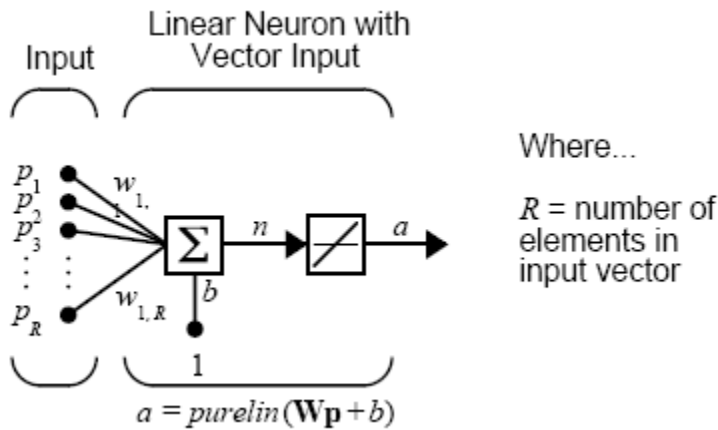


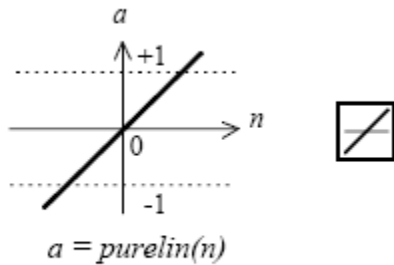
Laborator 9

Modelul neuronului

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



Această rețea are aceeași structură de bază ca rețeaua de tip perceptron. Singura diferență constă în utilizarea funcției de transfer liniară pe care o vom numi *purelin*.



Linear Transfer Function

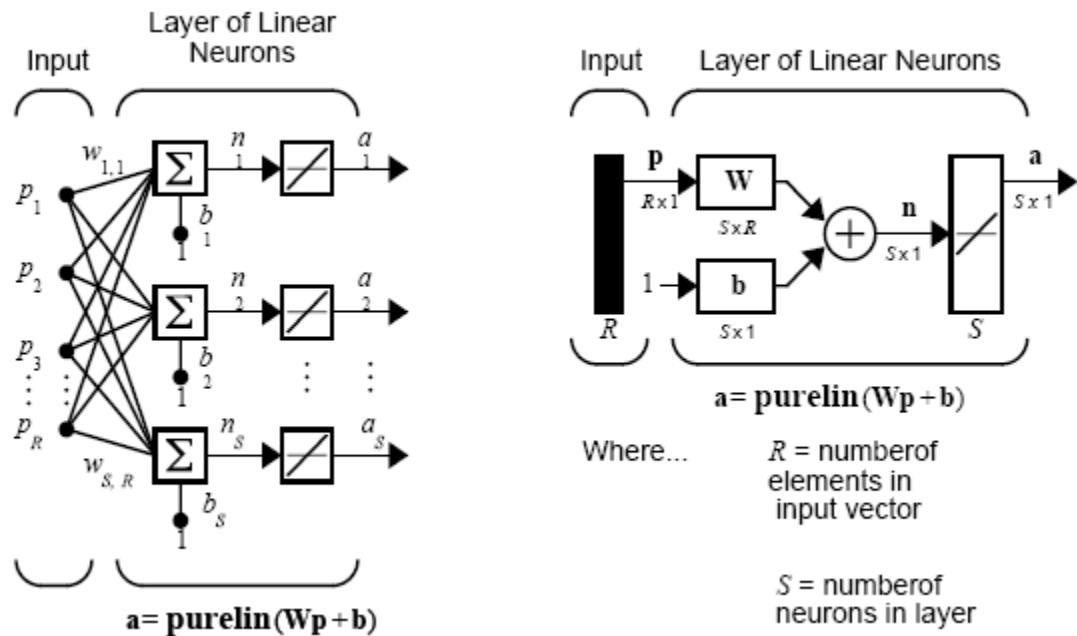
Rețeaua calculează ieșirile în modul următor:

$$a = \text{purelin}(n) = \text{purelin}(\mathbf{Wp} + b) = \mathbf{Wp} + b$$

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

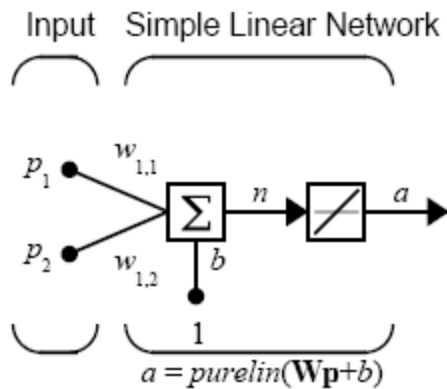
Arhitectura rețelei

Rețeaua liniară prezentată mai jos are un singur nivel cu S neuroni conectați la cele R componente ale vectorului de intrare prin matricea ponderilor \mathbf{W} .



Crearea unei rețele liniare (newlin)

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



Ieșirea rețelei este dată de formula:

$$a = \text{purelin}(n) = \text{purelin}(\mathbf{Wp} + b) = \mathbf{Wp} + b$$

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

sau

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

O rețea liniară este creată folosind funcția **newlin**:

$$\text{net} = \text{newlin}(P, S, ID, LR)$$

unde:

P – este o matrice $R \times Q$ conținând Q vectori de intrare de dimensiune $R \times 1$;

S – reprezintă numărul de ieșiri;

ID – vector conținând delay-urile inițiale, implicit $[0]$;

LR – rata de învățare, implicit 0.01 .

sau

$$\text{net} = \text{newlin}(P, T, ID, LR)$$

are aceleași argumente cu excepția lui

T – matrice de dimensiune $S \times Q$, având Q vectori de ieșire de dimensiune $S \times 1$.

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

$$\text{net} = \text{newlin}([2 \ 3; 3 \ 4], [-1 \ 1]);$$

$$Y = \text{sim}(\text{net}, [2; 3]);$$

După definire o rețea liniară poate fi simulată și antrenată exact ca în cazul rețelelor de tip perceptron.

Media pătratelor erorilor

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

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

în care p_i este una dintre intrările rețelei, iar t_i reprezintă ieșirea target corespunzătoare intrării p_i . În momentul în care este aplicată o intrare rețelei, rețeaua compară ieșirea rezultată cu ieșirea target. Eroarea este determinată ca fiind diferența dintre ieșirea target și ieșirea rețelei. 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 și bias-ul rețelei liniare astfel încât să se minimizeze media pătratelor erorilor. Indexul de performanță mse al rețelelor este o funcție pătratică. Astfel, indexul de performanță 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, rețelele liniare pot fi proiectate în mod direct dacă sunt cunoscute perechile de forma (**input, target**). Valorile specifice ale ponderilor și bias-ului pot fi obținute pentru a minimiza media pătratelor erorilor folosind funcția **newlind**.

Funcția **newlind** are sintaxa:

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

unde:

P - este o matrice de dimensiune $R \times Q$; reține cei Q vectori de intrare.

T - este o matrice de dimensiune $S \times Q$; reține 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 rețeaua folosim următoarea comandă:

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

Putem simula rețeaua pentru a verifica faptul că definirea rețelei este corectă în sensul îndeplinirii funcției dorite.

$$Y = sim(net,P)$$

$$Y =$$

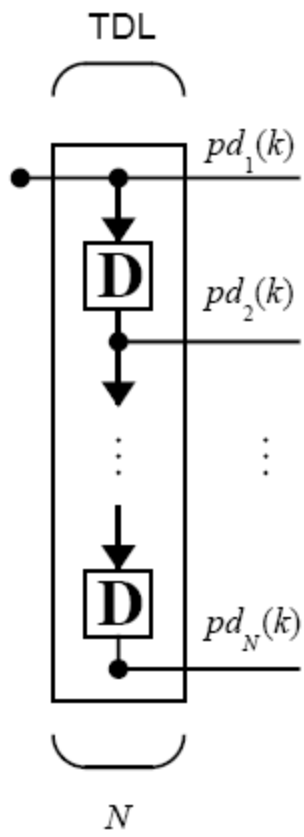
$$2.0500 \quad 4.0000 \quad 5.9500$$

Se observă că valorile ieșirilor rețelei sunt foarte apropiate de ieșirile target, deci rețeaua estimează corect valorile target în sensul minimizării mediei pătratelor erorilor.

Rețele liniare cu *delays*

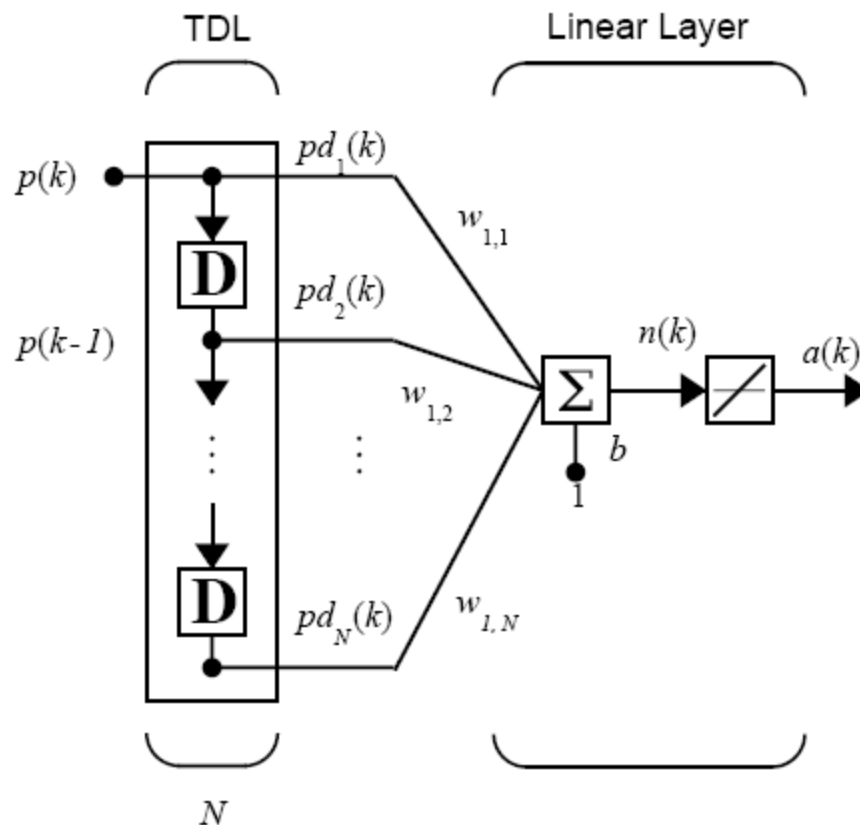
Șirul de interceptări ale semnalelor folosind delay-uri (tapped delay line (TDL))

Semnalul de input intră din partea stângă și trece prin $N - 1$ delay-uri. Ieșirea unui TDL este un vector N -dimensional format din semnalul de input la momentul curent, semnalul de input precedent, ș.a.m.d.



Filtre liniare

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



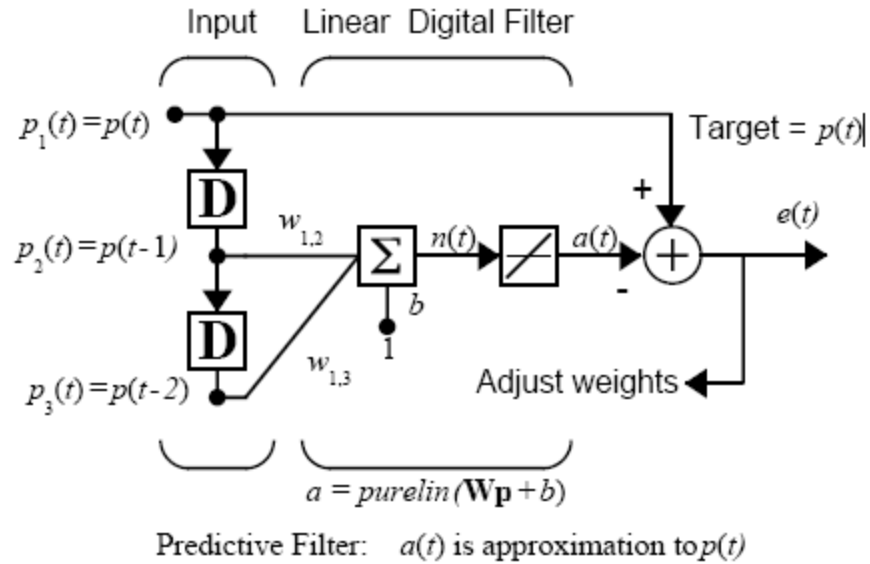
Ieșirea rețelei este dată de formula:

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

Modul în care funcționează o astfel de rețea este următorul: se consideră cunoscută o mulțime de stări inițiale aflate în unitățile de întârziere, se introduce în rețea o serie de semnale și pentru fiecare semnal introdus la un moment k se calculează și se returnează o ieșire, după care se introduce un nou semnal în rețea, iar semnalul anterior intră în următorul delay de sus în jos.

Exemplu de problemă de predicție:

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



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 ieșirile unităților din linia de delay-uri. Rețeaua va fi antrenată cu funcția *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 ieșirea $a(t)$ este exact $p(t)$, deci rețeaua a emis o ipoteză corectă.

Presupunem că vrem să definim un filtru liniar care să returneze secvența de ieșite T , fiind dată secvența de intrare P și două stări inițiale asociate delay-urilor P_i .

$$P = \{1\ 2\ 1\ 3\ 3\ 2\};$$

$$P_i = \{1\ 3\};$$

$$T = \{5\ 6\ 4\ 20\ 7\ 8\};$$

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

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

$$\text{net} = \text{newlind}(P, T, P_i);$$

Pentru a obține ieșirile dorite ale rețelei, vom simula rețeaua apelând funcția *sim* astfel:

$$Y = \text{sim}(\text{net}, P, P_i)$$

Și vom obține rezultatele:

$Y =$

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

Se poate observa că ieșirile rețelei nu sunt egale cu ieșirile target, dar sunt destul de apropiate; iar media pătratelor erorilor este minimizată (funcția de performanță pe care trebuia să o îndeplinească rețeaua).

Estimarea unei funcții liniare folosind o rețea liniară

Se consideră mulțimea de antrenare $S = \{(x_i, y_i) | i = \overline{1, m}\}$, unde $x_i \in \mathbb{R}^n$ și $y_i \in \mathbb{R}^p$. Se caută o aplicație liniară $h: \mathbb{R}^n \rightarrow \mathbb{R}^p$, $h(x) = Wx$, care să minimizeze diferențele dintre $h(x_i)$ și y_i (estimează funcția de atribuire a etichetelor).

Antrenarea rețelei se realizează pe baza algoritmilor lui Widrow Hoff care minimizează funcția de eroare (media pătratelor erorilor) pe mulțimea de antrenare. Cele două variante ale algoritmului, varianta incrementală (online) și cea globală (batch) sunt implementate prin funcțiile **adapt** și **train**.

În rețelele statice (cele fără delay) funcția **adapt** poate implementa atât antrenarea incrementală, cât și cea batch în funcție de formatul datelor de intrare. Astfel dacă datele de intrare sunt prezentate concurențial, sub forma unei matrice de vectori concurenți (de exemplu [-2 2; 1 2]) se va realiza antrenarea batch; dacă datele de intrare sunt introduse secvențial sub forma unei matrice de vectori secvențiali (de exemplu {[1;1] [2; 3]}) se va realiza antrenarea incrementală. Funcția **train** realizează o antrenare de tip incremental și returnează rezultatele obținute după parcurgerea întregii mulțimi de antrenare, indiferent de modalitatea prin care sunt transmise semnalele.

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

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

unde argumentele funcției sunt

net este rețeaua inițială pe care vrem să o antrenăm

P este vectorul intrărilor

T reprezintă vectorul target

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 conține reprezentarea grafică a funcției de performanță în funcție de numărul de epoci parcurse. Pentru a vizualiza această reprezentare și după ce antrenarea s-a terminat se poate folosi funcția *plotperf(tr, net.trainParam.goal)*, căreia i se transmit ca argumente variabila *tr* în care sunt reținute informațiile legate de procesul de învățare și marginea maximă față de care valoarea funcției de performanță trebuie să fie mai mică (*net.trainParam.goal*).

S-a constatat că eroarea poate fi minimizată doar în cazul în care rata de învățare este suficient de mică. Pentru a determina rata de învățare maximă acceptată folosim funcția *maxlinlr*, care returnează valoarea maximă a ratei de învățare pentru datele de intrare considerate:

lr = maxlinlr(intrări);

Aplicații

Aplicație 1:

Considerăm mulțimea 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 funcția de mai jos..

Fie funcția $f: [-1, 1] \rightarrow [-1, 1]$, $f(x) = 2x$ care atribuie etichetele corespunzătoare intrărilor. Presupunem că etichetele returnate de această funcție sunt afectate de un zgomot uniform distribuit pe $[-0.25, 0.25]$. Se cere să se definească o rețea liniară care să aproximeze valorile etichetelor. Să se reprezinte grafic datele din mulțimea de antrenare și datele obținute în urma antrenării rețelei în aceeași fereastră. Să se studieze influența asupra rezultatului obținut în urma antrenării a ratei de învățare și a numărului de iterații.

Aplicație 2:

Se consideră o mulțime 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 rețea cu 5 unități de intrare și o unitate de ieșire care va fi antrenată pentru a calcula media aritmetică a datelor de intrare.

Aplicație 3:

Se definește o serie de semnale T care durează 5 secunde și este definit folosind 40 de valori de selecție pe secundă.

$$time = 0:0.025:5;$$

Presupunem că semnalele sunt determinate conform valorilor returnate de funcția \sin astfel:

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

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

Aplicație 4:

Se cere să se prezică un semnal T care durează 6 secunde rată eșantion de 20 de exemple pe secundă. Se presupune că după 4 secunde frecvența semnalelor se dublează.

Indicație: Se pot reține momentele de timp astfel:

$$time1 = 0:0.05:4;$$

$$time2 = 4.05:0.025:6;$$

$$time = [time1 \ time2];$$

Considerăm secvența 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 rețele (funcția `adapt`) și să se studieze corectitudinea antrenării. Se dorește utilizarea unei rețele cu delay-uri care vor conține ca semnale inițiale valorile 1,2,3, 4,5.