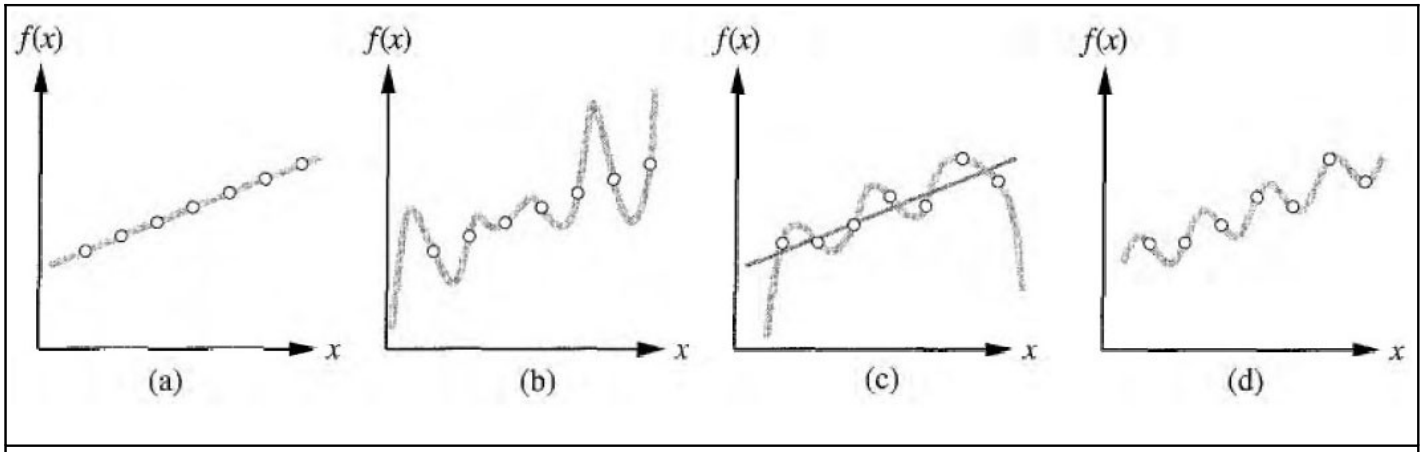
Zadaci iz ove paradigme su problemi kontrole[2] (balansiranje robota na 2 točka, cruise control sistem u autima) i igre (poslije određenog poteza, procijeni se koliko je bio dobar i doveo do cilja - pobjede).

2.3 Regresija i klasifikacija sa linearnim modelima

Traženje hipoteze unutar skupa funkcija sa diskretnim kodomenom {(sunčano, oblačno, kišno), (0, 1)} se naziva **klasifikacija.** U slučaju da diskretni skup sadrži samo dvije vrijednosti, tada je to Bulova ili binarna klasifikacija.

Traženje hipoteze unutar skupa funkcija sa realnim kodomenom (npr. temperatura za naredni dan) se naziva **regresija**.

Preciznost hipoteze u opštem slučaju mjerimo tako što mjerimo odstupanje na **skupu podataka za testiranje** (koji se razlikuje od skupa podataka za treniranje). Kažemo da je hipoteza zadovoljavajuća, ako je odstupanje rezultata hipoteze na skupu podataka za testiranje manje od neke date granice.



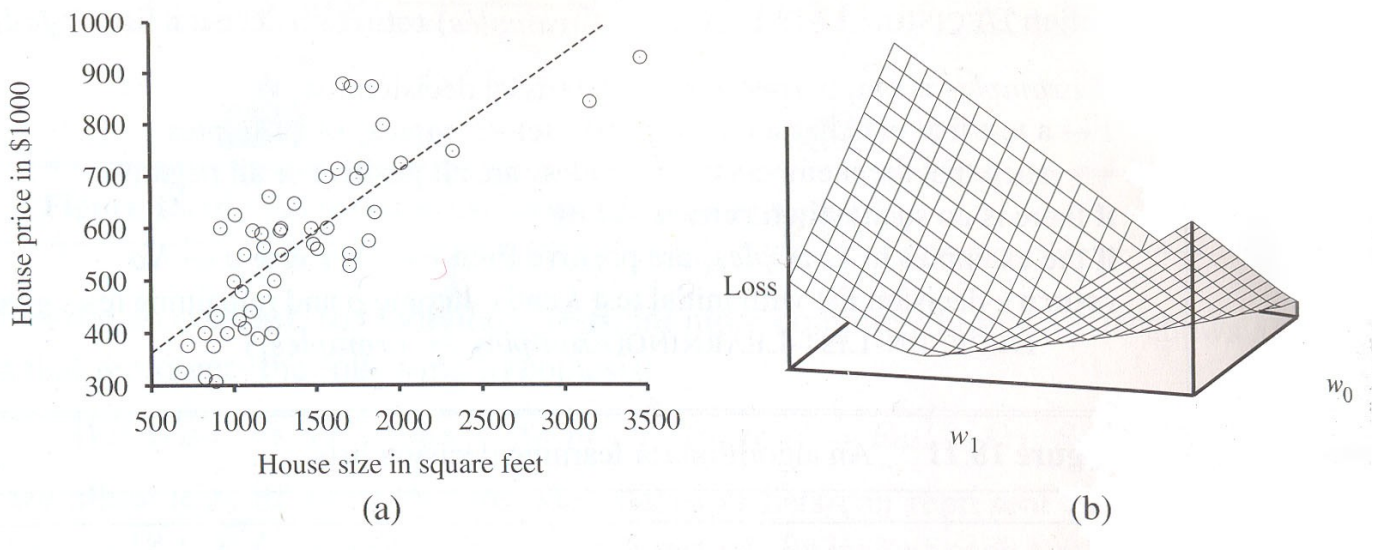
Slika 2.3 - (a) Skup podataka (x, f(x)) i konzistentna, linearna hipoteza. (b) Polinom 7-og stepena je takođe konzistentna hipoteza za dati skup. (c) Polinom 6-og stepena i linearna funkcija kao hipoteze za drugi skup podataka. (d) Sinusoidna hipoteza za isti skup.

Na slici 2.3 se vidi primjer: aproksimiranje funkcije od jedne promjenljive sa nekoliko tačaka. Primjerci su tačke u (x,y) ravni, gdje je $y=f(x)$. Ne znamo sta je f, ali ćemo je aproksimirati funckijom h iz **prostora hipoteza**, H, koji ćemo za ovaj primjer uzeti da je skup polinoma, npr. x^5+3x^3+2. Na slici 2.3 (a) su dati podaci koje možemo aproksimirati pravom (recimo polinom $0.4x+3$). Ova prava se naziva **konzistentna** hipoteza zato što odgovara svim podacima. Na slici 2.3 (b) imamo polinom visokog stepena, koji je takođe **konzistentan** sa podacima. Ovo predstavlja jedan od fundamentalnih problema induktivnog učenja, a to je pitanje: *kako izabrati iz skupa konzistentnih hipoteza?* Jedan od

1 Autonomni entitet koji posmatra okolinu i djeluje na nju, pokušavajući da postigne određene ciljeve.
2 Control problems

odgovora je izabrati *najjednostavniju*. Ovaj princip se naziva **Okamov žilet**, po Engleskom filozofu iz 14og vijeka, Vilijamu od Okama. Nije lako definisati jednostavnost, ali je jasno da je polinom 1og stepena jednostavniji od polinoma 7og stepena, te ćemo u slučaju na slici 2.3 izabrati (a) umjesto (b).

2.3.1 Jednodimenzionalna linearna regresija



Slika 2.4 - (a) Podaci o cijenama kuća naspram kvadrature kuća, sa linearnom hipotezom koja minimizuje kvadratnu grešku: $y = 0.232x + 246$. (b) Grafik funkcije greške za razne vrijednosti w_0 i w_1. Primijetimo da je funkcija konveksna, sa jedinstvenim globalnim minimumom.

Jednodimenzionalna linearna funkcija (prava) sa ulazom *x* i izlazom *y* ima oblik $y = w_1x + w_0$, gdje su w_1 i w_0 koeficijenti (iz skupa realnih brojeva) koje trebamo naučiti. Koristimo oznaku *w* jer posmatramo koeficijente kao **težinske faktore**; vrijednost *y* mijenjamo mijenjajući težinske faktore jednog ili drugog terma. Definišimo **w** kao vektor $[w_0, w_1]$ i definišimo

$$h_w(x) = w_1 x + w_0$$

Na slici 2.4 (a) je dat primjer skupa za obučavanje od n tačaka u (x, y) ravni, gdje svaka tačka predstavlja (na primjer) kvadraturu zemljišta koje se prodaje i njegovu cijenu. Funkcija gubitka hipoteze h_w se definiše:

$$Loss(h_w) = \sum_{j=1}^{N} L_2(y_j, h_w(x_j)) = \sum_{j=1}^{N} (y_j - h_w(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

Linearna funkcija (hipoteza) koja ima najmanju vrijednost funkcije gubitka hipoteze za dati skup podataka se naziva **linearna regresija**. Dakle, mi tražimo $\mathbf{w}^* = \text{argmin}_w \text{Loss}(h_\mathbf{w})$. Suma

$$\sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

se minimizuje tako što parcijalne izvode, za w_0 i w_1 izjednačimo sa nulom:

$$\frac{\partial}{\partial w_0}\sum_{j=1}^{N}(y_j-(w_1x_j+w_0))^2=0 \text{ i } \frac{\partial}{\partial w_0}\sum_{j=1}^{N}(y_j-(w_1x_j+w_0))^2=0$$

Ove jednačine daju jedinstveno rješenje:

$$w_1=\frac{N(\sum x_j y_j)-(\sum x_j)(\sum y_j)}{N(\sum x_j^2)-(\sum x_j)^2} \qquad w_0=(\sum y_j-w_1(\sum x_j))/N$$

Za primjer na slici 2.4 (a), rješenje je $w_1 = 0.232$, $w_0 = 246$, i prava je prikazana isprekidanom linijom na slici.

Mnogi oblici učenja uključuju podešavanja težinskih faktora da bi se minimizovao gubitak, te bi bilo korisno imati intiutivnu sliku o tome šta se dešava u **prostoru težinskih faktora** (prostor težina) – prostoru definisanom svim mogućim vrijednostima težina. Za jednodimenzionalnu linearnu regresiju, prostor težina definisan sa w_0, i w_1 je dvodimenzionalan, te možemo prikazati funkciju gubitka hipoteze na 3D grafiku kao funkciju od w_0 i w_1. Vidi se da je funkcija gubitka **konveksna,** odnosno da ne postoje lokalni minimumi.

Drugi način da dođemo do težinskih faktora koji minimizuju funkciju gubitka hipoteze je da se primijeni hill-climbing algoritam:

w ← bilo koja tačka u prostoru težina
ponavljaj konvergirajući ka minimumu
  za svako w_i **u w radi**

$$w_i \leftarrow w_i-\alpha\cdot\frac{\partial}{\partial w_i}Loss(w) \qquad (1)$$

Parametar α, koji se naziva **korak**, je zapravo **brzina učenja** kad pokušavamo minimizovati gubitak u problemu učenja. Može biti fiksna konstanta, ili može da se smanjuje kako proces učenja napreduje.

Za jednodimenzionalnu regresiju, funkcija gubitka je kvadratna funkcija, te će parcijalni izvodi biti linearne funkcije. Da izračunamo parcijalni izvod u opštem slučaju za jedan primjerak skupa za učenje, (x, y):

$$\begin{aligned}\frac{\partial}{\partial w_i}Loss(w)&=\frac{\partial}{\partial w_i}(y-h_w(x))^2\\&=2(y-h_w(x))\cdot\frac{\partial}{\partial w_i}(y-h_w(x))\\&=2(y-h_w(x))\cdot\frac{\partial}{\partial w_i}(y-(w_1x+w_0))\end{aligned}$$

I kad primijenimo na w_0 i w_1

$$\frac{\partial}{\partial w_0}Loss(w)=-2(y-h_w(x))$$

$$\frac{\partial}{\partial w_1} Loss(w) = -2(y - h_w(x)) \cdot x$$

I sve vratimo u jednačinu (1) koristeći parametar α, dobijamo da su pravila za učenje težina sledeča:

$$w_0 \leftarrow w_0 + \alpha \cdot (y - h_w(x))$$

$$w_1 \leftarrow w_1 + \alpha \cdot (y - h_w(x)) \cdot x$$

Ova pravila imaju intuitivnog smisla: ako je $h_w(x) > y$, izlaz hipoteze je veći od očekivanog, te w_0 umanjimo, kao i w_1 ako je x pozitivan ulaz, ili povećamo ako je x negativan ulaz.

Prethodna jednakost pokriva samo jedan primjerak skupa za treniranje. Za N primjeraka, želimo da minimizujemo sumu pojedinačnih gubitaka na svakom primjerku. Izvod sume je suma izvoda, te dobijamo:

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_w(x_j))$$

$$w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_w(x_j)) \cdot x_j$$

Ova se naziva pravilom **batch gradijentnog spusta** za jednodimenzionalnu linearnu regresiju. Konvergencija ka lokalnom minimumu je garantovana (sve dok biramo α dovoljno malo) ali može biti veoma spora.

2.3.2 Višedimenzionalna linearna regresija

Problem **linearne regresije** možemo uopštiti na višedimenzionalni slučaj, u kome je svaki primjerak x_j *n-dimenzionalni* vektor. Prostor hipoteza je u tom sličaju skup funkcija oblika:

$$h_{sw}(x_j) = w_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = w_0 + \sum_i x w_i x_{j,i}$$

Uz w_0 možemo staviti lažni ulaz $x_{j,0}$ koji će uvijek biti 1. Tada je h vektorski proizvod težina i ulaznog vektora (ili matrični proizvod transponovanog vektora težina i ulaznog vektora):

$$h_{sw}(x_j) = w \cdot x_j = w^T x_j = \sum_i w_i x_{j,i}$$

Vektor u kom se dostiže minimalna kvadratna greška, **w*** se računa:

$$w^* = argmin_w \sum_j L_2(y_j, w \cdot x_j)$$

Pravila izmjene težina za svaki w_i u cilju dostizanja minimuma funkcije gubitka hipoteze su:

$$w_i \leftarrow w_i + \alpha \sum_j x_{j,i} (y - h_w(x_j))$$

Kod jednodimenzionalne linearne regresije nismo morali da mislimo o overfittingu[1].

1 Overfitting je preveliko "navikavanje" modela na podatke iz skupa za treniranje. To znači da će hipoteza dati dobre rezultate na skupu za treniranje, ali će loše generalizovati nove primjerke.

Da bi se izbjegao overfitting kod višedimenzionalne linearne regresije se koristi **regularizacija**. Regularizacija minimizuje **cijenu hipoteze**:

$$Cost(h) = Loss(h) + \lambda Complexity(h)$$

gdje je

$$Complexity(h_w) = L_q(w) = \sum_j |w_i|^q$$

Kao i kod funkcije gubitka hipoteze, za *q=1* imamo L_1 regularizaciju, koja minimizuje sumu apsolutnih vrijednosti; za *q=2* imamo L_2 regularizaciju koja minimizuje sumu kvadrata.

2.3.3 Linearni klasifikatori sa jakim pragom

Podsjetimo se da je **klasifikacija** traženje hipoteze unutar skupa funkcija sa diskretnim kodomenom {(sunčano, oblačno, kišno), (0, 1)}.

Linearne funkcije se mogu koristiti i za klasifikaciju. Posmatrajmo skup hipoteza u sledećoj formi:

$$h_w(x) = \begin{cases} 1, & Lin_w(x) \geq 0 \\ 0, & Lin_w(x) < 0 \end{cases}$$
$$w - uređeni\ par\ (w_0, w_1)$$

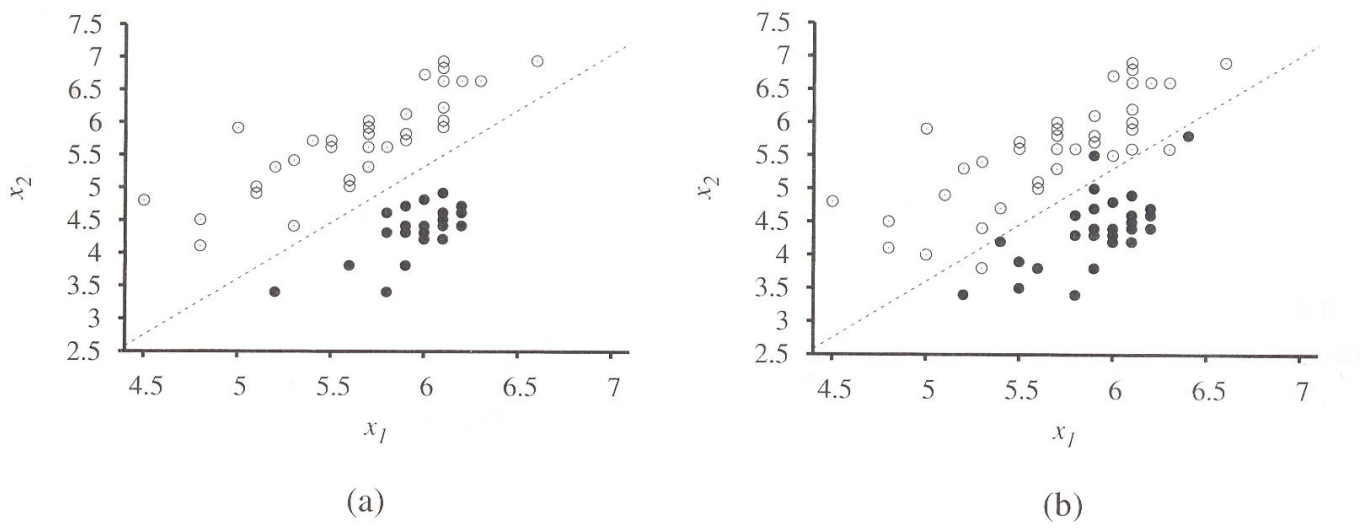
gdje je

$$Lin_w(x) = w_0 + w_1 x$$

Definišimo odstupanje u odnosu na dati skup podataka:

$$Error(h_w) = \sum_{i=1}^{N} \left| sgn(h_w(x) - y_i) \right|.$$

Na primjer, na slici 2.5(a) su prikazane tačke iz dvije klase: zemljotresi (interesantne seizmolozima) i podzemne eksplozije (interesantne vojnim ekspertima). Svaka tačka je definisana sa dvije ulazne vrijednosti, x_1 i x_2, koje ukazuju na jačinu potresa izračunatu koristeći seizmičke signale. Za ove podatke, zadatak klasifikacije je da se nađe hipoteza *h* koja će za nove tačke *(x_1, x_2)* da vrati 0 ako je zemljotres ili 1 ako je eksplozija.



Slika 2.5 - (a) Grafik seizmičkih parametara, za zemljotrese i podzemne eksplozije. Prikazan je i linearni separator. (b) Isti domen, sa više podataka. Zemljotresi i eksplozije više nisu linearno separabilne.

Granica odlučivanja je linija (ili površ, u prostorima veće dimenzije) koja dijeli ove dvije klase. Na slici 2.5(a), granica je prava. Linearna granica odlučivanja se naziva **linearni separator**, a za skup podataka na kojima je dobijen se naziva **linearno separabilnim**. Linearni separator za sliku 2.5(a) je definisan sa:,

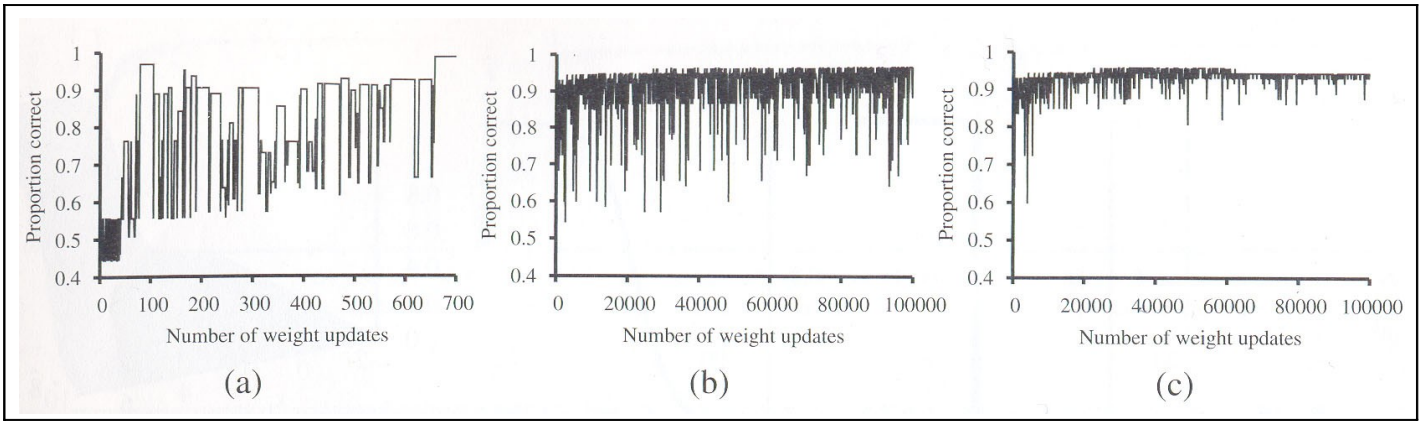
$$-4.9x_0+1.7x_1-x_2=0$$

Eksplozije koje želimo da klasifikujemo vrijednošću 1, se nalaze ispod prave, sa većim vrijednostima za x_1 nego x_2, te za njih važi da je $-4.9 + 1.7x_1 - x_2 > 0$, dok je za zemljotrese $-4.9 + 1.7x_1 - x_2 < 0$. Koristeći konvenciju da je $x_0 = 1$, možemo da zapišemo klasifikacionu hipotezu kao

$$h_w(x)=1\ ako\ w\cdot x\geq 0\ i\ 0\ u\ suprotnom$$
$$(w\cdot x-skalarni\ proizvod)$$

Alternativno, možemo posmatrati *h* kao rezultat kompozicije linearne funkcije **w*x** sa **funkcijom sa pragom**:

$$h_w(x)=Threshold(w\cdot x)\ gdje\ je\ Threshold(z)=1\ ako\ z\geq 0\ i\ 0\ u\ suprotnom$$



Slika 2.6 - (a) Grafik preciznosti naspram broja iteracija kroz skup za obučavanje koristeći pravilo učenja perceptrona za podatke sa slike 2.5. (b) Isti grafik za podatke sa šumom koji nisu linearno separabilno sa promjenom skale na *x-osi*. (c) Isti grafik kao (b) sa brzinom učenja $\alpha(t)=1000/(1000+t)$.

Postoji jednostavno pravilo za računanje težina koje konvergira linearnom separatoru sa najmanjim odstupanjem (Error(h_w)). Za jedan primjerak *(x,y)*[1], imamo:

$$w_i \leftarrow w_i + \alpha(y - h_w(x)) \cdot x_i$$

koja je jednaka pravilu za promjenu težina kod linearne regresije. Ovo pravilo se naziva **pravilo učenja perceptrona**. Obzirom da razmatramo 0/1 klasifikacioni problem ponašanje je drugačije. Odgovarajuća vrijednost *y* kao i rezultat hipoteze $h_w(x)$ su ili 0 ili 1, te imamo 3 mogućnosti.

Ako je izlaz tačan, tj. $y = h_w(x)$, ne mijenjamo težine
Ako je $y = 1$ ali je $h_w(x) = 0$, onda *povećamo* w_i ako je odgovarajući ulaz pozitivan ili *umanjimo* ako je x_i negativan. Ovo ima smisla, zato što povećavamo **w*x** kako bi izlaz $h_w(x)$ bio 1.
Ako je $y=0$ ali $h_w(x) = 1$, onda umanjimo w_i ako je odgovarajući ulaz x_i pozitivan ili povećamo ako je x_i negativan. Ovo ima smisla, zatko što umanjujemo **w*x** kako bi izlaz $h_w(x)$ bio 0.

Pravilo učenja se primjenjuje na po jednom primjerku u iteraciji, birajući primjerke slučajno. Na slici 2.6(a) je **kriva učenja** za ovo pravilo primijenjena na skupu zemljotresa/eksplozija. Kriva učenja mjeri performanse linearnog klasifikatora na fiksiranom skupu za treniranje dok proces učenja napreduje na istom skupu. Na grafiku se vidi da kriva konvergira ka linearnom separatoru sa minimalnim odstupanjem. Za ovaj slučaj potrebno je 657 koraka da bi se došlo do linearnog separatora sa minimalnim odstupanjem, dok je skup sadržao 63 primjerka, te je svaki primjerak učestvovao oko 10 puta.

Rekli smo da pravilo učenja perceptrona konvergira ka savršenom linearnom separatoru kada su podaci linearno separabilni, ali šta ako nisu? Ova situacija je česta u realnom svijetu. Kada primjeru sa zemljotresima/eksplozijama dodamo nekoliko novih tačaka koje će učiniti skup linearno neseparabilnim, pravilo učenja perceptrona čak i poslije 10,000 koraka neće uspjeti da nađe rješenje. Čak i ako uspije da nađe klasifikator sa minimalnim odstupanjem, nastaviće da mijenja težine. Generalno, pravilo učenja perceptrona ne mora konvergirati stabilnom rješenju za fiksnu brzinu učenja α. Međutim, ako pustimo da α opada (1/t, gdje je t broj iteracije), pokazuje se da pravilo konvergira rješenju sa najmanjom greškom kada se primjerci predaju slučajnim odabirom. Takođe je poznato da je nalaženje rješenja sa najmanjom greškom NP-težak problem, te je za očekivati veliki broj iteracija da bi smo dobili dobar rezultat. Na slici 2.6(c) se pokazuje da rezultat procesa učenja, sa brzinom učenja $\alpha(t)=1000/(1000+t)$, neće i dalje biti najbolje rješenje ni poslije 100,000 iteracija, ali je puno bolji od slučaja sa fiksnim α.

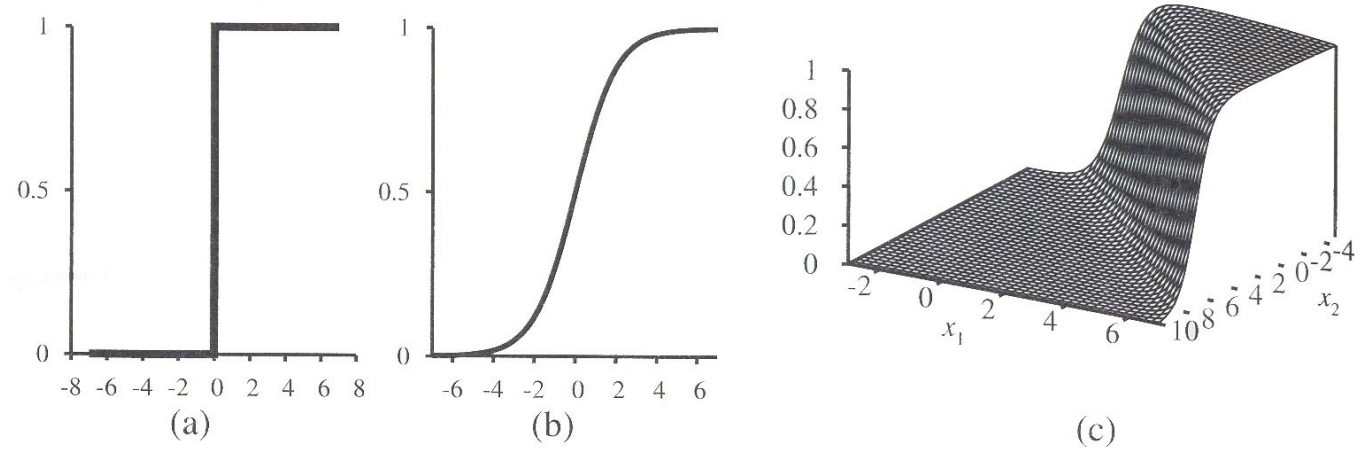
1 x - vektor ulaza, y - očekivana vrijednost

2.3.4 Linearna klasifikacija sa logističkom regresijom

Vidjeli smo da kompozicija linearne funkcije i funkcije sa pragom stvara linearni klasifikator. Međutim, ako koristimo klasifikator sa jakim pragom mogu nastati praktični problemi, jer odgovarajuća hipoteza $h_w(x)$ nije neprekidna funkcija ulaza i težina. To čini učenje perceptrona veoma nepredvidljivim. U tom slučaju, linearni klasifikator uvijek daje binarni odgovor (1 ili 0), čak i za slučajeve koji su blizu granice, a u velikom broju situacija nam je potreban precizniji odgovor. Ovi problemi se u velikoj mjeri mogu riješiti aproksimirajući jaki prag neprekidnom, diferencijabilnom funkcijom.

Logistička funkcija je data u sledećoj formi:

$$Logistic(z) = \frac{1}{1+e^{-z}}$$



Slika 2.7 - (a) Funkcija sa jakim pragom (na izlazu daje (0/1). (b) Logistička funkcija (naziva se često sigmoidna) $\frac{1}{1+e^{-x}}$. (c) Grafik hipoteze sa logističkom regresijom za podatke sa slike 2.6.

Zamjenom funkcije sa jakim pragom sa logističkom funkcijom dobijamo:

$$h_w(x) = Logistic(w \cdot x) = \frac{1}{1+e^{-w \cdot x}}$$

Primjer takve hipoteze za problem zemljotresa/eksplozija se može vidjeti na slici 2.7(c). Primjećujemo da izlaz, sada broj između 0 i 1, može biti interpretiran kao vjerovatnoća pripadnosti klasi 1.

Proces nalaženja težinskih faktora ovog modela minimizujući funkciju gubitka hipoteze na skupu za obučavanje je **logistička regresija**. Obzirom da hipoteza ne daje na izlazu samo 0 i 1, koristićemo L_2 funkciju greške; zbog čitljivosti, g je logistička funkcija dok je g' njen prvi izvod.

Za primjerak (**x**, y), nalaženje gradijenta je isto kao i za linearnu regresiju do tačke gdje se koristi prava forma hipoteze h. Imamo:

$$\frac{\partial}{\partial w_i} Loss(w) = \frac{\partial}{\partial w_i}(y - h_w(x))^2 = 2(y - h_w(x)) \cdot \frac{\partial}{\partial w_i}(y - h_w(x))$$

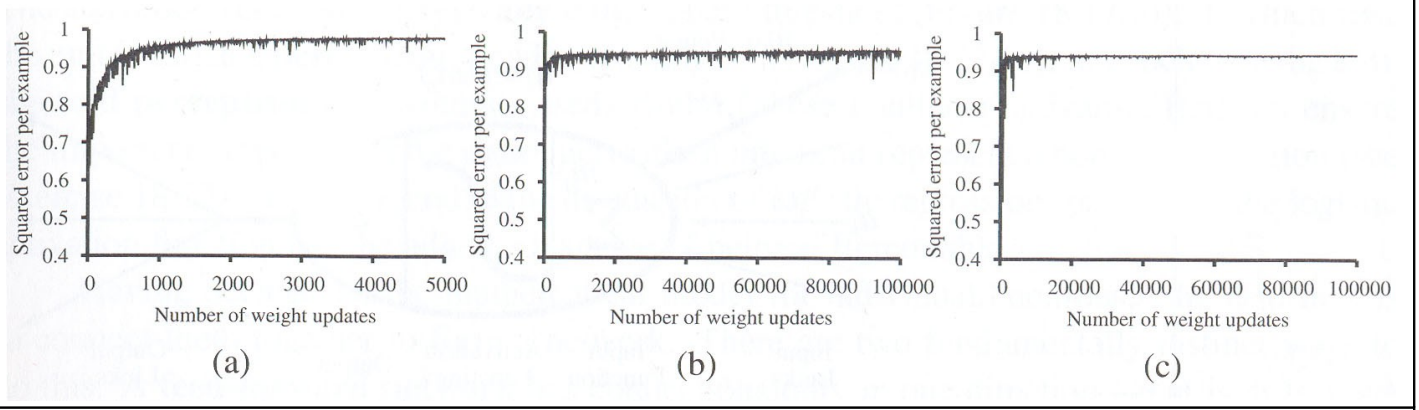
$$= -2(y - h_w(x)) \cdot g'(w \cdot x) \cdot \frac{\partial}{\partial w_i} w \cdot x = -2(y - h_w(x)) \cdot g'(w \cdot x) \cdot x_i$$

Izvod logističke funkcije g zadovoljava $g'(z)=g(z)(1-g(z))$, te imamo

$$g'(w \cdot x) = g(w \cdot x)(1 - g(w \cdot x)) = h_w(x)(1 - h_w(x))$$

pa pravilo računanja težina minimizujući gubitak je

$$w_i \leftarrow w_i + \alpha(y - h_w(x)) \cdot h_w(x)(1 - h_w(x)) \cdot x_i$$



Slika 2.8 - Ponavljanje eksperimenta sa slike 2.6 koristeći logističku regresiju i kvadratnu grešku.

Ponavljajući eksperiment sa slike 2.6 možemo vidjeti da logistička regresija sporije konvergira. Ako su nam ulazni podaci sa šumom ili nisu linearno separabilni, logistička regresija konvergira brže i pouzdanije. Ove prednosti se prenose i na podatke iz realnog svijeta, pa je samim tim logistička regresija postala jedna od najpopularnijih klasifikacionih tehnika za probleme u medicini, marketingu, analizi anketa, etc..

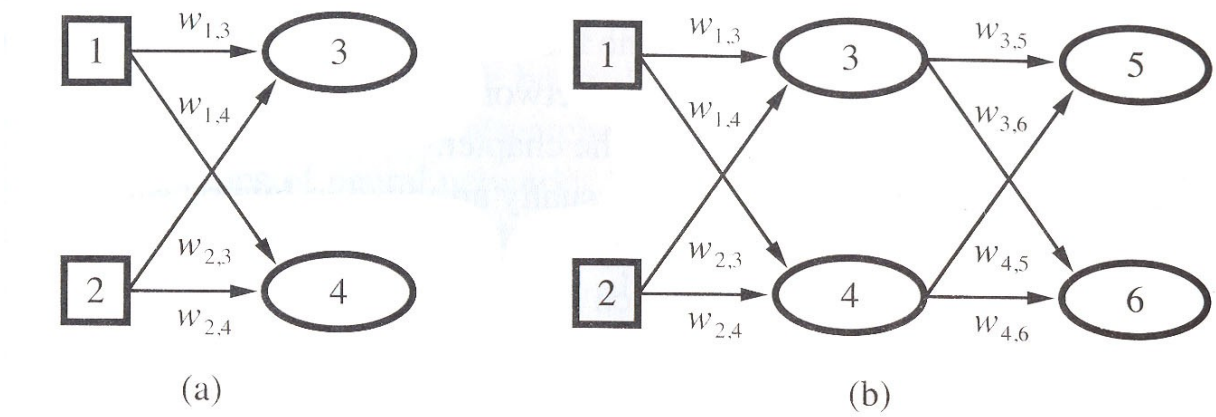
2.4 Jednoslojne feed-forward neuronske mreže (perceptroni)

Mreža u kojoj su svi ulazi direktno povezani na izlaze se zove **jednoslojna neuronska mreža**, ili **perceptron mreža**. Na slici 2.9(a) se može vidjeti primjer jednostavne perceptron mreže sa dva ulaza i dva izlaza. Sa ovakvom mrežom možemo probati, na primjer, da naučimo funkciju koja sabira 2 binarna broja. Podaci koji su nam potrebni:

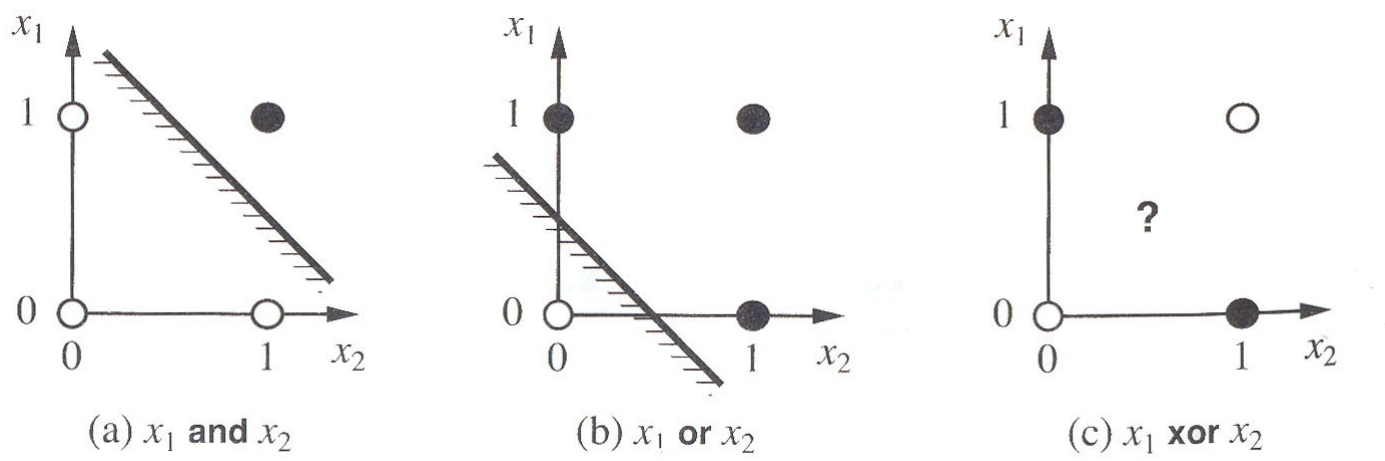
Tabela 1:

x_1	x_2	y_3 (carry)	y_4 (sum)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Primjećujemo da perceptron mrežu sa *m* izlaza sačinjava *m* različitih mreža, zato što pojedinačni težinski faktor utiče samo na jedan od izlaza. Prema tome, postojaće *m* odvojenih procesa učenja. Pored toga, zavisno od izbora aktivacione funkcije, proces učenja će biti ili **pravilo učenja perceptrona** ili gradijentni spust za **logističku regresiju**.
Ako probamo naučiti mrežu sa slike 2.9(a) da sabira dva binarna broja (tabela 1), dobijamo interesantne rezultate. Uočavamo da čvor 3 lako uči funkciju prenosa, dok čvor 4 ne uspijeva da nauči funkciju sume. Naime, funkcija prenosa (slika 2.10(a)), koja je logičko I, je linearno separabilna. Funkcija sume (slika 2.10(c)), koja je logičko XOR (ekskluzivno ILI), nije.



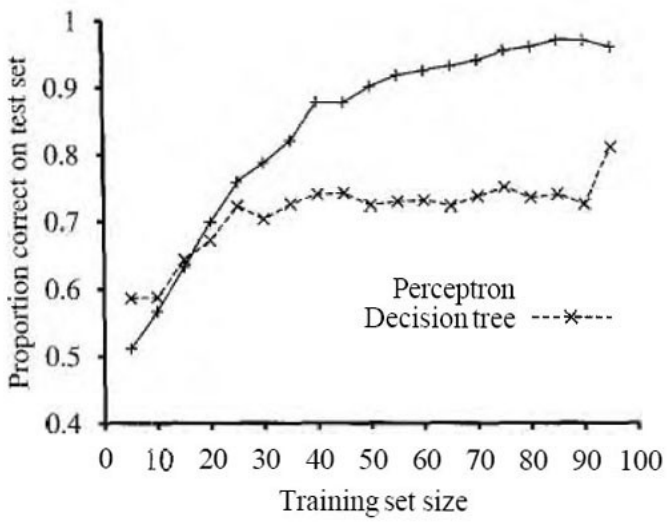
Slika 2.9 - (a) Preceptron mreža sa 2 ulaza i 2 izlaza. (b) Neuronska mreža sa 2 ulazna čvora, jednim skrivenim slojem sa 2 čvora i 2 izlazna čvora.



Slika 2.10 - Linearna separabilnost kod perceptron mreža sa aktivacionim funkcijama sa jakim pragom. Crne tačke predstavljaju podatke za koje je vrijednost funkcije 1, a bijele podatke za koje je vrijednost funkcije 0. Perceptron vraća 1 za region "iznad" prave. Za (c) ne postoji prava koja korektno klasifikuje ulaze.

Linearno separabilne funkcije čine samo mali dio Bulovih funkcija. Nemogućnost perceptrona da nauči jednostavne funkcije, kao što je XOR, je bila značajna prepreka zajednici koja je početkom šezdesetih istraživala neuronske mreže. Međutim, perceptroni su ipak korisni. Sigmoidni perceptron je i danas veoma popularna mreža. Perceptron može da predstavi neke "kompleksne" Bulovske funkcije veoma

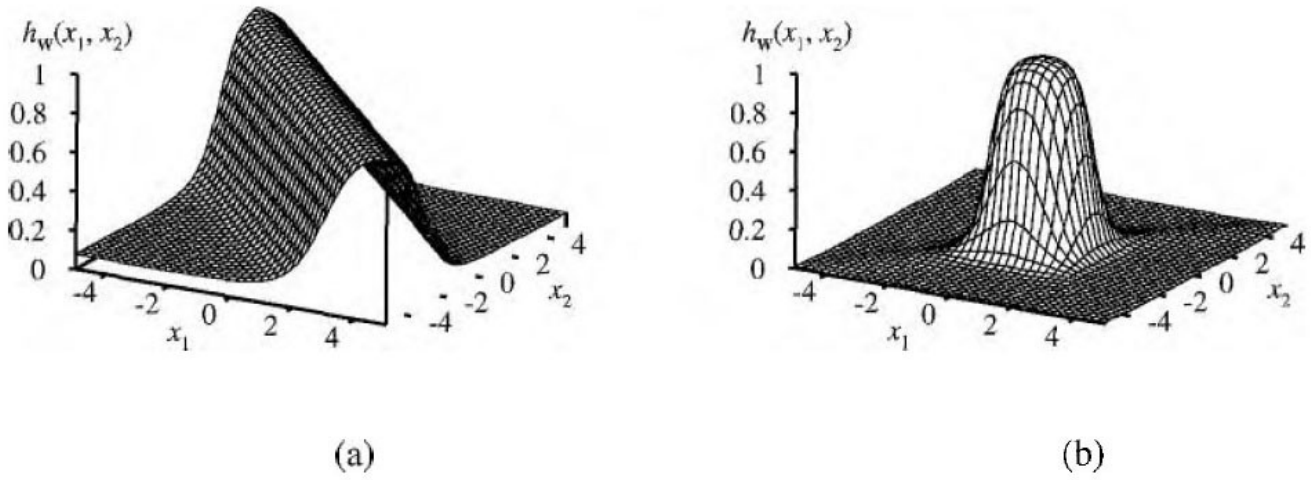
kompaktno. Na primjer, funkcija koja na izlazu daje 1 ako je više od pola ulaza jednako 1, se može predstaviti perceptronom kome su svi težinski faktori $w_1 = 1$ i $w_0 = n/2$.



Slika 2.11 - Poređenje perceptron mreže i drveta odlučivanja. Perceptroni su bolji za učenje većinske funkcije od 11 ulaza.

Na slici 2.11 se vidi kriva učenja za perceptron na dva različita problema. Lijevo je primjer krive za ucenje funkcije većine sa 11 Bulovskih ulaza (funckija daje 1 ako je više od 6 ulaza jednako 1). Kao što smo i objasnili, perceptron brzo uči ovu funkciju, jer je linarno separabilna.

2.5 Višeslojne feed-forward neuronske mreže



Slika 2.12 - (a) Rezultat kombinovanja 2 funkcije sa slabim pragom da bi se dobio greben. (b) Rezultat kombinovanja 2 grebena da bi se dobilo ispupčenje.

Od nastanka je bilo jasno da jednoslojna mreža neće moći riješiti sve probleme. Rad na tu temu

MekKuloha i Pitsa dokazuje da takva mreža može da predstavi osnovne Bulovske funkcije AND, OR i NOT i onda raspravlja da se bilo koja funkcionalnost može dobiti spajanjem velikog broja neurona u mreže (moguće rekurentne) sa proizvoljno mnogo slojeva. Problem je bio učenje takve mreže.

Ovo je zapravo jednostavan problem ako se mreža posmatra na pravi način: kao funkcija $h_w(x)$ parametrizovana težinskim faktorima **w**. Uzmimo u obzir jednostavnu mrežu sa slike 2.9(b), koja ima 2 ulaza, 2 skrivena čvora i 2 izlaza. (Podrazumijeva se da svaki čvor ima 1 lažan ulaz fiksiran na 1.) Za dati ulazni vektor $x = (x_1, x_2)$, aktivacije ulaznih jedinica su date sa $(a_1, a_2) = (x_1, x_2)$. Izlaz čvora 5 je dat sa

$$\begin{aligned} x_5 &= g(w_{0,5} + w_{3,5}a_3 + w_{4,5}a_4) \\ &= g(w_{0,5} + w_{3,5}g(w_{0,3} + w_{1,3}a_1 + w_{2,3}a_2) + w_{4,5}g(w_{0,4} + w_{1,4}a_1 + w_{2,4}a_2)) \\ &= g(w_{0,5} + w_{3,5}g(w_{0,3} + w_{1,3}x_1 + w_{2,3}x_2) + w_{4,5}g(w_{0,4} + w_{1,4}x_1 + w_{2,4}x_2)) \end{aligned}$$

Prema tome, imamo izlaz dat kao funkciju ulaza i težinskih faktora. Sličan izraz se dobija i za čvor 6. Sve dok možemo računati izvode ovih izraza po težinskim faktorima, možemo koristiti metodu minimizacije gubitka gradientnim spustom da bi naučili mrežu. Poglavlje 2.6 opisuje proces učenja. Obzirom da funkcija koju mreža predstavlja može biti jako nelinearna - složena od ugnježdenih nelinearnih funkcija sa slabim pragom - možemo koristiti neuronske mreže kao alat za **nelinearnu regresiju**.

Prije nego pređemo na pravila učenja, da prokomentarišemo na koji način neuronske mreže generišu komplikovane funkcije. Podsjetimo se da svaki čvor u sigmoidnoj mreži predstavlja funkciju sa slabim pragom u prostoru njegovih ulaza. (za ovo ce postojati slika u dijelu o regresiji) Sa jednim skrivenim slojem i jednim izlaznim slojem, svaki izlazni čvor računa linearnu kombinaciju više funkcija sa slabim pragom. Na primjer, sabirajuci dvije funkcije slabog praga suprotnih "smjerova" i primijenimo funkciju sa pragom, dobićemo "greben" funkciju kao na slici 2.12(a). Ukrštajući dva grebena pod određenim uglovima (kombinujući izlaze iz četiri čvora), možemo dobiti "ispupčenje" kao na slici 2.12(b).

Sa više skrivenih čvorova, možemo proizvesti više ispupčenja različitih veličina na različitim mjestima. Zapravo, sa jednim dovoljno velikim skrivenim slojem, moguće je predstaviti bilo koju neprekidnu funkciju ulaza sa proizvoljnom tačnošću; sa dva sloja je moguće predstaviti čak i funkcije sa prekidom. (Dokaz je kompleksan, ali je ideja da potreban broj sakrivenih čvorova eksponencijalno raste za broj ulaza. Na primjer, $2^n/n$ skrivenih čvorova je potrebno da bi se mogle predstaviti sve Bulovske funkcije sa n ulaza) Nažalost, za konkretnu strukturu mreže, težak problem je klasifikovati funkcije koje se mogu predstaviti datom mrežom i one koje ne mogu.

2.6 Učenje u višeslojnim mrežama

Jedan od manjih problema u višeslojnim mrežama jeste interakcija među problemima koji se pokušavaju naučiti kada mreža ima više izlaza. U ovom slučaju, trebamo uzeti u obzir da mreža implementira vektorsku funkciju $\mathbf{h}_w$ i da je izlaz mreže vektor **y**. Perceptron mrežu dekomponujemo na m problema u slučaju da ima m izlaza, ali to nije moguće sa višeslojnom mrežom. Srećom, ova zavisnost je veoma jednostavna u slučaju kad je funkcija greške aditivna po komponentama vektora greške $\mathbf{y} - \mathbf{h}_w(\mathbf{x})$. Za L_2 imamo, za bilo koju težinu w:

$$\frac{\partial}{\partial w} Loss(w) = \frac{\partial}{\partial w} |y - h_w(x)|^2 = \frac{\partial}{\partial w} \sum_k (y_k - a_k)^2 = \sum_k \frac{\partial}{\partial w} (y_k - a_k)^2 \quad (2)$$

a index k prolazi sve čvorove u izlaznom sloju. Svaki izraz u sumiranju je gradijent za gubitak k-og izlaza, izračunat kao da ostali izlazi ne postoje. Dakle, možemo dekomponovati problem učenja sa m-*izlaza* na

m zasebnih problema učenja, ali moramo uzeti u obzir da sumiramo gradijente svakog problema kad računamo nove težinske faktore.

Problem nastaje dodavanjem novih slojeva u mrežu. Iako grešku **y - h_w(x)** na izlazu lako računamo, greška u skrivenim slojevima je misterija obzirom da podaci za obučavanje nemaju podatak o tome koje izlazne vrijednosti bi skriveni čvorovi trebali imati. Srećom, možemo da **propagiramo unazad** grešku iz izlaznog sloja u skrivene slojeve. Proces propagiranja unazad direktno slijedi iz dobijanja ukupnog gradijenta greške.

U izlaznom sloju, pravilo računanja težinskih faktora je identično jednakosti (2). Imamo više izlaznih čvorova, i neka je Err_k k-ta komponenta vektora greške **y - h_w(x)**. Takođe ćemo definisati i modifikovanu grešku

$$\Delta k = Err_k \cdot g'(in_k)$$

te pravilo računanja težinskih faktora postaje

$$w_{j,k} \leftarrow w_{j,k} + \alpha \cdot a_j \cdot \Delta k$$

Da bi izračunali težine između ulaznih i skrivenih čvorova, trebamo definisati količinu analognu jednačini greške za izlazne čvorove. Ovdje propagiramo grešku unazad. Ideja je da je skriveni čvor j "odgovoran" za dio greške Δ_k u svakom izlaznom čvoru na koji je povezan. Prema tome, Δ_k se dijeli po težinama konekcija između skrivenih čvorova i izlaznih čvorova i propagira se unazad da bi smo dobili Δ_j vrijednosti u za skriveni sloj. Pravilo propagiranja za Δ vrijednosti je sledeće:

$$\Delta_j = g'(in_j) \sum_k w_{j,k} \Delta_k$$

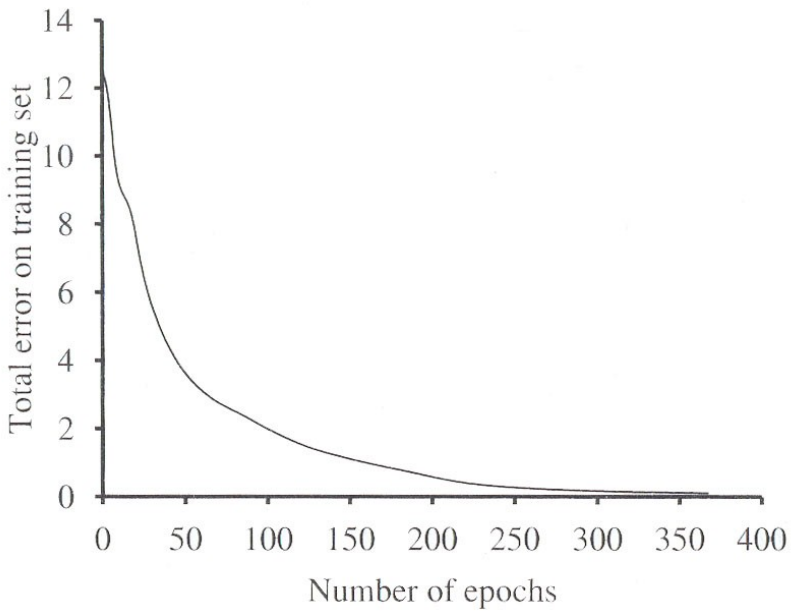
Sada je pravilo računanja za težinske faktore između ulaznih i skrivenih čvorova u suštini identično pravilu računanja za izlazne čvorove.

$$w_{i,j} \leftarrow w_{i,j} + \alpha \cdot a_j \cdot \Delta j$$

Proces propagiranja unazad je sledeći:

- Izračunati Δ vrijednostiza izlazni sloj, koristeći izračunatu grešku
- Počevši od izlaznog sloja, ponavljati za svaki sloj u mreži, dok se ne stigne do početnog skrivenog sloja
    - Propagirati Δ vrijednosti u prethodni sloj
    - Izračunati težine između 2 sloja

Za primjer ćemo uzeti ekskluzivno ILI (XOR) funkciju, koju perceptron mreža ne može da nauči. Prvo trebamo odrediti strukturu mreže. Imamo binarnu funkciju, pa ćemo imati 2 ulaza. Da li jedan skriveni sloj ili dva? Koliko neurona u skrivenoim slojevima? Potpuno povezana mreža? Ne postoji teorija koja će nam dati odgovor na ova pitanja. Možemo koristiti **unakrsnu validaciju**: probamo nekoliko različitih struktura, i zadržimo najbolju. Za ovaj problem, troslojna mreža je dovoljna, sa 2 neurona u skrivenom sloju. Na slici 2.13 se može vidjeti da mreža stvarno konvergira ka sve manjoj grešci na podacima.



Slika 2.13 - Kriva učenja koja pokazuje gradativno smanjenje greške kako se težine mijenjaju kroz epohe.

Neuronske mreže su u stanju da nauče daleko kompleksnije probleme od navedenog, s' time što je potrebno pronaći optimalnu strukturu mreže kako bi dobili najbolje rezultate. Do danas postoje na desetine hiljada radova objavljenih na temu primjene neuronskih mreža.

2.7 Određivanje strukture neuronske mreže

Sada smo utvrdili učenje težinskih faktora nad mrežom fiksne strukture. Treba da razumijemo i kako naći najbolju strukturu mreže. Ako izaberemo mrežu koja je prevelika, moći će da "upamti" sve primjere, ali to ne znači da će dobro generalizovati nove ulaze, koji nisu viđeni do tad. Drugim riječima, kao u svim statističkim modelima, neuronske mreže su podložne **overfitting-u** kada ima puno parametara u modelu. Ako se zadržimo na potpuno povezanim mrežama, jedini izbori koji se trebaju napraviti se tiču broja skrivenih slojeva i njihove veličine. Uobičajen pristup je da probamo par struktura, i zadržimo najbolju. Tehnike **unakrsne validacije** su dobar način za provjeru modela. Ovo znači da biramo arhitekturu mreže koja daje najveću tačnost predviđanja nad skupovima za testiranje.

Ako želimo uzeti u obzir mreže koje nisu potpuno povezane, onda moramo naći neku efektivnu metodu traženja kroz veoma veliki prostor mogućih topologija mreže. Algoritam **optimal brain damage** počinje od potpuno povezane mreže i napreduje uklanjajući veze iz nje. Pošto obučimo mrežu prvi put, pristup koristeći teoriju o informacijama identifikuje optimalni skup veza koje možemo ukloniti. Mreža se zatim ponovo obuči, i ako nema smanjenja performansi (smanjenja tačnosti predviđanja) ponavljamo proces. Uz uklanjanje konekcija, možemo pribjeći i uklanjanju čvorova koji ne doprinose rezultatu.

Nekoliko algoritama je takođe predloženo za dobijanje velike mreže od manje. Jedan od njih je algoritam **popločavanja**. Ideja je da se počne sa jednim čvorom koji sa najvećom tačnošću klasifikuje testni skup. Naredni čvorovi se dodaju da bi se obradili primjerci koje prethodni čvorovi nisu ispravno klasifikovali. Algoritam dodaje samo onoliko čvorova koliko je dovoljno da se pokriju svi primjerci testnog skupa.

3. Detekcija lica na slikama

Slike koje sadrže lica su veoma bitne u inteligentnoj čovjek-kompjuter interakciji baziranoj na percepciji, i napori istraživanja su usmjereni ka prepoznavanju lica, praćenju lica, procjene poze, kao i prepoznavanju izraza lica. Da bi se izgradio automatizovani sistem za analizu informacija koja lica sadrže, potreban je robustan i efikasan algoritam za detekciju lica. Za datu sliku, cilj detektora lica je da identifikuje sve regione na slici koji sadrže lice, bez obzira na poziciju, orijentaciju i osvjetljenje. Ovi problemi su izazovni zato što lica imaju veliki stepen varijabilnosti u veličini, obliku, boji i teksturi. Brojne tehnike su razvijene da bi se detektovala lica na slikama.

3.1 Lice?

Da li je moguće definisati pojam lica? Ova definicija bi bila veoma subjektivna, iz razloga što u realnom životu ljudi koriste različite "algoritme", kako za nalaženje objekata, tako i za prepoznavanje. Šta bi na primjeru iz *slike 3.1* trebalo klasifikovati kao lice?

Slika 3.1 - Lica?

Obzirom na ove poteškoće, preostaje nam da pokušamo da obučimo računar na osnovu primjera.

3.2 Formulacija problema

Za datu sliku, cilj detekcije lica je da zaključi da li postoje lica na slici, i ako postoje da vrati njihovu lokaciju.

Postoji puno usko povezanih problema sa detekcijom lica. *Lokalizacija lica* ima za cilj da nađe poziciju jednog lica na slici; ovo je pojednostavljen problem detekcije sa pretpostavkom da ulazna slika sadrži samo jedno lice. Cilj *detekcije svojstava lica* je da otkrije prisustvo i lokaciju očiju, nosa, obrva, usta, ušiju etc.. sa pretpostavkom da slika sadrži samo jedno lice. *Prepoznavanje lica* ili *identifikacija lica* upoređuje ulaznu sliku sa bazom i prijavljuje poklapanja, ako postoje. *Identifikacija lica* verifikuje identitet pojedinca za datu sliku, dok metode *praćenja lica* neprekidno procjenjuju lokaciju i moguću orijentaciju lica u sekvenci slika u realnom vremenu. *Prepoznavanje izraza lica* se bavi identifikovanjem emocionalnog stanja osoba (srećan, tužan, ljut, etc..). Očigledno da je detekcija lica prvi korak u bilo kom automatizovanom sistemu koji rješava pomenute probleme. Valja pomenuti da mnogi radovi koriste termin "detekcija lica", a metode nalaze samo jedno lice u ulaznoj slici.
Među brojnim algoritmima predloženim za detekciju lica, oni bazirani na algoritmima učenja su privukli puno pažnje zbog odličnih rezultata. Obzirom da se ove metode u velikoj mjeri oslanjaju na skup za treniranje, pomenućemo i neke baze koje odgovaraju ovom zadatku. Važan problem je kako vrednovati rezultate predloženih metoda za detekciju. Skoriji radovi na temu detekcije obično porede algoritme po stopi uspješne detekcije kao i stopi pogrešnih detekcija. Takođe treba pomenuti da postoje i neke opšte prihvaćene metrike za evaluaciju kao što su: brzina učenja, vrijeme izvršavanja, broj primjeraka potrebnih za obučavanje i odnos tačnih i pogrešnih detekcija. *Stopa detekcije* je odnos tačno detektovanih lica i broja lica koje prepoznaje čovjek. Dio slike identifikovan kao lice se uzima za tačan ako pokriva više od određenog procenta stvarnog lica. Detektori mogu da naprave dvije vrste grešaka: *false negatives* - nedetektovana lica na slici, *false positives* - detekcije koje zapravo nisu lica. Pravično vrednovanje bi trebalo da uzme u obzir sve navedene faktore, obzirom da se određeni parametri mogu podesiti da bi se povećala stopa detekcije ali se takođe povećava i broj pogrešnih detekcija.

Slika 3.2 - Na ovoj slici imamo 3 false positive detekcije, i 3 nedetektovana lica (false negative).

3.3 Problemi

Problemi kod detekcije lica se veže za sledeće faktore:

- **Poza** - Slike lica variraju, zavisno od relativne poze kamere i lica (frontalno, 45 stepeni, profil, odozgo), i određena svojstva lica kao što su oči ili nos mogu biti djelimično ili kompletno zaklonjeni.
- **Prisustvo ili odsustvo strukturnih komponenti** - Svojstva lica kao što su brada, brkovi i naočare mogu biti prisutni i postoji veliko variranje među ovim komponentama u obliku, boji i veličini.
- **Izraz lica** - Na detekciju lica direkno utiče izraz lica osobe.
- **Zaklonjenost** - Lica mogu biti djelimično zaklonjena drugim objektima. Na grupnim fotografijama , neka lica mogu biti zaklonjena drugim licima.
- **Orijentacija slike** - Slike lica variraju zavisno od rotacije u odnosu na optičku osu kamere.
- **Uslovi slikanja** - Kad se slika formira, faktori kao što je osvjetljenje (spektar, intenzitet) i karakteristike kamere (sočiva, odziv senzora) imaju uticaja na izgled lica.

Veoma je teško napraviti robustan model koji bi mogao da uspješno riješi sve od ovih problema. Sistemi za detekciju lica su obično kombinacija više algoritama, od kojih svaki prevazilazi određeni problem, kao i predparsiranje slike da bi se uklonile razlike u osvjetljenju i boje ujednačile (primjena određenih filtara na sliku).

4. Detekcija lica na slikama pomoću neuronskih mreža

4.1 Skup za obučavanje

Korišten je MIT CBCL skup slika za treniranje i testiranje. Slike su u PGM[1] formatu. Skup sadrži 30550 slika. Od toga 2429 su slike lica, dok ostatak čine slike koje ne sadrže lica.

Sliku u računaru predstavljamo kao matricu *MxN* gdje su M i N širina i visina respektivno. Obzirom da se radi o sivoskaliranim[2] slikama, vrijednosti u matrici su nijanse sive iz intervala [0-255]. Naše slike su veličine 19x19. Ova veličina je odgovarajuća jer je moguće detektovati lica malih dimenzija na velikim slikama. Smanjivanjem dimenzija, mogu se izgubiti određeni detalji bitni za detekciju lica, a na manjim dimenzijama čak i ljudi loše klasifikuju lica.

Obzirom da je ulaz u mrežu vektor, sliku 19x19 pretvaramo u vektor od 361 + 1 elementa, tako što zapisujemo vrijednosti piksela[3] , red po red u vektor, a posljednja vrijednosti je labela 0-1, tj. da li je slika lici ili ne. Radi lakšeg korištenja kompletan skup je pretvoren u CSV (Vrijednosti odvojene zarezom[4]) fajl.

Zbog numeričkih problema[5] koji se mogu javiti u toku obučavanja i loših performansi mreže kao klasifikatora, originalni podaci se skaliraju. U našem slučaju skaliramo podatke na interval [0-1] funkcijom:

$$f(x) = \frac{x}{255}$$

1 Portable Gray Map
2 Grayscale
3 Pixel
4 Comma separated values
5 Gubitak kod zaokruživanja, overflow i underflow

Na slike je takođe primijenjen filter za izjednačavanje histograma[1]. Slike ne moraju imati sve vrijednosti sive. Recimo da konkretna slika ima opseg vrijednosti [a-b]. Izjednačavanjem histograma vrijednosti slikamo u interval [0-255] funkcijom:

$$f(x) = \frac{x-a}{b-a} \cdot 255$$

Osim parametara, izbor primjeraka za trening skup kao i njihov broj može igrati važnu ulogu za performanse klasifikatora. *Bootstraping* znatno doprinosi poboljšanju skupa za obučavanje, a samim tim i boljem klasifikatoru. Ideja ovog procesa je da mrežu obučimo na početnom skupu, a poslije primjerke koje je pogrešno klasifikovala dodajemo skupu za obučavanje i ponovo je treniramo. Konkretno, u procesu detekcije lica želimo ovim putem da odstranimo jedan dio primjeraka koje mašina klasifikuje kao lica, iako ona u stvari to nisu (*false-positive*). Dakle, svaka slika koja nije lice a mašina je klasifikuje kao takvo moguće je dodati u skup za obučavanje sa labelom 0. Nakon toga ponavljamo proces obučavanja u nadi da ćemo ovakvim primjercima otkloniti jedan region slika koje nisu lica, a koji bi bili detektovani. Slično možemo uraditi i sa licima koja nisu detektovana.

Da bi izabrali parametre koji su optimalni za problem na kojem radimo potrebno je izvršiti unakrsnu validaciju[2]. Suština ove faze u obučavanju sastoji se u sledećem: naš skup podataka dijelimo na dva dijela, jedan na kojem ćemo obučavati mašinu i drugi na kojem ćemo testirati rezultate (podrazumijeva se da znamo labele svih primjera iz test skupa). Na startu određujemo opseg u kojem želimo da tražimo vrijednost parametara i vrijednost za koju povećavamo parametre u sledećoj iteraciji. Prvo obučavamo mašinu sa prvim parom parametara, onda testiramo njen rad na prethodno izdvojenom test skupu i pamtimo procenat tačnosti klasifikacije. Proces ponavljamo, s' tim što sada uzimamo nove vrijednosti parametara. Algoritam se završava kad se iscrpe sve vrijednosti parametara iz prethodno naznačenog opsega. Na kraju biramo one parametre za koje je preciznost klasifikacije bila najveća i sa njima obučavamo mrežu.

4.2 Struktura mreže

Probali smo nekoliko tipova 3-slojnih i 4-slojnih mreža. Testiranje ponašanja mreža je rađeno na manjim skupovima od oko 1000 primjeraka, tako što je mreža obučavana na 50 primjeraka, a na ostalih 950 testirana. Najbolje rezultate je dala 4-slojna mreža sa 256 neurona u prvom i 32 neurona u drugom skrivenom nivou. Zbog manjka računarske snage, aplikacija i prikazani rezultati su rađeni na 3-slojnoj mreži koja u skrivenom sloju ima 128 neurona.

Odabrana mreža je zatim obučavana na skupu od 371 primjerka izdvojenom iz skupa od 30550. Proces treniranja je trajao 33 minuta. (Na četvoroslojnoj mreži koja je dala najbolje rezultate ovaj proces bi trajao od 6-8 dana.)

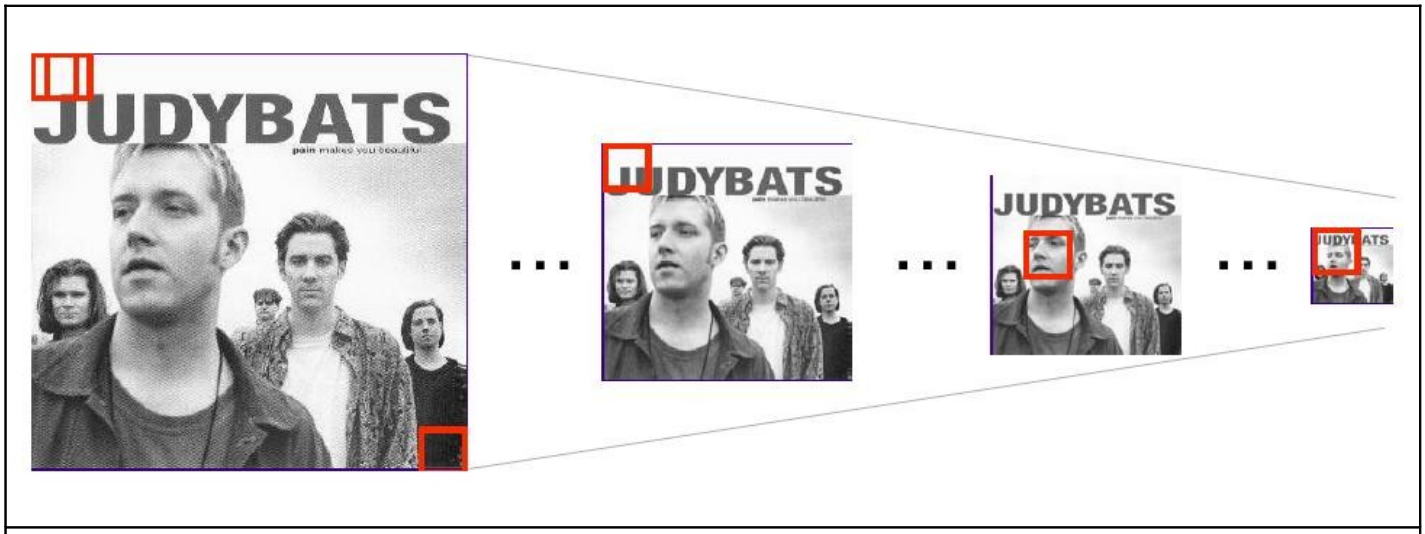
Aktivaciona funkcija je logistička funkcija. Algoritam učenja je propagacija unazad.

4.3 Rezultati

Detektor radi i sa slikama u boji i sa sivoskaliranim slikama. Za detekciju na većim slikama korištena je metoda klizajućeg prozora[3]. Ideja je da kvadratom dimenzija 19x19 "prođemo" cijelu sliku, tako što ga krećemo počevši od koordinate (0, 0) do koordinate (x_{max} - 19, y_{max} - 19). Svaki od kvadrata (regiona) predajemo klasifikatoru. Ako je dati region klasifikovan kao lice, uokvirimo ga i nastavljamo dalje.

1 Histogram equlization
2 Cross-validation
3 Sliding window

Ovim postupkom pronalazimo sva lica koja su dimenzija 19x19. Na slikama se najčešće nalaze lica većih dimenzija. Ovaj problem rješavamo umanjivanjem slike i ponavljanjem prethodno navedenog postupka. Umanjivanjem slike, lica koja su bila veća mogu odgovarati dimenzijama koje klasifikator prepoznaje. Slika se obično umanjuje za određeni faktor, na primjer dimenzije slike množimo sa 0.8 ili 0.9. U slučaju da na umanjenoj slici detektujemo lice treba da uokvirimo odgovarajući region na polaznoj slici. To postižemo time što koordinate piksela na kojem je detekcija vršena i dimenzije kvadrata koji ograničava region pomnožimo sa recipročnom vrijednošću faktora sa kojim smo umanjili sliku.



Slika 4.1 - Sliding window primjer

Konačan rezultat procesa je skup koordinata (x, y) i odgovarajućih veličina regiona na kojima je detektovano lice .

Data struktura je na skupu od 30279 primjeraka tačno klasifikovala: 29722 (98.16%).

Od 2314 lica tačno su klasifikovana: 2258 (97.57%).

Broj false-positive detekcija: 557 (1.79%).

5. Zaključak

Kao što se vidi iz rezultata, ovakav klasifikator daje veoma dobre rezultate na slike na kojima su lica prikazana frontalno.

Obzirom da je skup podataka sadržao samo frontalna lica, nije za očekivati da bi se rotirana lica mogla detektovati. Ovdje možemo pribjeći i drugim metodama koje se oslanjaju na strukturu lica (usta, nos, oči) i odnose između određenih elemenata, pa zatim region rotirati za određen ugao.

Jedan od, još uvijek, aktuelnih problema je prisustvo naočara i brade/brkova na slici. Ovi detalji mogu da poremete "strukturu" lica koju je mreža naulila i ne budu klasifikovana. Ovo zavisi i od skupa za obučavanje, tj. Koliki procenat lica sa naočarima/dlakama na licu je sadržan u skupu.

Sistemi za detekciju lica su, opet zavisno od skupa za obučavanje, osjetljivi na boju kože. Dakle, ako skup sadrži samo lica bijelaca, lica drugačije boje kože neće biti detektovana.

Još neki od problema su vezani za varijaciju osvijetljenosti na slikama, djelimičnog zaklanjanja lica, poze kao i izraza lica.

Neki od ovih problema mogu biti uklonjeni predprocesiranjem (predparsiranjem) slike, tako što se izjednači osvjetljenje i ujednači histogram (postoji pregršt metoda/filtera koji su korisni u ovoj fazi). Da bi se napravio rigidan model, dobar način je da se kombinuje nekoliko metoda. Na primjer smanjivanje oblasti pretraživanja detekcijom kože, rotiranje slika koristeći strukturne elemente lica i klasifikator baziran na neuronskim mrežama.

Detekcija lica je i dalje veoma izazovan i interesantan problem. Na nju se može posmatrati i kao na pokušaj rješavanja jednog od najvećih izazova kompjuterske percepcije, prepoznavanja objekata. Klasa lica ima veliki stepen varijabilnosti u obliku i boji zbog razlika između pojedinaca. Na to dodamo i varijabilnost pozadine, uslova slikanja i poze te imamo čitav spektar izazova koji su vezani za prepoznavanje objekata.

Iako je problem detekcije lica riješen na bezbroj mnogo načina, postoji puno detalja na kojima bi se moglo poraditi (algoritmi za smanjivanje regiona pretrage se line kao interesantan pravac, a i algoritmi zasnovani na strukturnim informacijama[1]). Zapostavljeni se čine i algoritmi koji rade sa slikama u boji. Boja je bitna informacija kad su u pitanju ljudski subjekti, te bi ona mogla igrati važnu ulogu u poboljšavanju postojećih algoritama.

Neuronske mreže se pokazuju kao dobar klasifikator, samim tim što su dijelom zasnovane na biološkim informacijama, iako još uvijek ne mogu dostići nivo ljudske preciznosti. Ostaje da vidimo da li će kombinovanje i primjena ovakvih algoritama dovesti do stvaranja vještačke inteligencije koja će biti u stanju sa da uči.

1 Odnosi i položaj bitnih djelova lica.

LITERATURA

1. Stuart Russel, Peter Norvig - A modern approach to artificial intelligence (3rd edition) ISBN-13: 978-0136042594, 2009.
2. Ming-Hsuan Zang, David Kriegman, Narendra Ahuja - Detecting Faces in Images: A Survey
3. Henry A. Rowley - Neural Network Based Face Detection (PhD Thesis)
4. Wilhelm Burger, Mark J. Burge - Digital Image Processing ISBN-13: 978-1846283796, 2007.
5. Marko Andrijašević — Detekcija lica na slikama (support vector machines), diplomski rad Prirodno-matematički fakultet, Univerzitet Crne Gore, Podgorica 2008.
6. Wikipedia