Multi-Layer Perceptron in MATLAB NN Toolbox

[Part 1]

**Yousof Koohmaskan, Behzad Bahrami,
Seyyed Mahdi Akrami, Mahyar AbdeEtedal**

Department of Electrical Engineering
Amirkabir University of Technology (Tehran Polytechnic)

Advisor: Dr. F. Abdollahi

Introduction

- Rosenblatt in 1961 created many variations of the perceptron
- One of the simplest was a single-layer network whose weights and biases could be trained to produce a correct target vector when presented with the corresponding input vector

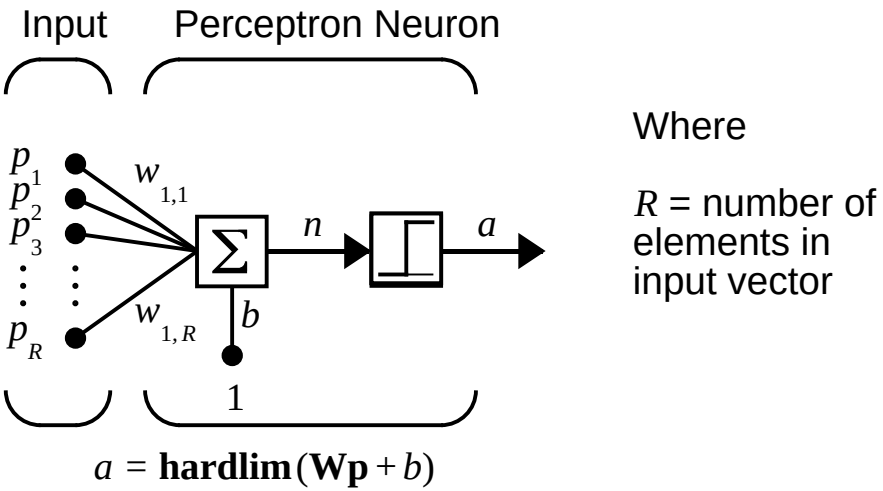


Figure: Perceptron Representation-[1]

Introduction

- There are many transfer function that can be used in the perceptron structure, *e.g.*

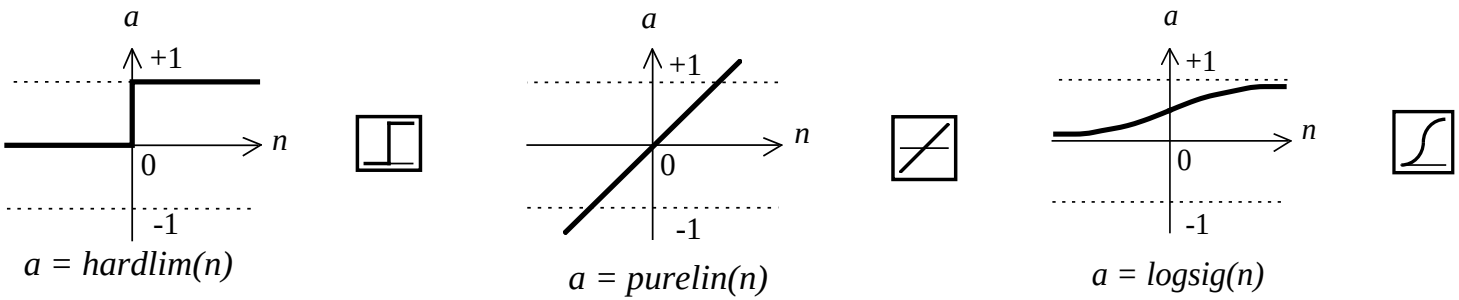


Figure: `hardlim, purelin, logsig`

- Usually `hardlim` is selected as default

Perceptron Architecture

- The perceptron network consists of a single layer
- *R* input vector
- *S* output scaler

Here, we consider just one-layer perceptron

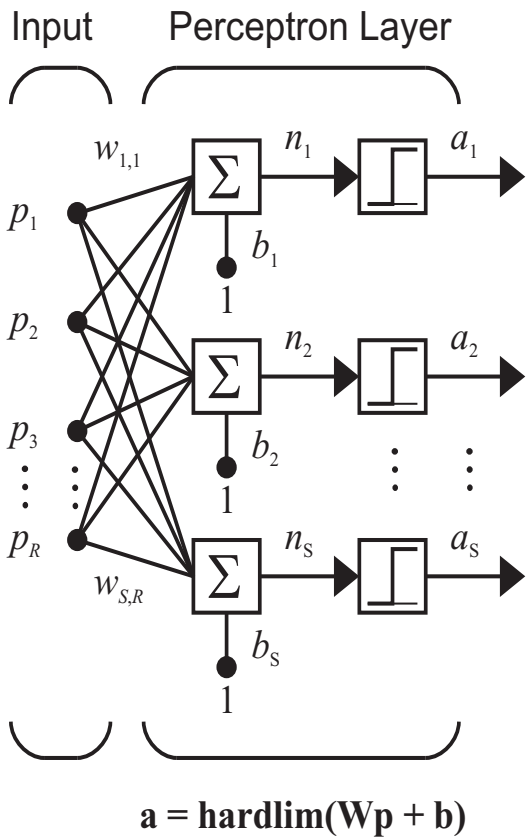


Figure: Perceptron Architecture

Mathematical Notation

- A single superscript is used to identify elements of a layer, *e.g.* $\mathbf{n}^3$ expresses input to layer 3
- Input weight matrix from layer k to layer l can be shown as $IW^{l,k}$, also for layer weight matrix, *i.e.*$LW^{l,k}$

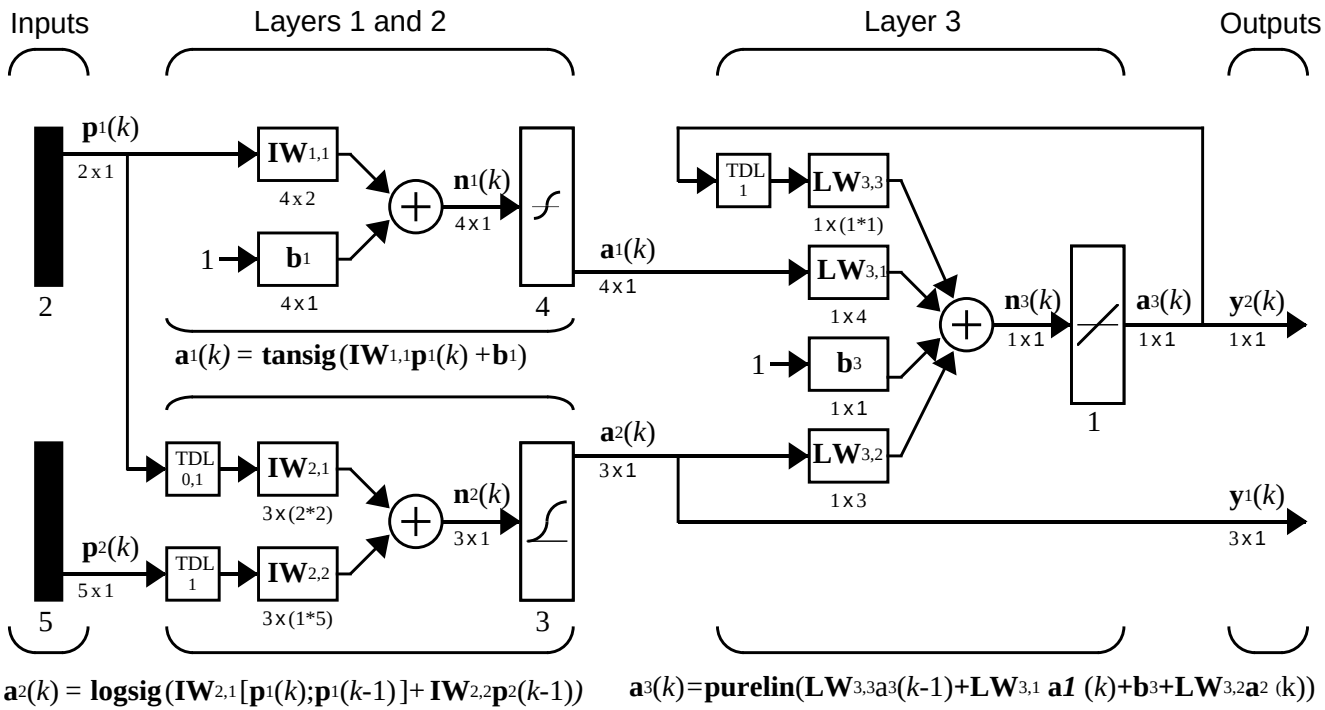


Figure: Example of a perceptron

Matlab Notation Considerations

- superscripts $\Leftrightarrow$ cell array indices, *e.g.* $p^1 \rightarrow p\{1\}$
- subscripts $\Leftrightarrow$ indices within parentheses, *e.g.* $p_2 \rightarrow p(2)$
- $\Rightarrow$ superscripts $+$ subscripts ,*e.g.* $p_2^1 \rightarrow p\{1\}(2)$
- indices within parentheses $\Leftrightarrow$ a second cell array index[1] *e.g.* $p^1(k-1) \rightarrow p\{1, k-1\}$

[1] also in the Matlab help, it should be corrected; $p\{1, k-1] \rightarrow p\{1, k-1\}$

Learning Rules

- It breaks into two parts, supervised and unsupervised learning, we consider the first topic
- The objective is to reduce the error **e**, which is the difference **t-a** between the neuron response a and the target vector **t**
- There are many algorithms that do this and we will introduce them
- All the algorithms compute Δw utilizing some methods

Learning Alghorithms

- Hebb weight learning rule[2]

$$\Delta w^k = c a^k p^k$$

MATLAB command `learnh`

- Perceptron weight and bias learning function

$$\Delta w^k = e^k (p^k)^T$$

MATLAB command `learnp`

- Normalized perceptron weight and bias learning function

$$pn = \frac{p}{\sqrt{1 + \|p\|^2}}, \quad \Delta w^k = e^k (pn^k)^T$$

MATLAB command `learnpn`

[2] in all, **e**: error, **p**: input and **a**: output

Learning Alghorithms

- Widrow-Hoff weight/bias learning function

$$\Delta w^k = ce^k (pn^k)^T$$

learning rate c,
MATLAB command `learnwh`

Training

- Training is using the learning rule to update the weights of network repeatedly

$$w_{new}^{k} = w_{old}^{k} + \Delta w^{k}$$

MATLAB command `train`

Utility Functions

- MATLAB command `init`, Initialize neural network
- MATLAB command `sim`, Simulate neural networks
- MATLAB command `adapt`
  Allow neural network to change weights and biases on inputs
- MATLAB command `newp`, create perceptron

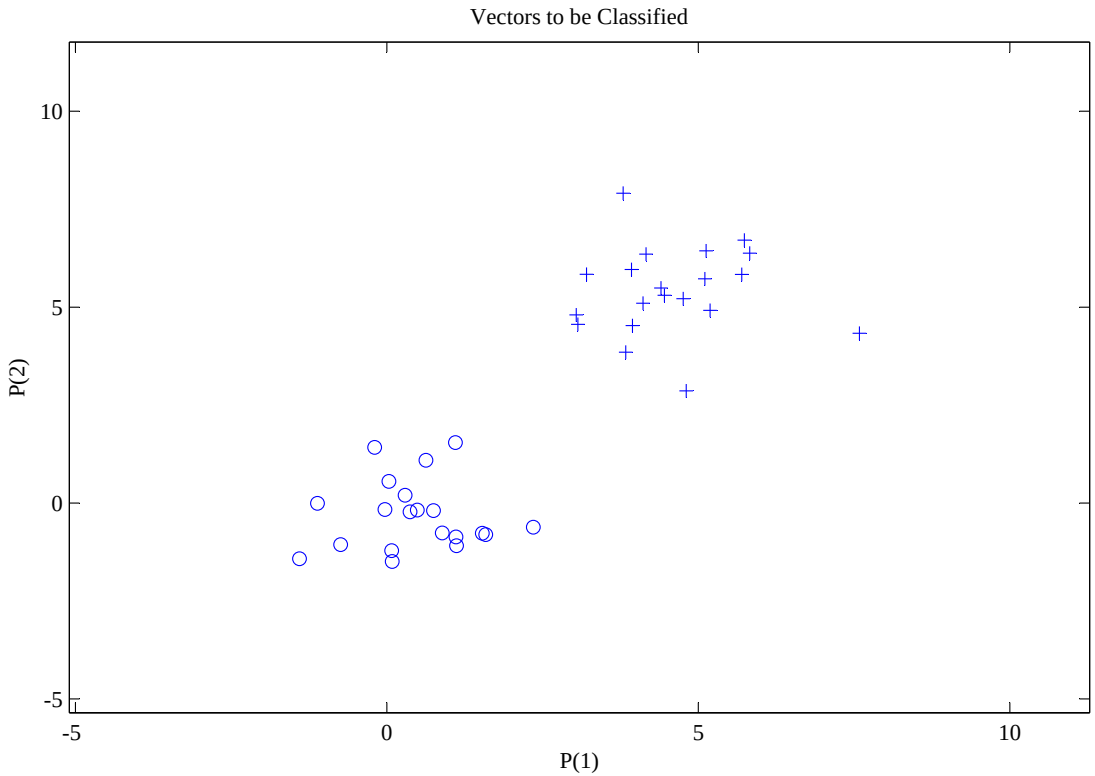
Perceptron problem, step by step

- Creating a Perceptron, `>> net=newp(P,T);`
- Assigning initial weight and bias,
  `>> net.IW{1,1}=` $[w1 \quad w2]$`;`
  `>> net.b{1} = [b];`
- Determining number of pass,
  `>> net.trainParam.epochs = pass;`
- Training Network, `>> net = train(net,p,t);`
- Test and Verifying, `>> a = sim(net,p);`

Perceptron problem, step by step

Example 1:

Classifying displayed data



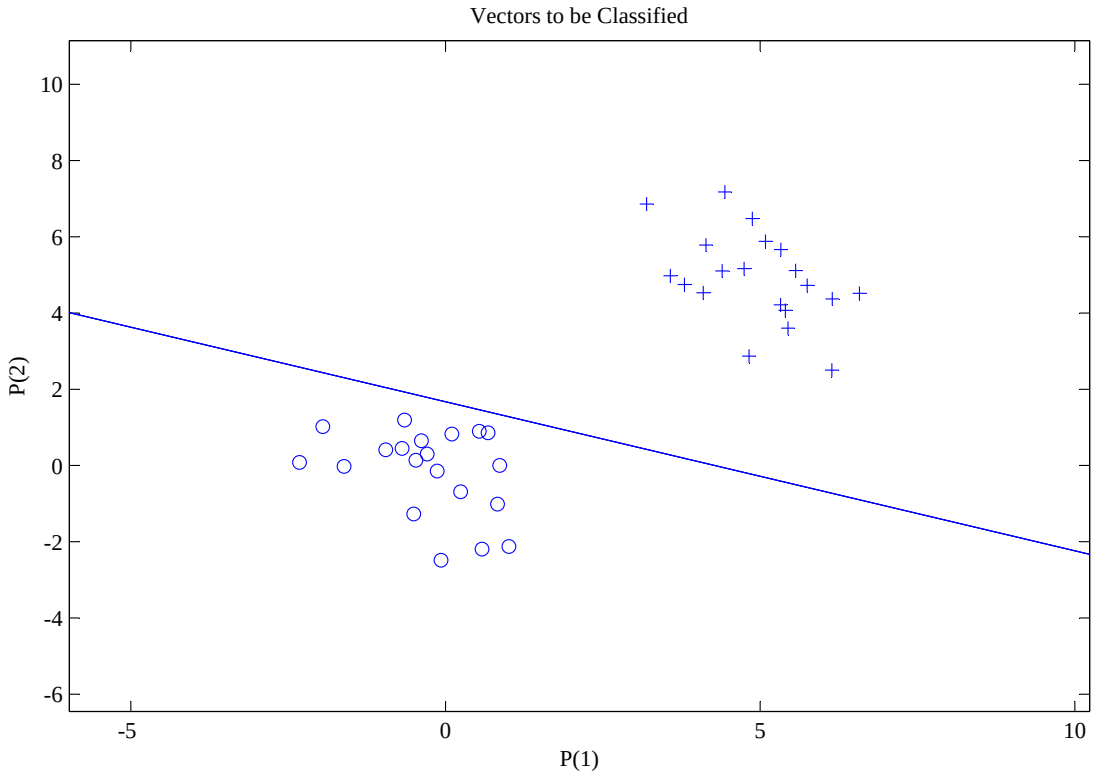
Perceptron problem, step by step

Example 1; Solution

```
% number of samples of each class
N = 20;
% define inputs and outputs
offset = 5; % offset for second class
x = [randn(2,N) randn(2,N)+offset]; % inputs
y = [zeros(1,N) ones(1,N)]; % outputs
% Plot input samples with PLOTPV (Plot perceptron
input/target vectors)
plotpv(x,y);
net=newp(x,y);
net = train(net,x,y); % train network
figure(1)
plotpc(net.IW{1},net.b{1});
```

Perceptron problem, step by step

Example 1; Solution



Note

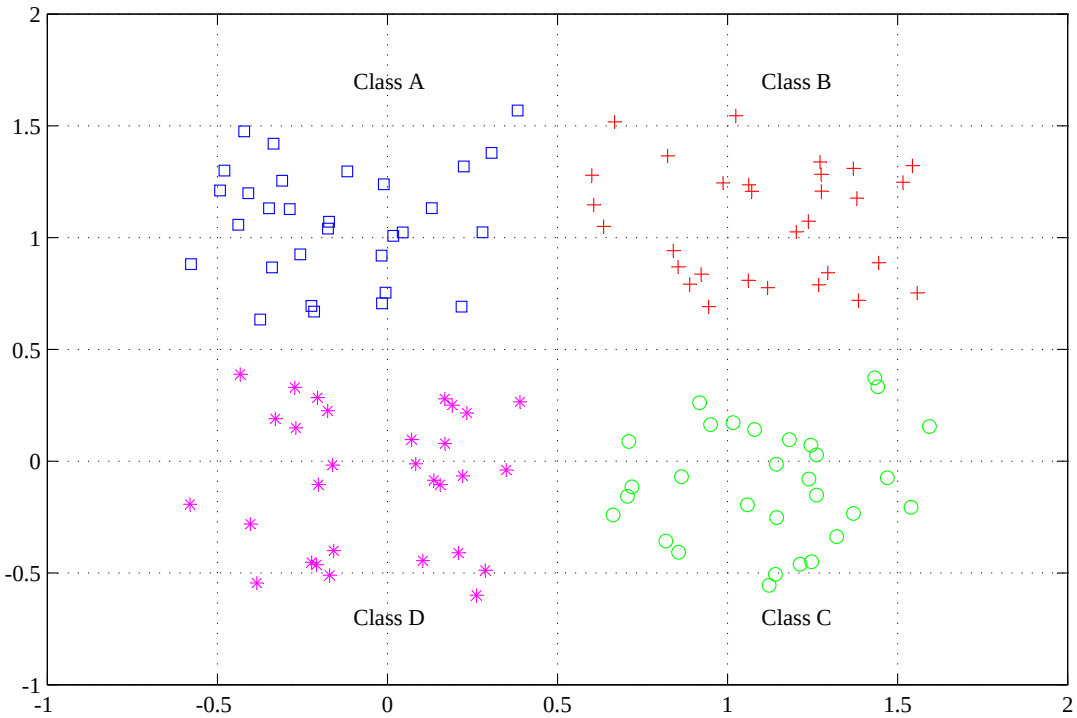
- `net.trainParam.epochs`
- `net.trainParam.goal`
- Random Weights Initialization

```
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net);
```

Perceptron problem

Example 2:

Classifying displayed data



Perceptron problem

Example 2; Solution

```
K = 30;
% define classes
q = .6; % offset of classes
A = [rand(1,K)-q; rand(1,K)+q]; B = [rand(1,K)+q;
rand(1,K)+q];
C = [rand(1,K)+q; rand(1,K)-q];  D = [rand(1,K)-q;
rand(1,K)-q];
% plot classes
plot(A(1,:),A(2,:),'bs')
hold on; grid on
plot(B(1,:),B(2,:),'r+'); plot(C(1,:),C(2,:),'go');
plot(D(1,:),D(2,:),'m*');
% text labels for classes
text(.5-q,.5+2*q,'Class A'); text(.5+q,.5+2*q,'Class B')
```

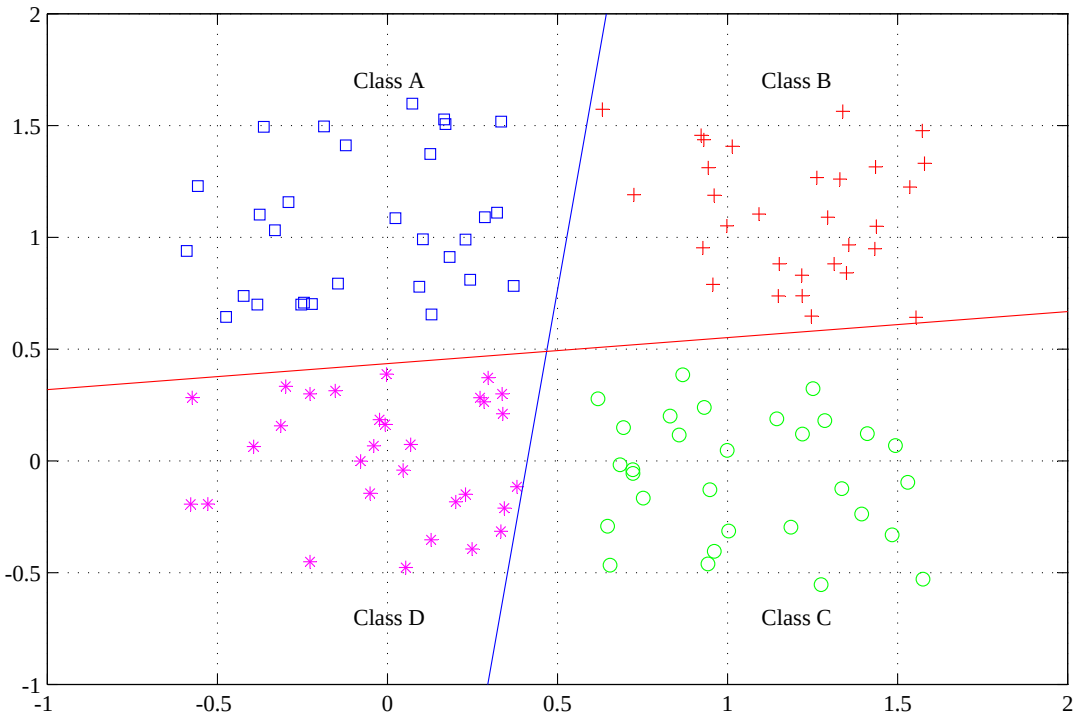
Perceptron problem

Example 2; Solution-Cont'd

```
text(.5+q,.5-2*q,'Class C'); text(.5-q,.5-2*q,'Class D')
a = [0 1]'; b = [1 1]';
c = [1 0]'; d = [0 0]';
P = [A B C D];
% define targets
T = [repmat(a,1,length(A)) repmat(b,1,length(B)) ...
repmat(c,1,length(C)) repmat(d,1,length(D)) ];
net=newp(P,T);
E = 1;
net.adaptParam.passes = 1;
linehandle = plotpc(net.IW{1},net.b{1}); n = 0;
while (sse(E) & n<1000)
n = n+1;
[net,Y,E] = adapt(net,P,T);
linehandle = plotpc(net.IW{1},net.b{1},linehandle);
drawnow;
end
```

Perceptron problem

Example 2; Solution- Cont'd



Thank You


Matlab 7.9, *Neural Network Toolbox Help* MathWorks Inc. R2009b


Click: LASIN - Laboratory of Synergetics Website, Slovenia


Dr. F. Abdollahi, *Lecture Notes on Neural Networks* Winter 2011