Inteligentni SAU

Žarko Zečević
Elektrotehnički fakultet
Univerzitet Crne Gore

Predavanje 2

Obučavanje jednoslojnih mreža

Ishodi učenja:

Nakon savladavanja gradiva sa ovog predavanja studenti će moći da:

- Definišu osnovne matematičke pojmove vezane za optimizaciju višedimenzionih funkcija
- Razumiju numeričke postupke za pronalaženje lokalnog minimuma funkcije
- Primjene metod najbržeg spusta/LMS algoritam/Njutnov metod na obučavanje jednoslojne linearne neuralne mreže
- Naprave razliku između različitih pristupa za obučavanje mreža

Obučavanje neuralnih mreža

Neuralna mreža predstavlja skup neurona organizovanih po slojevima. Izlazi jednog sloja su sinapsama (težinskim koeficijentima) povezani na ulaze drugih slojeva. Postoje razne arhitekture (strukture) neuralnih mreža. Obabir odgovarajuće arhitekture se vrši u skladu sa konkretnom primjenom.

Da bi neuralna mreža obavljala neku funkciju, nakon odabira arhitekture, potrebno je odrediti i vrijednosti njenih težinskih koeficijenata. Proces određivanja koeficijenata neuralne mreže se naziva *obučavanje neuralne mreže*. Postoji nekoliko načina za obučavanje neuralnih mreža:

- Nadgledano učenje (eng. supervised learning)
- Nenadgledano učenje (eng. unsupervised learning)
- Pojačano učenje (eng. reinforcement learning)

Kod nadgledanog učenja je dostupan set ulaznih i izlaznih podataka. Zadatak *algoritma za obučavanje* je da odredi koeficijente neuralne mreže tako da ona za zadate ulaze na svom izlazu generiše zadate izlaze.

Obučavanje neuralnih mreža

Algoritam za obučavanje adaptira koeficijente neuralne mreže tako što iterativno minimizuje zadatu *funkciju performanse.* Funkcija performanse je najčešće neka kvadratna funkcija koja zavisi od razlike između ulaznih i izlaznih podataka (signal greške).

Drugi tip učenja predstavlja nenadgledano učenje. Kod nenadgledanog su dostupni samo ulazni podaci, a neuralna mreža treba da nauči neke zakonitosti o ulaznim podacima. Kod ovog tipa učenja nije zadata funkcija na osnovu koje se ocjenjuju performanse učenja.

Kod pojačanog učenja takođe nijesu dostupni izlazni podaci, ali je poznata funkcija performanse. Kod ove vrste učenja se dobijaju nagrade/kazne za preuzete akcije, pro čemu se učenje vremenom „pojačava“ u smjeru dobijanja pozitivnih kritika.

U ovom predavanju biće prezentovani neki algoritmi za obučavanje neuralnih mreža koji iterativno minimizuju zadatu funkciju performanse. Prije toga biće napravljen kratak pregled osnovnih matematičkih pojmova koji su potrebni za dublje razumjevanje algoritama.

Funkcija performanse i njen minimum

Skalar λ nazivamo sopstvenom vrijednošću kvadratne matrice $\mathbf{A}$ ukoliko postoji nenulti vektor $\mathbf{q}$ takav da važi:

$$\mathbf{A}\mathbf{q} = \lambda\mathbf{q}.$$

Matrica $\mathbf{A}$ dimenzija $n \times n$ ima najviše n različitih sopstvenih vrijednosti koje se određuju na osnovu karakteristične jednačine:

$$\det(\lambda_i\mathbf{I} - \mathbf{A}) = 0.$$

Svakoj sopstvenoj vrijednost λ_i se pridružuje sopstveni vektor $\mathbf{q}_i$, koji se određuje na osnovu prve jednačine. Sopstveni vektori su linearno nezavisni.

Za matricu $\mathbf{A}$ kažemo da je pozitivno definitna $\left(\mathbf{A} \succ 0\right)$, ukoliko su sve njene sopstvene vrijednosti veće od 0. Matrica $\mathbf{A}$ je pozitivno semi-definitna ukoliko su joj sve sopstvene vrijednosti veće ili jednake 0. Matrica $\mathbf{A}$ je negativno definitna $(\mathbf{A} \prec 0)$, ukoliko su joj sve sopstvene vrijednosti manje od 0. Ako je matrica pozitivno definitna tada važi:

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} > 0 \text{ za bilo koje } \mathbf{x} \neq 0.$$

Funkcija performanse i njen minimum

Svaka multivarijabilna funkcija se u okolini neke tačke $\mathbf{x}_s$ može aproksimirati pomoću Tejlorovog reda:

$$f(\mathbf{x}) \approx f(\mathbf{x}_s) + \nabla f(\mathbf{x}_s)^T(\mathbf{x}-\mathbf{x}_s) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_s)^T\mathbf{H}(\mathbf{x}_s)(\mathbf{x}-\mathbf{x}_s),$$

gdje je $\nabla f(\mathbf{x}_s)$ gradijent funkcije $f(\boldsymbol{x})$ u tački $\mathbf{x}_s$:

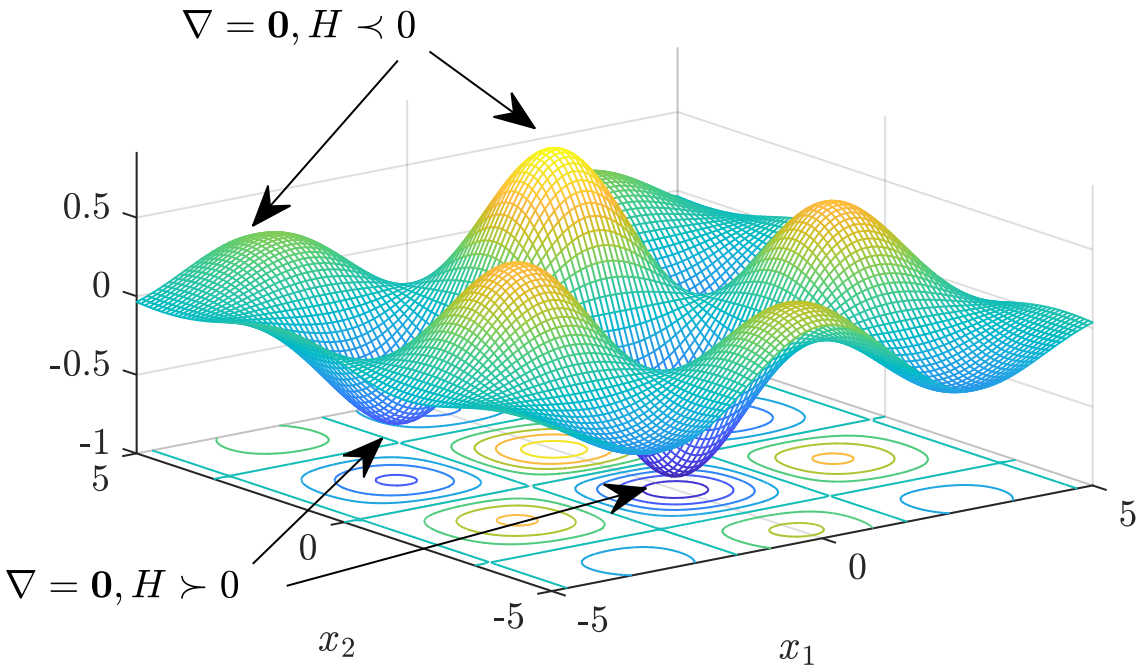
$$\nabla f(\mathbf{x}_s) = \left[\begin{array}{ccc} \dfrac{\partial f(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial f(\mathbf{x})}{\partial x_n} \end{array}\right]^T_{|\mathbf{x}=\mathbf{x}_s},$$

dok je $\mathbf{H}(\mathbf{x}_s)$ Hesijan funkcije $f(\mathbf{x})$ u tački $\mathbf{x}_s$:

$$\mathbf{H}(\mathbf{x}_s) = \left[\begin{array}{cccc} \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{array}\right]_{|\mathbf{x}=\mathbf{x}_s},$$

Funkcija performanse i njen minimum

Tačke funkcije u kojima je gradijent jednak nuli se zovu stacionarne tačke. Stacionarna tačka može da bude lokalni minimum, lokalni maksimum ili prevojna tačka funkcije. O karakteru stacionarne tačke možemo zaključiti na osnovu Hesijana. Ukoliko je Hesijan pozitivno definitan, tada stacionarna tačka predstavlja lokalni minimum. Ukoliko je Hesijan negativno definitan, tada stacionarna tačka predstavlja lokalni maksimum. Ukoliko Hesijan nije pozitivno definitan, niti negativno definitan, onda se radi o prevojnoj tački.



Funkcija performanse i njen minimum

Funkcija performanse često ima oblik takozvane *kvadratne forme* čiji je matematički zapis:

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c,$$

gdje je $\mathbf{A}$ simetrična matrica.

$$\nabla \mathbf{x}^T\mathbf{A}\mathbf{x} = 2\mathbf{A}\mathbf{x}$$

$$\nabla \mathbf{a}^T\mathbf{x} = \mathbf{a}$$

Gradijent kvadratne forme je:

$$\nabla F(\mathbf{x}) = \nabla\left(\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{d}^T\mathbf{x} + c\right) = \mathbf{A}\mathbf{x} + \mathbf{d}.$$

odakle se dobija stacionarna tačka:

$$\nabla F(\mathbf{x}_s) = \mathbf{0} = \mathbf{A}\mathbf{x}_s + \mathbf{d} \quad \rightarrow \mathbf{x}_s = -\mathbf{A}^{-1}\mathbf{d}.$$

Vrijednost funkcije $F(\mathbf{x})$ u stacionarnoj tački je:

$$F(\mathbf{x}_s) = -\frac{1}{2}\mathbf{d}^T\mathbf{A}^{-1}\mathbf{d} + c,$$

O prirodi stacionarne tačke se zaključuje na osnovu sopstvenih vrijednosti Hesijana:

$$\mathbf{H}(\mathbf{x}_s) = \nabla^2 F(\mathbf{x}_s) = \nabla\left(\mathbf{A}\mathbf{x} + \mathbf{d}\right) = \mathbf{A}.$$

Funkcija performanse i njen minimum

Primjer:

Funkcija $f(\mathbf{x}) = 2x_1^2 + 2x_2^2 + 2x_1x_2 - x_1 - x_2 + 2$ se može zapisati na sljedeći način:

$$f(\mathbf{x}) = \frac{1}{2}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \underbrace{\begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} -1 & -1 \end{bmatrix}^T}_{\mathbf{d}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 2.$$

Gradijent ove funkcije je:

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{d},$$

odakle se određuje stacionarna tačka:

$$\mathbf{x}_s = -\mathbf{A}^{-1}\mathbf{d} = \begin{bmatrix} 0.16 & 0.16 \end{bmatrix}^T.$$

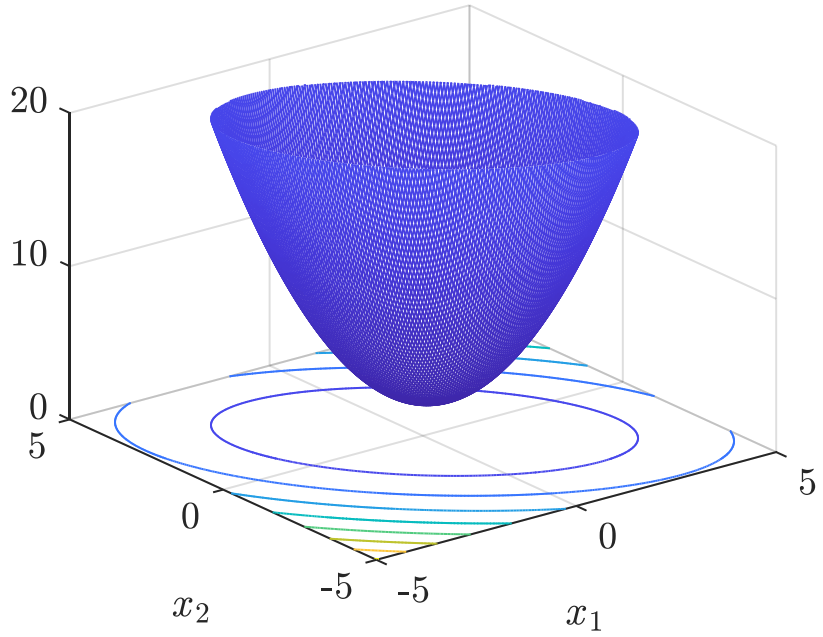
Sopstvene vrijednosti Hesijana su 2 i 6, što znači da u tački (0.16, 0.16) funkcija dostiže minimum koji je jednak:

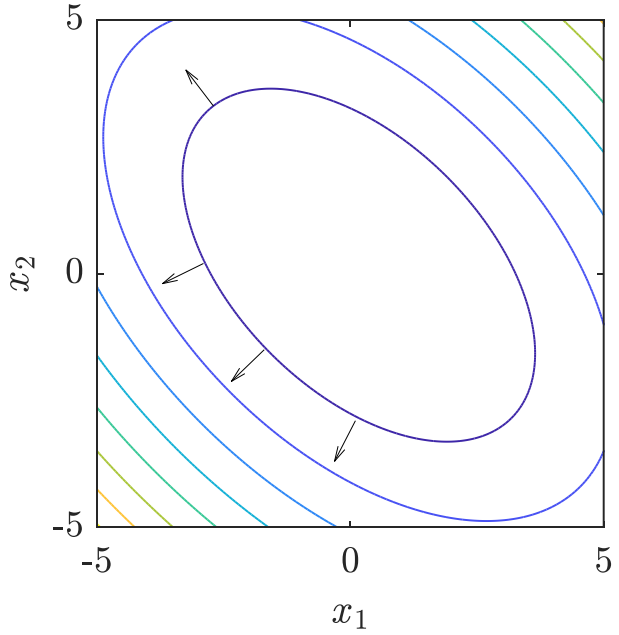
$$F(\mathbf{x}_s) = -\frac{1}{2}\mathbf{d}^T\mathbf{A}^{-1}\mathbf{d} + c = 1.83.$$

Primjer 1 – eliptički paraboloid

Na slici ispod (lijevo) je prikazana funkcija $f(\mathbf{x}) = 2x_1^2 + 2x_2^2 + 2x_1x_2 - x_1 - x_2 + 2$.

Na desnom grafiku su prikazane konturne funkcije. Konturne funkcije predstavljaju skup krivih $f(\mathbf{x})=c$, za različito c. Odnosno za dato c, to je skup tačaka u (x_1, x_2) ravni u kojima funkcija $f(\mathbf{x})$ ima vrijednost c. Na konturnom grafiku je prikazan gradijent za nekoliko različitih tačaka funkcije $f(\mathbf{x})$. Važno je uočiti da je gradijent zapravo vektor koji je usmjeren u pravcu maksimalnog rasta funkcije $f(\mathbf{x})$. Odnosno, funkcija $f(\mathbf{x})$ će se najviše povećati ukoliko se iz tačke $\mathbf{x}$ u sljedeću tačku pomjerimo u pravcu gradijenta. Gradijent predstavlja normalu na konturnu funkciju.



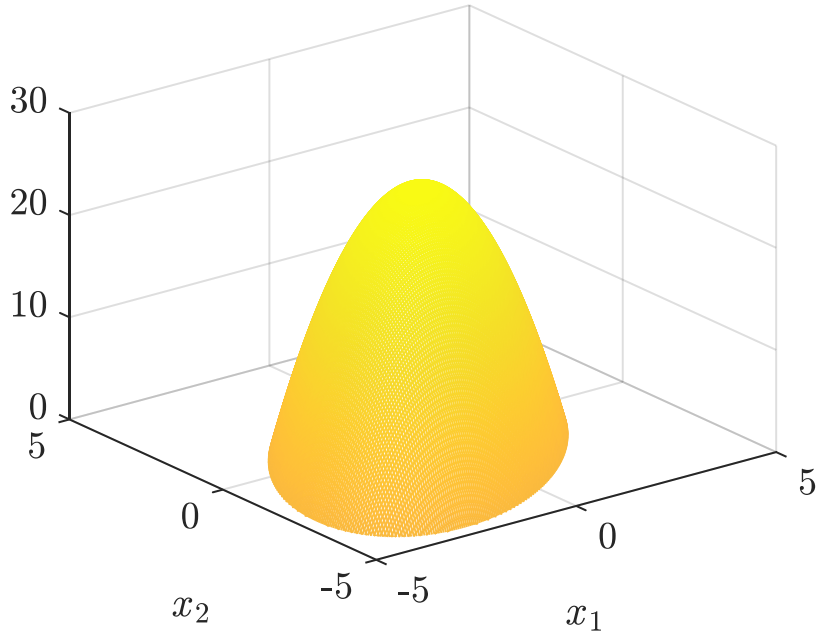


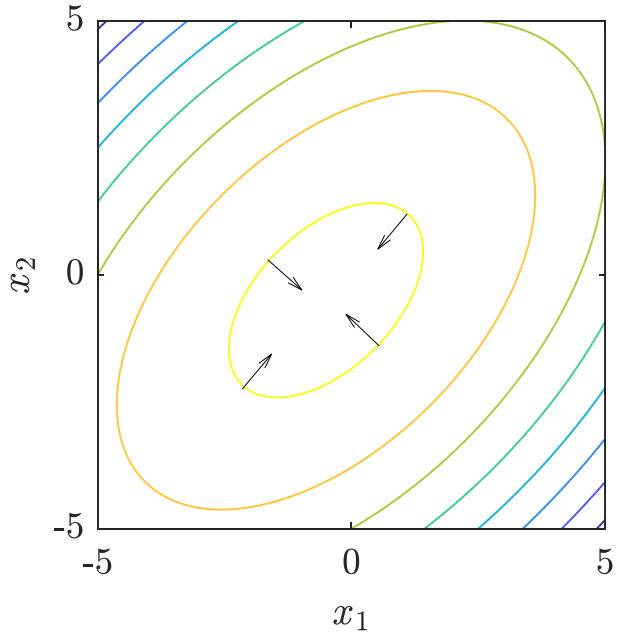
Primjer 2 – eliptički paraboloid

Ispod je prikazan grafik funkcije: $f(\mathbf{x}) = -2x_1^2 - 2x_2^2 + 2x_1x_2 - x_1 - x_2 + 2.$

Zadata funkcija takođe predstavlja eliptički paraboloid. Međutim za razliku od prethodne funkcije, ova funkcija ima globalni maksimum. Na konturnom grafiku se može uočiti da je gradijent usmjeren ka stacionarnoj tački, odnosno ka tački globalnog maksimuma. Gradijent funkcije i stacionarna tačka su:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} & \frac{\partial f(\mathbf{x})}{\partial x_2} \end{bmatrix}^T = \begin{bmatrix} -4x_1 + 2x_2 - 1, & 2x_1 - 4x_2 - 1 \end{bmatrix}^T \rightarrow \mathbf{x}_s = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$$



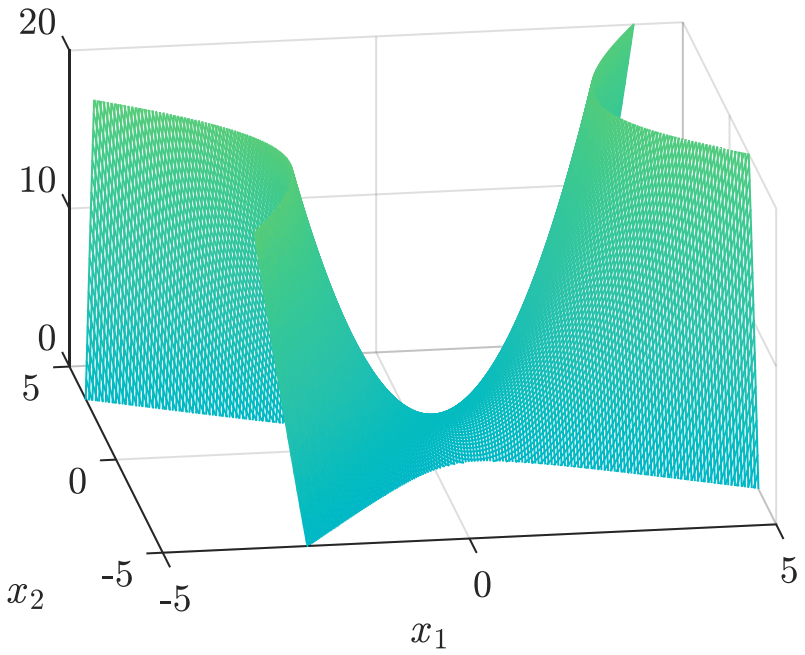


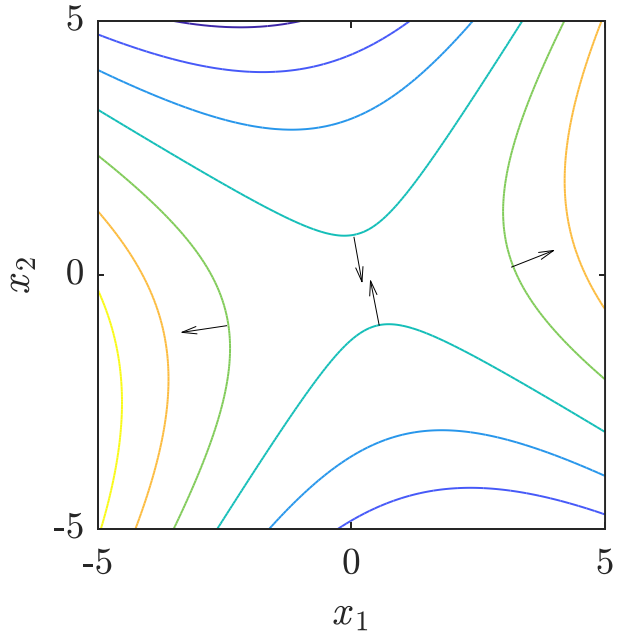
Primjer 3 – hiperbolički paraboloid

Na slici je prikazana funkcija $f(\mathbf{x}) = 2x_1^2 - 2x_2^2 + 2x_1x_2 - x_1 - x_2 + 2.$

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & -4 \end{bmatrix} = \mathbf{H}(\mathbf{x}_s).$$

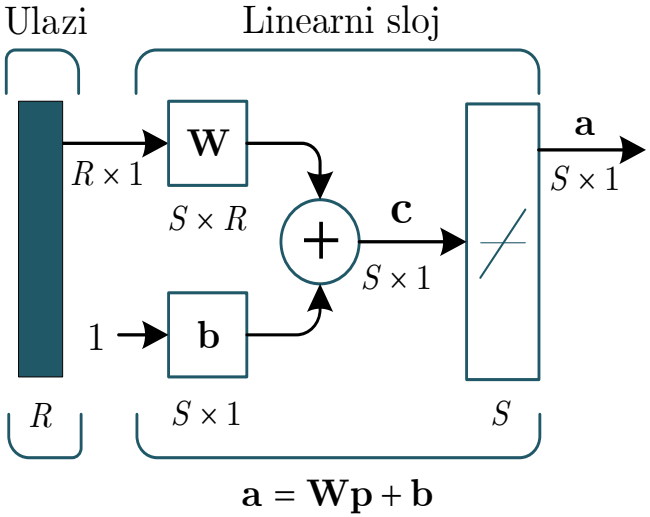
Sopstvene vrijednosti Hesijana su 4.47 i -4.47, što znači da stacionarna tačka u stvari predstavlja prevojnu tačku (`>> H=[4 2; 2 -4]; eig(H)`).





Obučavanje jednoslojne mreže

Adaline (adaptive linear neuron) predstavlja jednoslojnu neuralnu mrežu kod koje se koristi linearna aktivaciona funkcija. Njena struktura je prikazana na slici ispod.



Izlaz iz mreže je dat jednačinom:

$$\mathbf{a} = \mathbf{W}\mathbf{p} + \mathbf{b}.$$

$$\mathbf{w}_i = \begin{bmatrix} w_{i,1} & w_{i,2} & \cdots & w_{i,R} \end{bmatrix}^T$$

i-ta vrsta matrice $\mathbf{W}$

Odnosno, i-ti izlaz je jednak:

$$a_i = \mathbf{w}_i^T \mathbf{p} + b_i.$$

Obučavanje jednoslojne mreže

Pretpostavimo da je zadat set ulaznih i odgovarajućih izlaznih podataka:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}.$$

Cilj je podesiti koeficijente mreže tako da ona za ulazne podatke $\mathbf{p}_i$ na svom izlazu generiše izlaze $\mathbf{a}_i$ koji su što bliži željenim izlazima $\mathbf{t}_i$. Iz tog razloga koeficijenti neuralne mreže se određuju minimizacijom *funkcije performanse* koja se definiše kao srednja kvadratna razlika željenih izlaza i izlaza mreže (eng. mean square error, MSE):

$$\mathbf{F} = \mathbf{e}\mathbf{e}^T = \frac{1}{Q}\sum_{i=1}^{Q}(\mathbf{t}_i - \mathbf{a}_i)^T(\mathbf{t}_i - \mathbf{a}_i) = E\left((\mathbf{t} - \mathbf{a})^T(\mathbf{t} - \mathbf{a})\right).$$

Radi jednostavnosti, smatraćemo da mreža ima samo jedan neuron, odnosno jedan izlaz. Funkcija performanse u tom slučaju je skalarna funkcija:

$$F = e^2 = \frac{1}{Q}\sum_{i=1}^{Q}(t_i - a_i)^2 = E(t - a)^2.$$

Napomena: E je operator matematičkog očekivanja. Za veliki broj podataka $E(x)$ predstavlja srednju vrijednost vrijednost x.

Obučavanje jednoslojne mreže

Izlaz iz neurona se može zapisati na sljedeći način:

$$a = \mathbf{wp} + b = \underbrace{\begin{bmatrix} \mathbf{w} & b \end{bmatrix}}_{\mathbf{x}^T} \underbrace{\begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}}_{\mathbf{z}} = \mathbf{x}^T\mathbf{z}.$$

Funkcija performanse se sada može zapisati kao:

$$F = E\left(t - \mathbf{x}^T\mathbf{z}\right)^2 = E\left(t^2 - 2\mathbf{x}^T\mathbf{z}t + \mathbf{x}^T\mathbf{z}\mathbf{z}^T\mathbf{x}\right)$$

Kvadratna forma

$$= Et^2 - 2\mathbf{x}^T E\left(\mathbf{z}t\right) + \mathbf{x}^T E\left(\mathbf{z}\mathbf{z}^T\right)\mathbf{x} = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}.$$

Matrica $\mathbf{R}$ se naziva autokorelaciona matrica ulaznog signala, dok je $\mathbf{h}$ kroskorelacioni vektor između ulaznog i željenog signala:

$$\mathbf{R} = \frac{1}{Q}\sum_{i=1}^{Q}\mathbf{z}_i\mathbf{z}_i^T, \mathbf{h} = \frac{1}{Q}\sum_{i=1}^{Q}\mathbf{z}_i t_i.$$

Optimalna vrijednost vektora $\mathbf{x}$ se dobija minimizacijom funkcije performanse:

$$\nabla F(\mathbf{x}) = -2\mathbf{h} + 2\mathbf{R}\mathbf{x} \rightarrow \mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}.$$

Obučavanje jednoslojne mreže

Rješenje $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$ postoji ukoliko je matrica $\mathbf{R}$ nesingularna. Računanje optimalih vrijednosti koeficijenata na ovaj način je nepraktično, naročito u slučaju kada imamo veliki broj ulaznih podataka (inverzija matrice je računski zahtjevna operacija). Umjesto toga se može primijeniti metod najbržeg spusta (eng. steepest descend):

$$\mathbf{x}(n+1) = \mathbf{x}(n) - \mu \nabla F(\mathbf{x}(n))$$

$$= \mathbf{x}(n) - \mu \nabla \Big(c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x} \Big)_{\Big|_{\mathbf{x}_{(n)}}} = \mathbf{x}(n) - 2\mu(\mathbf{h} - \mathbf{R}\mathbf{x}(n))$$

Kod ovog metoda koeficijenti adaline mreže se iterativno ažuriraju u pravcu negativnog gradijenta. Kako je gradijent uvijek usmjeren u pravcu najbržeg rasta funkcije, na ovaj način će se koeficijenti iterativno približavati ka minimumu funkcije $F(\mathbf{x})$.

Za računanje gradijenta u prethodnom izrazu potrebno je poznavati autokorelacionu matricu $\mathbf{R}$ i kroskorelacioni vektor $\mathbf{h}$ koji se dobijaju usrednjavanjem svih dostupnih podataka. Ovaj način obučavanja se naziva *offline*, *batch* ili *serijsko učenje*.

Obučavanje jednoslojne mreže

Nekad ulazne podatke dobijamo u realnom vremenu, tj. nijesu unaprijed poznati svi podaci da bi mogli izračunali $\mathbf{R}$ i $\mathbf{h}$. Tada se adaptacija koeficijenata može vršiti na osnovu trenutne greške:

$$\mathbf{x}(n+1) = \mathbf{x}(n) - \mu \nabla e^2(\mathbf{x}(n)) = \mathbf{x}(n) + 2\mu e(n)\mathbf{z}(n).$$

Ovaj metod se zove stohastički gradijent ili LMS algoritam (eng. Least Mean Square). Primijetimo da je osnovna razlika u tome što se u posmatranom trenutku na ulaz mreže dovode podaci $\mathbf{p}(n)$, na osnovu kojih se računa izlaz mreže $a(n)$, greška $e(n)$ i gradijent. Na osnovu gradijenta se ažuriraju koeficijenti, a postupak se ponavlja u naredenoj iteraciji za nove ulazne podatke. Ovaj način obučavanja se zove *online* ili *iterativno učenje*. Jednačina za obučavanje koeficijenata se može razdvojiti na dvije zasebne jednačine:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\alpha e(n)\mathbf{p}(n),$$

$$b(n+1) = b(n) + 2\alpha e(n),$$

gdje $\mathbf{w}$ i b predstavljaju težinske koeficijente i bajas mreže.

Obučavanje jednoslojne mreže

Prethodni rezultati se mogu prilagoditi za slučaj kada mreža ima S izlaza/neurona. Tada se svaka vrsta matrice $\mathbf{W}$ i odgovarajući bajas ažuriraju na sljedeći način:

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + 2\alpha e_i(n)\mathbf{p}(n),$$
$$b_i(n+1) = b_i(n) + 2\alpha e_i(n),$$

gdje je $e_i(n)$ greška i-tog neurona. Jednačine za ažuriranje svih koeficijenata se mogu zapisati u matričnom obliku:

$$\mathbf{W}(n+1) = \mathbf{W}(n) + 2\alpha \mathbf{e}(n)\mathbf{p}^T(n),$$
$$\mathbf{b}(n+1) = \mathbf{b}(n) + 2\alpha \mathbf{e}(n).$$

Kod offline učenja n se koristi da označi redni broj epohe, dok kod online učenja n označava redni broj iteracije. Sem terminološke, osnovna razlika je što se u prvom slučaju gradijent računa na osnovu svih podataka, dok se u drugom slučaju on aproksimira na osnovu trenutnih podataka. Iz tog razloga LMS algoritam sporije konvergira u odnosu na metod najbržeg spusta.

Konvergencija LMS algoritma

Brzina konvergencije LMS algoritma zavisi od vrijednosti koraka μ. Što je veći korak, LMS će brže konvergirati ka minimumu. Međutim, korak ne može biti proizvoljno veliki, jer može doći do divergencije algoritma. Da bi izveli teorijsku granicu za maksimalni korak, pođimo od jednačine LMS algoritma:

$$\mathbf{x}(n+1) = \mathbf{x}(n) + 2\mu e(n)\mathbf{z}(n) = \mathbf{x}(n) + 2\mu\mathbf{z}(n)\left(t(n) - \mathbf{z}(n)^T\mathbf{x}(n)\right).$$

Ako prođemo operatorom očekivanja kroz prethodni izraz dobija se:

$$E\left\{\mathbf{x}(n+1)\right\} = E\left\{\mathbf{x}(n)\right\} + 2\mu E\left\{t(n)\mathbf{z}(n) - \mathbf{z}(n)\mathbf{z}(n)^T\mathbf{x}(n)\right\}.$$

$$= E\left\{\mathbf{x}(n)\right\} + 2\mu\mathbf{h} - 2\mu\mathbf{R}E\left\{\mathbf{x}(n)\right\} = \left[\mathbf{I} - 2\mu\mathbf{R}\right]E\left\{\mathbf{x}(n)\right\} + 2\mu\mathbf{h}$$

Prethodna vektorska diferencna jednačina će konvergirati jedino pod uslovom da je moduo svih sopstvenih vrijednosti matrice $\mathbf{I}$-$2\mu\mathbf{R}$ manji od 1. Ako je λ_i sopstvena vrijednost matrice $\mathbf{R}$, tada će 1-$2\mu\lambda_i$, biti sopstvena vrijednost matrice $\mathbf{I}$-$2\mu\mathbf{R}$. To znači da sopstvene vrijednosti treba da zadovolje uslov:

$$\left|1 - 2\lambda_i\right| < 1 \quad \rightarrow \quad 0 < \mu < \frac{1}{\lambda_{\max}}.$$ Najstrožiji uslov

Konvergencija LMS algoritma

Prethodni uslov garantuje konvergenciju srednje (očekivane) vrijednosti greške. Samim tim on važi kod metoda najbržeg spusta, dok se kod LMS-a koristi strožiji uslov (jer se gradijent ne računa na osnovu srednje, već na osnovu trenutne greške):

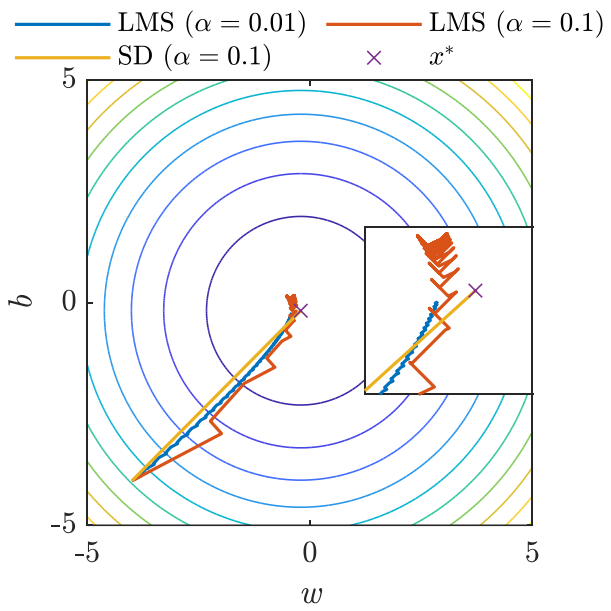
$$0 < \mu < \frac{1}{\lambda_{\min} + \lambda_{\max}}.$$

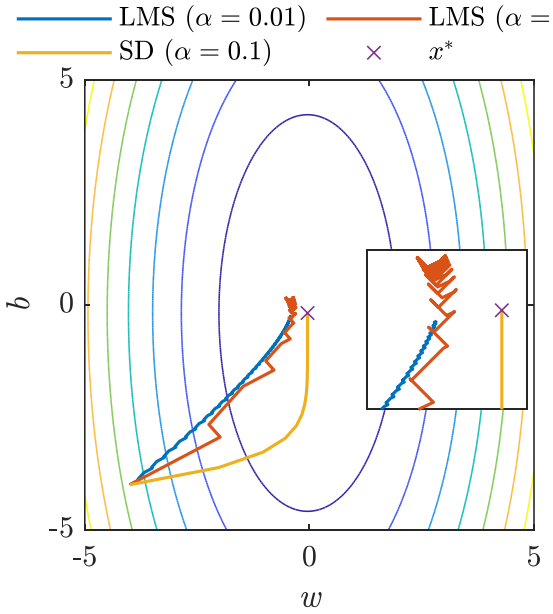
Treba napomenuti da se brzina konvergencije LMS algoritma povećava sa povećanjem koraka, dok se sa druge strane povećavaju oscilacije koeficijenata u stacionarnom stanju oko optimalnih vrijednosti.

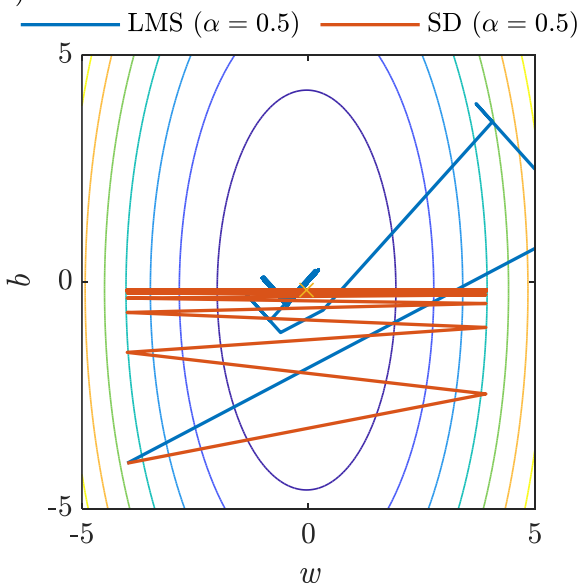
Takođe, treba zapamtiti da brzina konvergencije LMS algoritma zavisi od *raspona sopstvenih vrijednosti*, odnosno od odnosa maksimalne i minimalne sopstvene vrijednosti. Što je ovaj odnos bliži jedinici, konvergencija će biti brža (moguće je usvojiti veći korak).

Sopstvene vrijednosti će biti bliže jedna drugoj ukoliko su ulazni podaci raznovrsniji (time postižemo bržu konvergenciju)!

Konvergencija LMS algoritma







$$\mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \lambda_1 = \lambda_2 = 1$$

$$\mu_{\max}^{LMS} = \mu_{\max}^{SD} = 0.1.$$

$$\mathbf{R} = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, \lambda_1 = 1,\ \lambda_2 = 5$$

$$\mu_{\max}^{SD} = 0.2,\ \mu_{\max}^{LMS} = 0.16$$

- Jednake sopstvene vrijednosti
- Konturne krive su koncentrične kružnice
- Najbrža moguća konvergencija

- Različite sopstvene vrijednosti
- Konturne krive su koncentrične elipse (postaju izduženije sa povećanjem odnosa između sopstvenih vrijednosti)
- Sporija konvergencija

Primjer 1 – obučavanje Adaline mreže

Za adaline mrežu sa jednim ulazom su dostupni sljedeći podaci za treniranje:

$$\{p_i\} = \{0.9\},\ \{-0.9\},\ \{-0.9\},\ \{-0.9\},\ \{-0.9\},\ \{-1.1\},\ \{0.9\},\ \{-0.7\},\ \{0.5\},\ \{0.2\},$$
$$\{t_i\} = \{-0.5\},\ \{-1.0\},\ \{1.1\},\ \{-0.2\},\ \{0.6\},\ \{0.0\},\ \{-0.4\},\ -\{1.5\},\ \{-0.5\},\ \{0.6\}.$$

Definisati funkciju performanse, a zatim analitičkim putem odrediti optimalne koeficijente adaline mreže i minimum funkcije performanse. Na kraju, primjenom SD i LMS metoda pronaći optimalne koeficijente.

Adaline mreža ima jedan težinski koeficijent i bajas, koje možemo zapisati u vektorskom obliku:

$$\mathbf{x} = \begin{bmatrix} w & b \end{bmatrix}^T.$$

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \cdots & \mathbf{z}_{10} \end{bmatrix}$$

Autokorelaciona matrica je jednaka:

$$\mathbf{R} = \frac{1}{10}\sum_{i=1}^{10}\mathbf{z}_i\mathbf{z}_i^T = \frac{1}{10}\mathbf{Z}\mathbf{Z}^T = \begin{bmatrix} 0.685 & -0.29 \\ -0.29 & 1 \end{bmatrix},$$

pri čemu je:

$$\{\mathbf{z}_i\} = \left\{\begin{bmatrix} 0.9 \\ 1 \end{bmatrix}, \begin{bmatrix} -0.9 \\ 1 \end{bmatrix}, \begin{bmatrix} -0.9 \\ 1 \end{bmatrix}, \begin{bmatrix} -0.9 \\ 1 \end{bmatrix}, \begin{bmatrix} -0.9 \\ 1 \end{bmatrix}, \begin{bmatrix} -1.1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.9 \\ 1 \end{bmatrix}, \begin{bmatrix} -0.7 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}\right\}.$$

Primjer 1 – obučavanje Adaline mreže

Dalje, kroskorelacioni vektor $\mathbf{h}$ i skalar c su jednaki:

$$\mathbf{h} = \frac{1}{10}\sum_{i=1}^{10}\mathbf{z}_i t_i = \frac{1}{10}\mathbf{Z}\mathbf{T}^T = \begin{bmatrix} -0.034 \\ -0.18 \end{bmatrix},$$

$$\mathbf{T} = \begin{bmatrix} t_1 & t_2 & \cdots & t_{10} \end{bmatrix}$$

$$c = \frac{1}{10}\sum_{i=1}^{10} t_i = \frac{1}{10}\mathbf{T}\mathbf{T}^T = 0.588.$$

Funkcije performanse ima sljedeći oblik:

$$F = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}.$$

Konačno, optimalni koeficijenti, za koje gornja funkcija ima minimalnu vrijednost su jednaki:

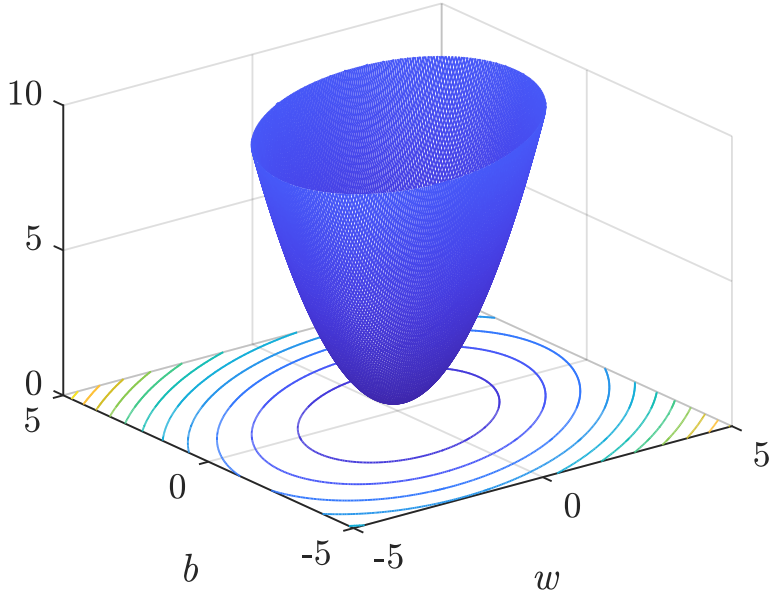
$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h} = \begin{bmatrix} -0.1435 \\ -0.2216 \end{bmatrix}.$$

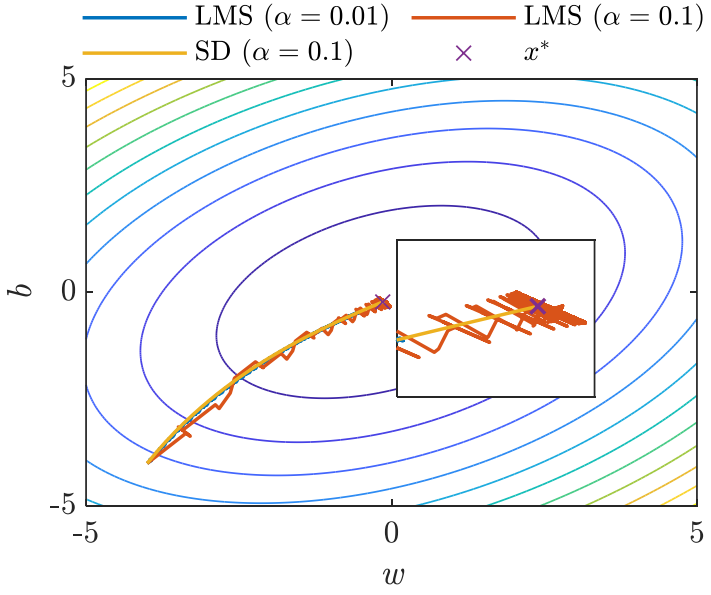
Minimum funkcije performanse iznosi:

$$F(\mathbf{x}^*) = c - 2\mathbf{x}^{*T}\mathbf{h} + \mathbf{x}^{*T}\mathbf{h} = c - \mathbf{x}^{*T}\mathbf{h} = 0.5432.$$

Primjer 1 – obučavanje Adaline mreže

Na slikama ispod su prikazani funkcija performanse i odgovarajući konturni grafik. U dva slučaja koeficijenti su trenirani pomoću LMS-a (stohastički gradijent), pri čemu su korišćena dva različita koraka. U trećem primjeru koeficijenti su trenirani pomoću metoda najbržeg spusta (egzaktni gradijent). Može se uočiti da LMS za veću vrijednost koraka algoritam brže konvergira, ali da koeficijenti prave veće oscilacije u stacionarnom stanju u odnosu na slučaj sa manjim korakom. U trećem slučaju se jasno vidi da koeficijenti konvergiraju „pravo“ ka optimalnim vrijednostima, iz razloga što se koristi egzaktna vrijednost gradijenta.



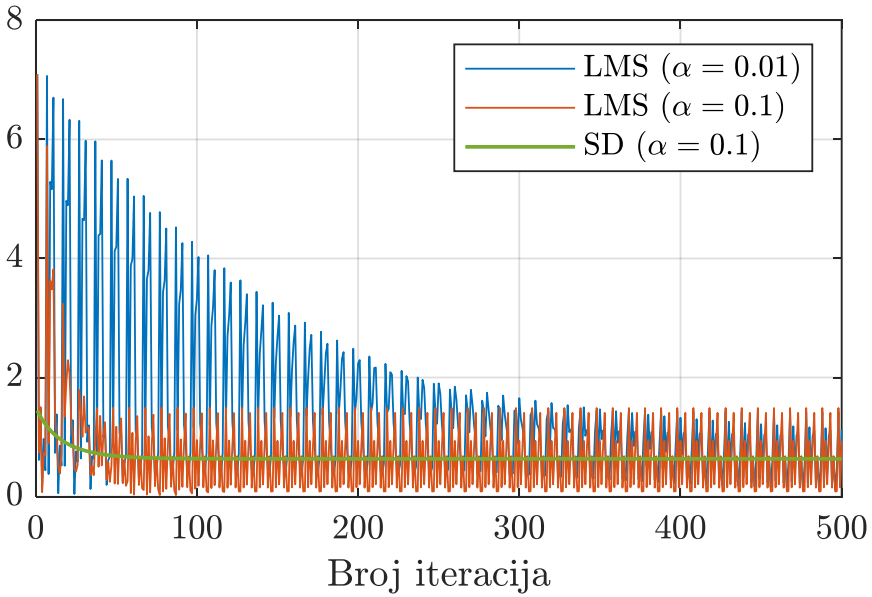


Primjer 1 – obučavanje Adaline mreže

Na slici ispod je prikazana apsolutna trenutna greška u zavisnosti od koeficijenata obučenih LMS algoritmom ($\widehat{F}(\mathbf{x}(k))$), kako i srednja greška u zavisnosti od koeficijenata obučenih metodom najbržeg spusta ($F(\mathbf{x}(k))$).

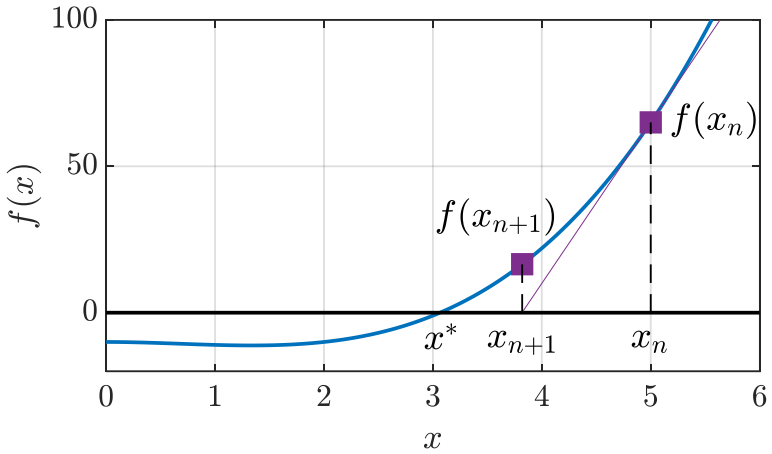
Može se uočiti da greška LMS brže konvergira za veću vrijednost koraka, ali da tada i pravi najveće oscilacije u stacionarnom stanju u odnosu na slučaj sa manjim korakom. Sa druge strane, metod najbržeg spusta konvergira ka optimalnim vrijednostima, pa se u stacionarnom stanju ne javljaju oscijacije.

Matlab kod za obučavanje koeficijenata i generisanje prikazanih slika se može preuzeti na: http://www.tsau.ac.me/ISAU/P2/pr1.m.



Njutnov metod

Posmatrajmo funkciju $f(x)$ čiju nulu želimo da odredimo iterativnim putem. Zamislimo da je x_n tačka u kojoj se trenutno nalazimo.



Jednačina tangente u tački x_n je:

$$y = f(x_n) + f'(x_n)(x - x_n).$$

Ako sa x_{n+1} označimo presjek tangente sa x-osom, tada taj presjek možemo odrediti na sljedeći način:

$$0 = f(x_n) + f'(x_n)(x_{n+1} - x_n) \;\rightarrow\; x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Sa slike se može uočiti da se $f(x_{n+1})$ nalazi bliže presjeku funkcije $f(x)$ sa x-osom u odnosu na $f(x_n)$. Ako istu proceduru nastavimo iterativno da ponavljamo, nakon određenog broja koraka doći ćemo do tačke x^* u kojoj funkcija $f(x)$ ima vrijednost 0. Opisana procedura pronalaženja nule funkcije se naziva Njutnov metod (ponekad Njutn-Rapsonov).

Njutnov metod

Kako je od interesa da pronađemo nulu izvoda funkcije (minimum), to možemo odraditi na sljedeći način:

$$x_{n+1} = x_n - \frac{f^{'}(x_n)}{f^{''}(x_n)}.$$

Prethodni postupak se može uopštiti na višedimenzione funkcije:

$$\mathbf{x}(n+1) \approx \mathbf{x}(n) - \mathbf{H}^{-1}(\mathbf{x}(n))\nabla F(\mathbf{x}(n)),$$

gdje su $\mathbf{H}(\mathbf{x}_n)$ i $\nabla F(\mathbf{x}_n)$ Hesijan i gradijent funkcije $F(\mathbf{x})$ u tački $\mathbf{x}(n)$:

$$\mathbf{H}(\mathbf{x}(n)) = \nabla^2 F(\mathbf{x}(n)) = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_1 \partial x_2} & \frac{\partial^2 F}{\partial x_2^2} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}_{|\mathbf{x}(n)}, \nabla F(\mathbf{x}(n)) = \begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \frac{\partial F}{\partial x_2} \\ \cdots \\ \frac{\partial F}{\partial x_n} \end{bmatrix}_{|\mathbf{x}(n)}$$

Njutnov metod

Konkretno za slučaj linearnog neurona, za koji je funkcija performanse definisana na sljedeći način:

$$F = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x},$$

koeficijenti neuralne mreže se ažuriraju pomoću jednačine:

$$\mathbf{x}(n+1) = \mathbf{x}(n) + \mathbf{R}^{-1}(\mathbf{h} - \mathbf{R}\mathbf{x}(n)).$$

Ako mreža ima S izlaza tada se koeficijenti koji odgovaraju različitim izlazima obučavaju nezavisno jedni od drugih:

$$\mathbf{x}_i(n+1) = \mathbf{x}_i(n) + \mathbf{R}^{-1}(\mathbf{h}_i - \mathbf{R}\mathbf{x}_i(n)).$$

Njutnov metod brže konvergira od SD metoda i LMS algoritma. Međutim, njegov glavni nedostatak je velika računska složenost, jer računanje Hesijana zahtijeva ogroman broj računskih operacija.

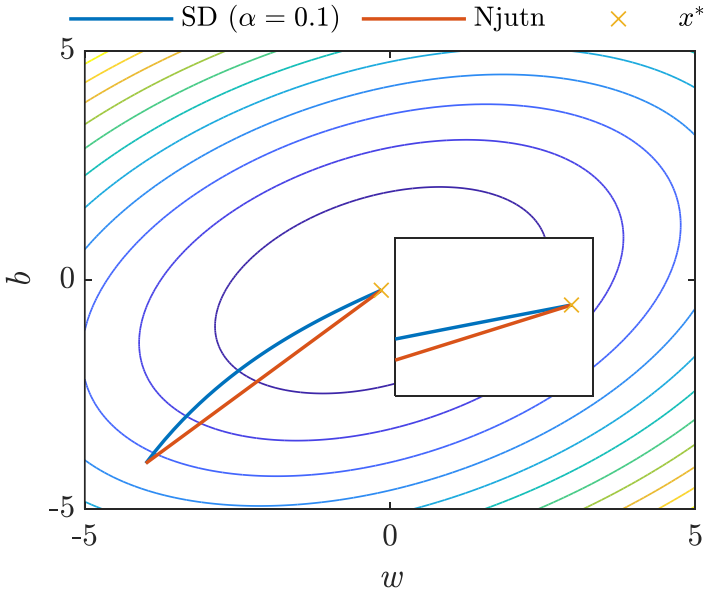
Njutnov algoritam se najčešće koristi za offline treniranje. Hesijan se određuje tako što se uzimaju u obzir svi poznati podaci. Postoje takozvane kvazi-Njutnove metode koje se mogu koristiti za online treniranje. Kod njih se Hesijan estimira iterativno, sa pristizanjem novih podataka.

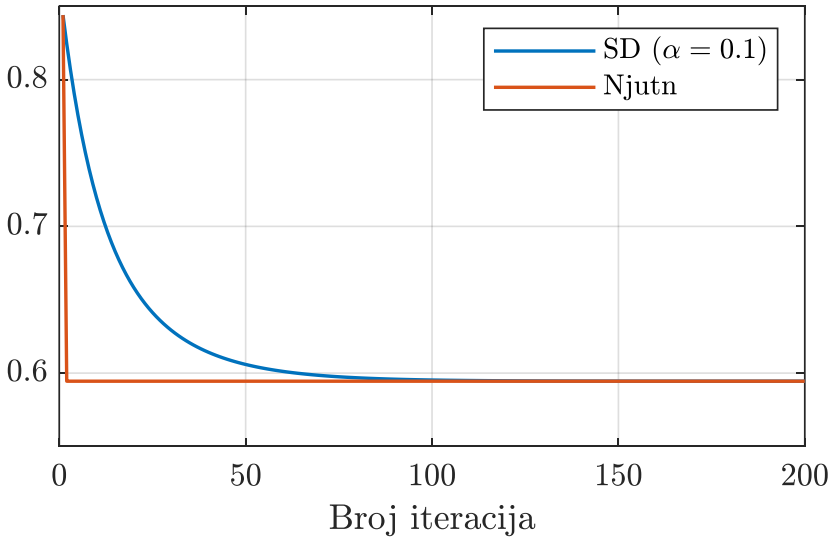
Primjer 2 – obučavanje Adaline mreže

U ovom primjeru su upoređeni SD i Njutnov metod. Korišćeni su podaci iz Primjera 1.

Sa desne slike se može uočiti da Njutnov metod konvergira u par iteracija, dok je SD metodu potrebno 60-ak iteracija. Sa konturnog grafika se vidi da se Njutnov metod približava optimalnoj tački najkraćom putanjom (bez obzira na „izduženost“ elipsi).

Matlab kod za obučavanje koeficijenata i generisanje prikazanih slika se može preuzeti na: http://www.tsau.ac.me/ISAU/P2/pr2.m.





Primjer 3 – Deep learning toolbox

U ovom primjeru će biti pokazano kako se DLT koristi za obučavanje nueralnih mreža. Korišćeni su podaci iz Primjera 1. Adaline mreža (linearni sloj) se definiše na sljedeći način:

```
>> net1=linearlayer
```

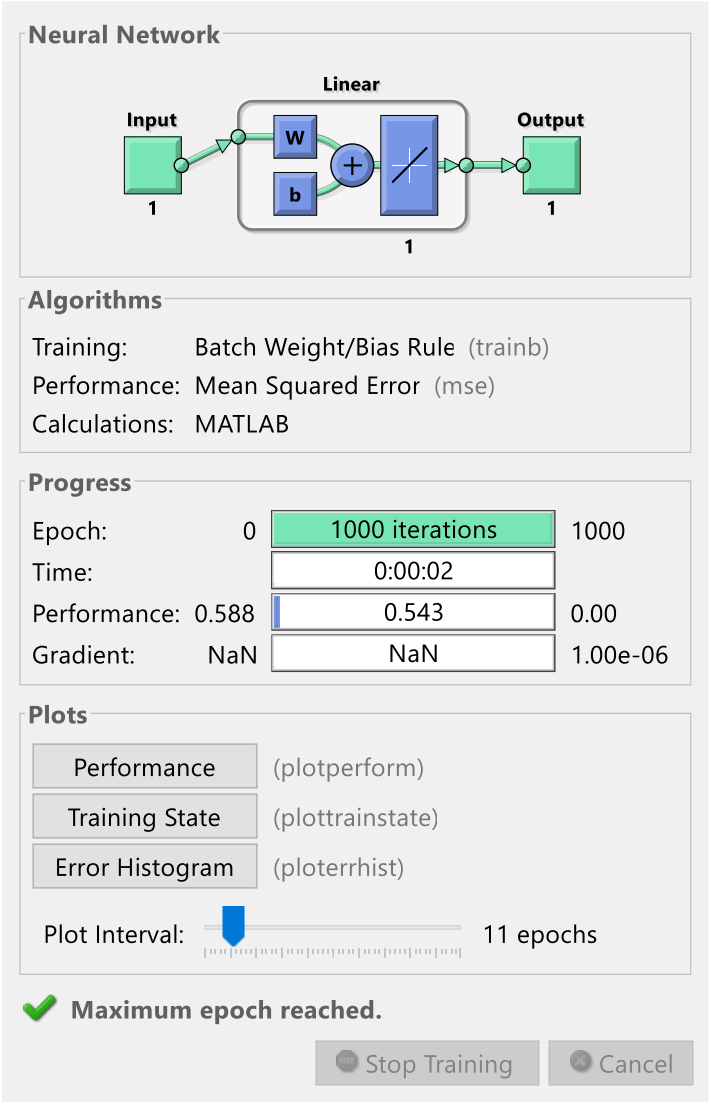
Neuralna mreža se obučava pomoću komande:

```
>> [net1 tr]=train(net1,p,t)
```

U promjenljivu `net1` se čuva istrenirana mreža, a u promjenljivu `tr` podaci koji su generisani tokom treniranja. Na primjer, vremenska zavisnost MSE od broja epoha se može prikazati komandom:

```
>> plot(tr.perf)
```

Funckija *train* obučava mrežu u offline modu. Dakle, u jednoj epohi se svi podaci propuštaju kroz mrežu, a zatim se računa gradijent na osnovu kojeg se ažuiraju koeficijenti. Isti postupak se ponavlja u narednim epohama.



Primjer 3 – Deep learning toolbox

Algoritam za obučavanje se podešava komandom:

```
>> net1.trainfcn
```

Deafult-ni algoritam za obučavanje linearnog sloja je metod najbržeg spusta. Korak algoritma se podešava na sljedeći način:

```
>> net1.inputWeights{1,1}.learnParam.lr = 0.1;
>> net1.biases{1,1}.learnParam.lr = 0.1;
```

Maksimalan broj epoha i dodatni parametri se podešavaju sa:

```
>> net2.trainParam
```

Neuralna mreža se u online modu trenira na sljedeći način:

```
>> [net1,y,e] = adapt(net1,pi,ti);
```

Parametri p_i i t_i predstavljaju trenutno dostupne podatke za treniranje. Promjenljiva y predstavlja izlaz iz mreže, dok je e trenutna vrijednost greške. Gornju komandu treba iterativno ponavljati za sve parove $(p_i,\ t_i)$ iz skupa podatka $(p_1,\ t_1)$.

Komanda `adapt` koristi LMS algoritam za treniranje koeficijenata.

Matlab kod za obučavanje koeficijenata i generisanje prikazanih slika se može preuzeti na: http://www.tsau.ac.me/ISAU/P2/pr3.m.