Neural network

Machine Learning II

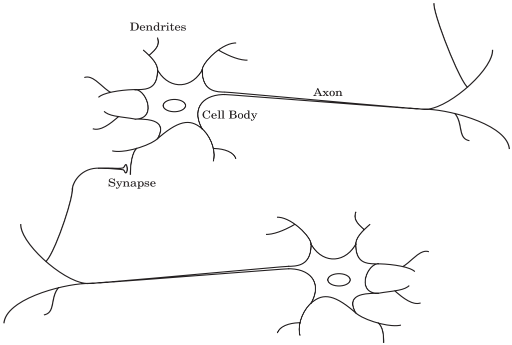
Lecture 2-d

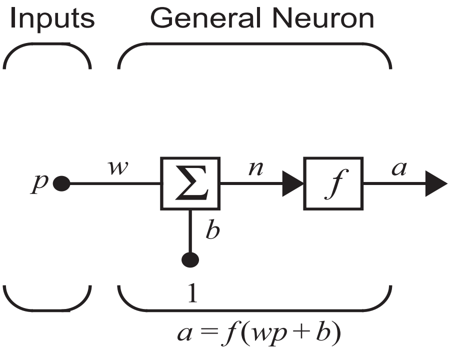
THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

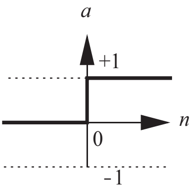
Brain Function

- Neurons Respond Slowly
  - 10^{-3} s compared to to 10^{-9} s for electrical circuits.
- The brain uses massively parallel comptation
  - $\approx 10^{11}$ neurons in the brain.
  - $\approx 10^{4}$ connections per neurons.



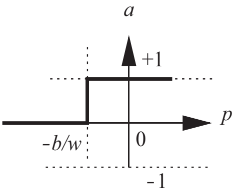

Inputs
General Neuron
p
w
Σ
n
f
a
b
1
a = f(wp + b)

Transfer function



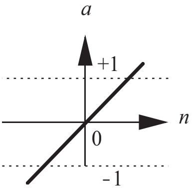
$a = hardlim(n)$

Hard Limit Transfer Function



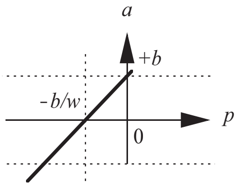
$a = hardlim(wp + b)$

Single-Input *hardlim* Neuron



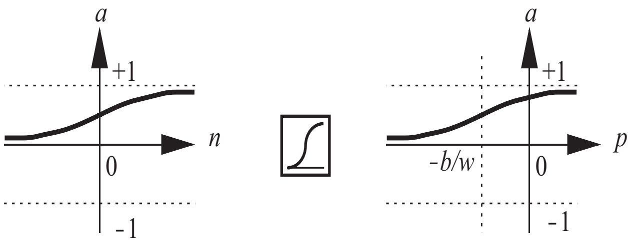
$a = purelin(n)$

Linear Transfer Function



$a = purelin(wp + b)$

Single-Input *purelin* Neuron



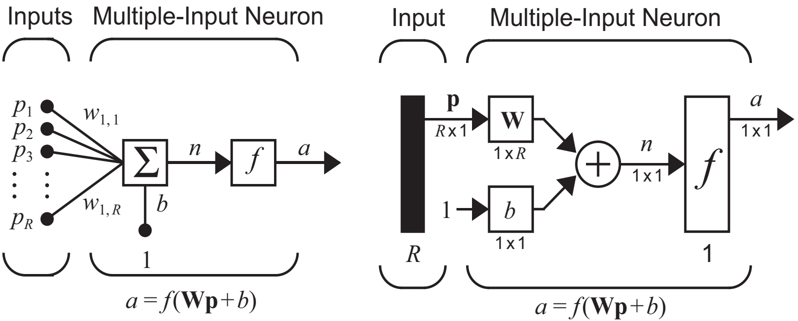
$$a = logsig(n)$$

Log-Sigmoid Transfer Function

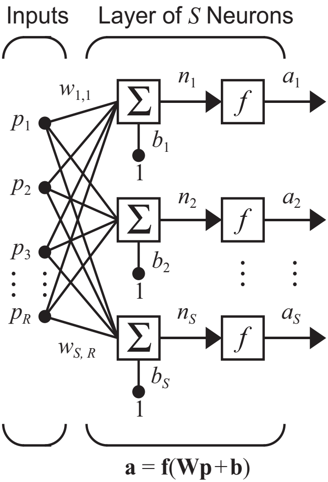
$$a = logsig(wp + b)$$

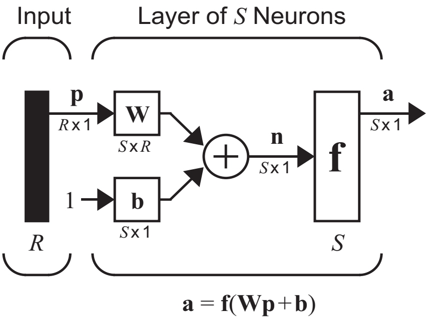
Single-Input *logsig* Neuron

Multiple input neuron



Abreviated Notation

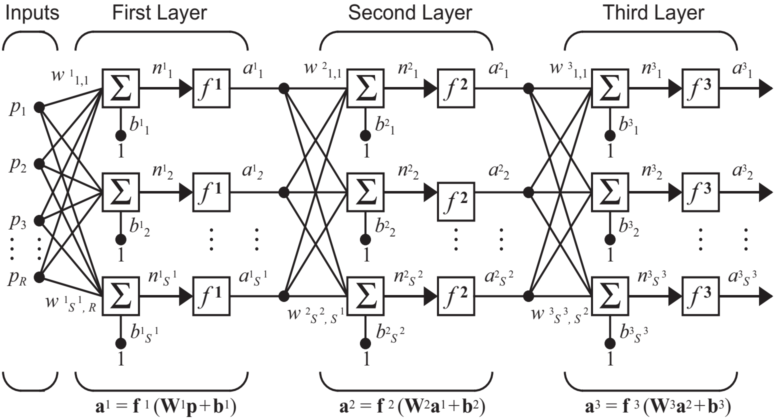

Inputs
Layer of S Neurons
$w_{1,1}$
p_1
p_2
p_3
p_R
$w_{S,R}$
Σ
n_1
n_2
n_S
f
a_1
a_2
a_S
b_1
b_2
b_S
1
$\mathbf{a} = \mathbf{f}(\mathbf{Wp}+\mathbf{b})$



$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

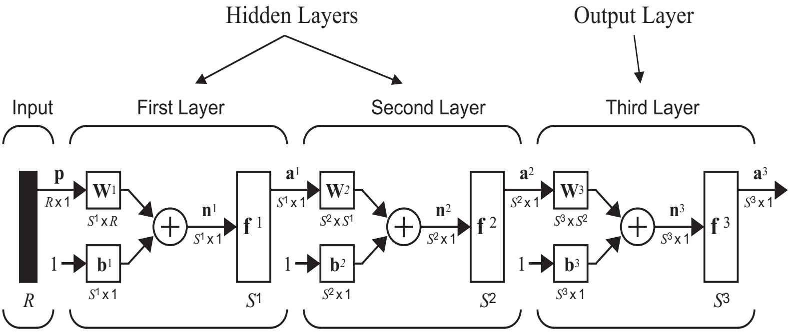
$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_S \end{bmatrix}$$

Multilayer network



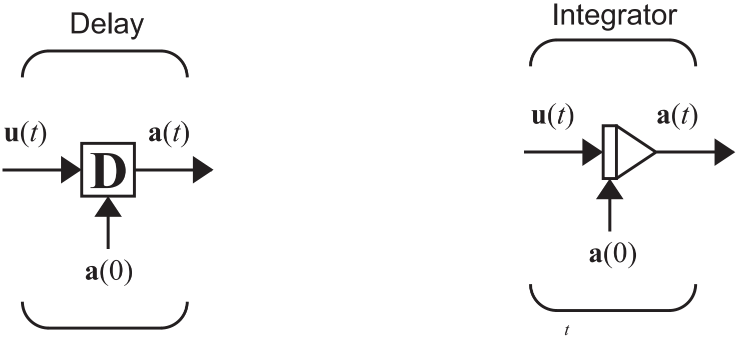
$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Abbreviated notation



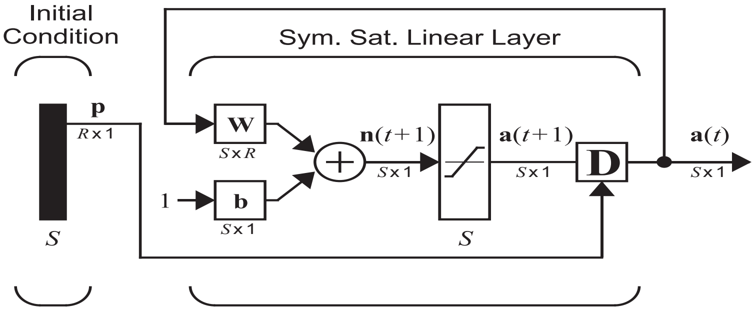
$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)$ $\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2)$ $\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\,\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$



$$\mathbf{a}(t) = \mathbf{u}(t-1)$$

$$\mathbf{a}(t) = \int_0^t \mathbf{u}(\tau)\, d\tau + \mathbf{a}(0)$$

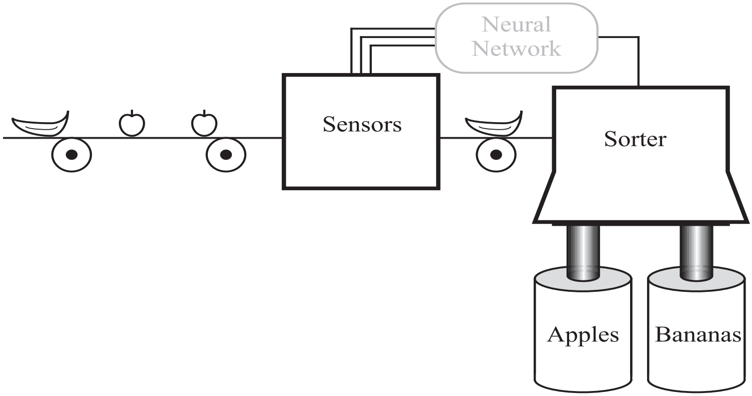


$$\mathbf{a}(0) = \mathbf{p} \qquad \mathbf{a}(t+1) = \mathbf{satlin}\,(\mathbf{W}\mathbf{a}(t) + \mathbf{b})$$

$$\mathbf{a}(1) = \mathbf{satlins}(\mathbf{W}\mathbf{a}(0) + \mathbf{b}) = \mathbf{satlins}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

$$\mathbf{a}(2) = \mathbf{satlins}(\mathbf{W}\mathbf{a}(1) + \mathbf{b})$$

Apple banana sorter



Prototype vectors

Measurement Vector

$$\mathbf{p} = \begin{bmatrix} \text{shape} \\ \text{texture} \\ \text{weight} \end{bmatrix}$$

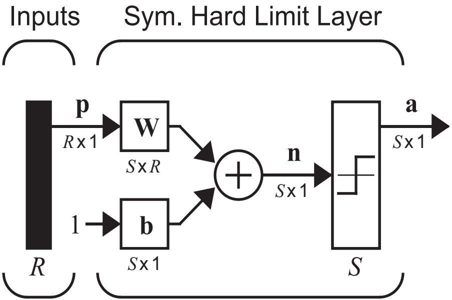
Shape: {1 : round ; -1 : eliptical}
Texture: {1 : smooth ; -1 : rough}
Weight: {1 : > 1 lb. ; -1 : < 1 lb.}

Prototype Banana

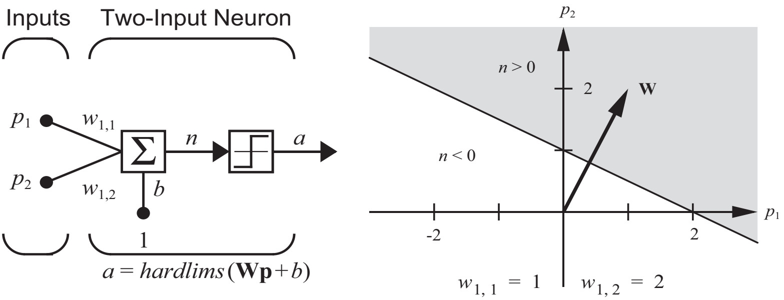
$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

Prototype Apple

$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$



$$\mathbf{a} = \mathbf{hardlims}(\mathbf{Wp} + \mathbf{b})$$

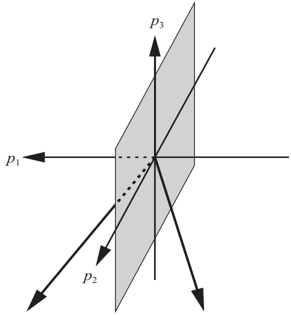


$$a = hardlims(n) = hardlims\left(\begin{bmatrix} 1 & 2 \end{bmatrix}\mathbf{p} + (-2)\right)$$

Decision Boundary

$$\mathbf{Wp} + b = 0 \qquad \begin{bmatrix} 1 & 2 \end{bmatrix}\mathbf{p} + (-2) = 0$$

$$a = hardlims\left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b\right)$$



- The decision boundary should separate the prototype vectors
- $p_1 = 0$
- The weight vector should be orthogonal to the decision boundary, and should point in the direction of the vector which should produce an output of 1. The bias determines the position of the boundary.
- $\begin{bmatrix} -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0$

Banana:

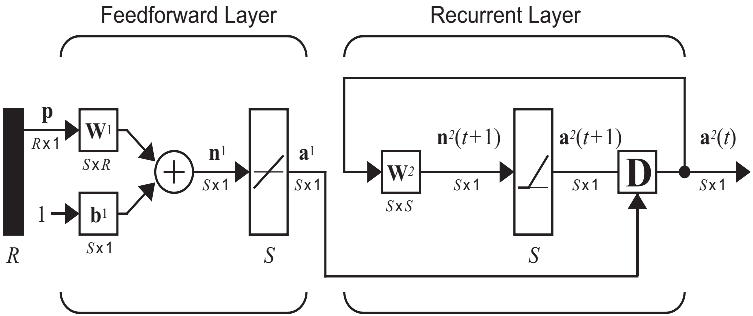
$$a = hardlims\left(\begin{bmatrix}-1 & 0 & 0\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0\right) = 1(\text{banana})$$

Apple:

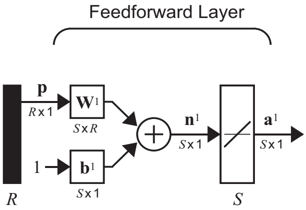
$$a = hardlims\left(\begin{bmatrix}-1 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + 0\right) = -1\ (\text{apple})$$

"Rough" Banana:

$$a = hardlims\left(\begin{bmatrix}-1 & 0 & 0\end{bmatrix}\begin{bmatrix}-1\\-1\\-1\end{bmatrix} + 0\right) = 1(\text{banana})$$



$$\mathbf{a}^1 = \mathbf{purelin}(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1) \qquad \mathbf{a}^2(0) = \mathbf{a}^1 \qquad \mathbf{a}^2(t+1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(t))$$



$$\mathbf{a}^1 = \mathbf{purelin}(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1)$$

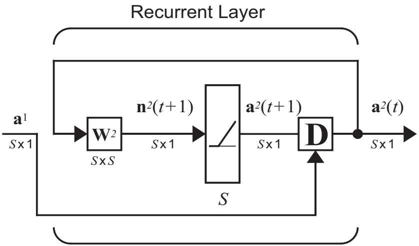
For Banana/Apple Recognition

$$S = 2$$

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix}$$

$$\mathbf{b}^1 = \begin{bmatrix} R \\ R \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

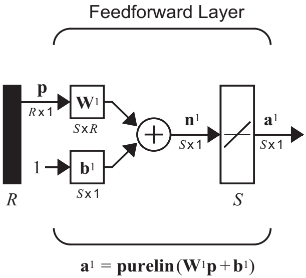
$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix}\mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T\mathbf{p} + 3 \\ \mathbf{p}_2^T\mathbf{p} + 3 \end{bmatrix}$$



$$\mathbf{a}^2(0) = \mathbf{a}^1 \qquad \mathbf{a}^2(t+1) = \mathbf{poslin}(\mathbf{W}^2\mathbf{a}^2(t))$$

$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix} \qquad \varepsilon < \frac{1}{S-1}$$

$$\mathbf{a}^2(t+1) = \mathbf{poslin}\left(\begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}\mathbf{a}^2(t)\right) = \mathbf{poslin}\left(\begin{bmatrix} a_1^2(t) - \varepsilon a_2^2(t) \\ a_2^2(t) - \varepsilon a_1^2(t) \end{bmatrix}\right)$$



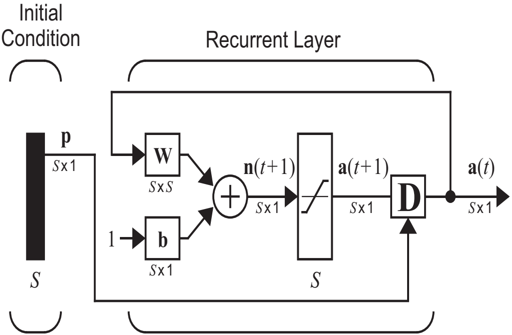
For Banana/Apple Recognition

$$S = 2$$

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix}$$

$$\mathbf{b}^1 = \begin{bmatrix} R \\ R \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix}\mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T\mathbf{p} + 3 \\ \mathbf{p}_2^T\mathbf{p} + 3 \end{bmatrix}$$



$$\mathbf{a}(0) = \mathbf{p} \qquad \mathbf{a}(t+1) = \mathbf{satlins}(\mathbf{W}\mathbf{a}(t)+\mathbf{b})$$

$$\mathbf{W} = \begin{bmatrix} 1.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 0.9 \\ -0.9 \end{bmatrix}$$

$$a_1(t+1) = satlins(1.2a_1(t))$$

$$a_2(t+1) = satlins(0.2a_2(t) + 0.9)$$

$$a_3(t+1) = \text{satlins}(0.2a_3(t) - 0.9)$$

Test: “Rough” Banana

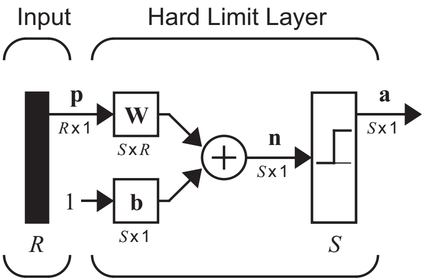
$$\mathbf{a}(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \quad \mathbf{a}(1) = \begin{bmatrix} -1 \\ 0.7 \\ -1 \end{bmatrix} \quad \mathbf{a}(2) = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \quad \mathbf{a}(3) = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \text{ (Banana)}$$

- Perceptron
  - Feedforward Network
  - Linear Decision Boundary
  - One neuron for each decision
- Hamming Network
  - Competitive Network
  - First Layer - Pattern Matching (Inner Product)
  - Second Layer - Competition (Winner - Take - All)
  - # Neurons = # Prototype Patterns
- Hopfield Network
  - Dynamic Associative Memory Network
  - Network Output Converges to Prototype Pattern
  - # Neurons = # Elements is each Prototype Pattern

- Supervised learning: Network is provided with a set of examples of proper network behavior (inputs and targets)
$\{p_1, t_1\}, \{p_2, t_2\}, \dots, \{p_Q, t_Q\}$

- Reinforcement learning: Network is only provided with a grade, or score, which indicates network performance.

- Unsupervised learning: Only network inputs are available to the learning algorithm. Network learns to categorize (cluster) the inputs.



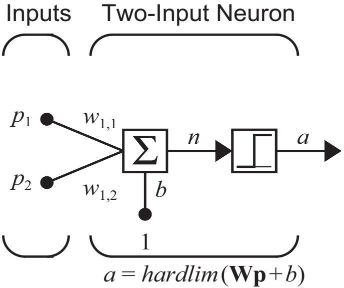
$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp} + \mathbf{b})$$

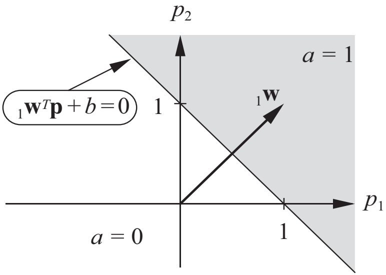
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$${}_{i}\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix} \qquad \mathbf{W} = \begin{bmatrix} {}_{1}\mathbf{w}^{T} \\ {}_{2}\mathbf{w}^{T} \\ \vdots \\ {}_{S}\mathbf{w}^{T} \end{bmatrix}$$

$$a_i = hardlim(n_i) = hardlim({}_{i}\mathbf{w}^{T}\mathbf{p} + b_i)$$

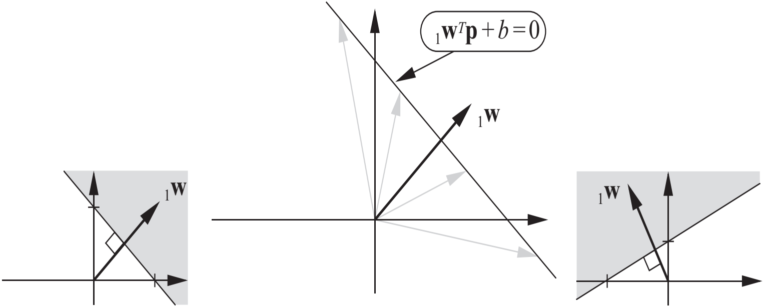
$$w_{1,1} = 1 \qquad w_{1,2} = 1 \qquad b = -1$$



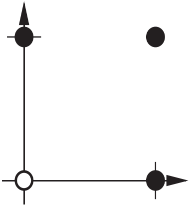


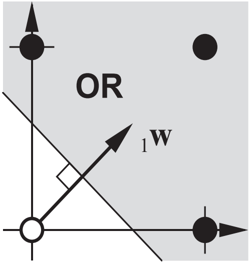
$$a = hardlim({}_1\mathbf{w}^T\mathbf{p} + b) = hardlim(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

- ${}_1\mathbf{w}^T\mathbf{p} + b = 0$, ${}_1\mathbf{w}^T\mathbf{p} = b$
- All points on the decision boundary have the same inner product with the weight vector.
- Therefore they have the same projection onto the weight vector, and they must lie on a line orthogonal tot he weight vector



$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\} \qquad \left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\} \quad \left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 1\right\}$$





- Weight vector should be orthogonal to the decision boundary.
- $_i\mathbf{w} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$
- Pick a point on the decision boundary to find the bias.
- $_i\mathbf{w}\mathbf{p} + b = [0.5 0.5] \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} + b = 0.25 + b = 0 \Rightarrow b = -0.25$

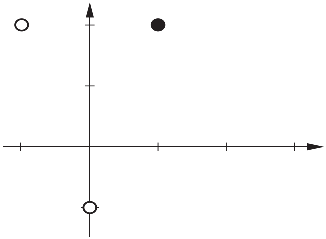
Multiple neuron perceptron

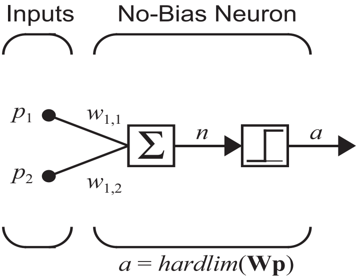
- Each neuron will have its own decision boundary.
- ${}_i\mathbf{w}^T\mathbf{p} + b_i = 0$
- A single neuron can classify input vectors into two categories.
- A multi neuron perceptron can classify input vectors into 2^S categories.

Learning rule test problem

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

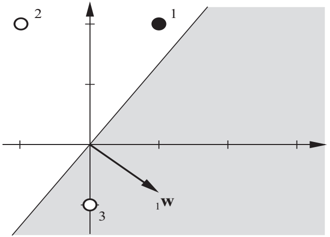
$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}$$





Random initial weight:

$$ {}_1\mathbf{w} = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} $$



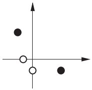
Present $\mathbf{p}_1$ to the network:

$$ a = hardlim({}_1\mathbf{w}^T\mathbf{p}_1) = hardlim\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) $$

$$ a = hardlim(-0.6) = 0 $$

Incorrect Classification.

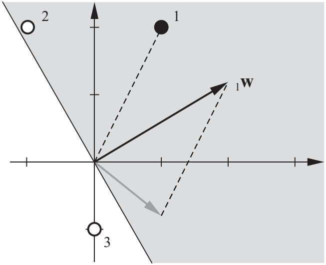
Set ${}_1\mathbf{w}$ to $\mathbf{p}_1$
– Not stable $\times$



Add $\mathbf{p}_1$ to ${}_1\mathbf{w}$ $\checkmark$

Tentative Rule: If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} $$



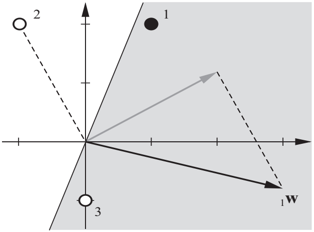
Second input vector

$$a = hardlim({}_1\mathbf{w}^T \mathbf{p}_2) = hardlim\left(\begin{bmatrix} 2.0 & 1.2 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix}\right)$$

$$a = hardlim(0.4) = 1$$ (Incorrect Classification)

Modification to Rule: If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

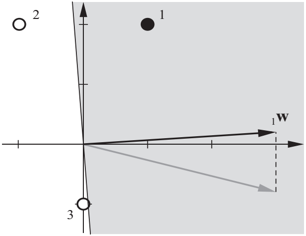
$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$



$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}_3) = hardlim\left(\begin{bmatrix}3.0 & -0.8\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\right)$$

$$a = \text{hardlim}(0.8) = 1 \qquad \text{(Incorrect Classification)}$$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix}3.0\\-0.8\end{bmatrix} - \begin{bmatrix}0\\-1\end{bmatrix} = \begin{bmatrix}3.0\\0.2\end{bmatrix}$$



Patterns are now correctly classified.

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$.

$$\text{If } t = 1 \text{ and } a = 0, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$$

$$\text{If } t = 0 \text{ and } a = 1, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$$

$$\text{If } t = a, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$$

$$e = t - a$$

$$\text{If } e = 1, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$$

$$\text{If } e = -1, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$$

$$\text{If } e = 0, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$$

$$\boxed{\begin{aligned} {}_1\mathbf{w}^{new} &= {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t - a)\mathbf{p} \\ b^{new} &= b^{old} + e \end{aligned}}$$

A bias is a weight with an input of 1.

To update the ith row of the weight matrix:

$$ {}_i\mathbf{w}^{new} = {}_i\mathbf{w}^{old} + e_i\mathbf{p} $$

$$ b_i^{new} = b_i^{old} + e_i $$

Matrix form:

$$ \mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T $$

$$ \mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e} $$

Training Set

$$\left\{\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, t_1 = \begin{bmatrix} 1 \end{bmatrix}\right\} \qquad \left\{\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = \begin{bmatrix} 0 \end{bmatrix}\right\}$$

Initial Weights

$$\mathbf{W} = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \qquad b = 0.5$$

First Iteration

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0.5\right)$$

$$a = hardlim(-0.5) = 0 \qquad e = t_1 - a = 1 - 0 = 1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} + (1)\begin{bmatrix} -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix}$$

$$b^{new} = b^{old} + e = 0.5 + (1) = 1.5$$

Second iteration

$$a = hardlim\,(\mathbf{W}\mathbf{p}_2 + b) = hardlim\left(\begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + (1.5)\right)$$

$$a = hardlim\,(2.5) = 1$$

$$e = t_2 - a = 0 - 1 = -1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix} + (-1)\begin{bmatrix} 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix}$$

$$b^{new} = b^{old} + e = 1.5 + (-1) = 0.5$$

Check

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim(1.5) = 1 = t_1$$

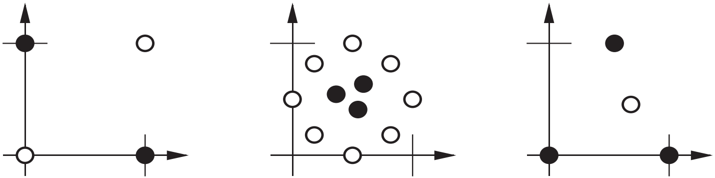
$$a = hardlim(\mathbf{W}\mathbf{p}_2 + b) = hardlim\left(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim(-1.5) = 0 = t_2$$

Linear Decision Boundary

$${}_1\mathbf{w}^T\mathbf{p} + b = 0$$

Linearly Inseparable Problems



- Some library has to be imported to implement the perceptron learning rule.

```
>>>import numpy as np
>>>import pandas as pd
>>>import matplotlib.pyplot as plt
>>>ppn = Perceptron(eta=0.1, n_iter=10)
>>>ppn.fit(X, y)
```

Summary

- We were touching the main concepts behind neural network in this lecture.
- Neural network is the back bone of machine learning algorithms.
- This lecture just briefly introduces the main and basic concepts of neural network.
- Read chapter 1 through 4 again and look at the solved problems from the following book, Neural Network Design.
- We are going to use these concepts in this lecture for the first mini project.

Next lecture

- Performance optimizations will be discussed in the next lecture.
- In order to training the network more than one layer we need to build some mathematical background in optimizations.
- Logistic regression will be discussed.
- Read ahead the Chapter 8 through 10 from the Neural Network Design book.