

Strojno učenje

Model = apstrakcija/preslikavanje iz ulaza na izlaz; u strojnom učenju parametriziran

Ideja -> Izgraditi model na temelju podataka tako da ugađamo parametre tog modela optimizirajući neki kriteriji (recimo točnosti ili pogreške) koristeći podatkovne primjere na ulazu

Cilj doći do modela (koji dobro generalizira)

Podatci -> Algoritam -> Model

Model mora moći predvidjeti svojstva neviđenih podataka – generalizirati; model radi indukciju

UI-potpuni problemi : Problemi koje ljudi rješavaju vrlo jednostavno a nama nije jasno kako, kada bi shvatili kako riješiti jedan takav problem onda bi mogli skoro svaki sličan

Strojno učenje korisno za:

- Probleme presložene za algoritamske metode

- Sustavi koji se dinamički mijenjaju – potrebna prilagodba

- Ogromne količine podataka (dana science, big data, dana mining)

Klasifikacija = ulaz se razvrstava u jednu od n klasa

Inteligentni sustavi se moraju moći prilagoditi svojoj okolini, sustav treba imati sposobnost učenja

Dubinska analiza podataka (dana mining) ili otkrivanje znanja u skupovima podataka je primjena strojnog učenja na (polu) strukturirane baze podataka

- npr.: analiza potrošačke košarice, određivanje kreditne sposobnosti, troubleshooting, postavljanje dijagnoza, optimizacija, analiza gena, klasifikacija teksta

Vrste strojnog učenja

1) Nadzirano

- Dajemo ulaz i **izlaz** i sustav na temelju tih parova treba naći preslikavanje na temelju ulaza x
- Ako je y diskretna/nebrojčana vrijednost -> klasifikacija
 - o Predviđanje ishoda izbora
- Ako je y kontinuirana/brojčana vrijednost -> regresija
 - o Predviđanje razine emisije NO2
- Najčešće korišteno

2) Nenadzirano

- Ne znamo što bi trebao biti izlaz, sustavu dajemo samo x-eve i nadamo se da će sustav sam pronaći neke pravilnosti
- Grupiranje (ulazni podatci koju su slični završavaju skupa)
 - o Analiza DNA-mikropolja
- Otkrivanje stršećih/novih vrijednosti (koji podatci od ulaznih nekako iskaču od ostalih?)
 - o Otkrivanje prijevare u kartičnom poslovanje
- Smanjenje dimenzionalnosti (ulazni skup podataka želimo sažeti na skup podataka koji možemo opisati s manje varijabli da vidimo koji su podatci na neki način povezani)

3) Podržano/ojačano

- Učenje optimalne strategije na temelju pokušaja s odgođenom nagradom
 - o Uspješni potezi se ojačavaju
 - o Šah, AlphaGo

Pet škola strojnog učenja:

- Symbolists (logika, indukcija),
- Connectionists (neuro, backpropagation),
- Evolutionaries (biologija, genetičko programiranje),
- Bayesians (statistika),
- Analogizers (psihologija, jezgri strojevi)

Nadzirano učenje

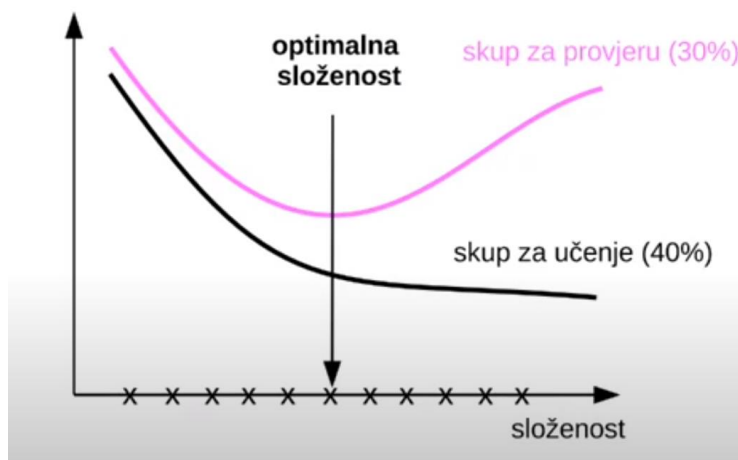
Treniranje i predikcija

Treniranje stvara model i onda taj model koristimo u predikciji

Kod treniranja je oznaka dana a kod predikcije oznaku dobijemo

Prenaučenost

- Ako model presložen previše se prilagodi podacima za treniranje a loše predikcije daje za neviđene primjere
- Glavni problem
- Izbjegavamo ga metodom unakrsne provjere
- Unakrsna provjera
 - o Skup podataka podijelimo na skup za učenje i skup za testiranje npr. 70:30
 - o Model se uči na skupu za učenje a zatim predikciju radimo na skupu za testiranje i na tom skupu računamo točnost (ili pogrešku)
- Ako naš model ima parametar kojim možemo odrediti njegovu složenost onda bi smo trebali odabrati optimalnu složenost – kako bi smo nju odredili treba nam dodatan skup -> skup za provjeru
 - o 40:30:30 skup za učenje, skup za provjeru, skup za testiranje
 - o Napravimo npr. 10 modela različite složenosti, svaki od njih ispitujemo na skupu za provjeru (odabiremo optimalni model) i onda njega testiramo na skupu za testiranje



Načelo parsimonije – treba dati prednost jednostavnijoj hipotezi, složenost dovoditi u hipotezu/model samo kada je nužno

Bayesov klasifikator

$$P(H_i|E) = \frac{P(E|H_i)P(H_i)}{\sum_j P(E|H_j)P(H_j)}$$

Bayesovo pravilo

Bayesovo pravilo se izravno može upotrijebiti kao klasifikacijski model

Hipoteza H = klasa y

Dokazi E1, ..., En = primjer x = (x1, ..., xn)

P(y) apriorna vjerojatnost klase

P(y|x) aposteriorna vjerojatnost klase – ova nas zanima

P(x|y) izglednost klase (vjerojatnost da klasa y ima primjer x)

Optimalna klasifikacijska odluka – primjer klasificiramo u klasu s najvećom aposteriornom vjerojatnošću

Takvu klasifikaciju nazivamo hipoteza maksimum aposteriori ili MAP-hipoteza

Hipoteza hMAP je klasa (oznaka) kojoj primjer najvjerojatnije pripada

Pretpostavka uvjetne nezavisnosti – $P(x_j, x_k | y) = P(x_j | y)P(x_k | y)$

$$h_{\text{MAP}} = \underset{y}{\operatorname{argmax}} P(y|\mathbf{x}) \stackrel{u.n.}{=} \underset{y}{\operatorname{argmax}} P(y) \prod_{j=1}^n P(x_j|y)$$

Ovaj model naziva se naivan Bayesov klasifikator.

Procjena najveće izglednosti (MLE) = procjena vjerojatnosti kao relativnih funkcija

$$P(y = v) = \frac{|D_{y=v}|}{|D|}$$
$$P(x_j = w | y = v) = \frac{|D_{y=v \wedge x_j=w}|}{|D_{v=y}|}$$

- Podložna je prenaučivosti
 - Izglednosti koje računamo će biti **nula** za one kombinacije koje se nisu pojavile u skupu za učenje
 - Rješenje: Zaglađivanje

- Preraspodijelimo masu vjerojatnosti s kombinacija koje smo vidjeli na kombinacije koje nismo vidjeli
- Najjednostavnije – Laplaceovo zaglađivanje

$$P(x_j = w | y = v) = \frac{|D_{y=v \wedge x_j=w}| + \alpha}{|D_{v=y}| + \alpha|V(x_j)|}$$

- - V(xj) je skup mogućih vrijednosti značajke xj, alfa je parametar zaglađivanja
 - alfa = 0 ne zaglađuje, alfa = 1 radi „dodaj jedan“ zaglađivanje

- model se puno lakše prenauci ako je skup za učenje malen (potreban veći alfa)
- ako je skup za učenje vrlo velik svejedno je zaglađujemo li ili ne
- podljev – ako imamo puno značajki za koje smo dobili male vjerojatnosti i sve njih množimo skupa dobit ćemo vrlo mali broj
 - rješenje -> računati u logaritamskoj domeni

$$h_{\text{MAP}} = \underset{y}{\operatorname{argmax}} P(y) \prod_{j=1}^n P(x_j|y)$$

$$= \underset{y}{\operatorname{argmax}} \left(\ln P(y) + \sum_{j=1}^n \ln P(x_j|y) \right)$$

- jednostavan i brz algoritam, linearan broju značajki (radi dobro s velikim brojem značajki)
- nedostatak što su u stvarnosti značajke rijetko nezavisne a što su zavisnije to je točnost klasifikacije lošija
- klasifikacija dokumenata, filtriranje neželjene pošte
- porodica je probabilističkog grafičkog modela
- ako pretpostavka o uvjetnoj nezavisnosti ne vrijedi može se koristiti polunavisan Bayesov klasifikator
- ako su značajke kontinuirane koristimo Gaussov Bayesov klasifikator jer koristimo Gaussove distribucije

Stablo odluke

Unutarnji čvorovi odgovaraju značajkama (atributima), grane ispod svakog čvora vrijednostima dotične značajke a listovi odgovaraju klasifikacijskim odlukama (oznakama klase).

ID3

U praksi preferiramo stabla sa što manje čvorova – izbjegavanje prenaučivosti

Nalaženje minimalnog stabla koje ispravno klasificira sve primjere iz skupa za učenje je NP-težak problem

- treba nam heuristički pristup

U korijenskom čvoru biramo između n značajki, zatim n-1, zatim ...

Preferiramo značajku koja što bolje diskriminira između primjera za učenje prema ciljnoj oznaci y.

Entropija je mjera neuređenosti sustava. Mi ju želimo što manju, savršena podjela ima 0, a najgora $\log_2(K)$ (1?).

$$E(D) = - \sum_{i=1}^K P(y = i) \log_2 P(y = i)$$

$P(y=i)$ je vjerojatnost klase $y = i$, dogovorno $0 \cdot \log_2 0 = 0$

"Koja mi značajka što bolje dijeli primjere po klasi"

Rekurzijom je entropija sve manja i konačno u listu 0

IG = informacijska dobit

= entropija početnog skupa - suma(udio podskupa u početnom skupu * entropija podskupa D za vrijednost v značajke x)

npr. za sunčano, oblačno, kišno

$$IG(D, x) = \underbrace{E(D)}_{\text{entropija početnog skupa}} - \sum_{v \in V(x)} \underbrace{\frac{|D_{x=v}|}{|D|} E(D_{x=v})}_{\text{očekivana vrijednost entropije nakon podjele skupa na temelju značajke}}$$

D je skup primjera, D_p je podskup primjera koji zadovoljava uvjet P

poslije minusa je očekivana entropija, želimo da je što manja -> da je IG što veća

Prema IG tražimo najbolju značajku

Računamo entropiju za svaku granu jer nam treba za IG.

Izračunamo IG za svaku značajku i odabiremo onu koja ima najveću.

Nakon što je odabrana mičemo ju iz skupa značajki i sad se spuštamo u jednu granu i gledamo podskup primjera u kojima prethodna značajka ima vrijednost trenutne grane.

Čvorovi koji savršeno diskriminiraju između primjera možemo zamijeniti listom (s oznakom odluke).

Algoritam

`D.isEmpty()` znači da otvorena grana nema nikakvih primjera, pa radimo fallback

odnosno staviti ćemo najčešću oznaku klase na cijelom podskupu

`x.isEmpty()` znači da nema više značajki po kojima mogu razdvojiti primjere

or

$E=0$, svi su primjeri u jednoj klasi i nijednoj drugoj

u x je najdiskriminativnija značajka

po toj značajci gradimo podgrane

idemo po svim vrijednostima značajke x (v)

Stablo uvijek savršeno klasificira sve primjere iz skupa D ako ne postoje identični primjeri s različitim oznakama

- To lako dovodi do prenaučivosti, loše klasificira neviđene primjere.

- Do nje dolazi ako primjeri za učenje imaju pogrešne vrijednosti značajki ili pogrešnu oznaku klase.

Ne želimo da stablo raste i raste jer mu ne možemo potpuno vjerovati.

Mi želimo izbjeći savršenu klasifikaciju jer nas savršena klasifikacija na skupu koji ima šum vodi do prenaučivosti.

Šum = pogrešna vrijednost značajki ili pogrešna oznaka klase u skupu za učenje

Kako spriječiti prenaučivost?

Ograničavanje rasta stabla prije nego što dosegne savršenu klasifikaciju

ili

Naknadno podrežemo prenaučeno stablo.

zamijenimo neka podstabla listovima ili

pretvorimo stabla u ako-onda pravila te uklanjanje uvjeta u antecedentima pravila

Kako odrediti optimalnu veličinu stabla?

Intrinzični (unaprijed definirana maksimalna dubina, br. primjera u nekom čvoru manji od unaprijed zadanog,

- pad entropije manji od unaprijed definiranog praga)

i ekstrinzični kriteriji (pad točnosti ili porast pogreške na skupu primjera za provjeru)

Postoje još C4.5 i QUEST algoritmi.

Ne mora se koristiti IG, može i hi kvadrat ili Gini indeks.

Stabla odluke mogu se koristiti i za regresiju (ciljna vrijednost nije klasa nego broj) (CART)

Ovo nije vjerojatnosni model, za razliku od Bayesovog klasifikatora, ali ga je lako interpretirati.

Sklono prenaučivosti, visoka varijanca (može se ublažiti primjenom ansambla stabala - model slučajne šume)

Stablo odluke rekurzivno particionira skup primjera prema vrijednostima značajki koje se odabiru na temelju informacijske dobiti (IG).

Umjetne neuronske mreže

Neuroračunarstvo grana je mekog računarstva.

Dva pristupa razvoju inteligentnih sustava:

Simbolistički pristup – znanje iz neke domene nastoji se obuhvatiti skupom simbola i zatim manipulirati tim simbolima pomoću algoritamskih pravila

Konektivistički pristup – izgraditi sustav koji ima arhitekturu sličnu arhitekturi mozga kojeg umjesto da se programira on uči samostalno na temelju iskustva

-> skup međusobno povezanih jednostavnih procesnih elemenata (neurona) čija se funkcionalnost temelji biološkom neuronu i koji služe distribuiranoj paralelnoj obradi podataka

- omogućuju robusnu obradu podataka

- mogu se koristiti za probleme klasifikacije te probleme regresije

- sposobne su učiti iz podataka

Dvije faze:

- faza učenja

- faza obrade podataka / iskorištavanje

Učenje je iterativan postupak predočavanja ulaznih primjera (i eventualno očekivanih izlaza) pri čemu dolazi do

postpunog prilagođavanja težina veza između neurona.

Epoha je jedno predočavanje svih uzoraka.

Razlikujemo:

Pojedinačno učenje (on-line): učenje se događa nakon svakog pojedinačnog uzorka

Učenje s minigrupama: nakon jedne minigrupe uzoraka

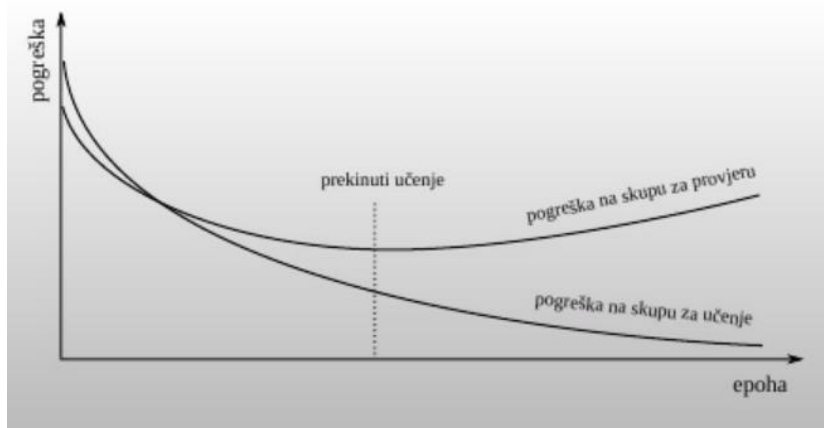
Grupno učenje: nakon svih predošćenih uzoraka

Znanje o izlazu je pohranjeno implicitno u težinama veza neurona.

Postupak učenja može biti: nadzirano (znamo izlaz), nenadzirano i podržano.

Skup primjera dijelimo na: skup za učenje, provjeru i testiranje.

Pratimo točnost i pazimo na pretreniranost (prekidamo učenje kad pogreška počne rasti na skupu za provjeru).



TLU-perceptron

- vrijednost sa svakog ulaza množi se s osjetljivošću tog ulaza w_i i akumulira u tijelu
- još se dodaje bias/pomak b/w_0

- suma se naziva net
$$net = \left(\sum_{i=1}^n x_i \cdot w_i \right) + w_0$$

- net se propušta kroz prijenosnu funkciju i time dobijemo izlaznu vrijednost

- step funkcija / funkcija skoka

- 0 ako $net < 0$ else 1

- ponekad -1 1

- funkcija identiteta $f(x) = x$; ADALINE-neuron

- sigmoidalna

- $1 / (1 + e^{-net})$

- 0 else 1, kontinuirani prijelaz između 0 i 1

- tangens hiperbolni

- -1 else 1

- zglobnica

- $\max(0, net)$

- propusna zglobnica

- if ≥ 0 net else $\alpha \cdot net$

- ima linearnu decizijsku granicu

- ne može riješiti problem klasifikacije linearno-nerazdvojivih razreda

Klasifikacija

- binarna ako 2 razreda

- dovoljan 1 neuron? funkcija skoka

- primjerice funkciju XOR ne može

- više razreda -> imamo toliko i izlaza, i-ti izlaz je 1 ako uzorak pripada i-tom razredu, ostali su 0 (1 hot encoding)
- decizijska granica kod TLU neurona je linearna, dijeli 2D prostor pravcem na dva dijela
- u 3D je decizijska granica ravnina a u višedimenzijalnim prostorima hiperravnina

Pravilo učenja perceptrona

Ciklički prolazi kroz svih N uzoraka za učenje, jedan po jedan.

Klasificiraj uzorak.

Ako se klasificira korektno, ne mijenjaj težine

Ako je to bio N-ti uzastopni uzorak prekini učenje

else

korigiraj težine

$$w_i(k+1) \leftarrow w_i(k) + \eta \cdot (t - o) \cdot x_i$$

Parametar η (eta) naziva se stopa učenja

- regulira u kojoj će se mjeri ažurirati trenutne vrijednosti težina
- mali η , sporo uči
- veliki η , postupak može divergirati

Epoha	Uzorak	(x_2, x_1, x_0)	t	(w_2, w_1, w_0)	<i>net</i>	o	Korekcija	$\eta \cdot (t - o)$
1	1	(2, 5, 1)	1	(1, 1.3, -5.85)	2.65	1	ne	×
1	2	(5, 2, 1)	1	(1, 1.3, -5.85)	1.75	1	ne	×
1	3	(1, 5, 1)	-1	(1, 1.3, -5.85)	1.65	1	da	-0.04
1	4	(5, 1, 1)	-1	(0.96, 1.1, -5.89)	0.01	1	da	-0.04
2	1	(2, 5, 1)	1	(0.76, 1.06, -5.93)	0.89	1	ne	×
2	2	(5, 2, 1)	1	(0.76, 1.06, -5.93)	-0.01	-1	da	0.04
2	3	(1, 5, 1)	-1	(0.96, 1.14, -5.89)	0.77	1	da	-0.04
2	4	(5, 1, 1)	-1	(0.92, 0.94, -5.93)	-0.39	-1	ne	×
3	1	(2, 5, 1)	1	(0.92, 0.94, -5.93)	0.61	1	ne	×
3	2	(5, 2, 1)	1	(0.92, 0.94, -5.93)	0.55	1	ne	×
3	3	(5, 1, 1)	-1	(0.92, 0.94, -5.93)	-0.31	-1	ne	×

Kada dobijemo <uzorakbroj> dobrih predikcija (korekcija ne) stajemo.

Neka neuron ima n ulaza; iste ćemo označiti s $\vec{x} = (x_1, x_2, \dots, x_n)$

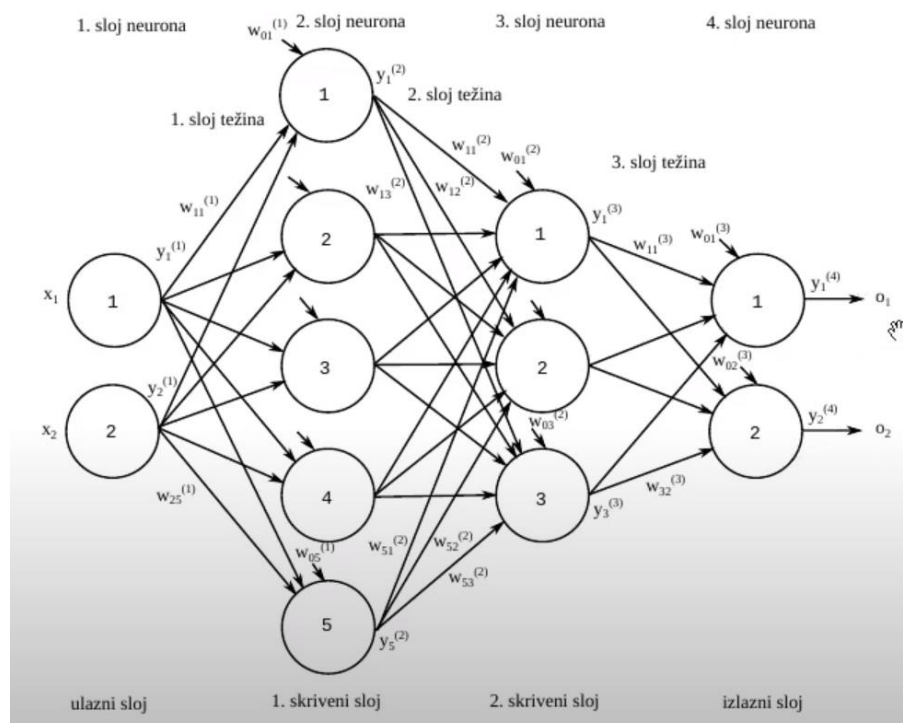
Tada ima i n težina $\vec{w} = (w_1, w_2, \dots, w_n)$ i jedan prag b (tj. w_0)

Tada vrijedi:

$$o = f(\text{net}) = f(\vec{w}^T \cdot \vec{x} + b)$$

Arhitekture neuronskih mreža

- kako su neuroni međusobno povezani
- ulazni sloj, skriveni sloj(evi), izlazni sloj
- slojevita
 - neuron iz jednog sloja dobiva na svoje ulaze izlaze samo prethodnog sloja
 - korisnik vidi ulazni i izlazni
 - ulazni neuroni ništa ne računaju
- neslojevita
 - mreža ima povratnu vezu zbog koje više nije unaprijedna
- mreža nije unaprijedna (povratna je) ako imamo petlju
- slojevita je po definiciji unaprijedna
- nije slojevita ali je unaprijedna, ako prima od prethodnog sloja i od n-1-1 sloja
- ako koristimo linearnu prijenosnu funkciju, ekspresivna moć izlaznog neurona je kao cijele mreže
- linearna kombinacija linearnih kombinacija je opet linearna kombinacija
 - da bi povećali ekspresivnost mreže moraju se koristiti prijenosne funkcije koje su nelinearne
 - do nedavno je to bila sigm, danas se prefeira ReLU (bolje za dublje mreže)



Pogledajmo sada prvi neuron u prvom skrivenom sloju. On računa:

$$y_1^{(2)} = f \left(\begin{bmatrix} w_{11}^{(1)} & w_{21}^{(1)} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{01}^{(1)} \right)$$

Drugi neuron u prvom skrivenom sloju računa:

$$y_2^{(2)} = f \left(\begin{bmatrix} w_{12}^{(1)} & w_{22}^{(1)} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + w_{02}^{(1)} \right)$$

Poslužimo li se matičnim zapisom, kompletan izlaz sloja možemo zapisati kao:

$$\begin{bmatrix} y_1^{(2)} \\ y_2^{(2)} \\ y_3^{(2)} \\ y_4^{(2)} \\ y_5^{(2)} \end{bmatrix} = f \left(\begin{bmatrix} w_{11}^{(1)} & w_{21}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} \\ w_{13}^{(1)} & w_{23}^{(1)} \\ w_{14}^{(1)} & w_{24}^{(1)} \\ w_{15}^{(1)} & w_{25}^{(1)} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} w_{01}^{(1)} \\ w_{02}^{(1)} \\ w_{03}^{(1)} \\ w_{04}^{(1)} \\ w_{05}^{(1)} \end{bmatrix} \right)$$

$$\vec{h}_i = f(\mathbf{W}_i \cdot \vec{h}_{i-1} + \vec{b}_i)$$

Izlaz i-tog skrivenog sloja

Funkcija pogreške E – srednje kvadratno odstupanje

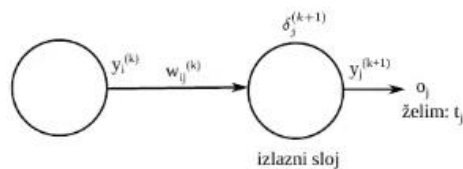
$$E = \frac{1}{2} \sum_{s=1}^N E(s) = \frac{1}{2} \sum_{s=1}^N \frac{1}{N} \sum_{i=1}^{N_o} (t_{s,i} - o_{s,i})^2$$

Učenje = podešavanje težina

Postupak propagacije pogreške unatrag (backpropagation) – Postupak učenja neuronskih mreža koji se temelji na učinkovitom izračunu svih parcijalnih derivacija i njihovoj primjeni na određivanje iznosa kojim korigiramo svaku od težina

Početno se sve težine postavi na slučajne vrijednosti.

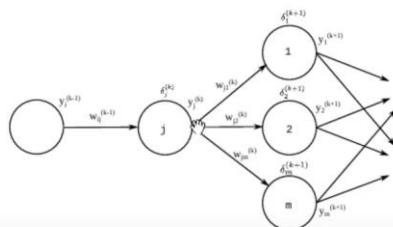
Izračun pogreške u neuronu izlaznog sloja.



Pogreška: umnožak derivacije prijenosne funkcije neurona $(y_j^{(k+1)} \cdot (1 - y_j^{(k+1)}))$ i stvarne pogreške $(t_j - o_j)$.

$$\delta_j^{k+1} = o_{s,j} \cdot (1 - o_{s,j}) \cdot (t_{s,j} - o_{s,j}).$$

Izračun pogreške u neuronu skrivenog sloja: $\delta_j^{(k)}$.



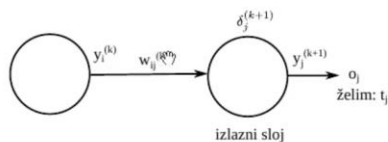
Pogreška: umnožak derivacije prijenosne funkcije promatranog neurona i težinske sume pogrešaka neurona kojima on šalje svoj izlaz:

$$\delta_j^{(k)} = y_j^{(k)} \cdot (1 - y_j^{(k)}) \cdot (w_{j,1}^{(k)} \cdot \delta_1^{(k+1)} + \dots + w_{j,m}^{(k)} \cdot \delta_m^{(k+1)}).$$



Postupak propagacije pogreške unatrag: algoritam

Korekcija: proporcionalno umnošku stope učenja η , izlaza neurona lijevo od težine i pogreške neurona desno od težine:



$$\Delta w_{ij}^k = \eta \cdot y_i^{(k)} \cdot \delta_j^{(k+1)}$$

$$w_{ij}^k \leftarrow w_{ij}^k + \Delta w_{ij}^k$$

Za pragove "lijevo" je konstanta 1, pa imamo:

$$\Delta w_{0j}^k = \eta \cdot \delta_j^{(k+1)}$$



Prirodom inspirirani optimizacijski algoritmi

Optimizacija – postupak pronalaženja najboljeg rješenja problema, rješenja s najmanjom cijenom

Constraint Satisfaction Problem (CSP) = vrsta pretraživanja prostora stanja kod kojeg put od početnog do konačnog stanja nije bitan, rješenje je isključivo konačno stanje

Problem trgovačkog putnika – NP težak problem (faktorijska složenost)

Heuristike – algoritmi – konstrukcijske (rješenje grade segment po segment), lokalne pretrage (kreću od početnog rješenja i pokušavaju ga inkrementalno poboljšavati)

Metaheuristike – heuristika koja vodi problemski specifične heuristike

- primjenjive na širok skup problema
- postupak pretraživanja dovode u dobar prostor rješenja na koji možemo primijeniti neku specifičnu heuristiku
- simulirano kaljenje, Tabu pretraživanje, evolucijski algoritmi, mravlji algoritmi, algoritmi rojeva, algoritmi umjetnih imunoloških sustava
- ^prve dvije rade s jednim rješenjem kojeg modificiraju a ove ostale istovremeno rade s čitavim nizom rješenja

Evolucija

Generiraj slučajnu populaciju od VEL_POP jedinki, evaluiraj svaku

Ponavljamo dok nije kraj

 Biram jedinke

 Dijete = križaj + mutacija

 Vrednuj dijete i ubaci u populaciju

Genetski algoritam

- radimo s populacijom kromosoma, svaki kromosom je jedno rješenje
- svako rješenje ima svoju dobrotu (fitness) ili kaznu
- ako tražimo minimum funkcije veći $f(x)$ znači manja dobrota dakle $f(x)$ odgovara kazni, a npr. $-f(x)$ dobroti; ako tražimo maksimum onda nam je dobrota pozitivna
- selekcija – što veći selekcijski pritisak to veću prednost dajemo boljim roditeljima – to brža konvergencija, ako prevelik onda konvergiraš **lokalnom** (loše) optimumu
 - najjači selekcijski pritisak bi bio odaberi najboljeg roditelja i to je to
 - ako jako slab onda efektivno imaš nasumično pretraživanje
- križanje – ustvari pretraživanje okoline roditelja
- mutacija – služi bijegu iz lokalnog ekstrema; veliki skokovi
 - isto ne smije biti preslaba ni prejaka
- bolje koristiti Grayev kod umjesto prirodnog binarnog jer je to kod s minimalnom promjenom
- uniformno križanje – za svaki bit biraš hoćeš iz prvog ili drugog roditelja
- izbor roditelja – više tehnika; jedna je proporcionalna selekcija (Roulette-wheel selection)

Mravlji algoritmi

- početno na svakom bridu jednak feromonski trag
- vjerojatnost da ćeš otići u neki drugi čvor s kojim si spojen se računa ovako

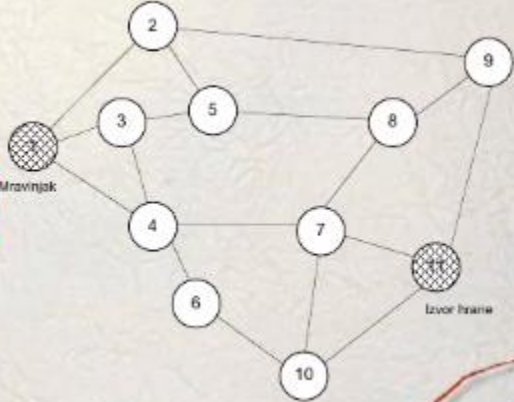
$$\tau_0 = konst$$

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha}{\sum_{l \in N_i^k} \tau_{il}^\alpha}, & \text{ako } j \in N_i^k \\ 0, & \text{ako } j \notin N_i^k \end{cases}$$


52

- Taj od kud si došao dodaš u listu zabranjenih
- dođemo do cilja i stanemo, sad ga možemo vrednovati
- zatim puštamo idućeg mrava odnosno ponavljamo proces
- ...
- ako si pustio 30 mrava imaš 30 staza, vrednuješ ih
- imaš matricu koja ti kaže koliko feromona ima gdje, primjerice red 1 stupac 3 ti kaže koliko na bridu između 1 i 3
- isparavaš ih tako da ih pomnožiš s brojem manjim od 1 – ro = stopa isparavanja
 - množimo sa 1 – ro
- onaj mrav koji je našao dugi put će ostavit malu količinu feromona a ovaj koji je kratak obrnuto
 - količina feromona proporcionalna dobroti rješenja
- parametar alfa utječe na mjeru kojom apsolutna vrijednost feromonskih tragova utječe na vjerojatnost
- poboljšanje

$$\tau_0 = \frac{m}{C^{nn}}$$

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in N_i^k} (\tau_{il}^\alpha \cdot \eta_{il}^\beta)}, & \text{ako } j \in N_i^k \\ 0, & \text{ako } j \notin N_i^k \end{cases}$$


53

m = broj mrava koji koristimo nakon svake iteracije kako bi deponirali feromonske tragove
 C^{nn} = nn je nearest neighbor

- Procedura: Ispari tragove

- Funkcija feromonske tragove na svim bridovima umanji za određeni iznos

$$\tau_{ij} \leftarrow \tau_{ij} \cdot (1 - \rho)$$

$$\Delta\tau_{ij}^k = \begin{cases} 1/C^k, & \text{ako je brid } i - j \text{ na stazi } k - \text{tog mrava} \\ 0, & \text{inaĉn} \end{cases}$$

- Novo stanje je tada:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k$$

58

Ažuriranje

$1/C^k$ je 1 kroz duljina mrava k