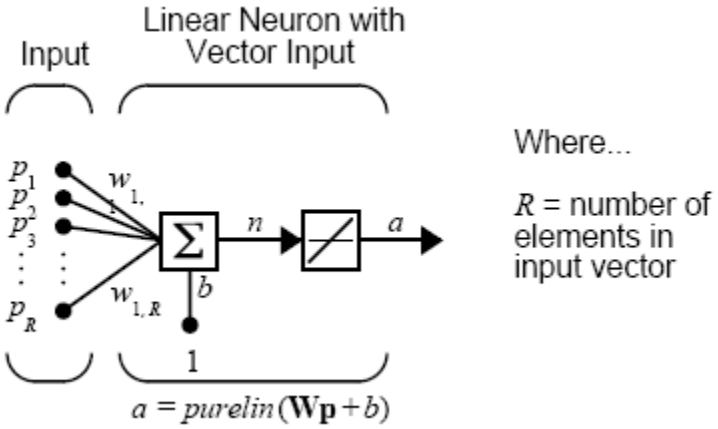
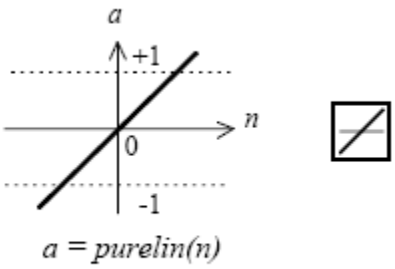
Laborator 9

Modelul neuronului

Un neuron liniar având ca intrare un vector cu R componente este reprezentat în figura de mai jos:



Această reţea are aceeaşi structură de bază ca reţeaua de tip perceptron. Singura diferenţă constă în utilizarea funcţiei de transfer liniară pe care o vom numi *purelin*.



Linear Transfer Function

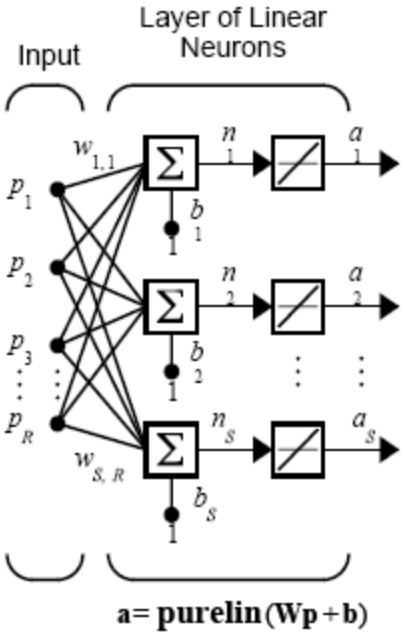
Reţeaua calculează ieşirile în modul următor:

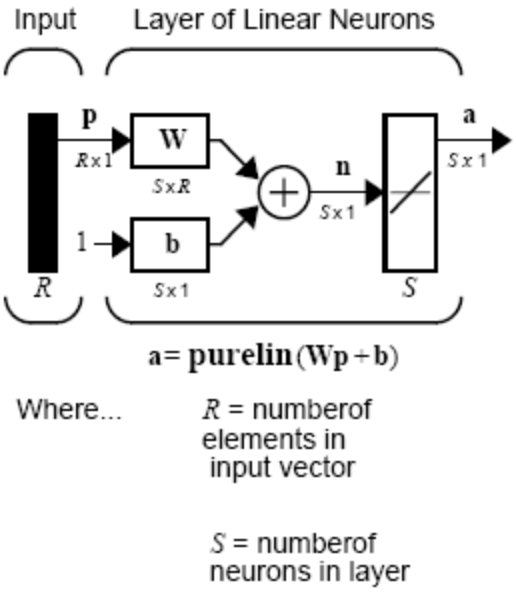
$a = purelin(n) = purelin(\mathbf{W}\mathbf{p} + b) = \mathbf{W}\mathbf{p} + b$

Acest neuron poate fi antrenat pentru a învăţa funcţii afine aplicate intrărilor sau pentru a găsi o aproximare liniară a unei funcţii neliniare.

Arhitectura reţelei

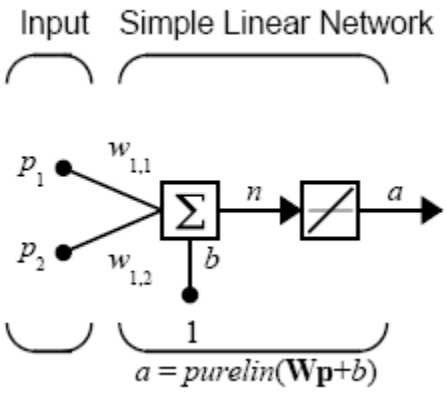
Reţeaua liniară prezentată mai jos are un singur nivel cu S neuroni conectaţi la cele R componente ale vectorului de intrare prin matricea ponderilor **W**.





Crearea unei reţele liniare (newlin)

Considerăm un neuron liniar având două intrări. Diagrama asociată acestei reţele este prezentată mai jos:



Ieşirea reţelei este dată de formula:

$$a = purelin(n) = purelin(\mathrm{W}p + b) = \mathrm{W}p + b$$

$$a = w_{1,1}p_1 + w_{1,2}p_2 + b$$

sau

$$a = w_{1,1}p_1 + w_{1,2}p_2 + b\,.$$

O reţea liniară este creată folosind funcţia ***newlin***:

net = newlin(*P, S, ID, LR*)

unde:

P – este o matrice R × Q conţinând Q vectori de intrare de dimensiune R × 1;

S – reprezintă numărul de ieşiri;

ID – vector conţinând delay-urile iniţiale, implicit [0];

LR – rata de învăţare, implicit 0.01.

sau

net = newlin(*P, T, ID, LR*)

are aceleaşi argumente cu excepţia lui

T – matrice de dimensiune S × Q, având Q vectori de ieşire de dimensiune S × 1.

Exemplu: O reţea ca cea prezentată mai sus poate fi definită folosind comanda:

net = newlin([2 3; 3 4], [-1 1])*;*

Y = sim(net, [2; 3]);

După definire o reţea liniară poate fi simulată şi antrenată exact ca în cazul reţelelor de tip perceptron.

Media pătratelor erorilor

Ca şi în cazul regulii de învăţare a perceptronului, metoda celor mai mici pătrate (least mean square error (LMS)) este un exemplu de antrenare supervizată, în care regula de învăţare (aproximarea relaţiei din cadrul datelor de antrenare) se bazează pe o mulţime de exemple . Fie următoarea mulţime de antrenare:

$$\{(p_1,t_1),(p_2,t_2),\ldots,(p_q,t_q)\}$$

în care p_i este una dintre intrările reţelei, iar t_i reprezintă ieşirea target corespunzătoare intrării p_i. În momentul în care este aplicată o intrare reţelei, reţeaua compară ieşirea rezultată cu ieşirea target. Eroarea este determinată ca fiind diferenţa dintre ieşirea target şi ieşirea reţelei. Astfel, scopul nostru este de a minimiza media pătratelor acestor erori.

$$mse = \frac{1}{q}\sum_{k=1}^{q} e(k)^2 = \frac{1}{q}\sum_{k=1}^{q} (t(k)-a(k))^2$$

Algoritmul LMS ajustează ponderile şi bias-ul reţelei liniare astfel încât să se minimizeze media pătratelor erorilor. Indexul de performanţă *mse* al reţelelor este o funcţie pătratică. Astfel, indexul de performanţă poate fi minimizat fie până la un minim global, fie până la un minim mai slab sau poate să nu aibă un minim, acest fapt depinzând de caracteristicile vectorilor de intrare.

Proiectarea sistemelor liniare (newlind)

Spre deosebire de cele mai multe dintre arhitecturile neurale, reţelele liniare pot fi proiectate în mod direct dacă sunt cunoscute perechile de forma **(input, target)**. Valorile specifice ale ponderilor şi bias-ului pot fi obţinute pentru a minimiza media pătratelor erorilor folosind funcţia **newlind**.

Funcţia **newlind** are sintaxa:

net = newlind(*P, T*)*;*

unde:

P - este o matrice de dimensiune *R* × *Q*; reţine cei *Q* vectori de intrare.

T - este o matrice de dimensiune *S* × *Q*; reţine cele Q valori target ale vectorilor de intrare

Exemplu: Presupunem că intrările, respectiv valorile target corespunzătoare sunt:

P = [1 2 3];

T= [2.0 4.1 5.9];

Pentru a defini reţeaua folosim următoarea comandă:

net = newlind(P,T);

Putem simula reţeaua pentru a verifica faptul că definirea reţelei este corectă în sensul îndeplinirii funcţiei dorite.

Y = sim(net,P)

Y =

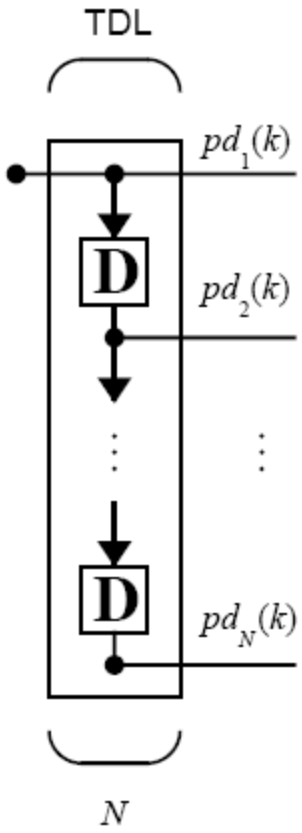
2.0500 4.0000 5.9500

Se observă că valorile ieşirilor reţelei sunt foarte apropiate de ieşirile target, deci reţeaua estimează corect valorile target în sensul minimizării mediei pătratelor erorilor.

Reţele liniare cu *delays*

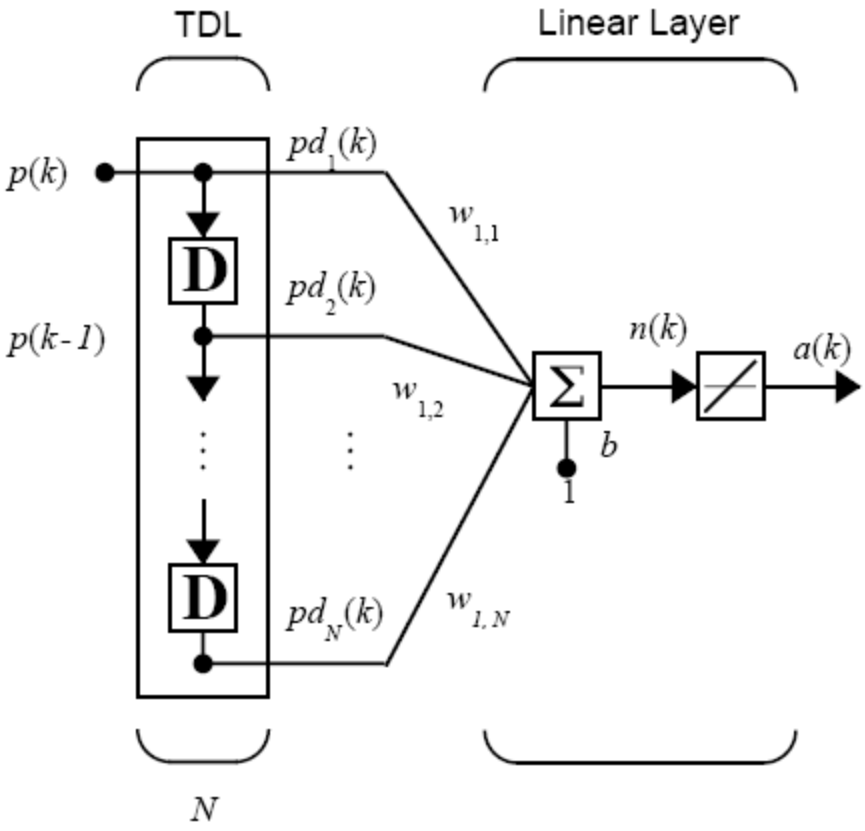
Şirul de interceptări ale semnalelor folosind delay-uri (tapped delay line (TDL))

Semnalul de input intră din partea stângă şi trece prin *N* – 1 delay-uri. Ieşirea unui TDL este un vector *N*-dimensional format din semnalul de input la momentul curent, semnalul de input precedent, ş.a.m.d.



Filtre liniare

Definim un *filtru liniar* ca fiind o reţea liniară cu un şir de delay-uri, având modelul prezentat mai jos:



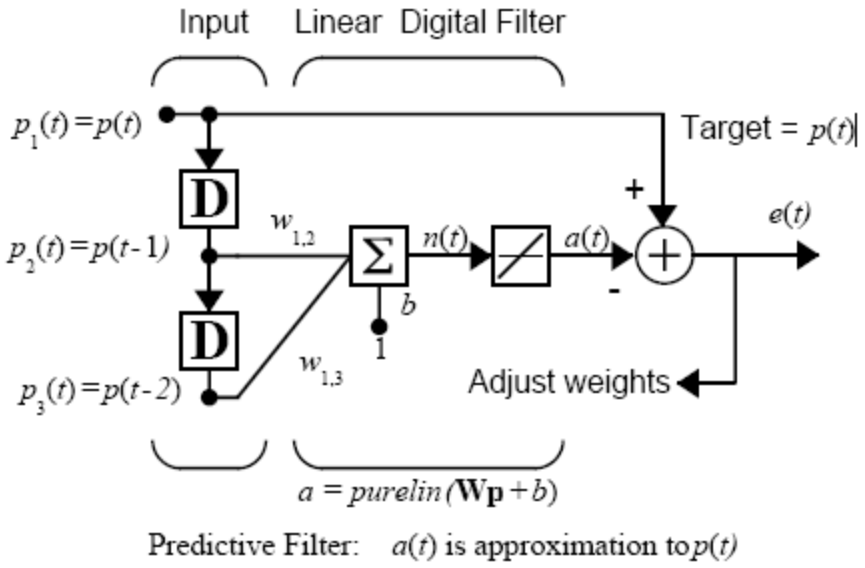
Ieşirea reţelei este dată de formula:

$$a(k) = \sum_{i=1}^{N} w_{1,i}\, pd_i(k) + b$$

Modul în care funcţionează o astfel de reţea este următorul: se consideră cunoscută o mulţime de stări iniţiale aflate în unităţile de întârziere, se introduce în reţea o serie de semnale şi pentru fiecare semnal introdus la un moment *k* se calculează şi se returnează o ieşire, după care se introduce un nou semnal în reţea, iar semnalul anterior intră în următorul delay de sus în jos.

Exemplu de problemă de predicţie:

Presupunem că vrem să folosim un filtru adaptiv pentru a prezice valoarea următoare a unui proces aleator staţionar, $p(t)$. În acest scop vom utiliza reţeaua reprezentată mai jos:



Predictive Filter: $a(t)$ is approximation to $p(t)$

Semnalul pe care vrem să-l prezicem, $p(t)$, intră de la stânga la dreapta în linia de delay-uri. Cele două valori anterioare ale lui $p(t)$ vor reprezenta ieşirile unităţilor din linia de delay-uri. Reţeaua va fi antrenată cu funcţia *adapt* pentru a modifica ponderile după fiecare pas astfel încât să se minimizeze eroarea $e(t)$. Dacă această eroare poate fi minimizată la zero, atunci ieşirea $a(t)$ este exact $p(t)$, deci reţeaua a emis o ipoteză corectă.

Presupunem că vrem să definim un filtru liniar care să returneze secvenţa de ieşite *T*, fiind dată secvenţa de intrare *P* şi două stări iniţiale asociate delay-urilor P_i.

P = {1 2 1 3 3 2};
Pi = {1 3};
T = {5 6 4 20 7 8};

Se poate folosi funcţia *newlind* pentru a crea o reţea cu delay-uri care să returneze ieşiri potrivite asociate unor intrări date. Numărul de întârzieri asociate delay-urilor vor fi reţinute de cel de al treilea argument.

Reţeaua având parametrii de mai sus poate fi definită folosind următoarea comandă:

net = newlind(*P, T, Pi*)*;*

Pentru a obţine ieşirile dorite ale reţelei, vom simula reţeaua apelând funcţia *sim* astfel:

Y = sim(*net, P, Pi*)

Şi vom obţine rezultatele:

Y =

[2.73] [10.54] [5.01] [14.95] [10.78] [5.98]

Se poate observa că ieşirile reţelei nu sunt egale cu ieşirile target, dar sunt destul de apropiate; iar media pătratelor erorilor este minimizată (funcţia de performanţă pe care trebuia să o îndeplinească reţeaua).

Estimarea unei funcţii liniare folosind o reţea liniară

Se consideră mulţimea de antrenare $S = \left\{(x_i, y_i) \mid i = \overline{1,m}\right\}$, unde $x_i \in \mathbb{R}^n$ şi $y_i \in \mathbb{R}^p$. Se caută o aplicaţie liniară $h: \mathbb{R}^n \to \mathbb{R}^p$, $h(x) = Wx$, care să minimizeze diferenţele dintre $h(x_i)$ şi y_i (estimează funcţia de atribuire a etichetelor).

Antrenarea reţelei se realizează pe baza algoritmilor lui Widrow Hoff care minimizează funcţia de eroare (media pătratelor erorilor) pe mulţimea de antrenare. Cele două variante ale algoritmului, varianta incrementală (online) şi cea globală (batch) sunt implementate prin funcţiile **adapt** şi **train**.

În reţelele statice (cele fără delay) funcţia **adapt** poate implementa atât antrenarea incrementală, cât şi cea batch în funcţie de formatul datelor de intrare. Astfel dacă datele de intrare sunt prezentate concurenţial, sub forma unei matrice de vectori concurenţi (de exemplu [-2 2; 1 2]) se va realiza antrenarea batch; dacă datele de intrare sunt introduse secvenţial sub forma unei matrice de vectori secvenţiali (de exemplu {[1;1] [2; 3]}) se va realiza antrenarea incrementală. Funcţia **train** realizează o antrenare de tip incremental şi returnează rezultatele obţinute după parcurgerea întregii mulţimi de antrenare, indiferent de modalitatea prin care sunt transmise semnalele.

Funcţia **adapt** poate fi apelată folosind sintaxa:

[net, Y, E] = adapt(net, P, T, Pi)

unde argumentele funcţiei sunt

net este reţeaua iniţială pe care vrem să o antrenăm

P este vectorul intrărilor

T reprezintă vectorul target

Pi sunt valorile iniţiale ale liniei de delay-uri (dacă reţeaua conţine delay-uri) şi returnează

net - reţeaua în urma antrenării

Y – ieşirile corespunzătoare intrărilor după antrenare

E – vectorul erorilor după antrenare

Înainte de utilizarea funcţiei *adapt* trebuie setat parametrul care indică numărul paşilor în care se va parcurge mulţimea de antrenare (*net.adaptParam.passes*).

Exemplu:

P = {[-2; 2] [-2; 3] [-1; 1] [-1; 4] [0; 0] [0; 1] [0; 2] [0; 3] [1; 0] [1; 1] [2; 1] [2; 2] [3; -1] [3; 0] [3; 1] [3; 2] [4; -2] [4; 1] [5; -1] [5; 0]};

T = {-1 -1 -1 -1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 1 1 1 1 1}

net.adaptParam.passes = 100;

[net, Y, E] = adapt(net, P, T);

net

Y

E
mse(E)

Funcţia **train** se apelează astfel:
[net, tr, Y, E] = train(net, P, T, Pi)

unde argumentele sunt aceleaşi ca cele ale funcţiei *adapt* şi returnează
net – reţeaua după modificarea ponderilor
tr – informaţii legate de procesul de antrenare (epoca, performanţa)
Y – ieşirile corespunzătoare intrărilor după antrenare
E – vectorul erorilor după antrenare

Parametrii pe care trebuie să îi setăm în general sunt:
net.trainParam.epochs – numărul de epoci
net.trainParam.goal – eroarea medie pătratică pe care ne propunem să o atingem

Exemplu:
net.trainParam.epochs = 100;
net.train.goal = 0.001;
[net, tr] = train(net, P, T)

În timpul antrenării este generată o figură care conţine reprezentarea grafică a funcţiei de performanţă în funcţie de numărul de epoci parcurse. Pentru a vizualiza această reprezentare şi după ce antrenarea s-a terminat se poate folosi funcţia *plotperf*(*tr, net.trainParam.goal*), căreia i se transmit ca argumente variabila *tr* în care sunt reţinute informaţiile legate de procesul de învăţare şi marginea maximă faţă de care valoarea funcţiei de performanţă trebuie să fie mai mică (*net.trainParam.goal*).

S-a constatat că eroarea poate fi minimizată doar în cazul în care rata de învăţare este suficient de mică. Pentru a determina rata de învăţare maximă acceptată folosim funcţia *maxlinlr*, care returnează valoarea maximă a ratei de învăţare pentru datele de intrare considerate:
lr = maxlinlr(intrări);

Aplicaţii

Aplicaţie 1:
Considerăm mulţimea de antrenare formată din perechi de forma: prima componentă este dată de elemente ale vectorului x = -1:0.1:1, iar cea de a doua de valorile corespunzătoare determinate de funcţia de mai jos..
Fie funcţia f : [-1, 1]$\rightarrow$[-2, 2], f(x) = 2x care atribuie etichetele corespunzătoare intrărilor. Presupunem că etichetele returnate de această funcţie sunt afectate de un zgomot uniform distribuit pe [-0.25, 0.25]. Se cere să se definească o reţea liniară care să aproximeze valorile etichetelor. Să se reprezinte grafic datele din mulţimea

de antrenare şi datele obţinute în urma antrenării reţelei în aceeaşi fereastră. Să se studieze influenţa asupra rezultatului obţinut în urma antrenării a ratei de învăţare şi a numărului de iteraţii.

Aplicaţie 2:
Se consideră o mulţime de antrenare de dimensiune 10 având perechi de forma: (un vector de dimensiune 5, media aritmetică a valorilor din vector). Vom considera elementele vectorului din intervalul (-1, 1). Se va defini o reţea cu 5 unităţi de intrare şi o unitate de ieşire care va fi antrenată pentru a calcula media aritmetică a datelor de intrare.

Aplicaţie 3:
Se defineşte o serie de semnale T care durează 5 secunde şi este definit folosind 40 de valori de selecţie pe secundă.

time = 0:0.025:5;

Presupunem că semnalele sunt determinate conform valorilor returnate de funcţia *sin* astfel:

*T = sin(time*4*pi);*

Să se definească o reţea liniară care să prezică un semnal pe baza ultimelor 5 semnale. Să se compare rezultatele obţinute cu valorile target reprezentând grafic semnalele la fiecare moment de timp (atât cele pe care vrem să le prezicem, cât şi cele returnate de reţea) şi să se ploteze eroarea de prezicere a fiecărui semnal.

Aplicaţie 4:
Se cere să se prezică un semnal T care durează 6 secunde rată eşantion de 20 de exemple pe secundă. Se presupune că după 4 secunde frecvenţa semanlelor se dublează.

Indicaţie: Se pot reţine momentele de timp astfel:

time1 = 0:0.05:4;
time2 = 4.05:0.025:6;
time = [time1 time2];

Considerăm secvenţa de semnale pe care dorim să le prezicem astfel:

*T = [sin(time1*4*pi) sin(time2*8*pi)];*

Să se prezică semnalul T folosind un algoritm de antrenare incrementală a unei reţele (funcţia adapt) şi să se studieze corectitudinea antrenării. Se doreşte utilizarea unei reţele cu delay-uri care vor conţine ca semnale iniţiale valorile 1,2,3, 4,5.