CS5805 : Machine Learning I
Lecture # 14

Reza Jafari, Ph.D

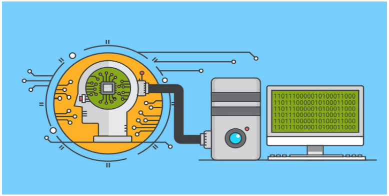
Collegiate Associate Professor
rjafari@vt.edu

COLLEGE OF ENGINEERING
COMPUTER SCIENCE
VIRGINIA TECH.

Outline of Lecture

What is Machine learning?

- A discipline within the field of Artificial Intelligence, by means of algorithms, provides computers with the ability to identify patterns from mass data in order to make predictions.
- Learning is the process of gaining knowledge or skills through experience.
- The input to this learning process is training data and the output is some expertise that can perform some task.

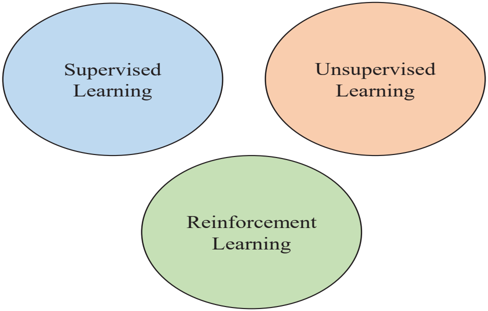


Machine Learning Applications

- Aerospace
  - Autopilot
- Intelligent vehicles
  - Driverless cars.
- Social networks
  - Spam detection.
- Medicine
  - Early breast cancer detection.
- Computer vision
  - Object detection.
- Speech
  - Translation from one language to another

Machine Learning Algorithms

- Machine Learning algorithms are divided into three categories:

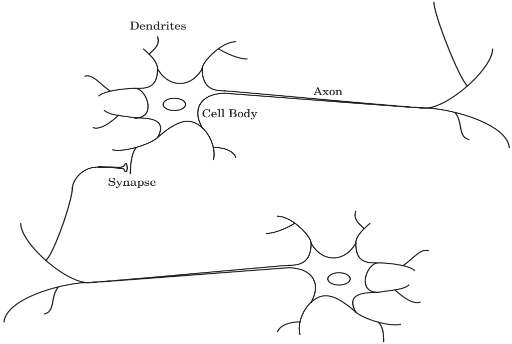


Learning rules

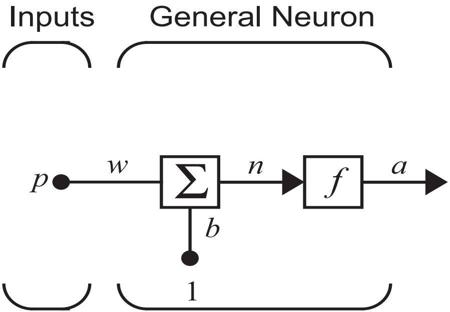
- Supervised learning: Network is provided with a set of examples of proper network behavior inputs and targets.
  $\{p_1, t_1\}, \{p_2, t_2\}, ..., \{p_Q, t_Q\}$
- Reinforcement learning: Network is only provided with a grade, or score, which indicates network performance.
- Unsupervised learning: Only network inputs are available to the learning algorithm. Network learns to categorize (cluster) the inputs.

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.

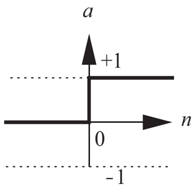


Single input model



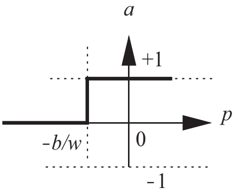
$$a = f(wp + b)$$

Transfer function



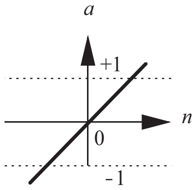
$a = hardlim(n)$

Hard Limit Transfer Function



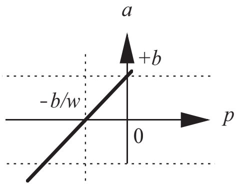
$a = hardlim(wp + b)$

Single-Input *hardlim* Neuron



$a = purelin(n)$

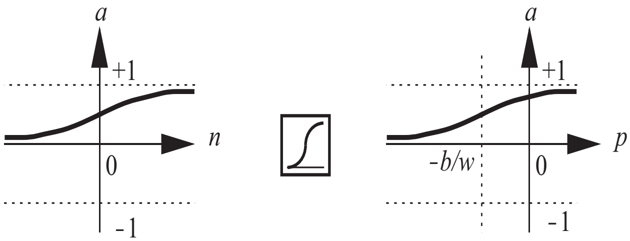
Linear Transfer Function



$a = purelin(wp + b)$

Single-Input *purelin* Neuron

Transfer function



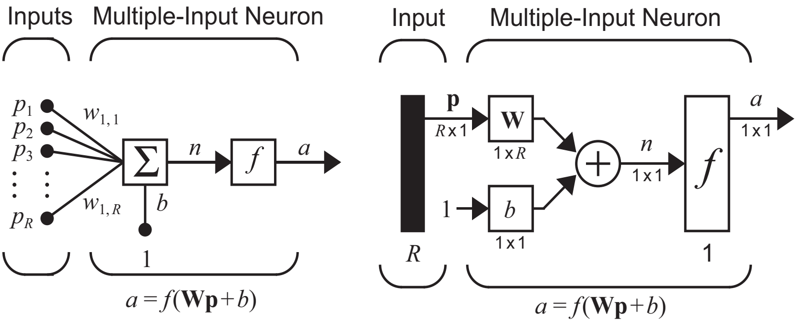
$a = logsig(n)$

Log-Sigmoid Transfer Function

$a = logsig(wp + b)$

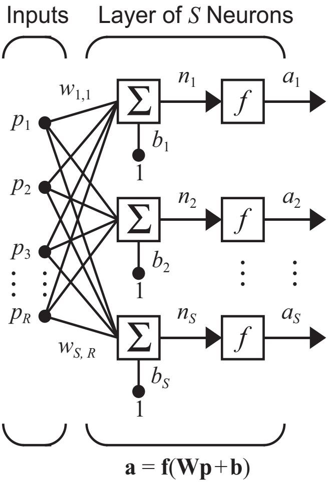
Single-Input *logsig* Neuron

Multiple input neuron

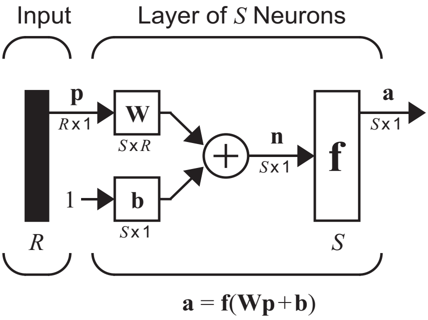


Abreviated Notation

Layer of neurons



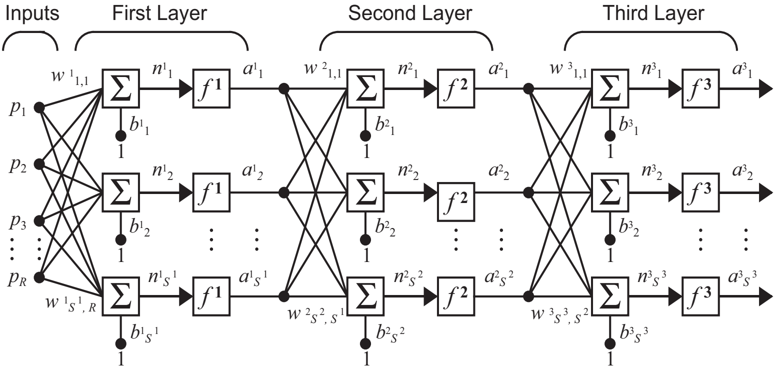
Abbreviated notation



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

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

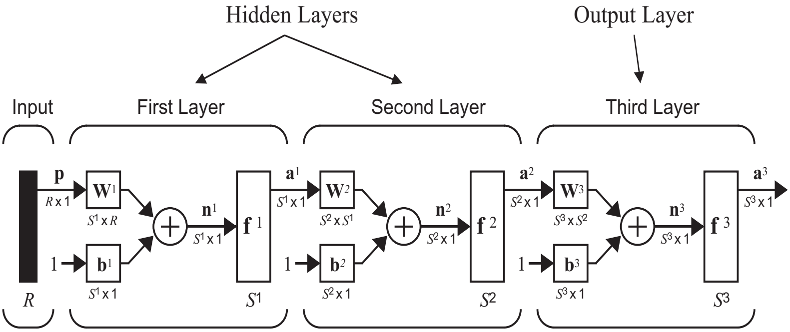
Multilayer network



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

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

Abbreviated notation

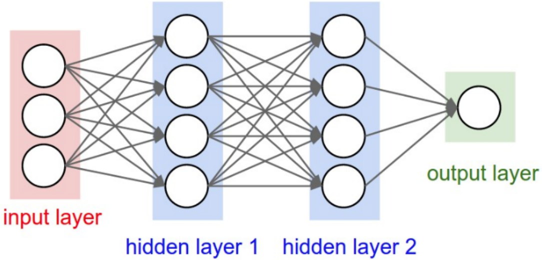


$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)$ $\quad$ $\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2)$ $\quad$ $\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$

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

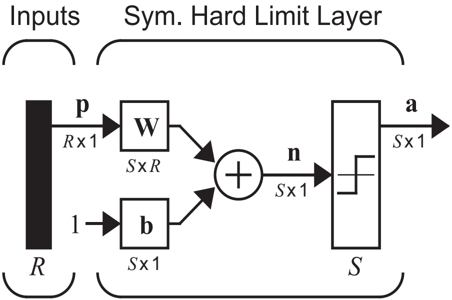
Diagram representation

- input layer = 3 units (feature space dimension)
- hidden layer 1 = 4 units
- hidden layer 2 = 4 units
- output layer = 2 units



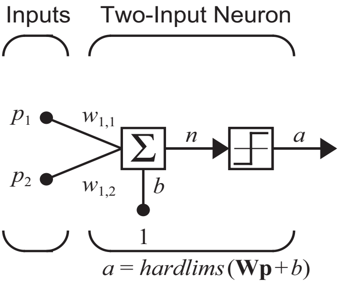

```
from keras.models import Sequential
from keras.layers import *

model = Sequential()

model.add(Input(shape=(3,))) # Input tensor
model.add(Dense(units=4)) # hidden layer 1
model.add(Dense(units=4)) #hidden layer 2
model.add(Dense(units=1)) #output layer
```

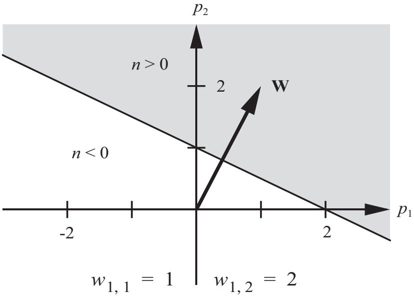
Perceptron



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

Two input case



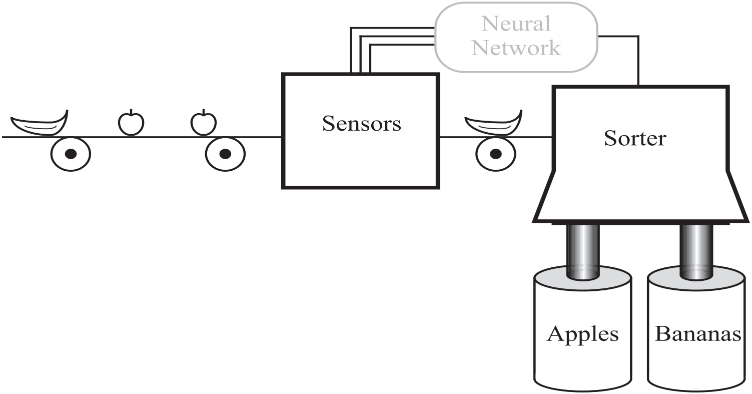


$$a = hardlims(n) = hardlims(\begin{bmatrix}1 & 2\end{bmatrix}\mathbf{p} + (-2))$$

Decision Boundary

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

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

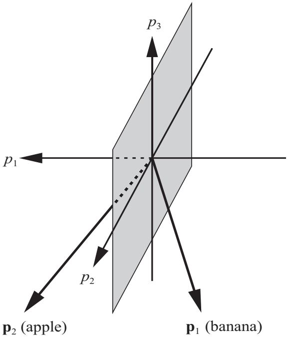
$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

Prototype Apple

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

Apple banana example

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



- 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.
- $\begin{bmatrix} -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0$

Testing the network

Banana:

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

Apple:

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

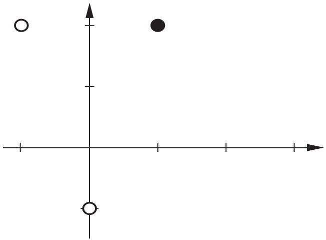
“Rough” Banana:

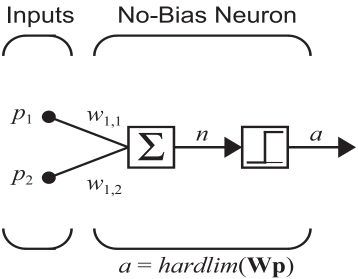
$$a = hardlims\left(\begin{bmatrix}-1 & 0 & 0\end{bmatrix}\begin{bmatrix}-1\\-1\\-1\end{bmatrix}+0\right) = 1(\text{banana})$$

Learning rule test problem

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

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

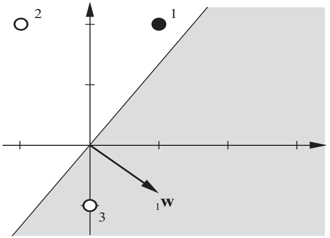




Starting point

Random initial weight:

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



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

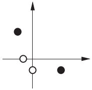
$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}_1) = hardlim\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$$

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

Incorrect Classification.

Tentative learning rule

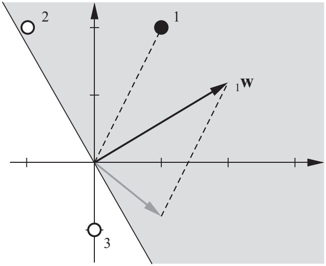
Set ${}_1\mathbf{w}$ to $\mathbf{p}_1$
– Not stable ×



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

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

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



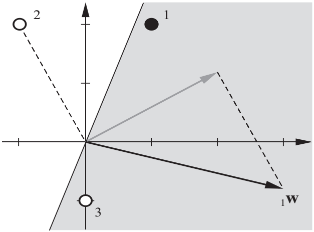
Second input vector

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

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

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

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

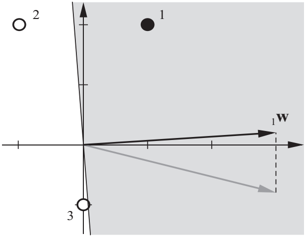


Third input vector

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

$$\mathrm{a = hardlim(0.8) = 1}$$ (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 ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$.

Unified learning rule

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

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

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

$$e = t - a$$

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

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

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

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

A bias is a weight with an input of 1.

Multiple neuron perceptrons

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:

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

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

Apple banana example

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

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

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

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

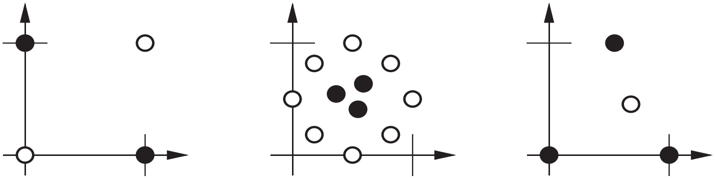
$$a = hardlim(-1.5) = 0 = t_2$$

Perceptron limitations

Linear Decision Boundary

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

Linearly Inseparable Problems



In-class Activity-Demo

- Generate two sets of normally distributed dataset [random].
- Label the dataset with target value of 0 and 1.
- Implement the perceptron rule that classifies the dataset and draw the decision boundary.