Neuronske mreže: LMS algoritam

Prof. dr. sc. Sven Lončarić

Fakultet elektrotehnike i računarstva
https://www.fer.unizg.hr/predmet/neumre_c

Pregled predavanja

- Uvod
- Wiener-Hopf jednadžbe
- Metoda najbržeg spusta
- LMS algoritam učenja
- Krivulja učenja
- Diskusija
- Zadaci

Uvod

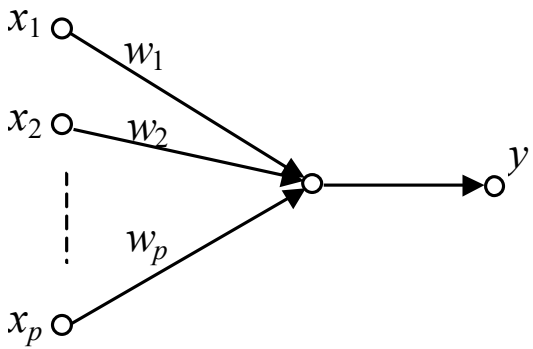
- U ovom poglavlju govorimo o klasi neuronskih mreža s jednim neuronom koje rade u linearnom režimu (nema nelinearnog bloka)
- Ovakve mreže su važne zbog toga što:
  - linearni adaptivni filtri su dobro proučeni i primjenjeni u području komunikacija, upravljanja, radarima i biomedicini
  - predstavljaju prvi korak ka višeslojnim neuronskim mrežama

Uvod

- LMS = least mean squares
- LMS algoritam je algoritam učenja
- LMS algoritam su razvili Widrow i Hoff, 1960
- LMS algoritam se koristi u adaptivnoj obradi signala:
  - za adaptivnu ekvalizaciju telefonskih kanala,
  - uklanjanje jeke na telefonskim linijama,
  - adaptivnu detekciju signala u šumu

Problem optimalnog filtriranja

- Neka imamo *p* senzora postavljenih u prostoru
- Neka su x_1, x_2, …, x_p signali koje daju senzori i koji se množe s težinama w_1, w_2, …, w_p
- Treba odrediti težine w_1, w_2, …, w_p tako da se minimizira razlika dobivenog odziva *y* i željenog odziva *d* u smislu srednje kvadratne pogreške



$$y = \sum_{k=1}^{p} w_k x_k$$

Problem optimalnog filtriranja

- Signal pogreške dan je kao:

$$e = d - y$$

- Pretpostavimo da je d slučajna varijabla
- Pretpostavimo da su ulazni uzorci x_k slučajne varijable, odnosno da je taj niz slučajnih varijabli jedan slučajni proces (slučajni niz)
- Slijedi da su i y i e također slučajne varijable

Problem optimalnog filtriranja

- Mjera kvalitete odziva je srednja kvadratna pogreška:

  $$J = 1/2\ E\ [e^2]$$

  gdje je E operator statističkog očekivanja
- Problem optimalnog linearnog filtriranja glasi: Odrediti optimalni skup težina w_1, w_2, …, w_p za koji je srednja kvadratna pogreška J minimalna
- Rješenje ovog problema u području digitalne obrade signala naziva se Wienerov filtar

Problem optimalnog filtriranja

- Izraz za srednju kvadratnu pogrešku može se raspisati kao:

$$J = \frac{1}{2} E\left[d^2\right] - E\left[\sum_{k=1}^{p} w_k x_k d\right] + \frac{1}{2} E\left[\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k x_j x_k\right]$$

- Nadalje izraz se može pojednostavniti kao:

$$J = \frac{1}{2} E\left[d^2\right] - \sum_{k=1}^{p} w_k E\left[x_k d\right] + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k E\left[x_j x_k\right]$$

gdje su težine w_i kao konstante izlučene ispred znaka sumacije i operatora očekivanja

Problem optimalnog filtriranja

- Možemo uvesti slijedeću notaciju:
- Srednja kvadratna vrijednost željenog odziva *d*:

$$r_d = E\left[d^2\right]$$

- Kros-korelacijska funkcija željenog odziva *d* i signala x_k:

$$r_{dx}(k) = E\left[dx_k\right], \quad k = 1,2,\ldots,p$$

- Autokorelacijska funkcija ulaznih signala:

$$r_x(j,k) = E\left[x_j x_k\right], \quad j,k = 1,2,\ldots,p$$

Problem optimalnog filtriranja

- Koristeći uvedenu notaciju možemo pisati izraz za srednju kvadratnu pogrešku kao:

$$J = \frac{1}{2} r_d - \sum_{k=1}^{p} w_k r_{dx}(k) + \frac{1}{2} \sum_{j=1}^{p} \sum_{k=1}^{p} w_j w_k r_x(j,k)$$

- Multidimenzionalni prikaz funkcije *J* u ovisnosti o težinama w_1, w_2, …, w_p kao slobodnim parametrima daje tzv. plohu pogreške
- Ploha pogreške je ploha drugog reda koja ima udubljeni oblik s jednim globalnim minimumom gdje se postiže minimalna pogreška

Minimizacija pogreške

- Da bi pronašli minimum funkcije pogreške *J* koja je funkcija *p* varijabli w_1, w_2, …, w_p treba odrediti parcijalne derivacije funkcije *J* po varijablama w_1, w_2, …, w_p i izjednačiti ih s nulom
- Parcijalne derivacije funkcije pogreške *J* dane su izrazima:

$$\frac{\partial J}{\partial w_k} = -r_{dx}(k) + \sum_{j=1}^{p} w_j r_x(j,k), \quad k = 1,2,\ldots,p$$

Wiener-Hopf jednadžbe

- Izjednačavajući parcijalne derivacije s nulom dobivamo Wiener-Hopf jednadžbe:

$$\sum_{j=1}^{p} w_j r_x(j,k) = r_{dx}(k), \quad k = 1,2,\ldots,p$$

- Težine w_1, w_2, …, w_p koje zadovoljavaju Wiener-Hopf jednadžbe definiraju filtar koji se zove Wienerov filtar
- Za rješenje Wiener-Hopf jednadžbi potrebno je izračunati inverziju matrice dimenzija $p \times p$
- Da se izbjegne računanje inverzne matrice mogu se koristiti iterativne gradijentne metode

Metoda najbržeg spusta

- engl. steepest descent method
- Korištenjem najbržeg spusta težine filtra su vremenski promjenjive i njihove vrijednosti određuju se iterativno uzduž plohe pogreške s ciljem kretanja prema globalnom minimumu
- Korekcija težine u iteraciji *n* jednaka je produktu konstante η i gradijenta:

$$\Delta w_k(n) = -\eta \frac{\partial J}{\partial w_k}, \quad k = 1,2,\ldots,p$$

gdje je η pozitivna konstanta koja određuje brzinu učenja

Metoda najbržeg spusta

- Uz danu staru vrijednost težine u iteraciji *n* nova vrijednost računa se kao:

$$w_k(n+1) = w_k(n) + \Delta w_k(n), \quad k = 1,2,\ldots,p$$

- Ako se za gradijent uvrsti ranije dobiveni izraz dobivamo konačni izraz:

$$w_k(n+1) = w_k(n) + \eta\left[r_{dx}(k) - \sum_{j=1}^{p} w_j(n) r_x(j,k) \right], \quad k = 1,2,\ldots,p$$

LMS algoritam

- engl. least-mean-square algorithm
- Problem kod Wienerovog filtra je da moramo znati kros-korelacijsku funkciju $r_{dx}(k)$ i autokorelacijsku funkciju $r_x(j,k)$
- LMS algoritam je specijalni slučaj ranije izvedenog algoritma s najbržim spustom
- LMS algoritam je temeljen na *trenutnim procjenama* autokorelacijske i kros-korelacijske funkcije:

$$\hat{r}(j,k;n) = x_j(n)x_k(n)$$
$$\hat{r}_{dx}(k;n) = x_k(n)d(n)$$

LMS algoritam

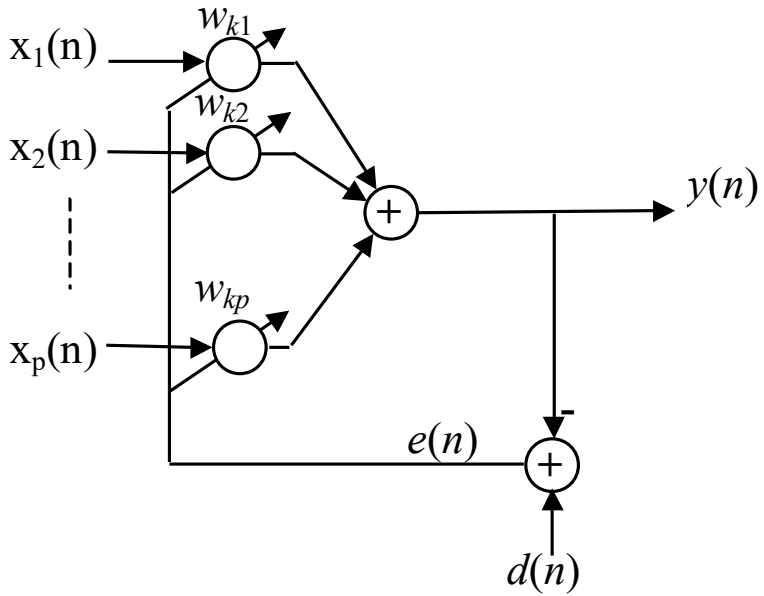
- Ako uvrstimo trenutne procjene u ranije izvedeni izraz dobivamo izraz za LMS algoritam učenja (delta pravilo učenja, Widrow-Hoff pravilo učenja) :

$$
\begin{aligned}
w_k(n+1) &= w_k(n) + \eta\left[x_k(n)d(n) - \sum_{j=1}^{p} w_j(n)x_j(n)x_k(n) \right] \\
&= w_k(n) + \eta\left[d(n) - \sum_{j=1}^{p} w_j(n)x_j(n) \right] x_k(n) \\
&= w_k(n) + \eta\left[d(n) - y(n) \right] x_k(n), \quad k = 1,2,\ldots,p
\end{aligned}
$$

$$
\Delta w_k(n) = \eta e(n) x_k(n)
$$

LMS algoritam

- Slika prikazuje adaptivni filter
- Težine se dinamički izračunavaju u realnom vremenu pomoću LMS algoritma



LMS algoritam - sažetak

1. Inicijalizacija. Postaviti početne vrijednosti

$$w_k(1) = 0, \quad k = 1,2,\ldots, p$$

2. Filtriranje. Za trenutke *n*=1, 2, ... izračunaj

$$y(n) = \sum_{1=1}^{p} w_j(n) x_j(n)$$
$$e(n) = d(n) - y(n)$$
$$w_k(n+1) = w_k(n) + \eta e(n) x_k(n), \quad k = 1,2,\ldots, p$$

Krivulja učenja

- Krivulja učenja je prikaz ovisnosti pogreške $J(n)$ o indeksu iteracije n
- Krivulja učenja pokazuje karakteristike procesa učenja
- Za neki LMS algoritam karakteristična vrijednost je konačna vrijednost pogreške
- Konačna vrijednost pogreške je uvijek veća od stacionarne pogreške za pripadni Wienerov filtar: $J(\infty)$
- Razlika ovih dvaju grešaka zove se preostala pogreška: $J_{\min}$

$$J_{\text{ex}} = J(\infty) - J_{\min}$$

Krivulja učenja

- Kvocjent preostale pogreške i Wienerove pogreške zove se razdešenost:

$$M = \frac{J_{\text{ex}}}{J_{\text{min}}}$$

- Razdešenost se izražava u postocima (npr. razdešenost od 10% smatra se prihvatljivom u praksi)
- Važna karakteristika LMS algoritma je srednje vrijeme konvergencije
  - Srednje vrijeme konvergencije može se definirati npr. kao vremenska konstanta eksponencijalne funkcije pomoću koje možemo aproksimirati krivulju $J(n)$

Krivulja učenja

- Razdešenost je proporcionalna parametru učenja η
- Srednje vrijeme konvergencije je obrnuto proporcionalno parametru brzine učenja η
- To su kontradiktorni zahtjevi jer ako smanjimo brzinu učenja η da bi smanjili razdešenost onda srednje vrijeme konvergencije postaje veće
- Prilikom projektiranja potrebno je pažljivo odabrati brzinu učenja η da bi se zadovoljili svi zahtjevi

Promjena brzine učenja

- Najjednostavniji način učenja je kada se konstanta učenja η ne mijenja s indeksom iteracije *n*:

  $\eta(n)=\eta_{0,}$ za svaki *n*

- Konvergencija LMS algoritma može se popraviti ako se koristi promjenjiva konstanta učenja npr:

  $\eta(n)=c/n,$ gdje je *c* konstanta

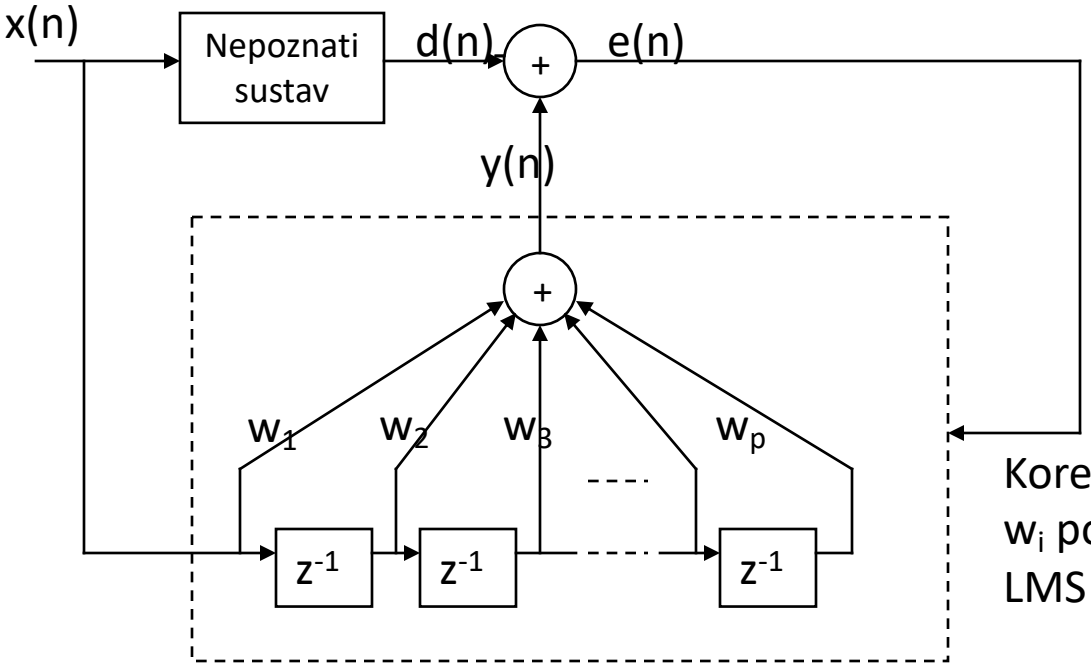
- Zbog problema velikih vrijednosti gornjeg izraza za male *n* koristi se i modificirani izraz:

$$\eta(n)=\frac{\eta_0}{1+(n/\tau)}$$

Primjene LMS algoritma

- LMS algoritam ima puno primjena uključujući
  - Identifikacija sustava
  - Aktivno upravljanje i uklanjanje šuma (engl. active noise control)

Primjer: Identifikacija sustava

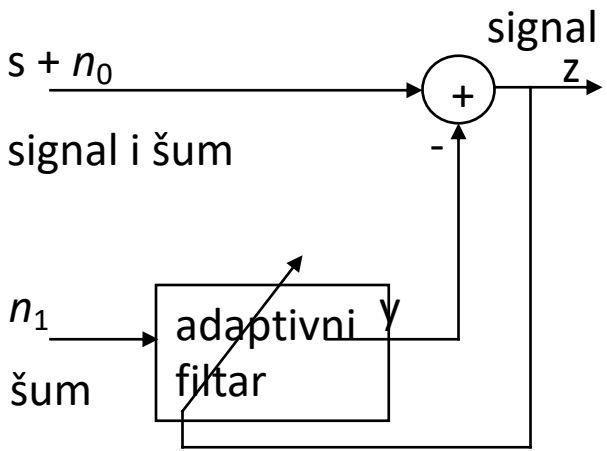


Model sustava (adaptivni FIR filtar)

Primjer:
Adaptivno uklanjanje buke

- Engl. adaptive noise cancellation
- Primjeri:
- Uklanjanje buke iz pilotske kabine
  - Buka mlaznog motora do 140 dB, govor pilota 30 dB
  - Govorna komunikacija nije moguća bez poništavanja buke
  - Buka motora nije konstantna nego ovisi o režimu leta
  - Poništavanje buke mora biti adaptivno, mora ovisiti o buci
- Slušalice koje poništavaju šum

Adaptivno uklanjanje buke



- *s* je signal, a n_0 je šum koji nije koreliran sa signalom *s*
- n_1 je referentni šum koji je koreliran sa šumom n_0
- *y* je procjena šuma n_0
- Npr. n_1 je šum motora snimljen izvan kabine, a s + n_0 je korisni signal u kabini zagađen šumom

Adaptivno uklanjanje buke

- Signal y nije koreliran sa signalom s
- Šum n_0 nije koreliran sa signalom s
- Dakle vrijedi:

$$z = s + n_0 - y$$

$$E[z^2] = E[s^2] + E[(n_0 - y)^2]$$

- Kad adaptivni filter mijenjamo tako da minimiziramo E[z^2], onda minimiziramo i E[$(n_0 - y)^2$]

Adaptivno uklanjanje buke

- Dakle signal *z* možemo koristiti kao signal pogreške u adaptivnom filtru koji uči LMS algoritmom
- Za adaptaciju filtra koristimo LMS pravilo učenja:

$$\Delta w_k(i) = \eta z(i) n_1(i)$$

Diskusija

- LMS algoritam je široko prihvaćen algoritam u adaptivnoj obradi signala zbog svojih svojstava:
  - Jednostavnost implementacije
  - Radi dobro u nepoznatoj okolini
  - Može pratiti vremenske promjene statistika ulaznog signala

Zadaci

- Problem 5.1
  - Neka koristimo metodu najbržeg spusta i jednu težinu w(n). Treba

  a) odrediti srednju kvadratnu pogrešku $J(n)$ kao funkciju od $w(n)$

  b) Odrediti optimalno rješenje $w_o(n)$ i minimalnu pogrešku J_{min}