第二讲 深度前馈神经网络

目录

- 感知机
- 前馈神经网络与反向传播算法
- 反向传播算法分析
- 过度拟合与正则化
- 深度神经网络的优化与扩展

感知机

- 前馈网络与递归网络结构及数学模型
- **How to mimic some intelligent behaviors?**
- **Perceptron: Data Classification**
- **An Intuitive Example**

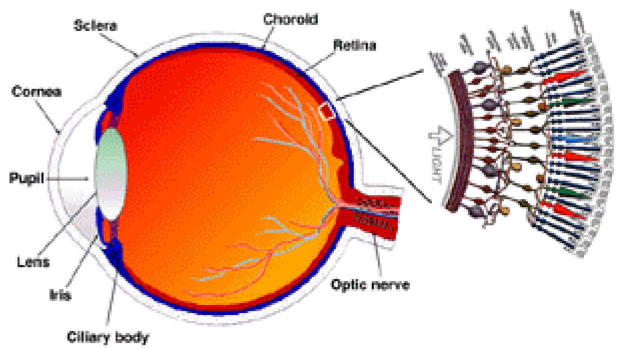
Neural Network Structures

A Neural Network＝Neurons＋Connection

<u>神经网络分类:</u>

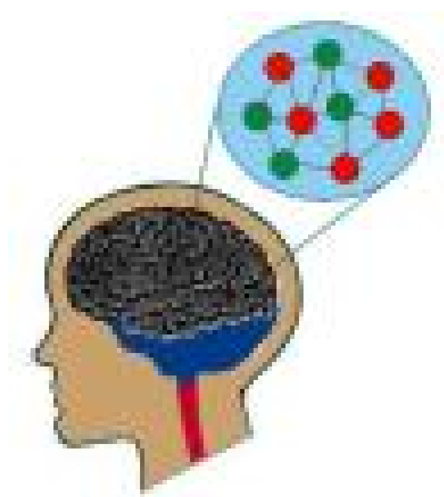
- 前馈神经网络（**Feed forward NNs**）：没有后向反馈连接
- 递归神经网络（**Recurrent NNs**）：有后向反馈连接

Neural Networks



Feed forward NNs (FNNs):

No feedback connection

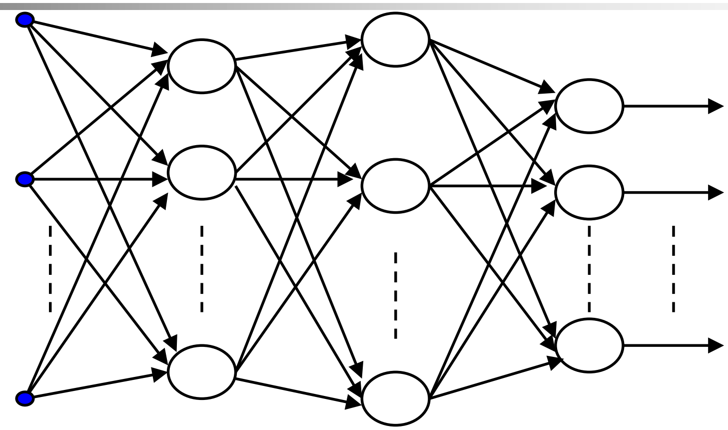


Recurrent NNs (RNNs):

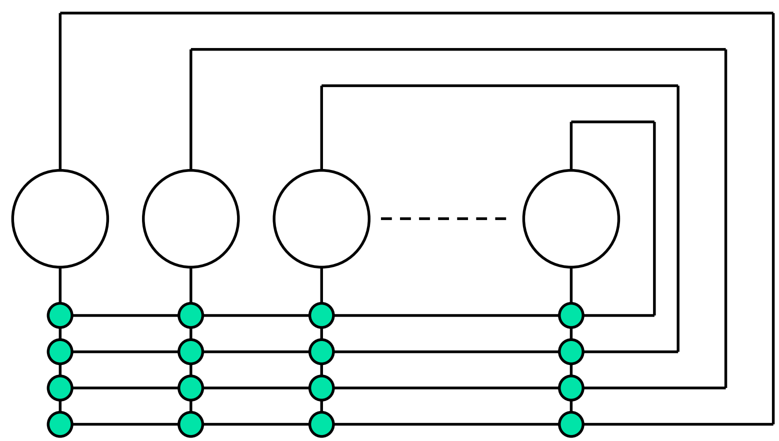
have feedback connection

Neural Networks

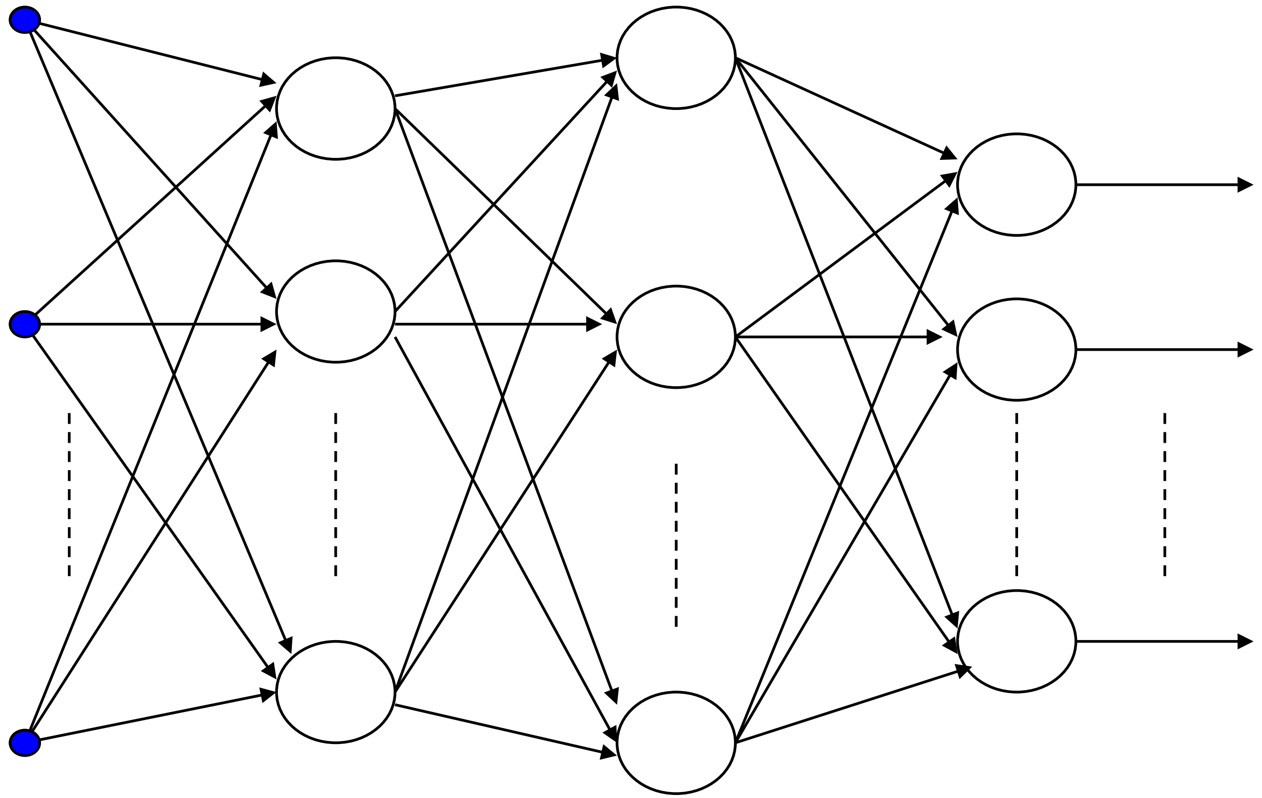
Feed forward NNs



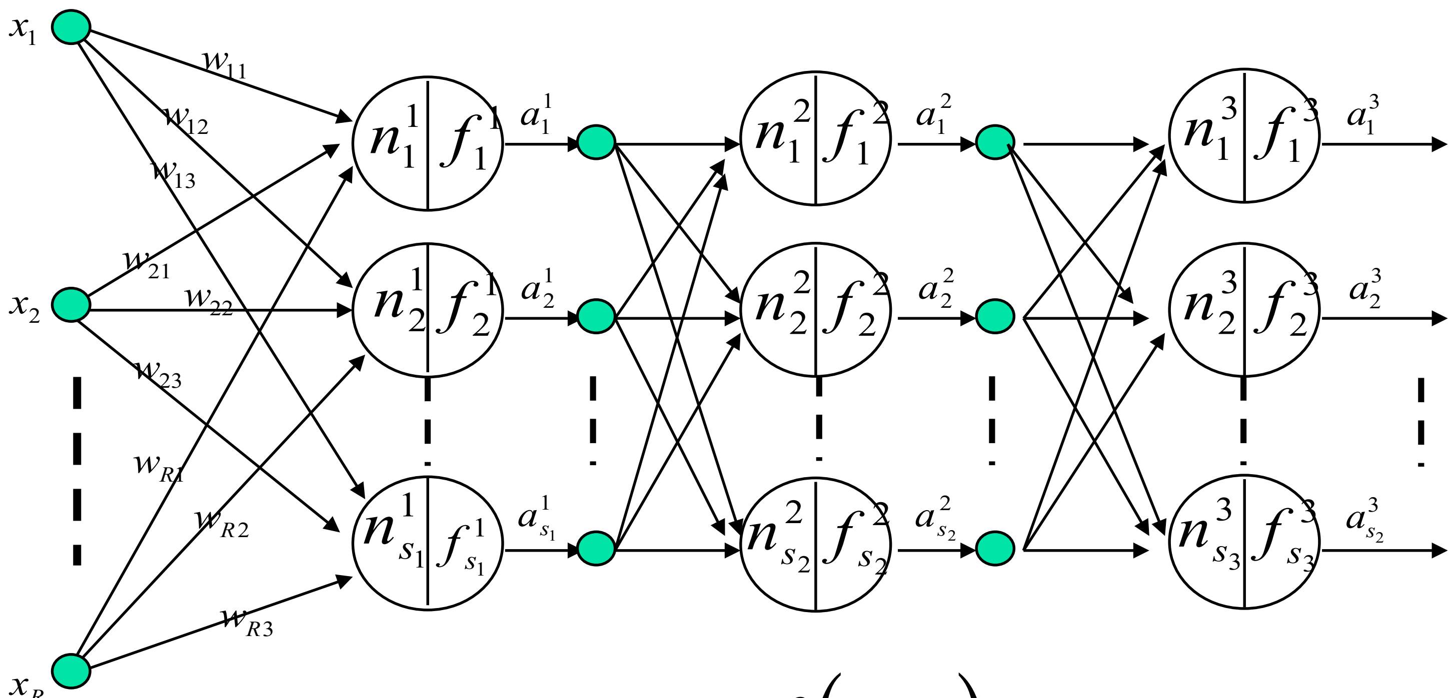
Recurrent NNs



Feed Forward NNs

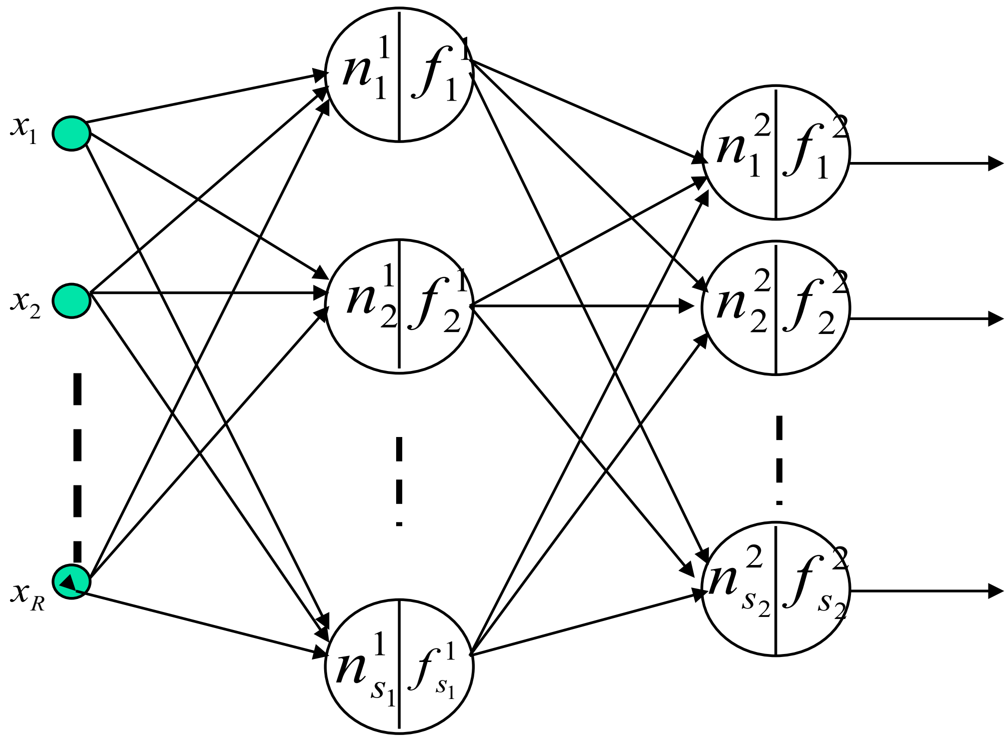


Math Model of FNNs



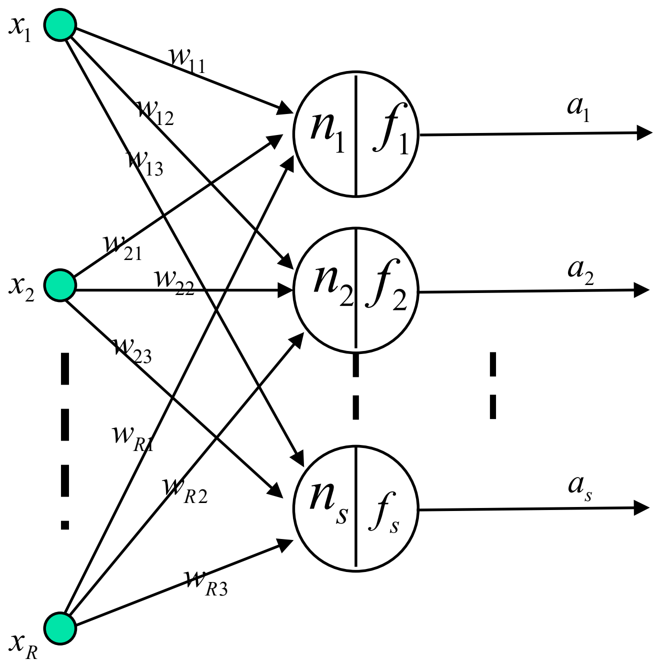
$$a = f(Wx)$$

Math Model of FNNs



$$a = f(Wx)$$

Math Model of FNNs



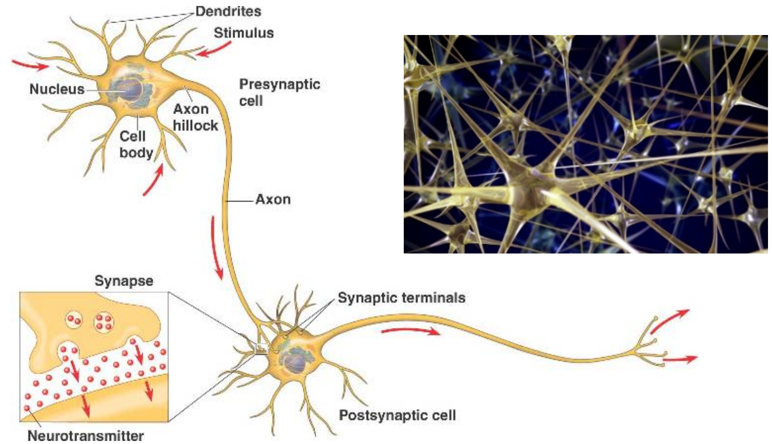
$$a = f\left(Wx\right)$$

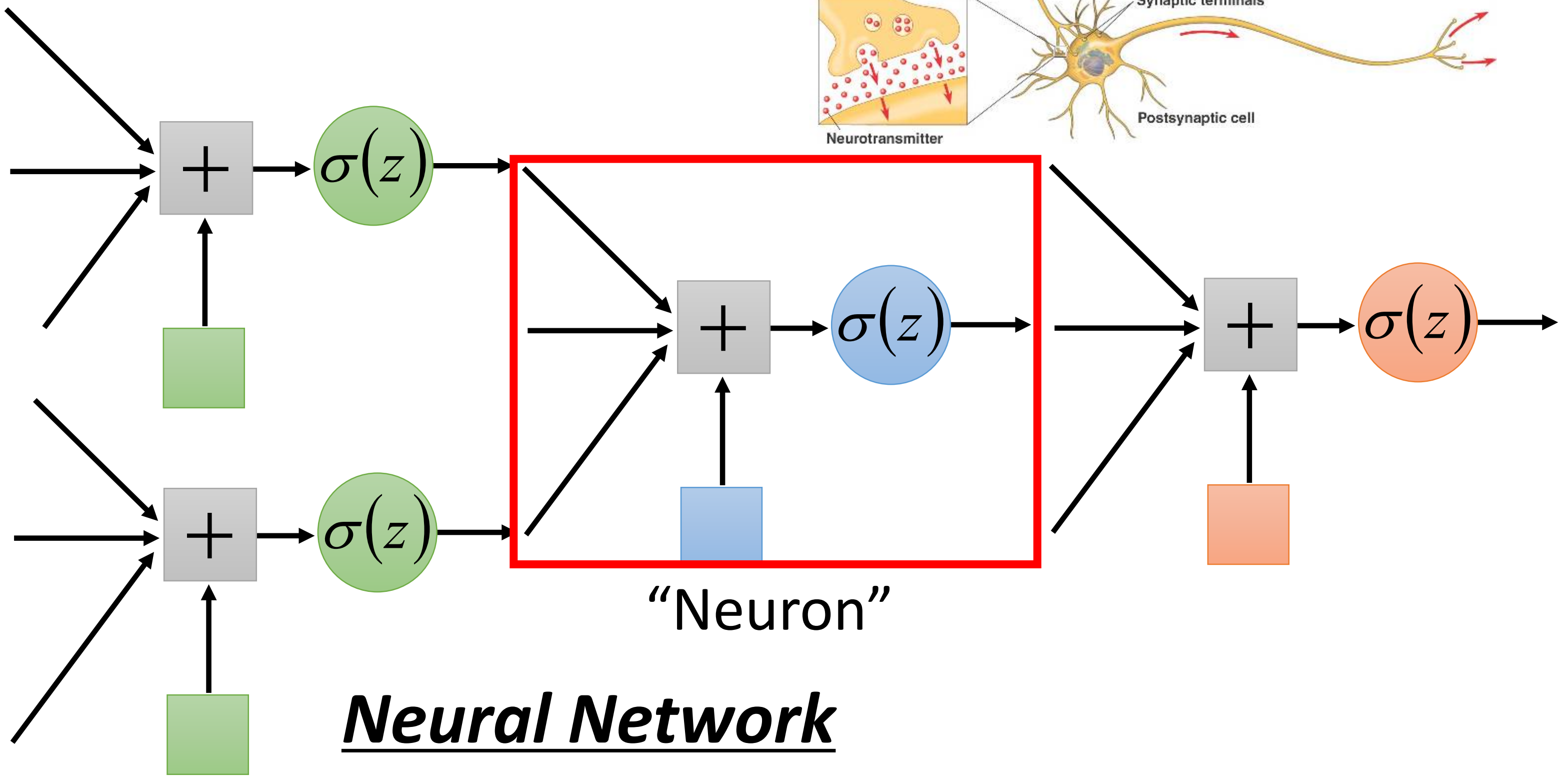
$$a_i = f_i\left(\sum_{j=1}^{R} w_{ij} x_j\right)$$

Three Steps for Neural Network

Step 1: Neural Network → Step 2: goodness of function → Step 3: pick the best function

Neural Network





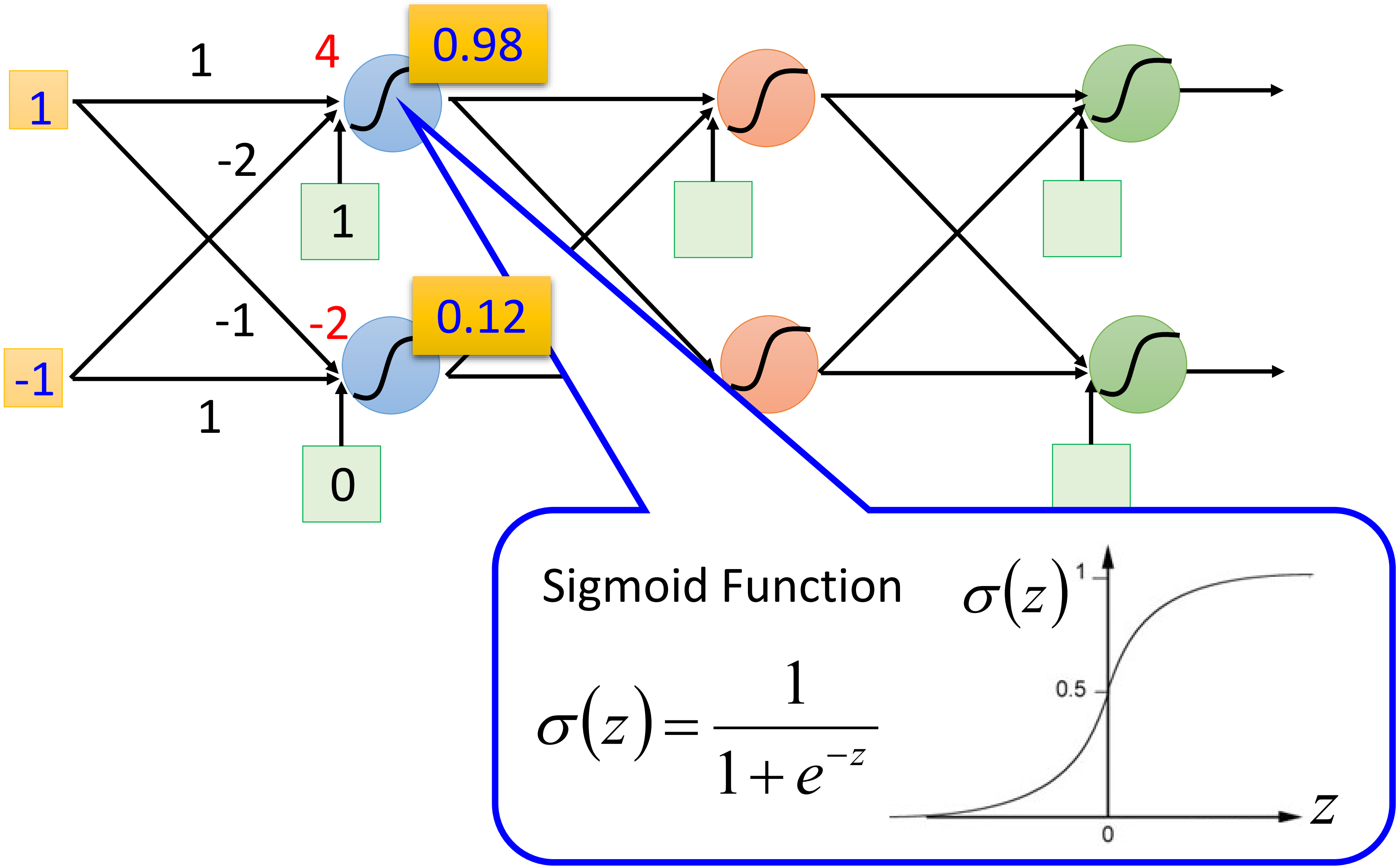
"Neuron"

Neural Network

Different connection leads to different network structures

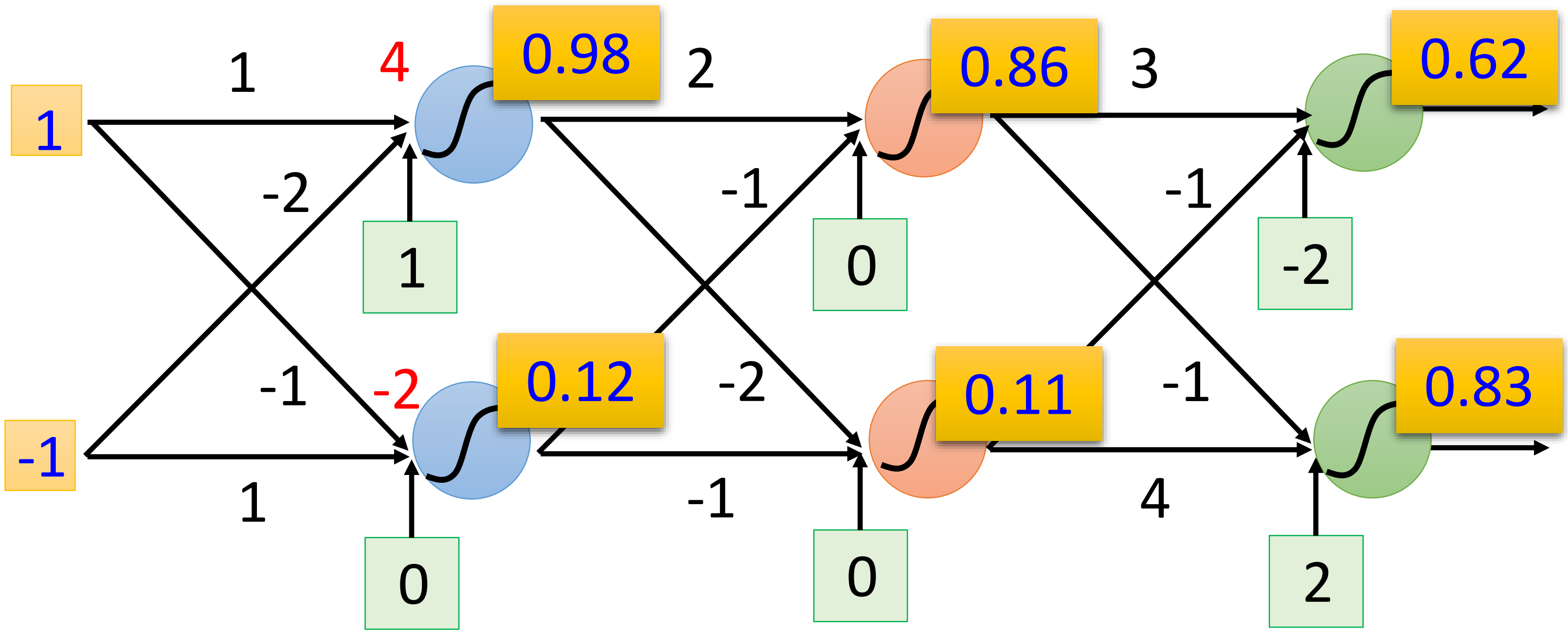
Network parameter θ: all the weights and biases in the "neurons"

Fully Connect Feedforward Network

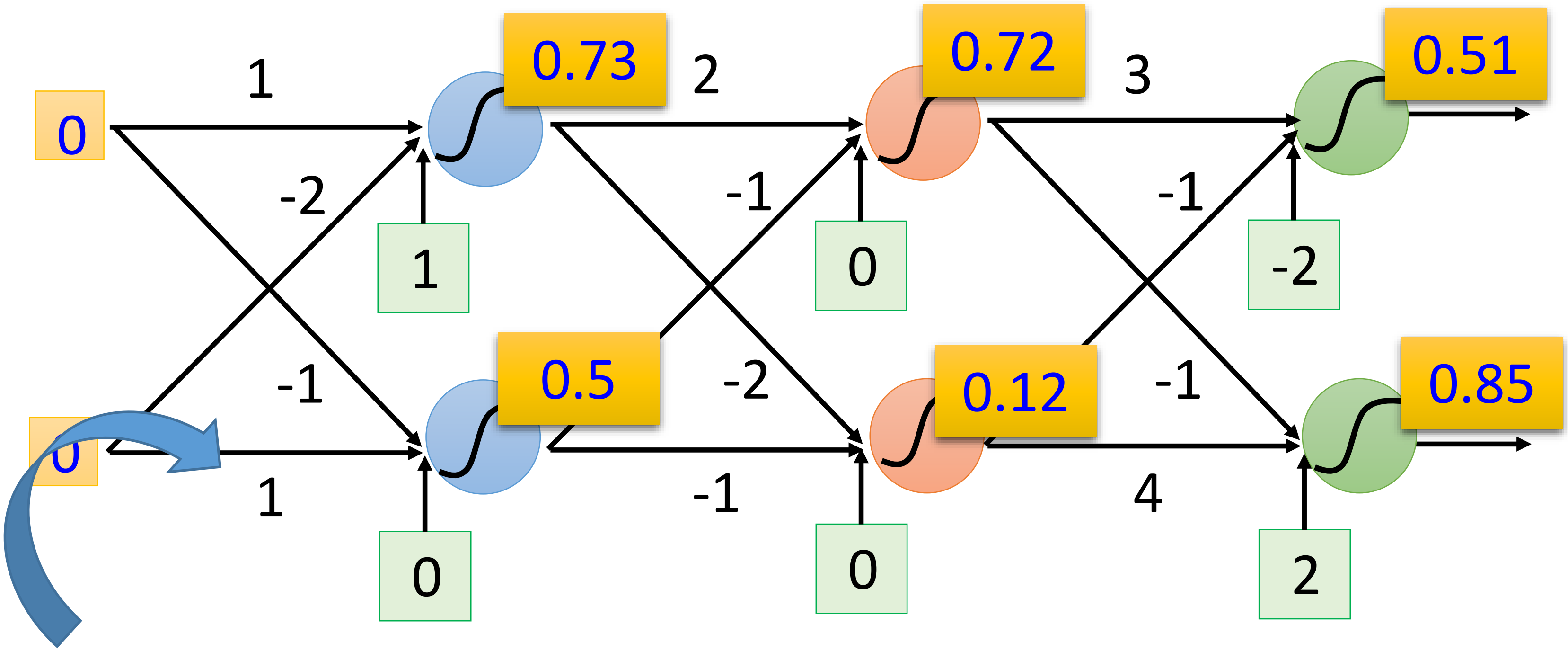


$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Fully Connect Feedforward Network



Fully Connect Feedforward Network



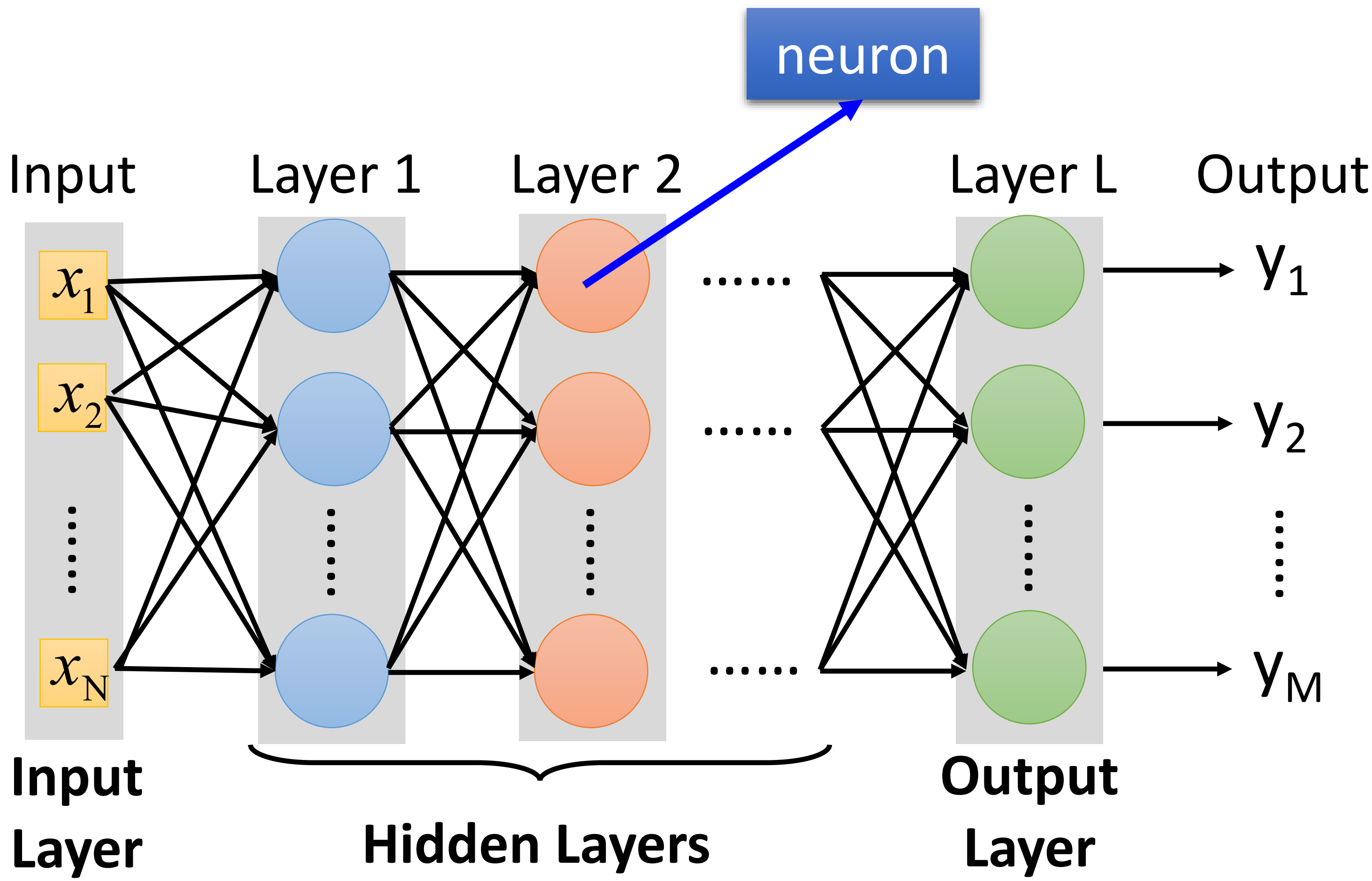
This is a function.

Input vector, output vector

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)=\begin{bmatrix}0.62\\0.83\end{bmatrix} \quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right)=\begin{bmatrix}0.51\\0.85\end{bmatrix}$$

Given network structure, define ***a function set***

Fully Connect Feedforward Network

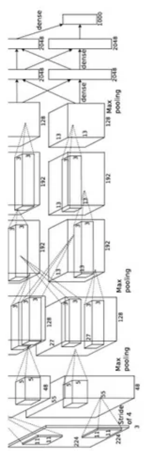


Deep = Many hidden layers

http://cs231n.stanford.edu/slides/winter1516_lecture8.pdf

8 layers

16.4%



AlexNet (2012)

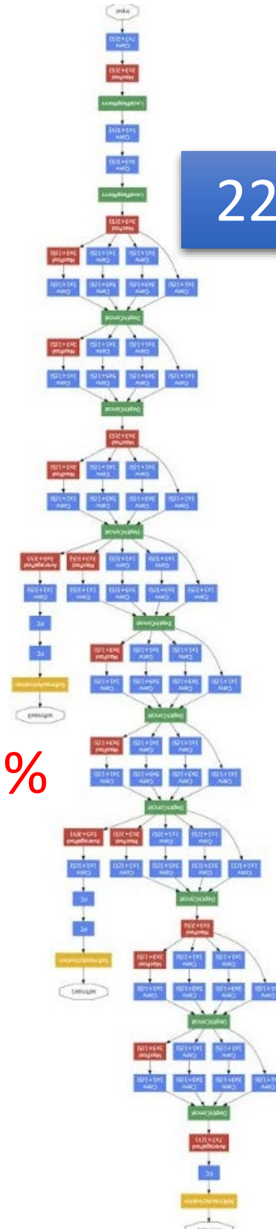
19 layers

7.3%

VGG (2014)

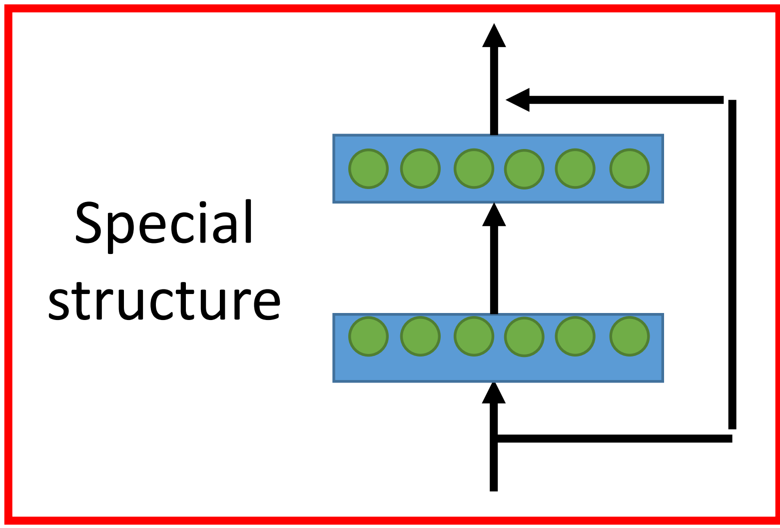
22 layers

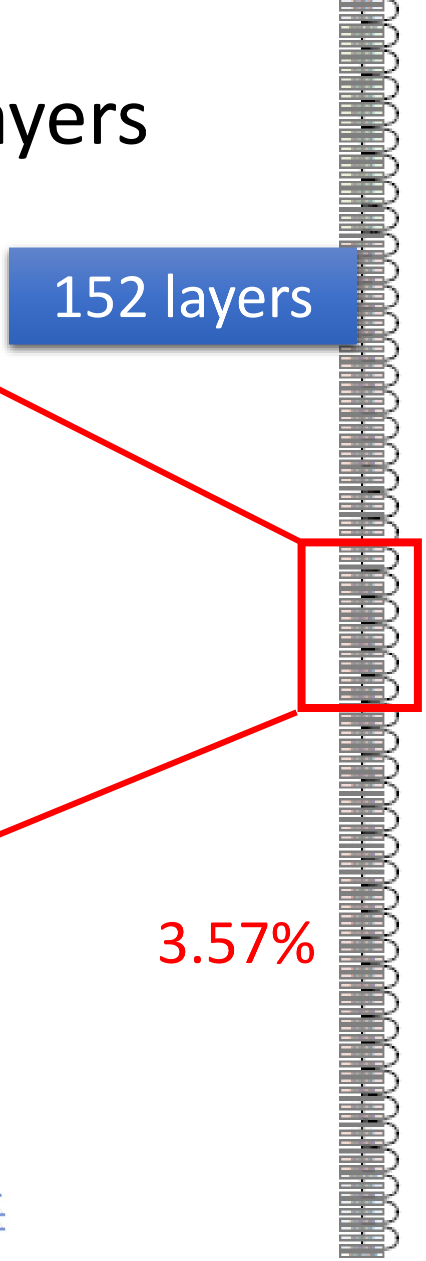
6.7%



GoogleNet (2014)

Deep = Many hidden layers





Ref: https://www.youtube.com/watch?v=dxB6299gpvI

16.4%

AlexNet (2012)

7.3%

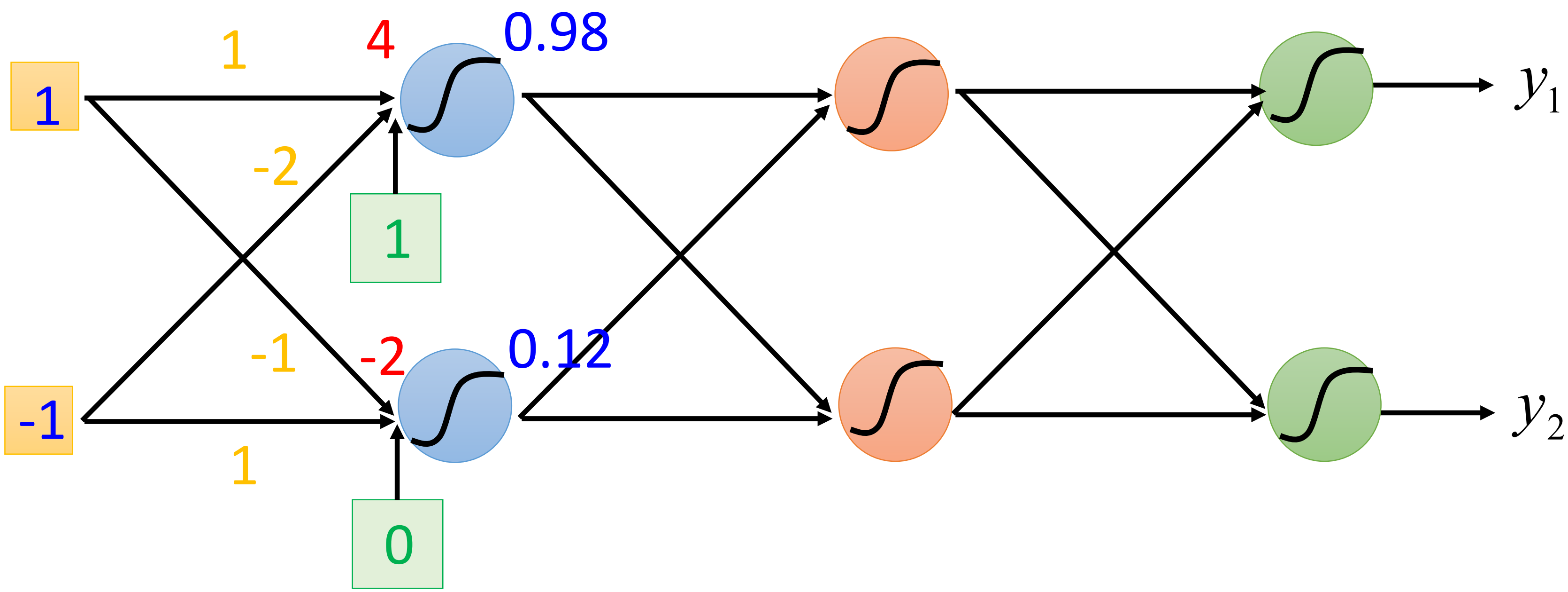
VGG (2014)

6.7%

GoogleNet (2014)

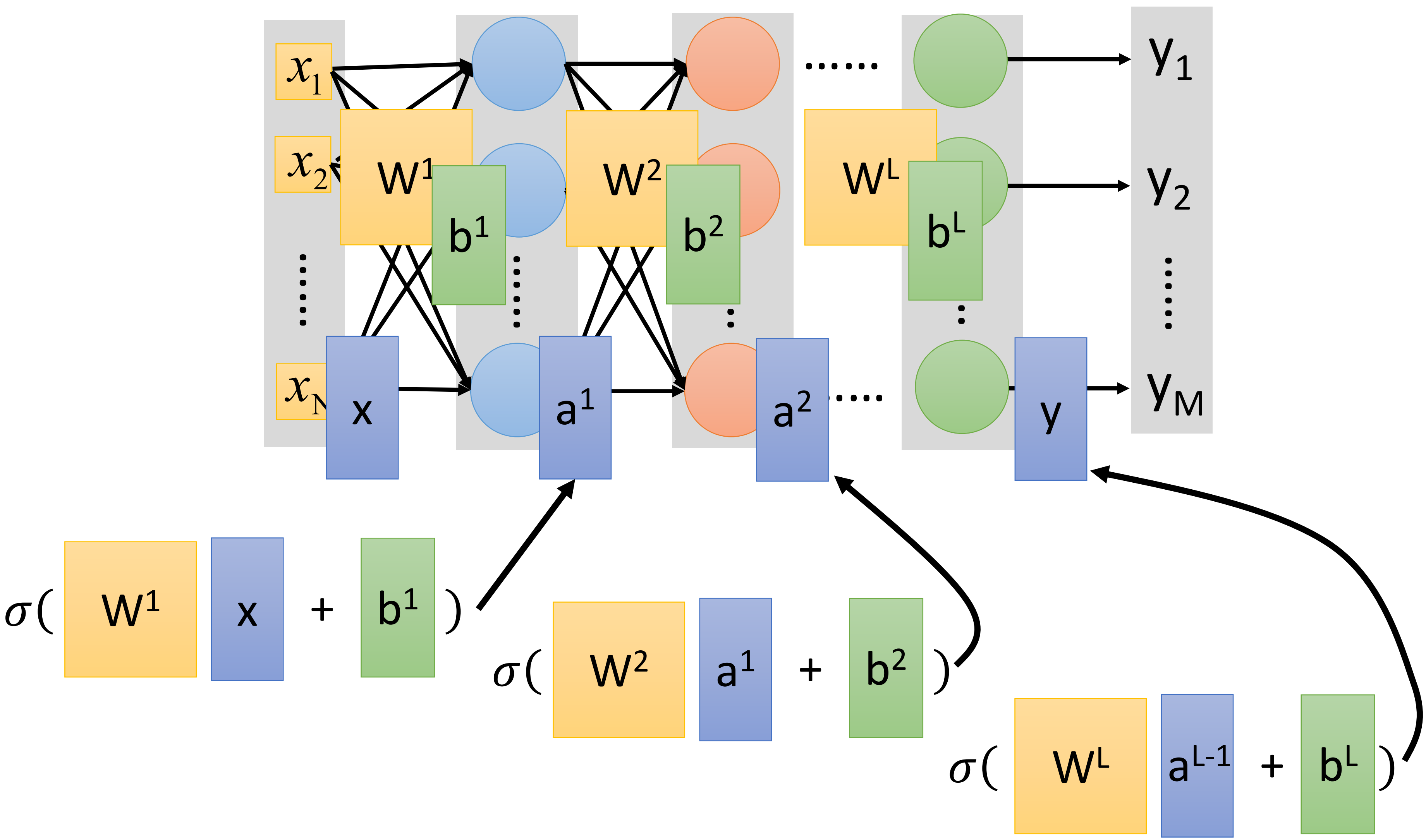
Residual Net (2015)

Matrix Operation

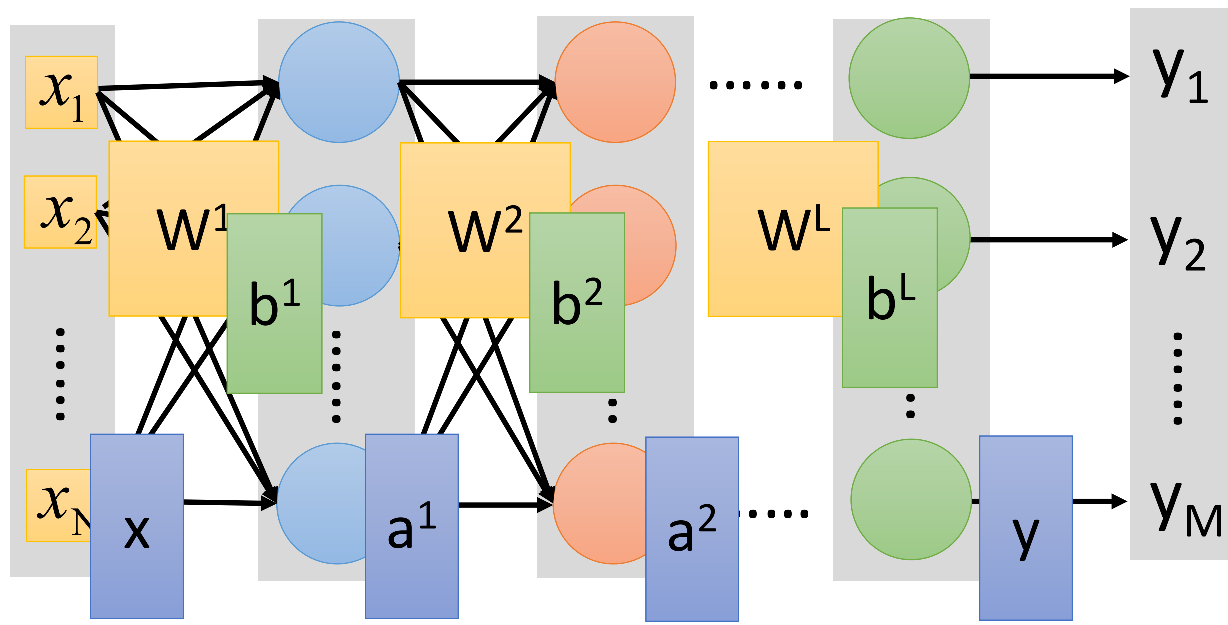


$$\sigma\Bigg(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}\Bigg) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Neural Network



Neural Network

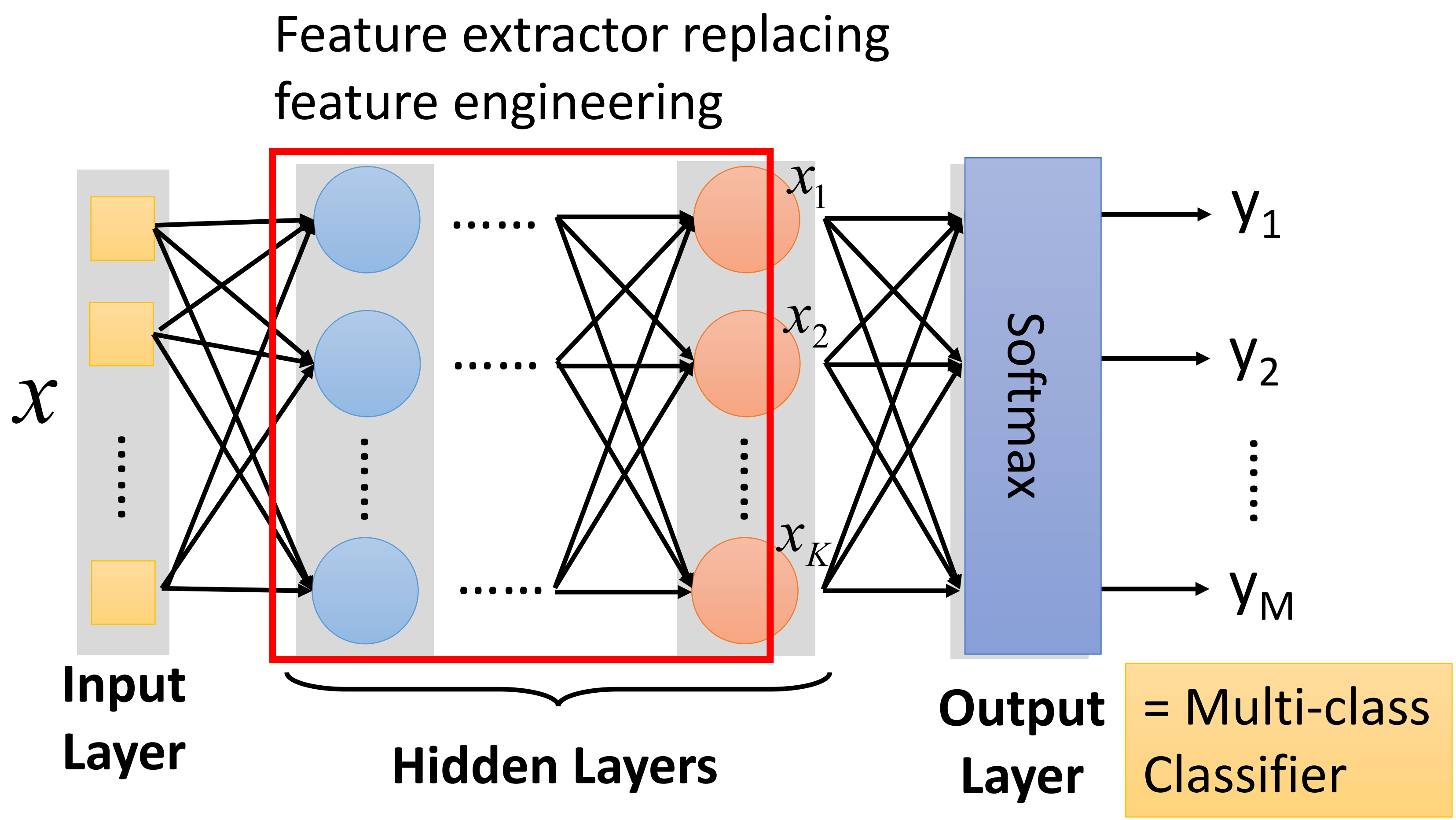


$$y = f(x)$$

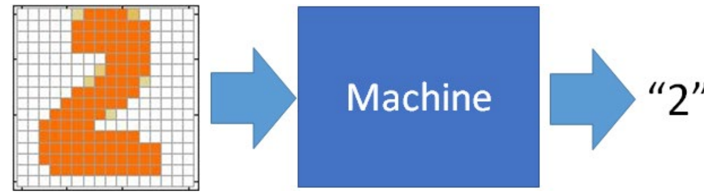
Using parallel computing techniques to speed up matrix operation

$$= \sigma\left(W^L \cdots \sigma\left(W^2 \, \sigma\left(W^1 x + b^1\right) + b^2\right) \cdots + b^L\right)$$

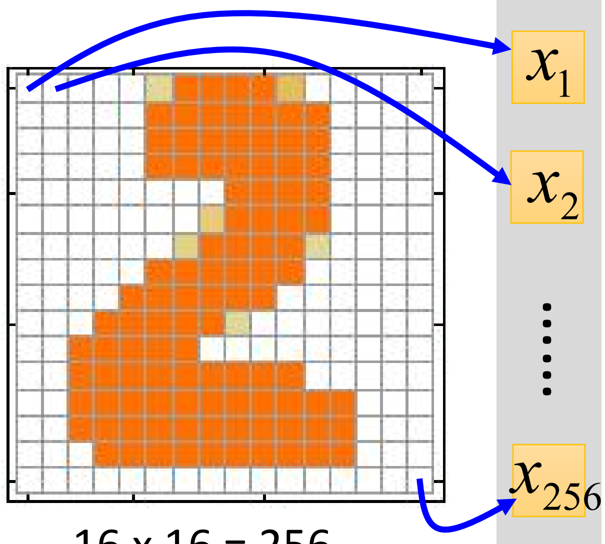
Output Layer as Multi-Class Classifier



Example Application



Input

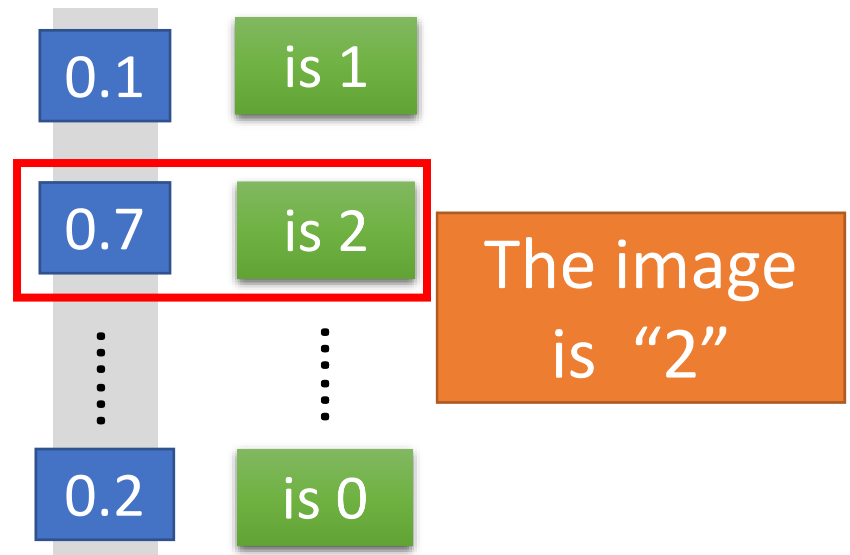


16 x 16 = 256

Ink → 1

No ink → 0

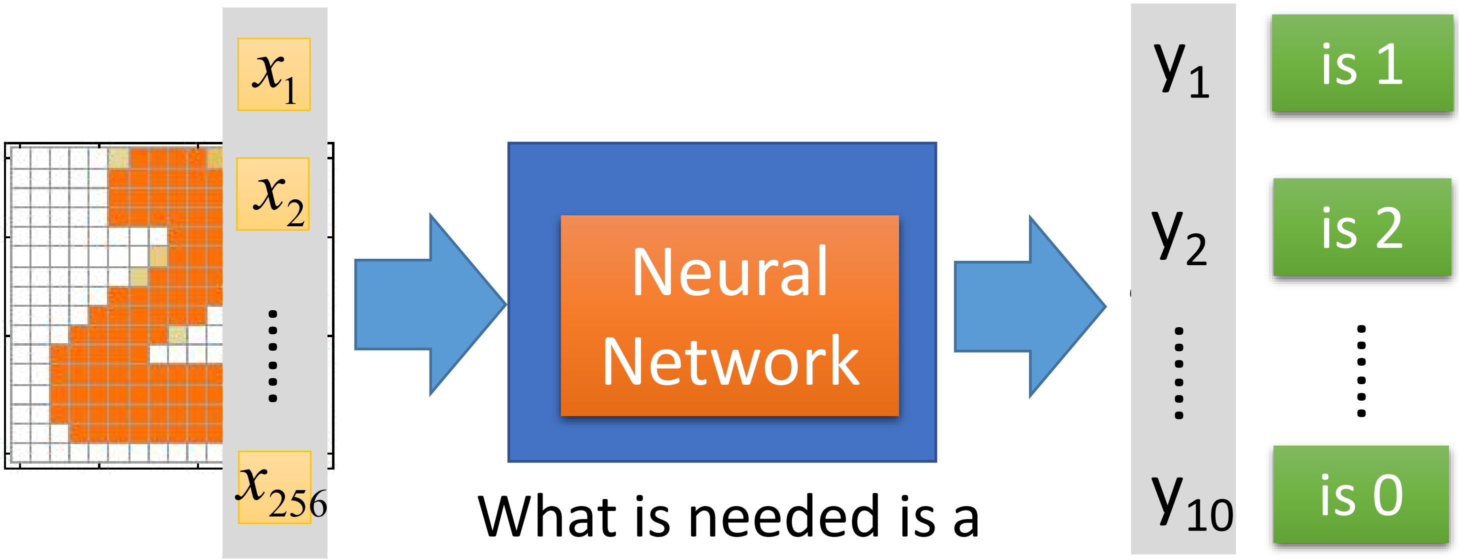
Output



Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition

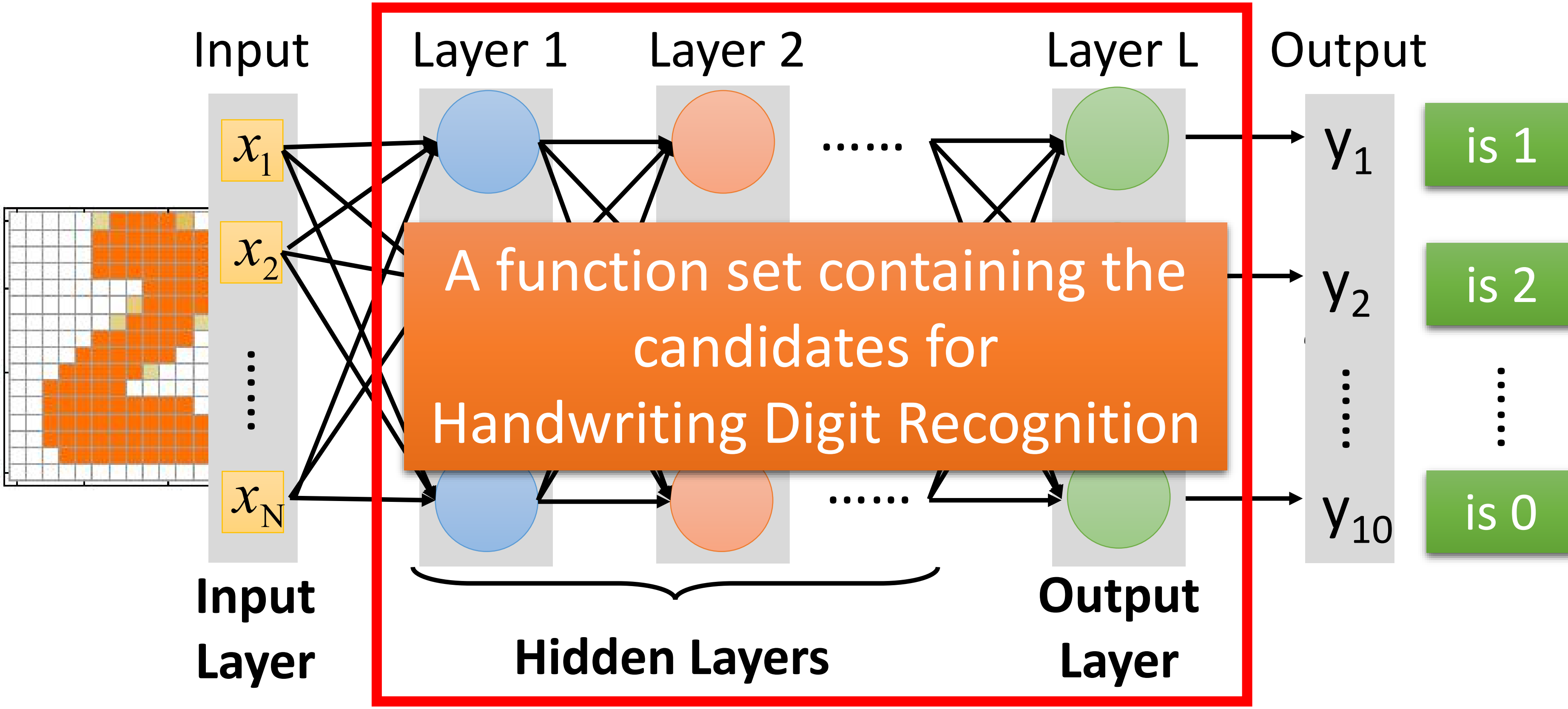


What is needed is a function ……

Input:
256-dim vector

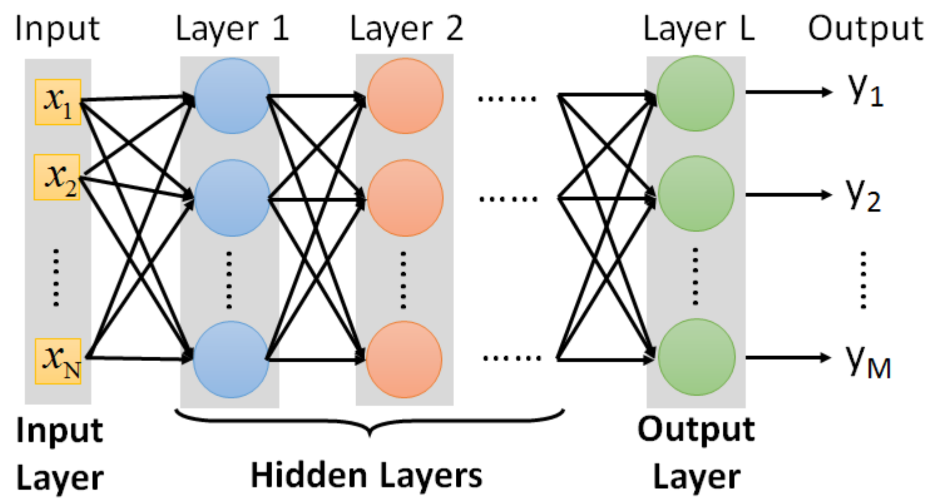
output:
10-dim vector

Example Application



You need to decide the network structure to let a good function in your function set.

FAQ



- Q: How many layers? How many neurons for each layer?

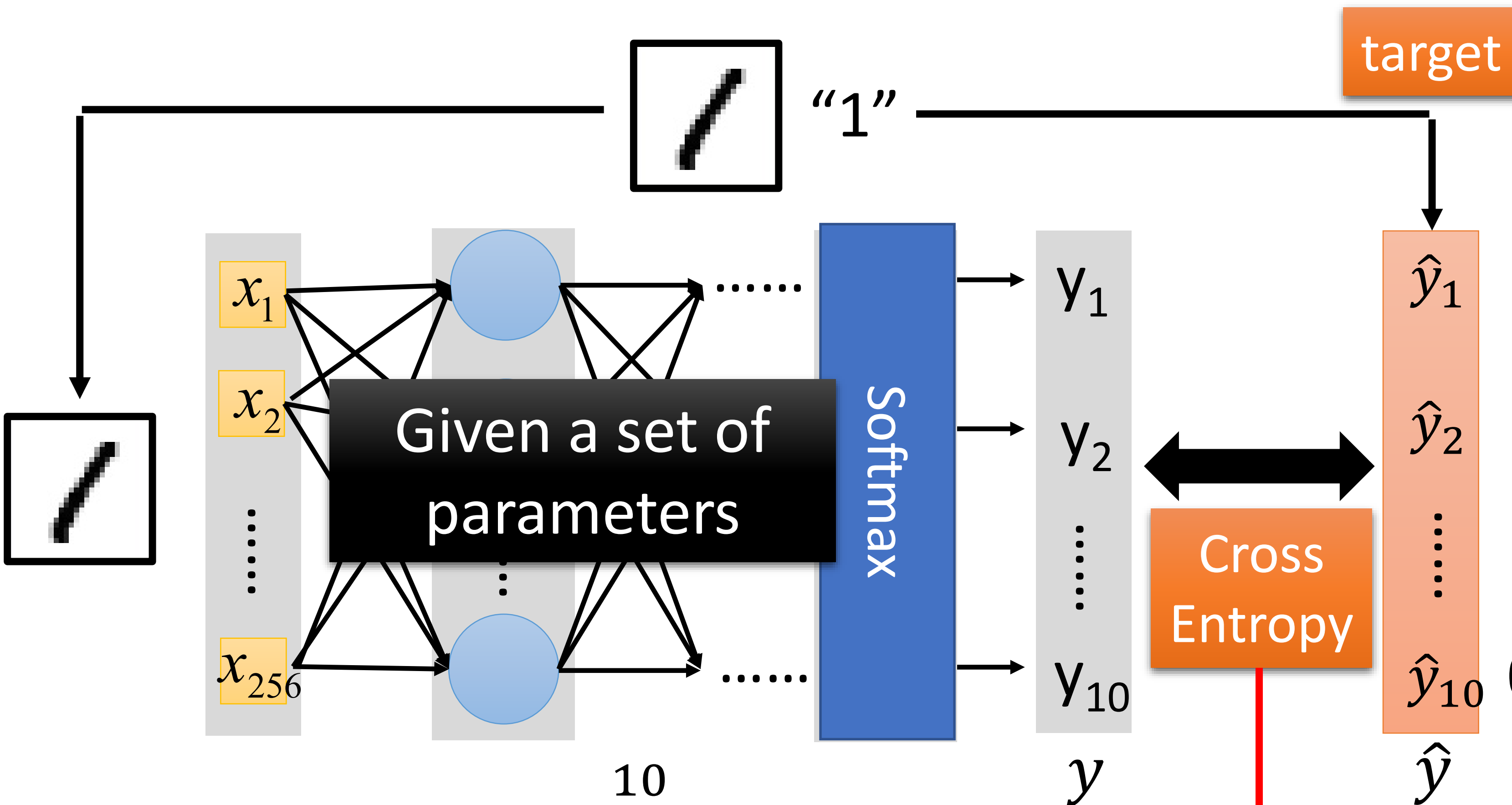
Trial and Error + Intuition

- Q: Can the structure be automatically determined?
  - E.g. Evolutionary Artificial Neural Networks
- Q: Can we design the network structure?

Convolutional Neural Network (CNN)

Three Steps for Deep Learning

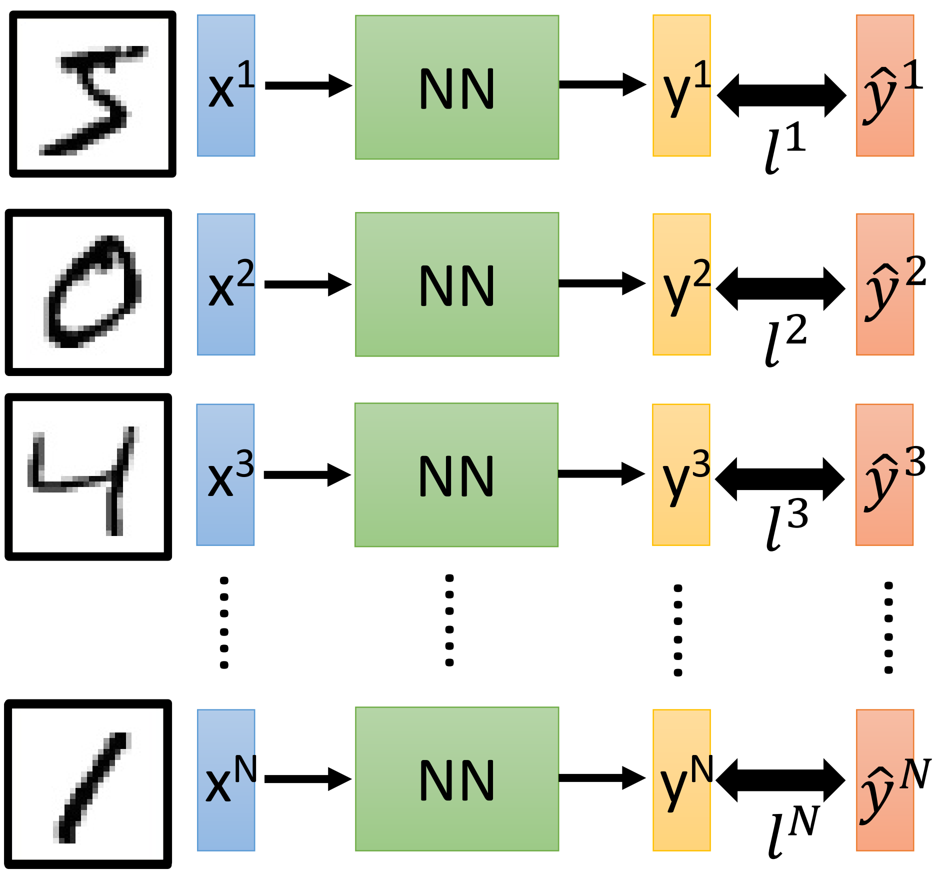
Loss for an Example



$$l(y, \hat{y}) = -\sum_{i=1}^{10} \hat{y}_i \ln y_i$$

Total Loss

For all training data …



Total Loss:

$$\boxed{L = \sum_{n=1}^{N} l^n}$$

Find ***a function in function set*** that minimizes total loss L

Find ***the network parameters θ^**** that minimize total loss L

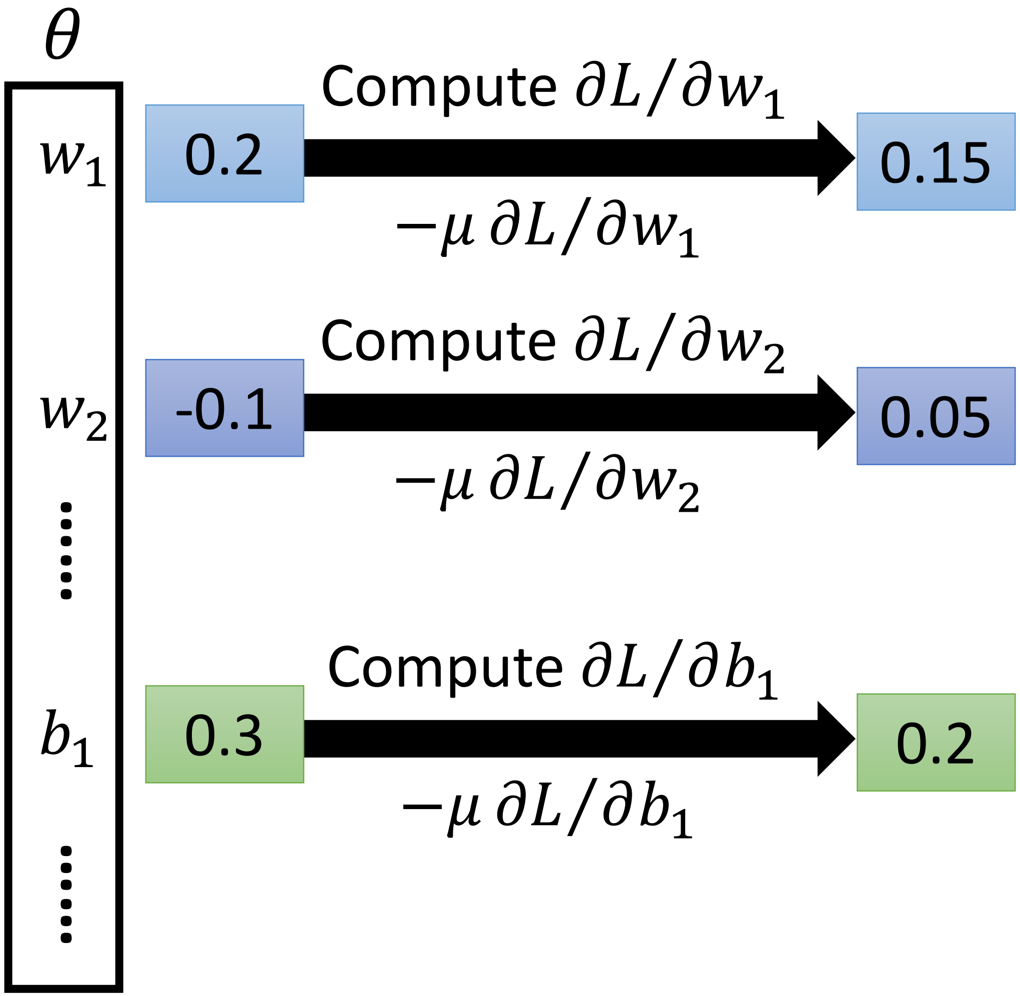
Three Steps for Deep Learning

Step 1: Neural Network

Step 2: goodness of function

Step 3: pick the best function

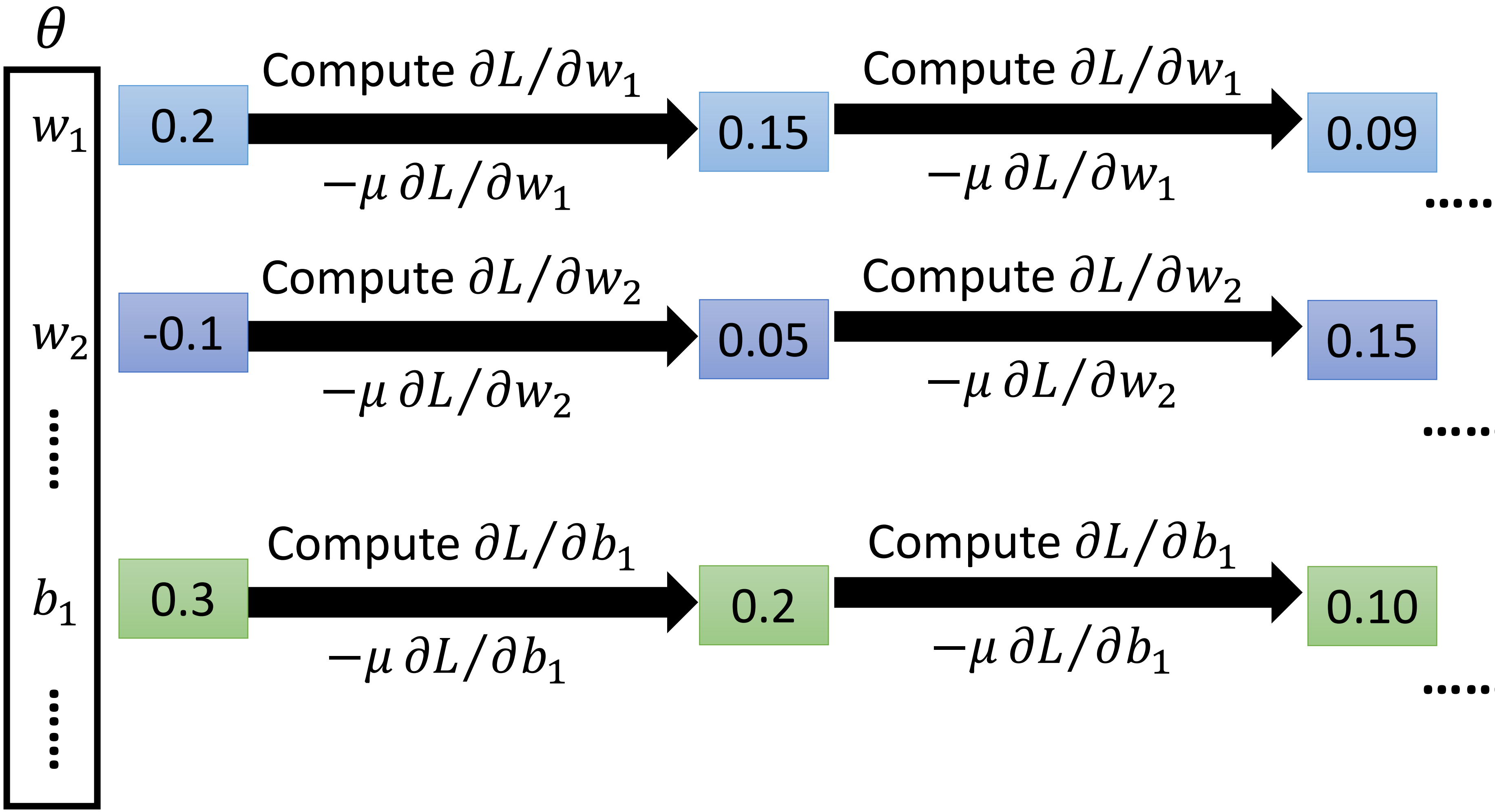
Gradient Descent



$$\nabla L = \begin{bmatrix} \dfrac{\partial L}{\partial w_1} \\ \dfrac{\partial L}{\partial w_2} \\ \vdots \\ \dfrac{\partial L}{\partial b_1} \\ \vdots \end{bmatrix}$$

gradient

Gradient Descent

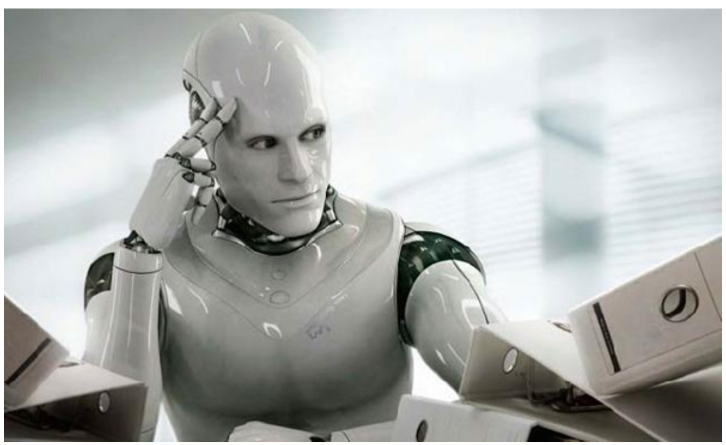


Gradient Descent

This is the “learning” of machines in deep learning ......

Even alpha go using this approach.

People image ......



Actually .....

I hope you are not too disappointed :p

Backpropagation

- Backpropagation: an efficient way to compute $\partial L / \partial w$ in neural network

Three Steps for Deep Learning

感知机

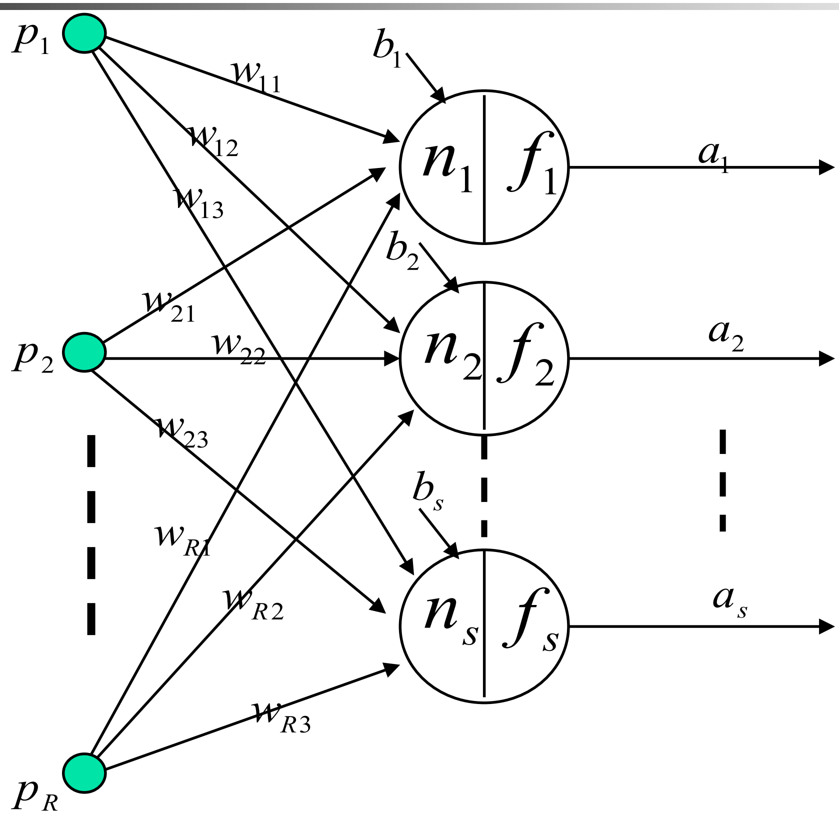
1.感知机的网络结构

2.例子

3.感知机的训练规则

4.感知机的局限

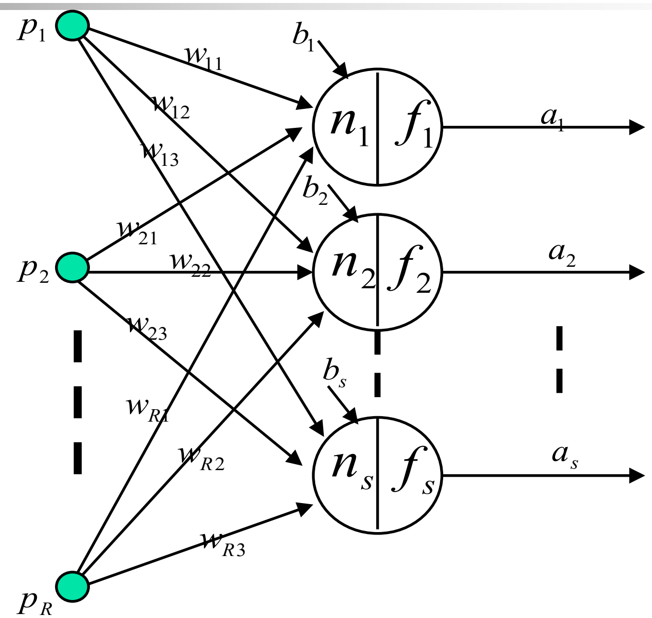
Single Layer FNNs

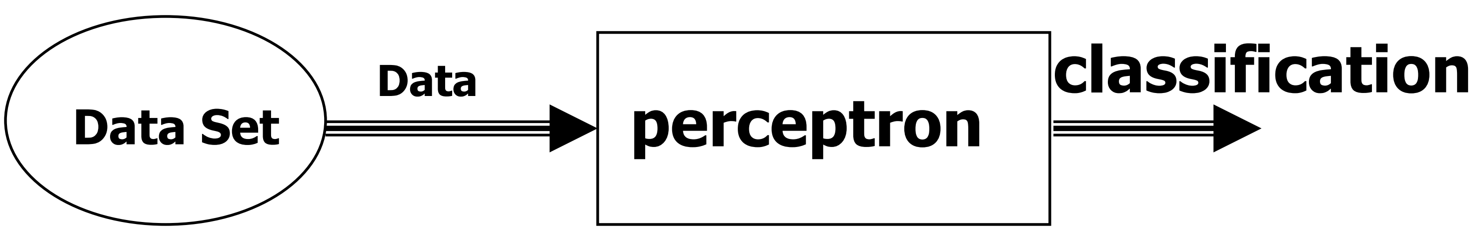


How to mimic intelligent behaviors?

Perceptron

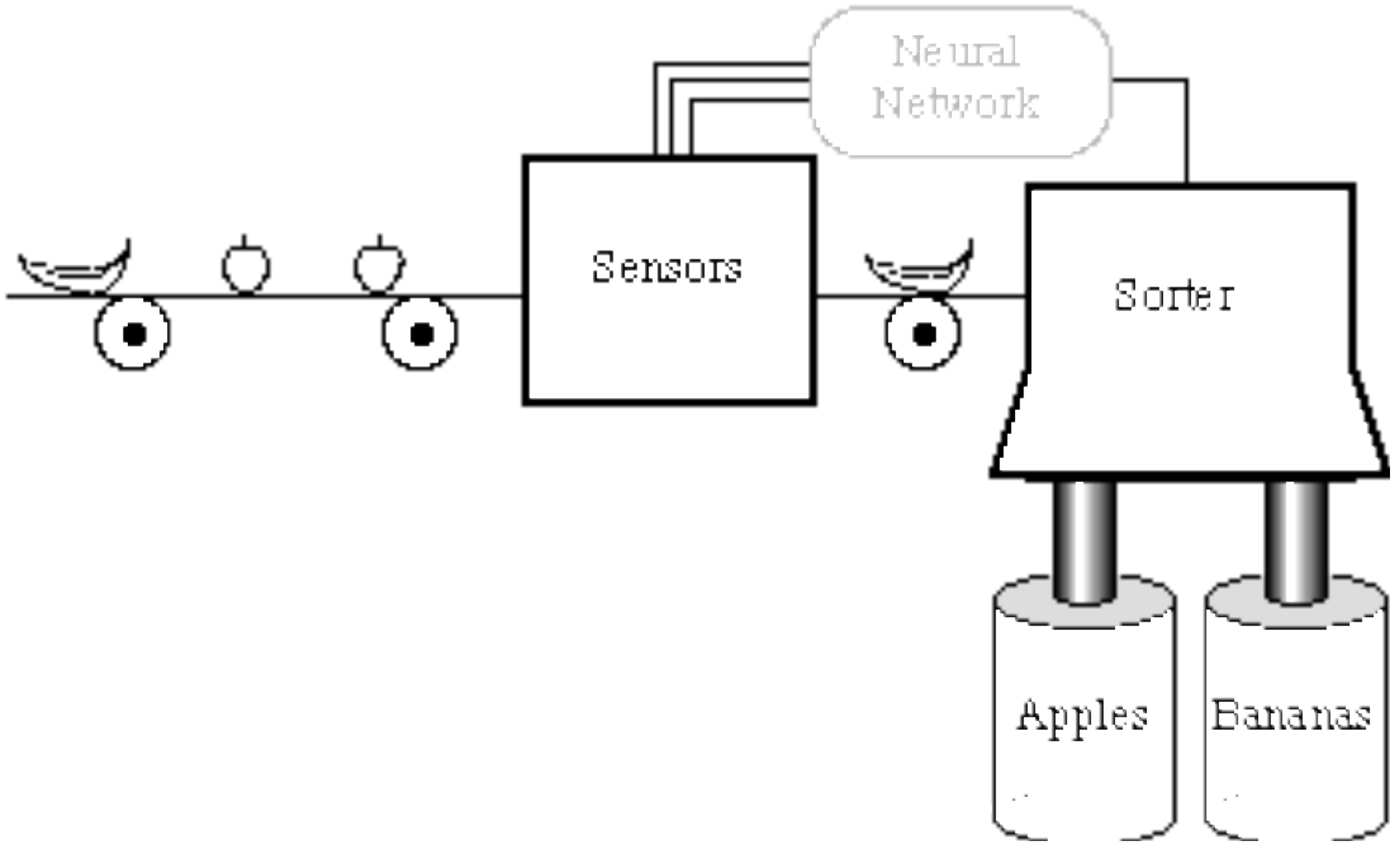
Data Classification





An Intuitive Example

Apple/Banana Classifier



Apple/Banana Classifier

1. 特征描述（Measurement Vector）
2. NN结构设计
3. NN学习，以具备分类能力
4. NN性能分析

Prototype Vectors

Banana **Apple**

Measurement Vector

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

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \qquad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Perceptron → **1**

perceptron → $-$**1**

Shape: {1 : round ; -1 : eliptical}

Texture: {1 : smooth ; -1 : rough}

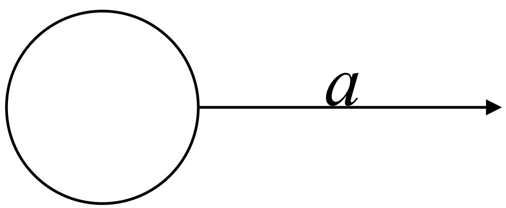
Weight: {1 : > 1 lb. ; -1 : < 1 lb.}

Apple/Banana Classifier

1. 特征描述（Measurement Vector）
2. NN结构设计
3. NN学习，以具备分类能力
4. NN性能分析

选择网络结构：多少神经元？多少层网络？

Perceptron Design



Banana

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

Perceptron

1

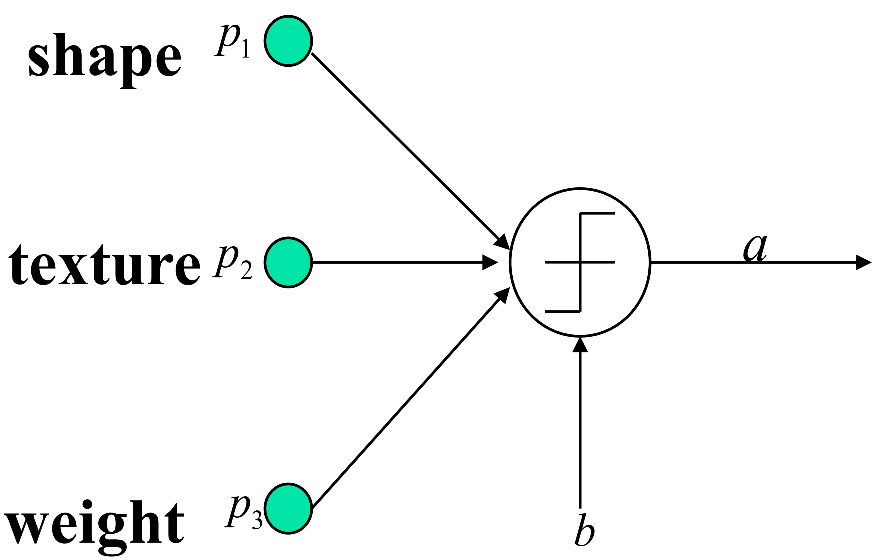
Apple

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

perceptron

−1

Perceptron Design



Banana

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

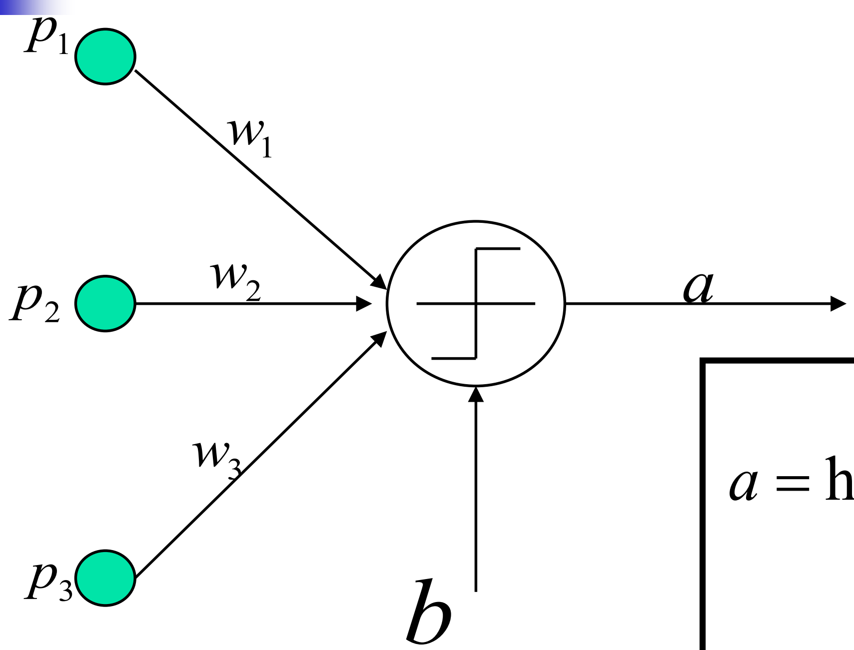
Perceptron $\Downarrow$ **1**

Apple

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

perceptron $\Downarrow$ **−1**

Perceptron Design



$$a = \text{hardlims}(wp + b)$$

$$= \text{hardlims}\left(\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b \right)$$

Perceptron Design

$$a = \text{hardlims}(wp + b)$$

$$= \text{hardlims}\left(\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b \right)$$

Decision plan

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b = 0$$

Decision Plan

Banana **Apple**

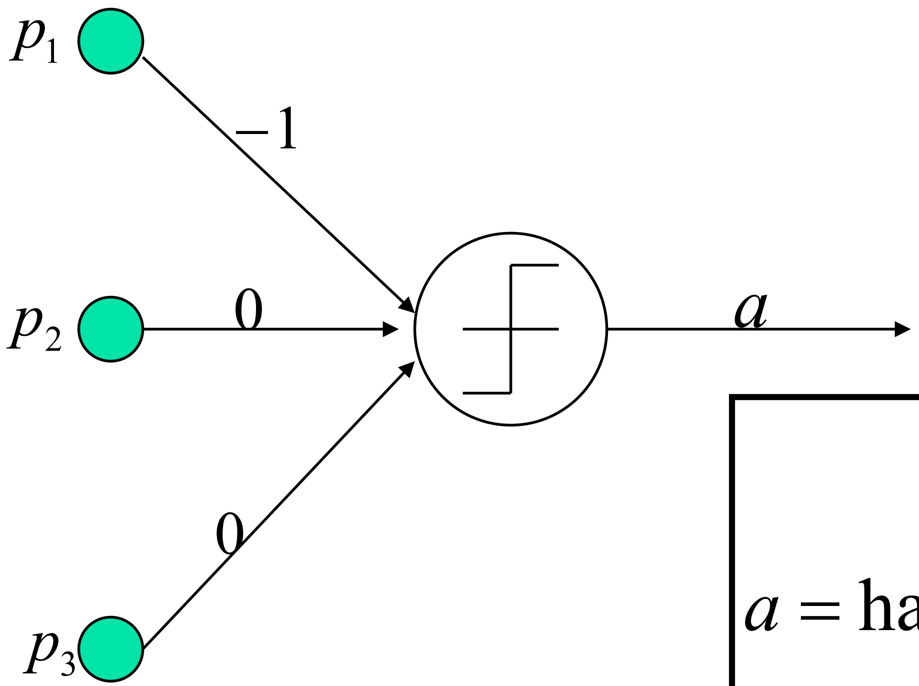
$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \qquad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

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

$$p_1 = 0$$

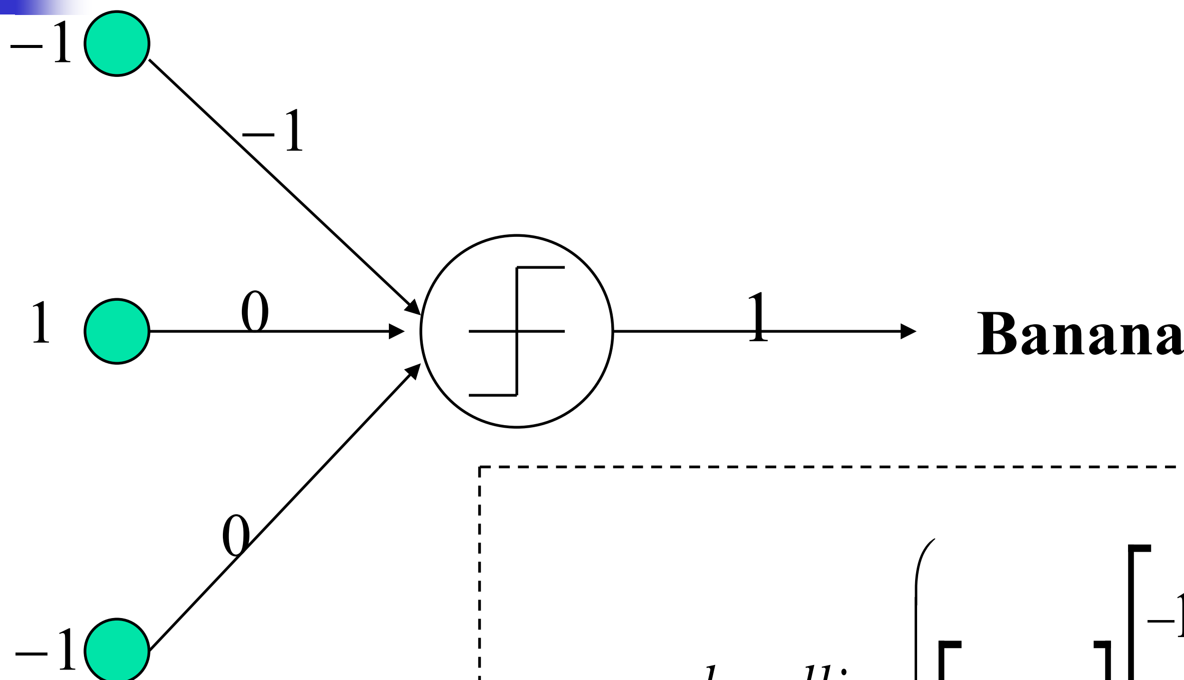
$$\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0$$

Perceptron Design



$$a = \text{hardlims}\left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \right)$$

Check



Banana

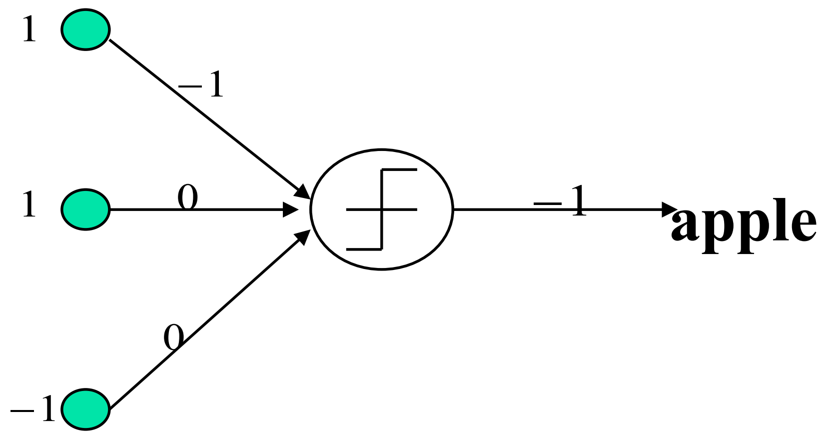
$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

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

Check

Apple

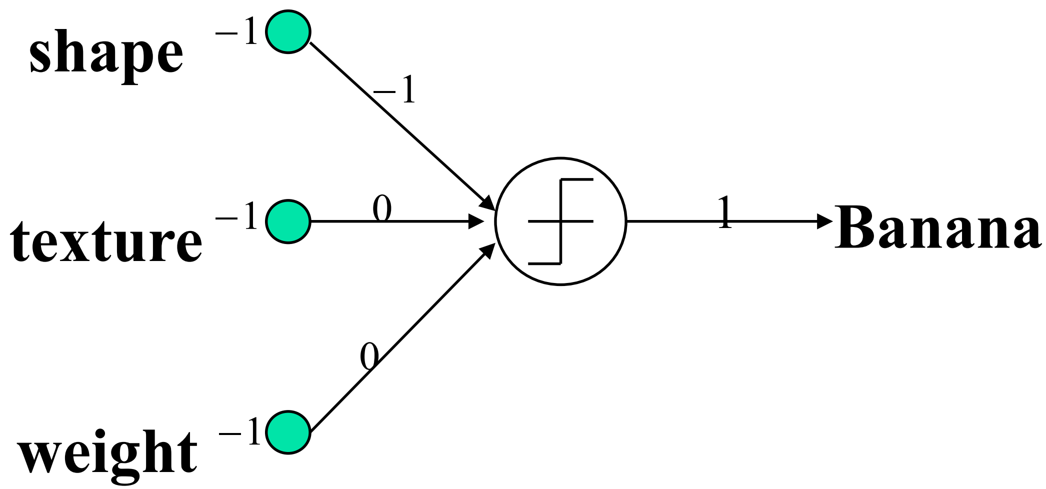
$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$



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

Check

“Rough” Banana:



Banana

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

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

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:

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

Perceptron

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

$$p_1 = 0$$

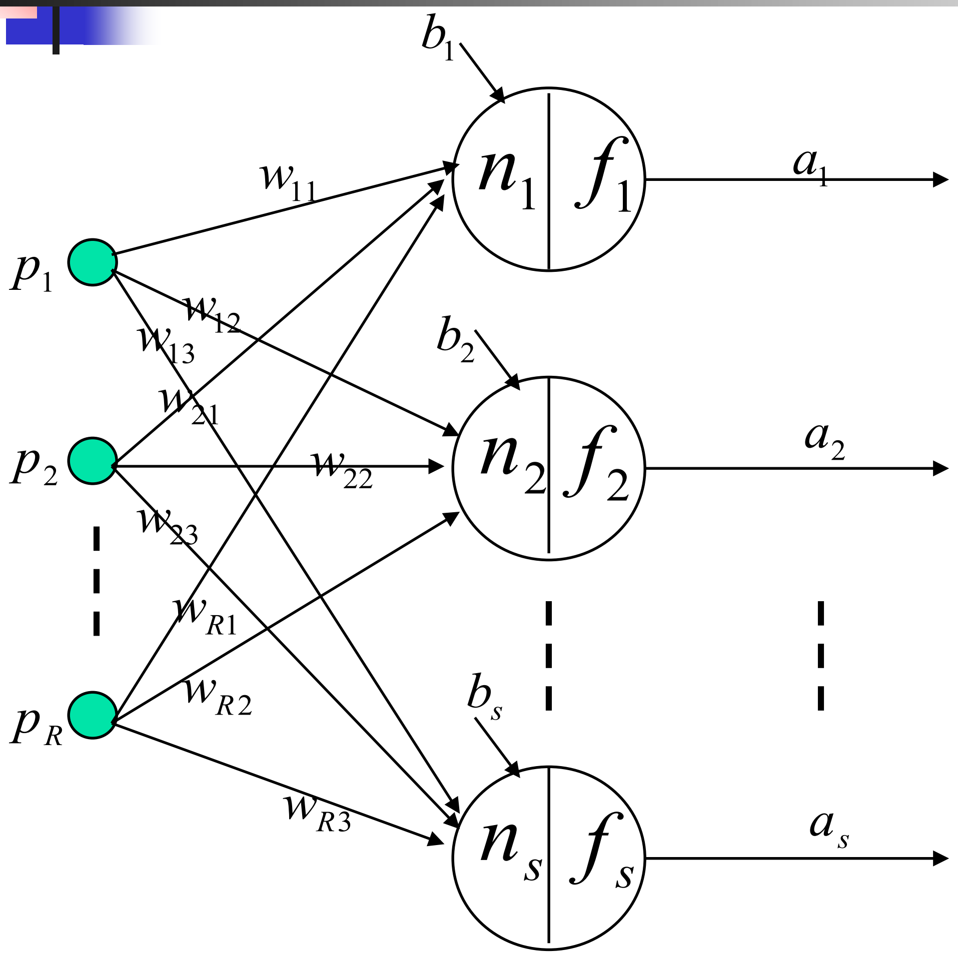
$$\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0$$

How to adjust W by learning?

Perceptron – Part Two

Perceptron Learning

Perceptron



Math Model

$$a_i = f_i\left(\sum_{j=1}^{R} w_{ij} p_j + b_i\right)$$

$$n_i = \sum_{j=1}^{R} w_{ij} p_j + b_i$$

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

Learning Rules

target

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

input

- **Supervised Learning**
  - **- Network is provided with a set of examples of proper network behavior (inputs/targets)**

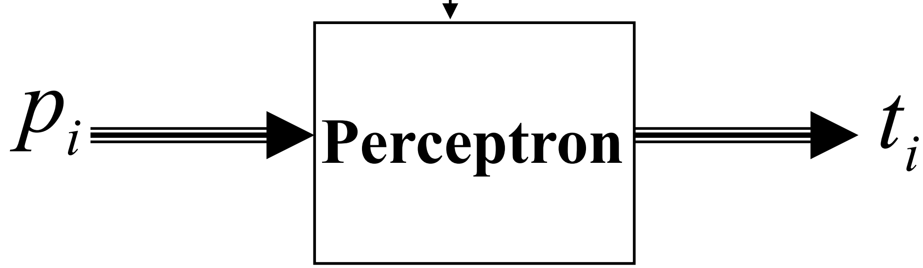
Learning Rules

target

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

input

Learning aim: adjusting connection weights



Decision Plan

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

为了获得决策面，令净输入为0：

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

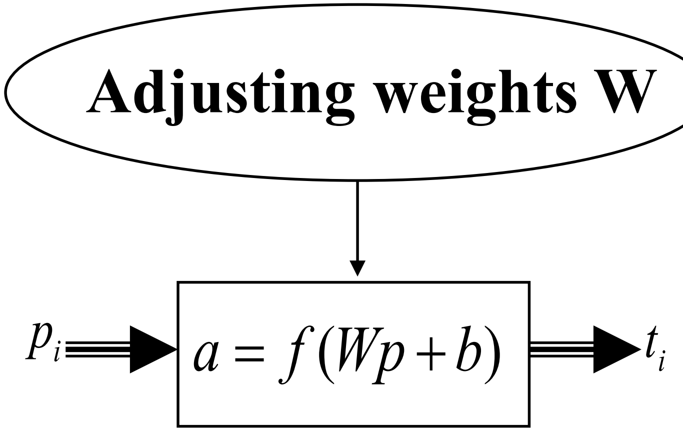
决策面的法向量即W

Learning Rules

target

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

input

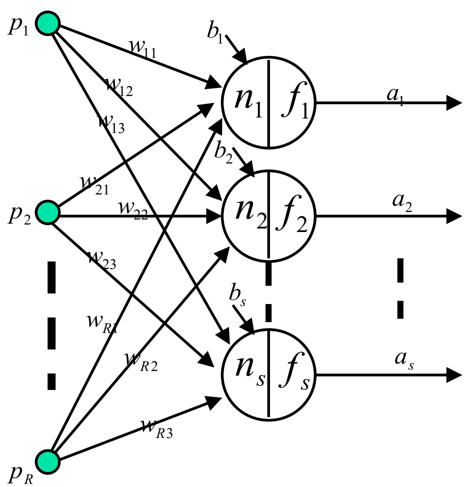


Learning Rules

target

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

input

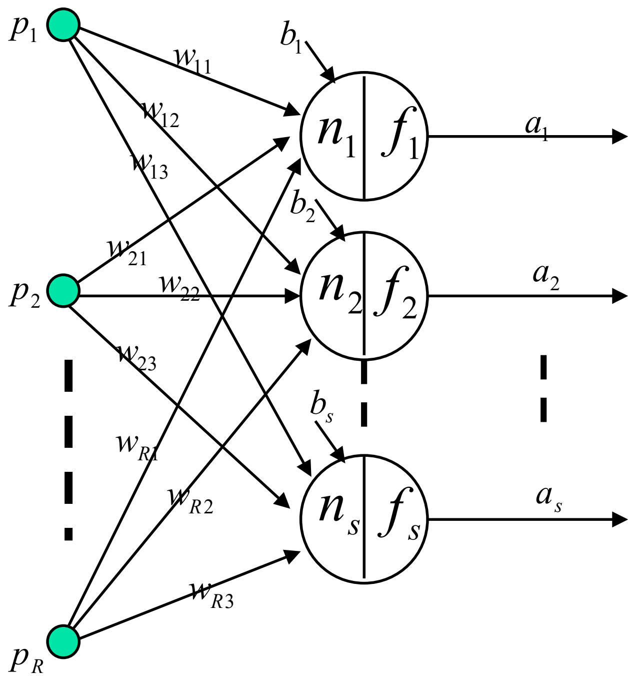


Question : How to adjust W?

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

Learning Rules

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



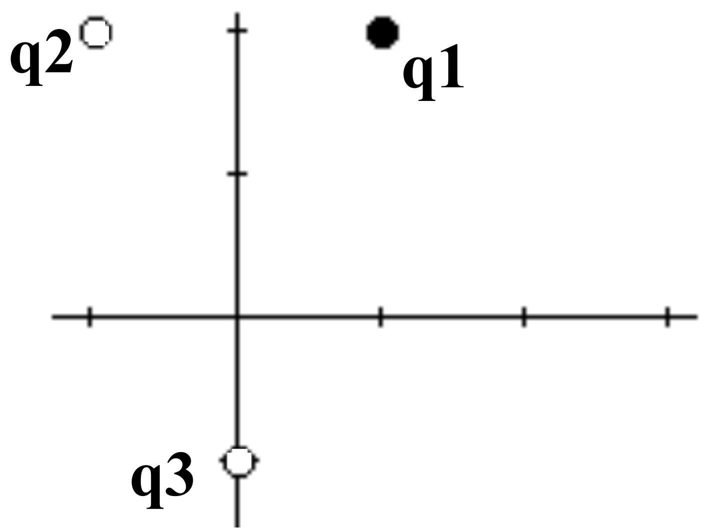
$$a = f(Wp + b)$$

$$E(W) = \sum_{i=1}^{Q} (t_i - a_i)^2$$

$$= \sum_{i=1}^{Q} [t_i - f(Wp_i + b)]^2$$

Learning Rule

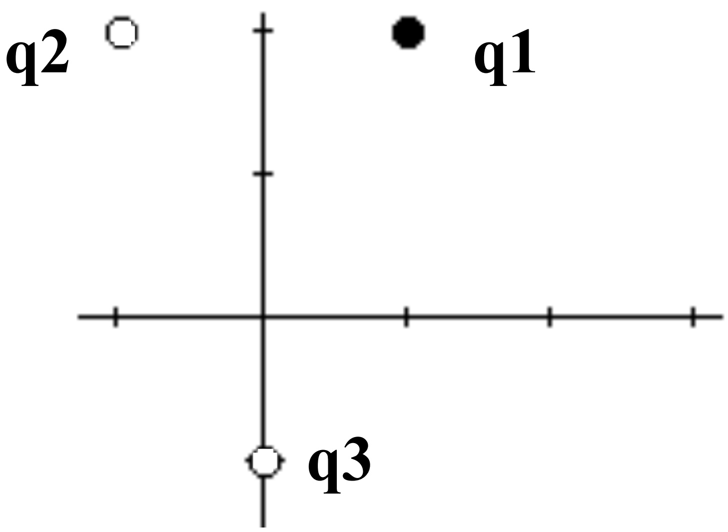
$$\left\{ q_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \quad \left\{ q_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \quad \left\{ q_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, t_3 = 0 \right\}$$

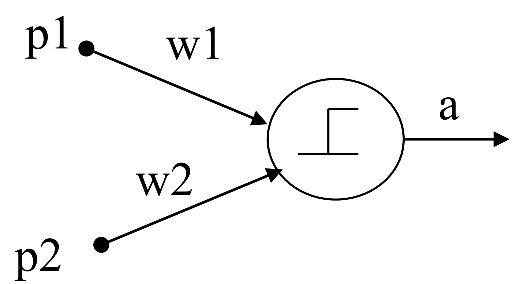


Classification:
2 classes

Learning Rule

$$\left\{ q_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \quad \left\{ q_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \quad \left\{ q_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, t_3 = 0 \right\}$$





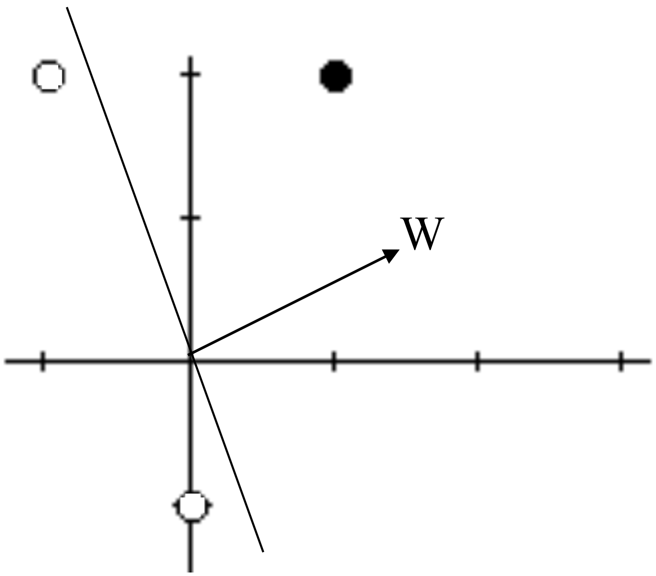
$$a = f\left(w^T p\right) = f\left(w_1 p_1 + w_2 p_2\right)$$

Learning Rule

$$\left\{ q_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \quad \left\{ q_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \quad \left\{ q_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, t_3 = 0 \right\}$$

$$a = f\left(w^T p\right) = f\left(w_1 p_1 + w_2 p_2\right)$$

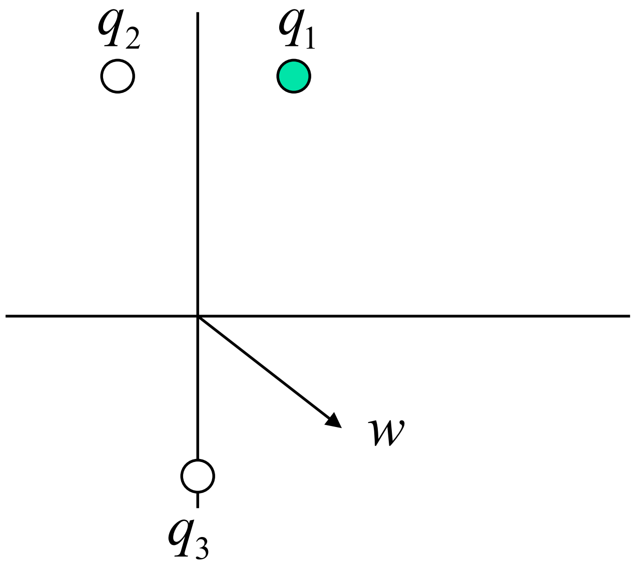
$$w_1 p_1 + w_2 p_2 = 0$$



Starting Point

Random initial weight:

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



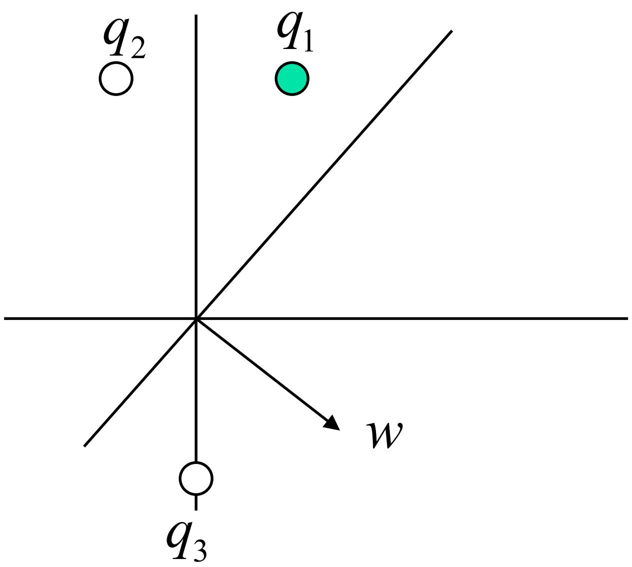
Starting Point

Random initial weight:

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

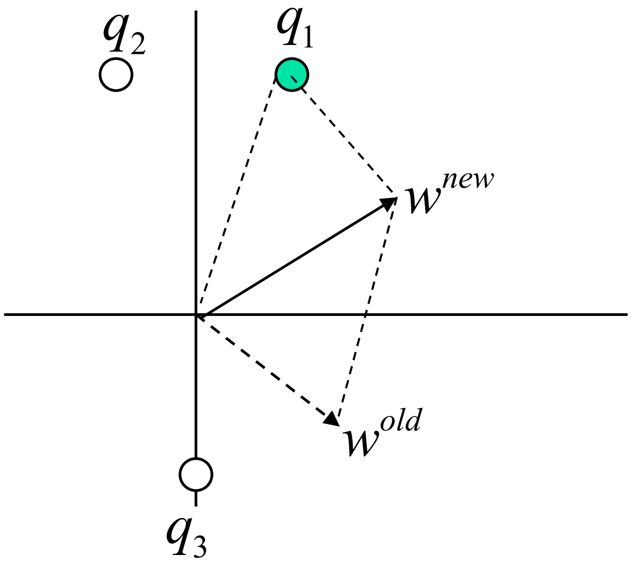
Present q_1 to the network: $\left\{ q_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\}$

$$a = f\left(w^T q_1\right) = f\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) = f(-0.6) = 0$$



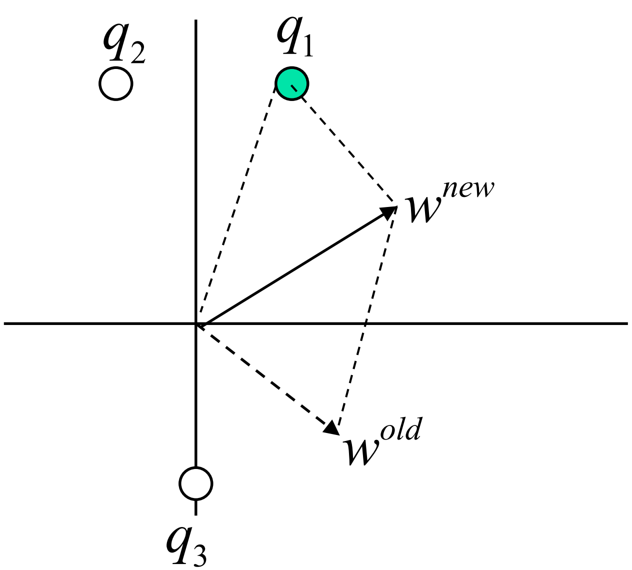
Incorrect Classification !

Adjusting W



First Iteration

$$w^{new} = w^{old} + q_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$

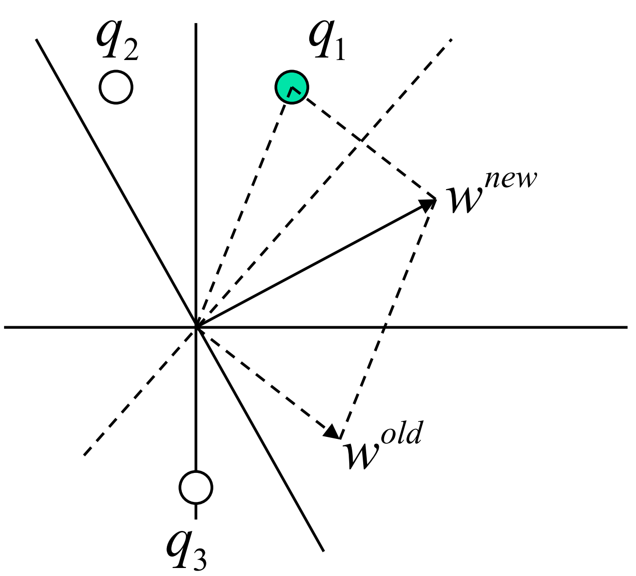


Learning Rule:

If $t = 1$ and $a = 0$, then $w^{new} = w^{old} + q$

First iteration

$$w^{new} = w^{old} + q_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$

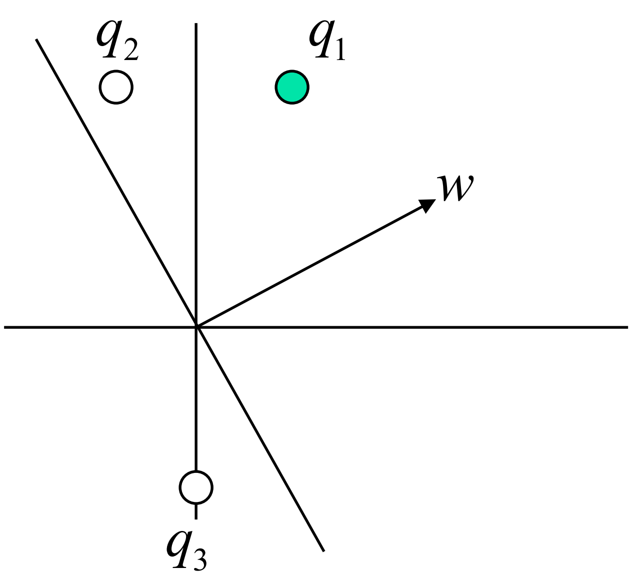


Learning Rule:

If $t = 1$ and $a = 0$, then $w^{new} = w^{old} + q$

First Iteration

$$w^{new} = w^{old} + q_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$



Learning Rule:

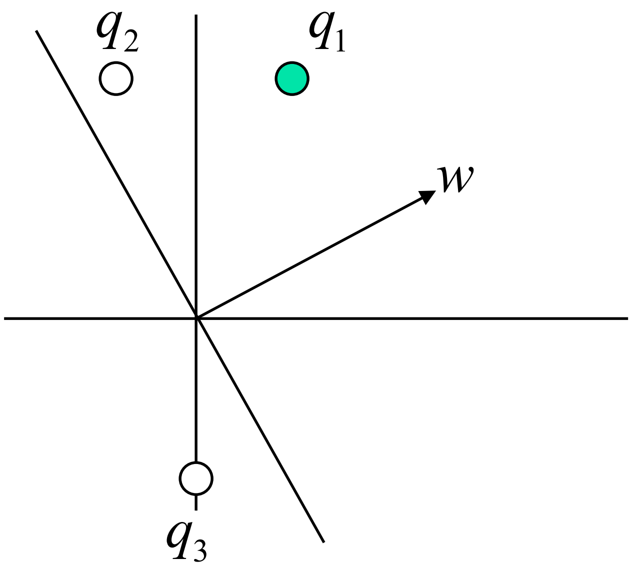
If $t = 1$ and $a = 0$, then $w^{new} = w^{old} + q$

Checking

$$w = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$

Present q_2 to the network: $\left\{ \; q_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \; \right\}$

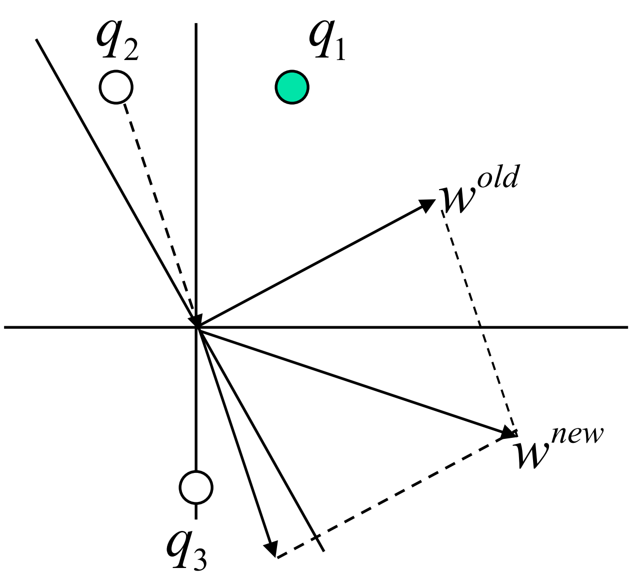
$$a = f\left(w^T q_2\right) = f\left(\begin{bmatrix} 2.0 & 1.2 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix} \right) = f\left(0.4\right) = 1$$



Incorrect Classification !

Second Iteration

$$w^{new} = w^{old} - q_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$



Learning Rule:

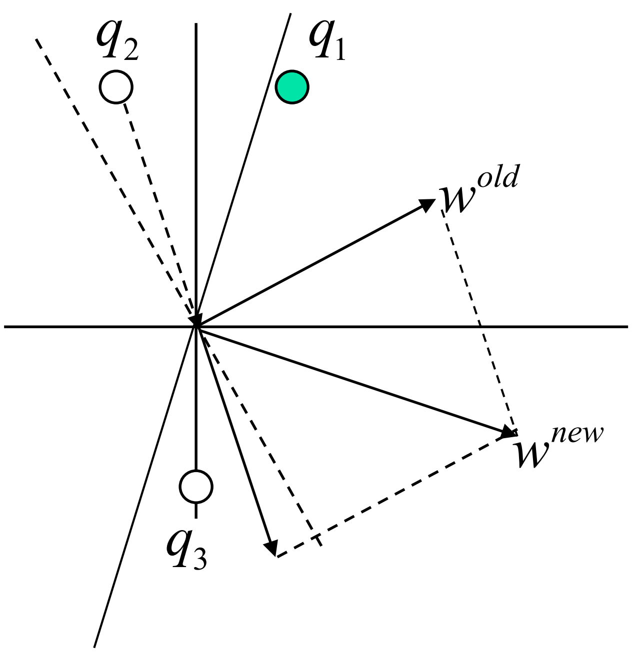
If $t = 0$ and $a = 1$, then $w^{new} = w^{old} - q$

Second Iteration

$$w^{new} = w^{old} - q_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$

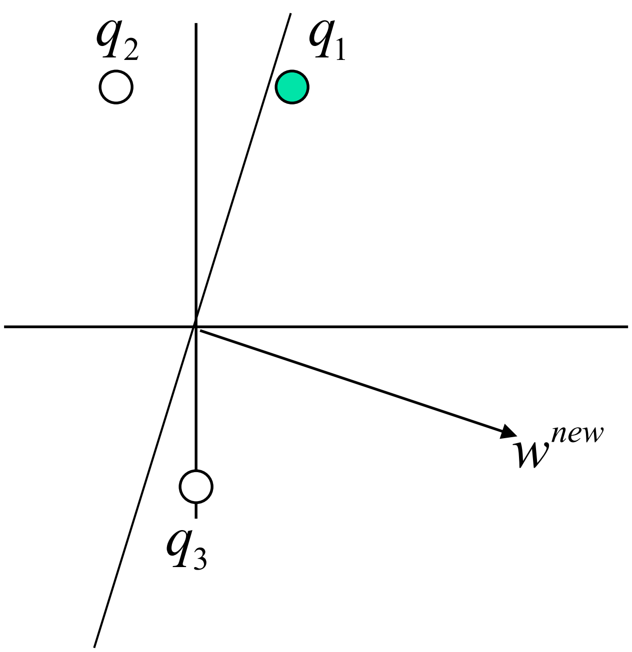
Learning Rule:

If $t = 0$ and $a = 1$, then $w^{new} = w^{old} - q$



Second Iteration

$$w^{new} = w^{old} - q_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$



Learning Rule:

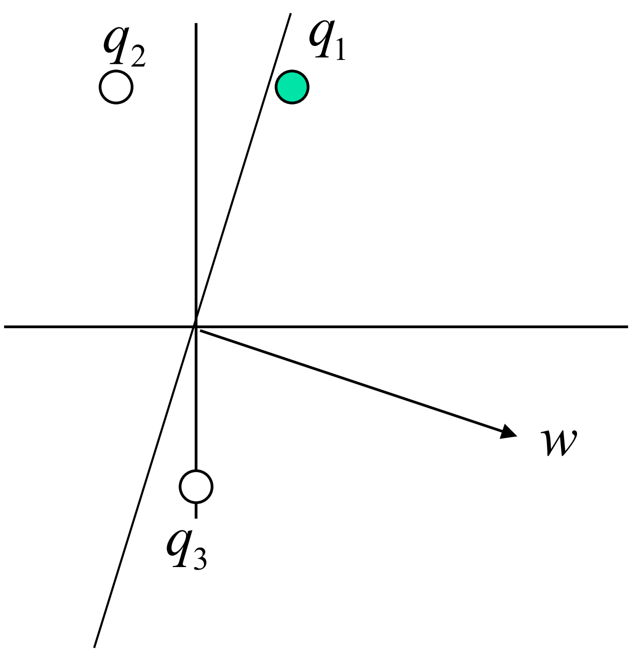
If $t = 0$ and $a = 1$, then $w^{new} = w^{old} - q$

Checking

$$w = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$

Present q_3 to the network: $\left\{ q_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, t_3 = 0 \right\}$

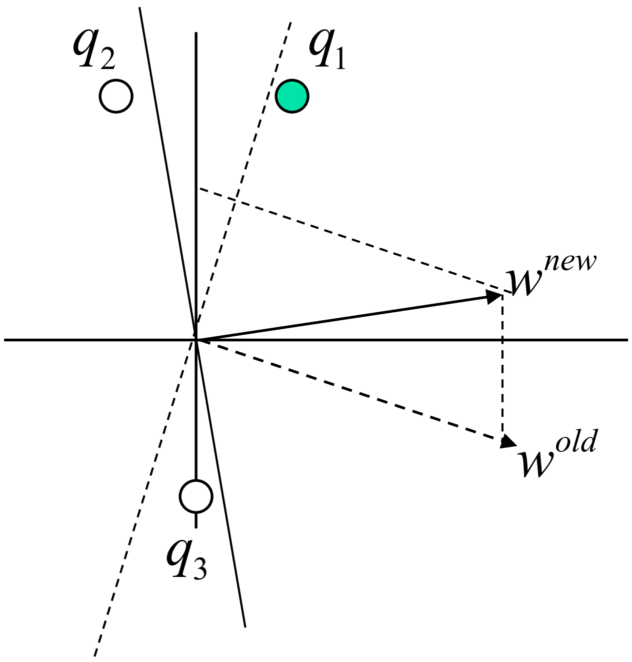
$$a = f\left(w^T q_3\right) = f\left(\begin{bmatrix} 3.0 & -0.8 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right) = f(0.8) = 1$$



Incorrect Classification !

Third Iteration

$$w^{new} = w^{old} - q_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix}$$

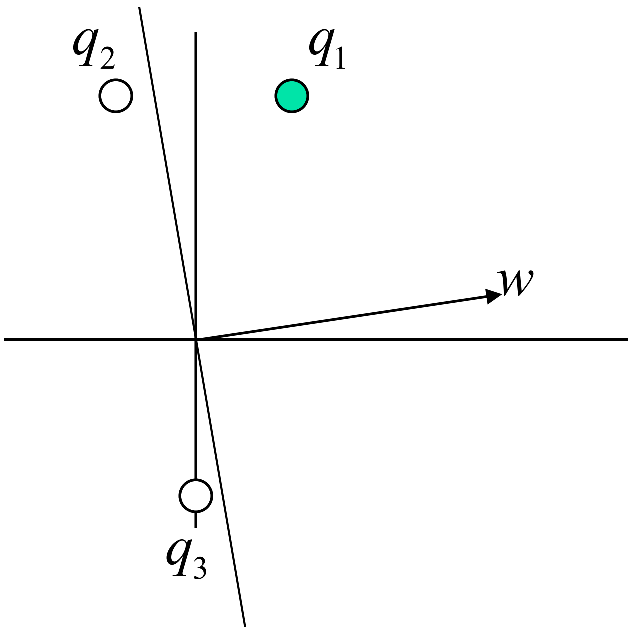


Learning Rule:

If $t = 0$ and $a = 1$, then $w^{new} = w^{old} - q$

Third Iteration

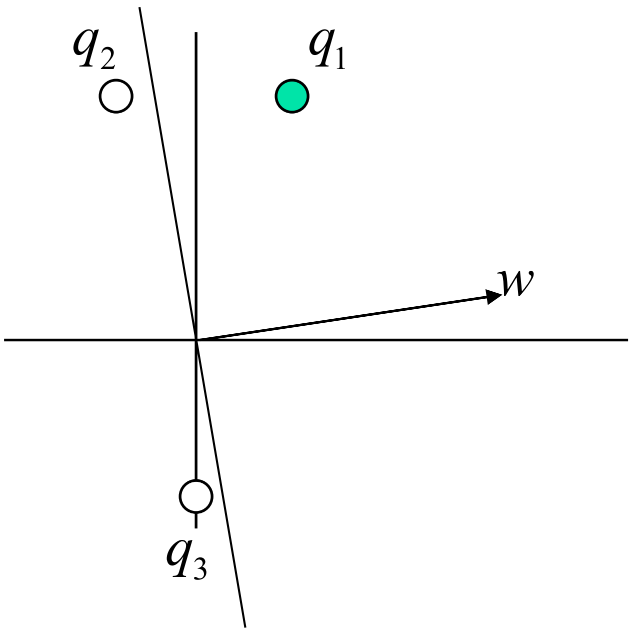
$$w^{new} = w^{old} - q_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix}$$



Learning Rule:

If $t = 0$ and $a = 1$, then $w^{new} = w^{old} - q$

Algorithm Converged



Patterns are now correctly classified.

Learning Rule:

If $t = a$, then $w^{new} = w^{old}$

Unified Learning Rule

$$\left\{ q_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \quad \left\{ q_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \quad \left\{ q_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, t_3 = 0 \right\}$$

Learning Rule:

If $t = 1$ and $a = 0$, then $w^{new} = w^{old} + q$

If $t = 0$ and $a = 1$, then $w^{new} = w^{old} - q$

If $t = a$, then $w^{new} = w^{old}$

Unified Learning Rule

Learning Rule:

Define $e = t - a$

If $e = 1$, then $w^{new} = w^{old} + q$

If $e = -1$, then $w^{new} = w^{old} - q$

If $e = 0$, then $w^{new} = w^{old}$

Unified Learning Rule

Learning Rule:

Define $e = t - a$

$$w^{new} = w^{old} + eq = w^{old} + (t - a)q$$

Unified Learning Rule

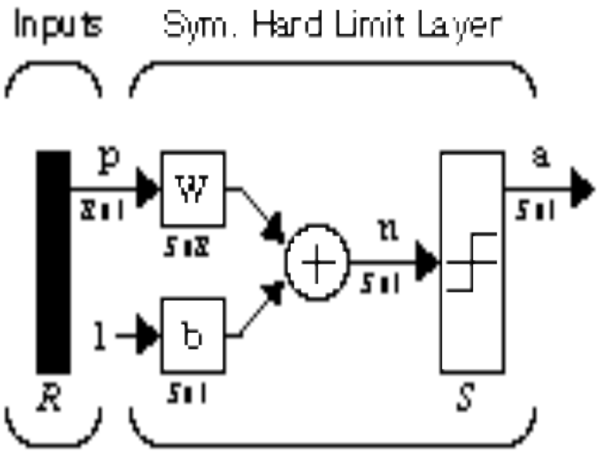
Since, a bias is a weight with an input of 1.

<u>Learning Rule</u>:

$$w^{new} = w^{old} + eq = w^{old} + (t - a)q$$

$$b^{new} = b^{old} + e = w^{old} + (t - a)b$$

Perceptron



a = hardlims(Wp+b)

Multiple-Neuron Perceptrons

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

Learning Algorithm

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

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

Multiple-Neuron Perceptrons

Step 1: Design the perceptron structure.

$$a = \mathrm{hardlim}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

Step 2: Training the network by learning algorithm.

Learning Algorithm

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

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

Apple/Banana Example

Training Set

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

Apple/Banana Example

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 Ability

- **The perceptron rule will always converge to weights which accomplish the desired classification, assuming that such weights exist.**

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

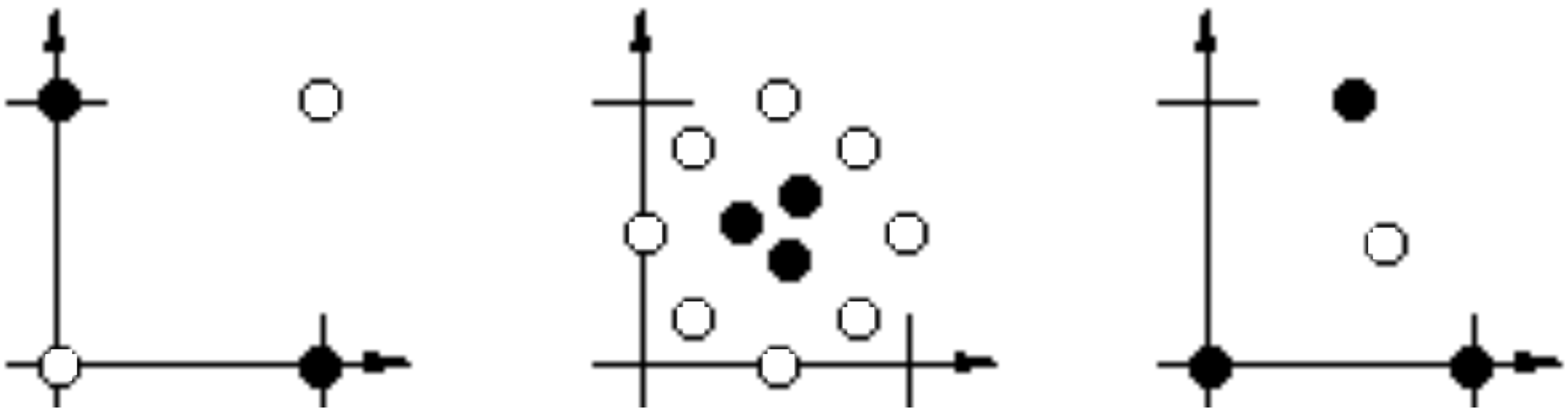
$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$$

Perceptron limitation

Linear Decision Boundary:

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

Linearly Inseparable Problems:



Exercise 1

设有4类，其输入向量分别如下所示：

$$C1: \left\{ p_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, p_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\} \qquad C2: \left\{ p_3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, p_4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right\}$$

$$C3: \left\{ p_5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, p_6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \right\} \qquad C4: \left\{ p_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, p_8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix} \right\}$$

试设计一种感知机网络，进行分类。

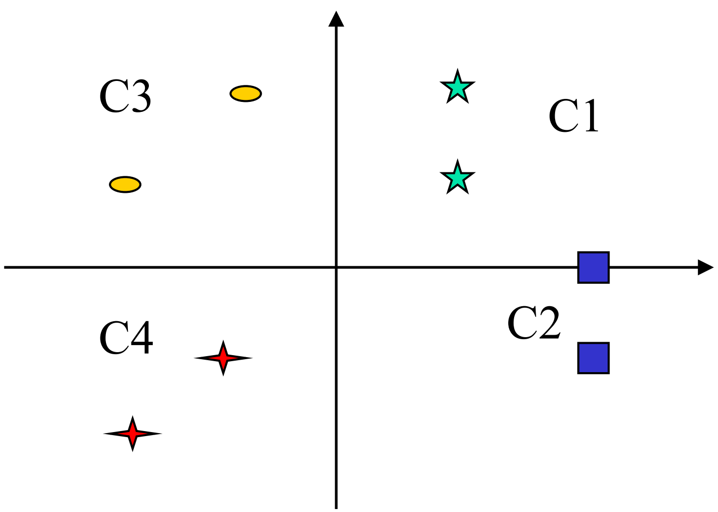
Solution to exercise 1

C1: $\left\{ p_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, p_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}$

C2: $\left\{ p_3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, p_4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right\}$

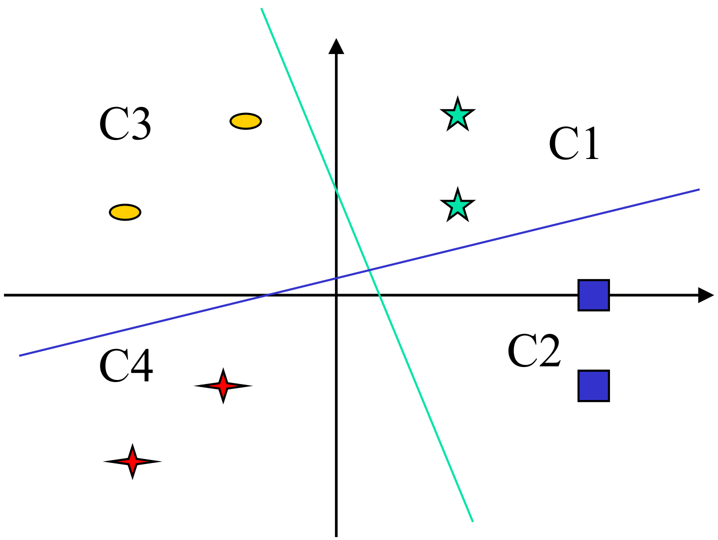
C3: $\left\{ p_5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, p_6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \right\}$

C4: $\left\{ p_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, p_8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix} \right\}$



由于，2^S个线性可分类别可以用具有S个神经元的感知机进行分类，因此，此类问题至少需要具有2个神经元的感知机。

Solution to exercise 1



定义4个类别的目标输出：

$$\text{C1:}\ \left\{ t_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}$$

$$\text{C2:}\ \left\{ t_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

$$\text{C3:}\ \left\{ t_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}$$

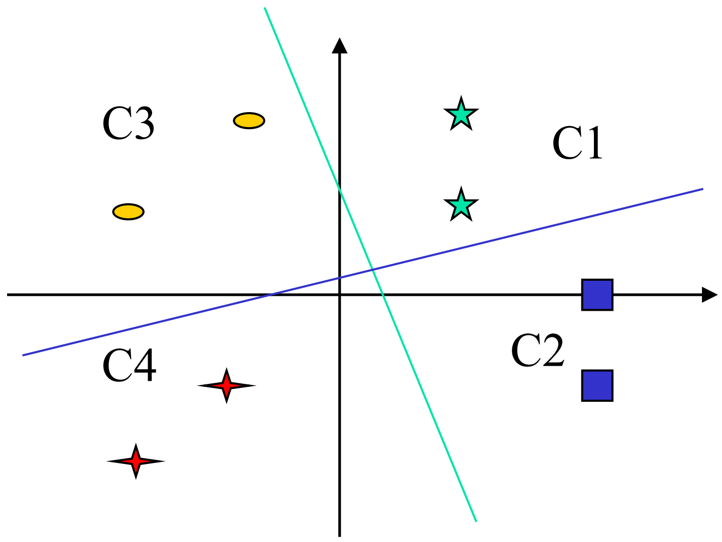
$$\text{C4:}\ \left\{ t_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$$

2个神经元的感知机可以生成两条决策线

Solution to exercise 1

可以设定2个权值向量为：

$$w_1 = \begin{bmatrix} -3 \\ -1 \end{bmatrix} \qquad w_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$



权值向量的矩阵表示形式：

$$W = \begin{bmatrix} w_1{}^T \\ w_2{}^T \end{bmatrix} = \begin{bmatrix} -3 & -1 \\ 1 & -2 \end{bmatrix}$$

Solution to exercise 1

可以设定2个权值向量为：

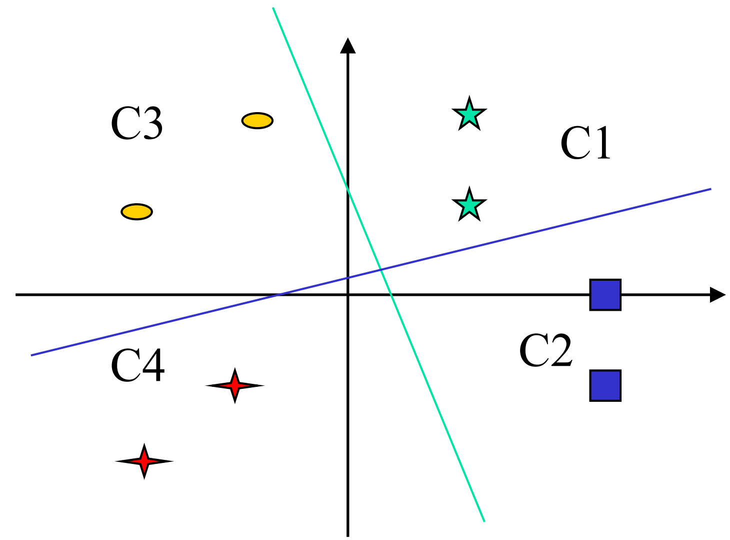
$$w_1 = \begin{bmatrix} -3 \\ -1 \end{bmatrix} \qquad w_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

计算偏移量，令：

$$w_i^T P + b_i = 0$$

$$b_1 = -w_1^T P = -[-3 \quad -1]\begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1$$

$$b_2 = -w_2^T P = -[1 \quad -2]\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$$



Exercise 2

请利用感知机学习规则训练一个感知机网络，求解上述分类问题。

Solution to exercise 2

利用上述设定的目标向量以及输入向量，构成的训练集为：

$$\left\{ p_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} \qquad \left\{ p_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}$$

$$\left\{ p_3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, t_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \qquad \left\{ p_4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, t_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

$$\left\{ p_5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \qquad \left\{ p_6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix}, t_6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}$$

$$\left\{ p_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \qquad \left\{ p_8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, t_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$$

Solution to exercise 2

设初始权值和偏移向量为：

$$W(0)=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \qquad \mathrm{b}(0)=\begin{bmatrix}1\\ 1\end{bmatrix}$$

Solution to exercise 2

The 1st Iteration：

$$a = hard\lim\left(W(0)P_1 + b(0)\right) = hard\lim\left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}+\begin{bmatrix}1\\1\end{bmatrix}\right) = \begin{bmatrix}1\\1\end{bmatrix}$$

$$e = t_1 - a = \begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}-1\\-1\end{bmatrix}$$

$$w(1) = w(0) + ep_1^T = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} + \begin{bmatrix}-1\\-1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix}$$

$$b(1) = b(0) + e = \begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}-1\\-1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

Solution to exercise 2

The 2nd Iteration：

$$a = hard\lim\left(W(1)P_2 + b(1)\right) = hard\lim\left(\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$e = t_2 - a = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$w(2) = w(1) + ep_2{}^T = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

$$b(2) = b(1) + e = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solution to exercise 2

The 3rd Iteration：

$$a = hard\lim\left(W(2)P_3 + b(2)\right) = hard\lim\left(\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} 2 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$e = t_3 - a = \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$w(3) = w(2) + ep_3^{\,T} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}\begin{bmatrix} 2 & -1 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 1 & -1 \end{bmatrix}$$

$$b(3) = b(2) + e = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Solution to exercise 2

继续迭代，直至第8次：

$$W(8)=W(7)=W(6)=W(5)=W(4)=W(3)=\begin{bmatrix}-2 & 0\\ 1 & -1\end{bmatrix}$$

$$b(8)=b(7)=b(6)=b(5)=b(4)=b(3)=\begin{bmatrix}-1\\ 1\end{bmatrix}$$

Solution to exercise 2

The 9th Iteration :

$$a = hard\lim\left(W(8)P_1 + b(8)\right) = hard\lim\left(\begin{bmatrix} -2 & 0 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$e = t_1 - a = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$w(9) = w(8) + ep_1^T = \begin{bmatrix} -2 & 0 \\ 1 & -1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix}\begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}$$

$$b(9) = b(8) + e = \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

Solution to exercise 2

至此，算法收敛。最终决策线如图所示：