Week 8

Linear Discrimination
Multilayer Perceptrons

Emre Ugur, BM 33

emre.ugur@boun.edu.tr

http://www.cmpe.boun.edu.tr/~emre/courses/cmpe462

cmpe462@listeci.cmpe.boun.edu.tr

Exercise (histogram, k-nn, etc.)

Assume you are given the following training sample composed of 10 instances, where the first column corresponds to the feature value (x) and the second column corresponds to the class (c) of the instance

x	c
2	0
2	0
4	0
6	0
6	0
3	1
3	1
4	1
5	1
5	1

Predict the class of a new instance (x=5) using all the methods below.

x	c
5	?

Projects

- By April 9
- Email to instructor
    - Team members and
    - Description of the project
- Penalty: -5% x late-day
    - If not provided by deadline.

Likelihood- vs. Discriminant-based Classification

- Likelihood-based: Assume a model for $p(\boldsymbol{x}|\mathcal{C}_i)$, use Bayes' rule to calculate $P(\mathcal{C}_i|\boldsymbol{x})$

  $$g_i(\boldsymbol{x}) = \log P(\mathcal{C}_i|\boldsymbol{x})$$

- Discriminant-based: Assume a model for $g_i(\boldsymbol{x}|\Phi_i)$; no density estimation
- Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries

Linear Discriminant

- Linear discriminant:

$$g_i(\boldsymbol{x} \mid \boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$$

- Advantages:
  - Simple: O(d) space/computation
  - Knowledge extraction: Weighted sum of attributes; positive/negative weights, magnitudes (credit scoring)

Generalized Linear Model

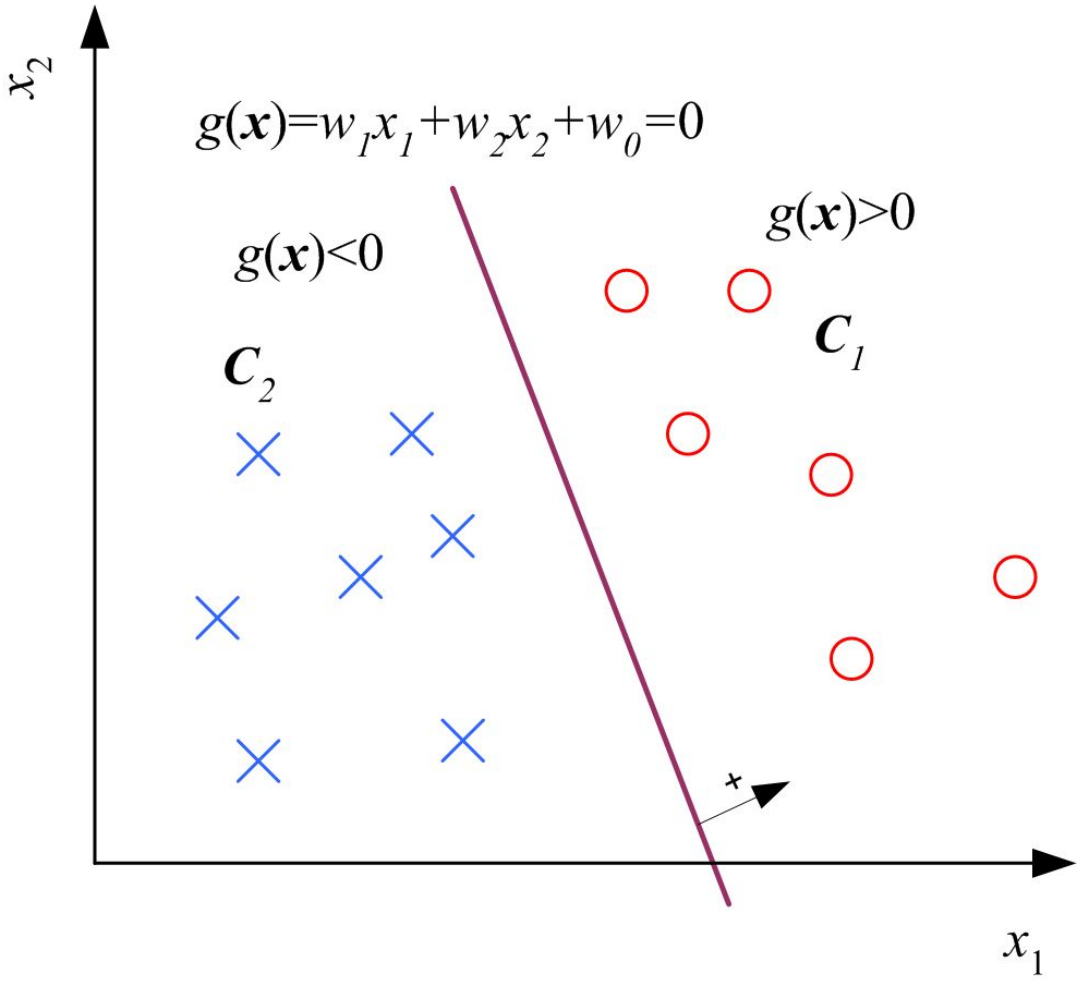
- Higher-order (product) terms:

$$z_1 = x_1,\ z_2 = x_2,\ z_3 = x_1^2,\ z_4 = x_2^2,\ z_5 = x_1 x_2$$

Map from $\boldsymbol{x}$ to $\boldsymbol{z}$ using nonlinear basis functions and use a linear discriminant in $\boldsymbol{z}$-space

$$g_i(\boldsymbol{x}) = \sum_{j=1}^{k} w_{ij}\phi_j(\boldsymbol{x})$$

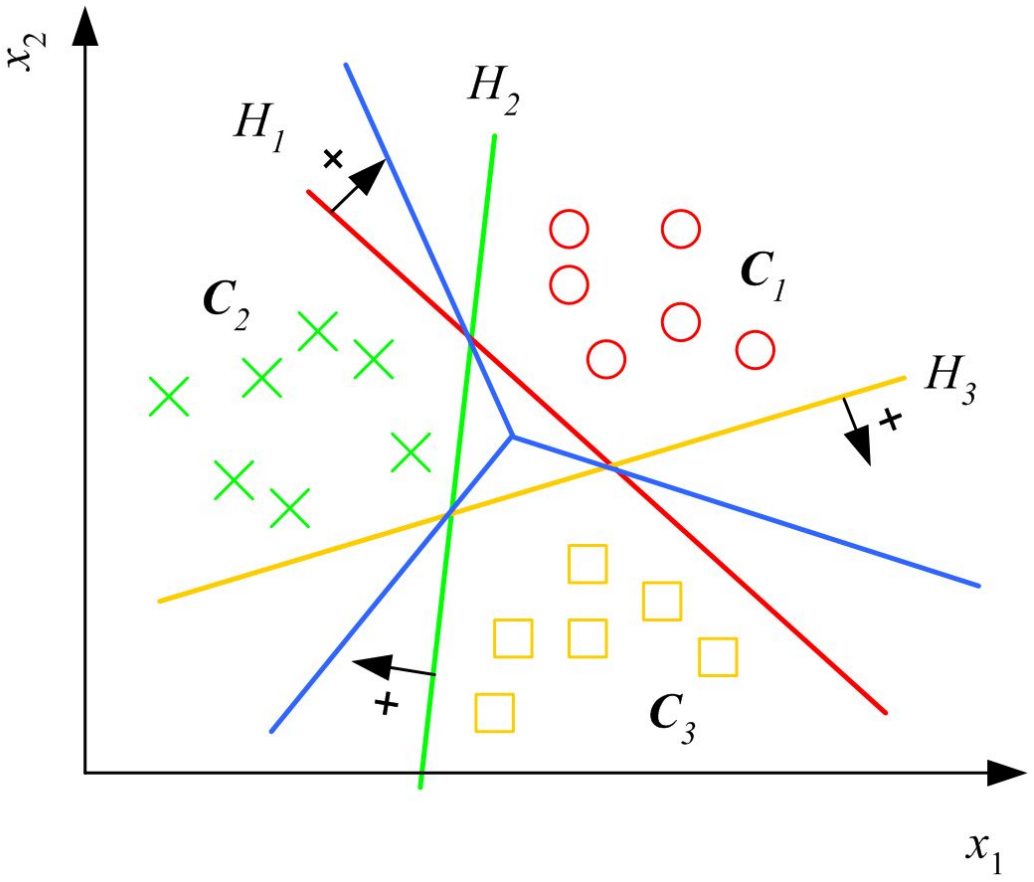
Two Classes



$$\text{choose}\begin{cases} C_1 & \text{if } g(\boldsymbol{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}$$

Multiple Classes

$$g_i(\boldsymbol{x} \mid \boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}$$

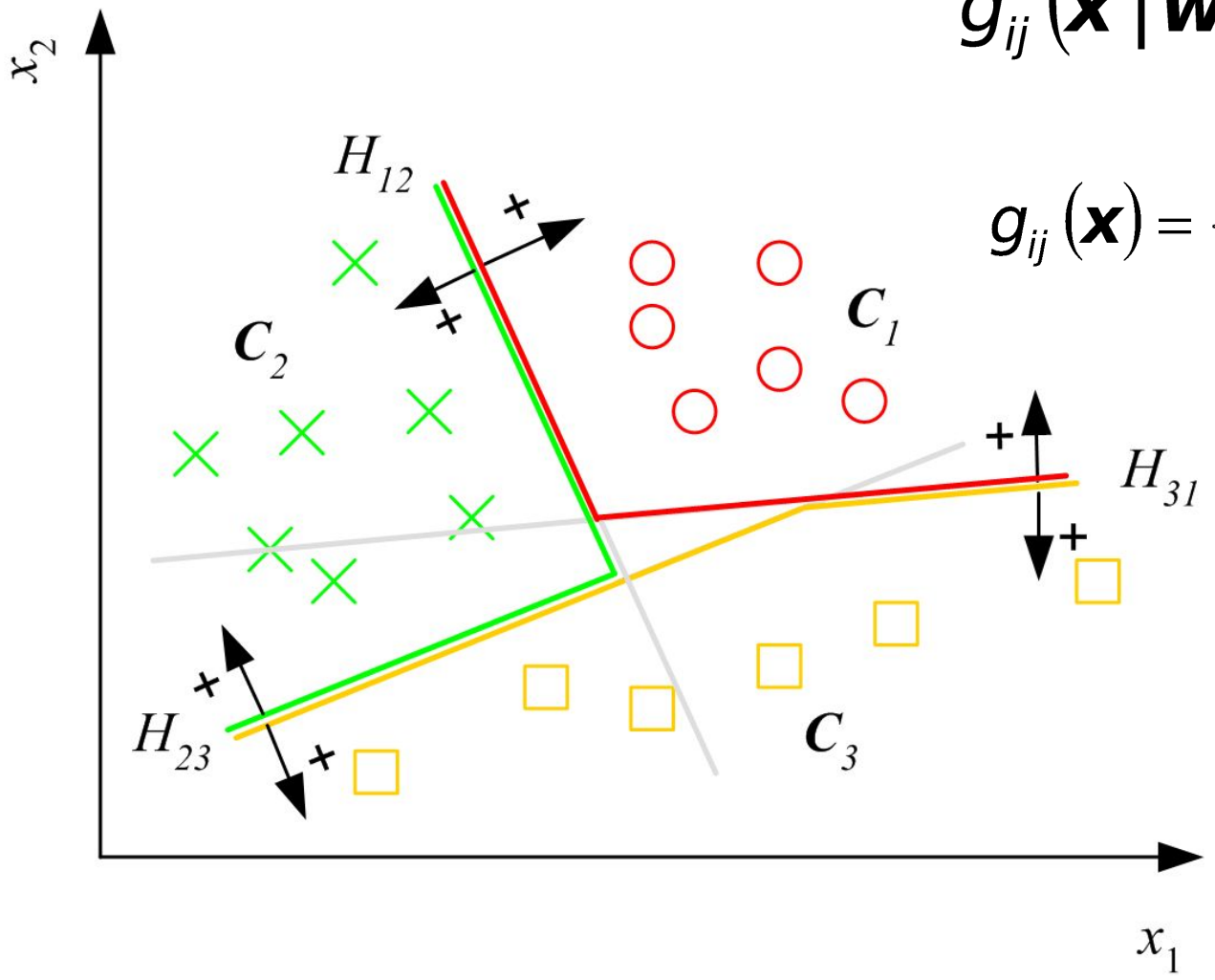


Choose C_i if

$$g_i(\boldsymbol{x}) = \max_{j=1}^{K} g_j(\boldsymbol{x})$$

Classes are linearly separable

Pairwise Separation



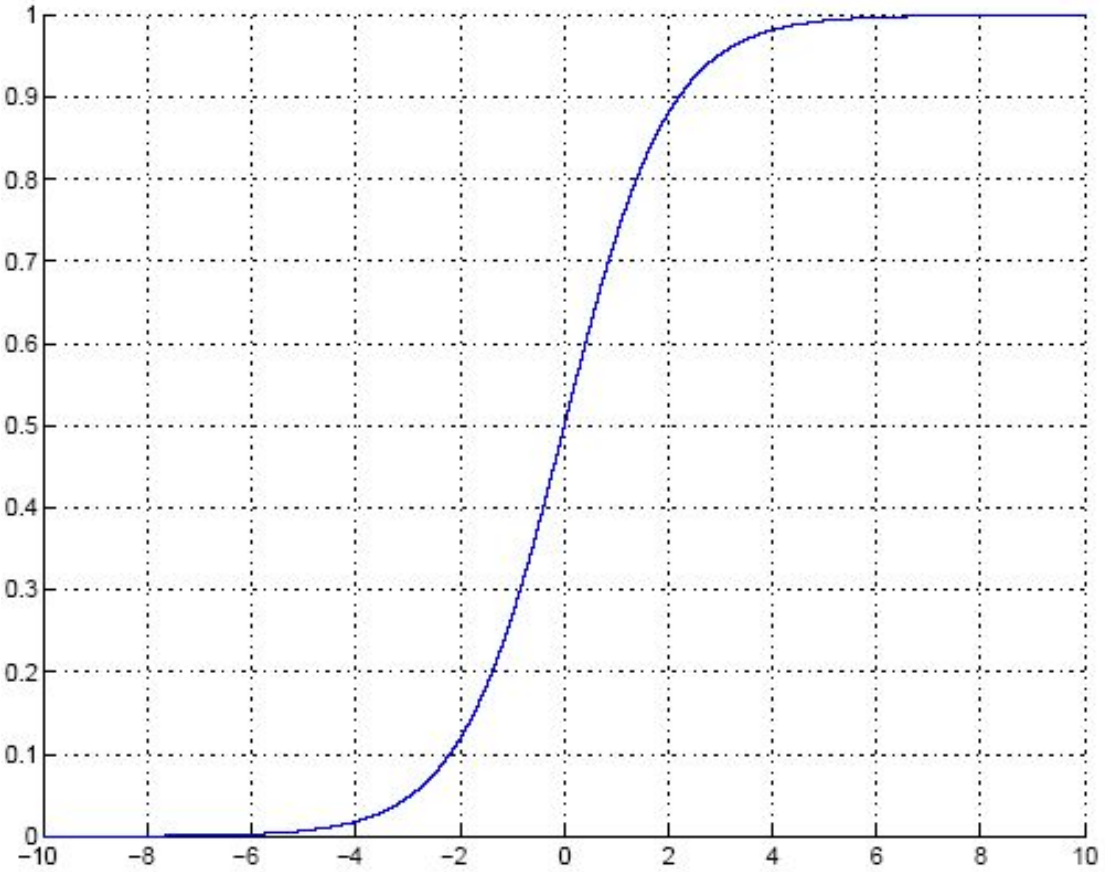
$$g_{ij}\left(\boldsymbol{x} \mid \boldsymbol{w}_{ij}, w_{ij\,0}\right) = \boldsymbol{w}_{ij}^T \boldsymbol{x} + w_{ij\,0}$$

$$g_{ij}(\boldsymbol{x}) = \begin{cases} > 0 & \text{if } \boldsymbol{x} \in C_i \\ \leq 0 & \text{if } \boldsymbol{x} \in C_j \\ \text{don't care} & \text{otherwise} \end{cases}$$

choose C_i if

$$\forall j \neq i, g_{ij}(\boldsymbol{x}) > 0$$

Sigmoid (Logistic) Function



1. Calculate $g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$ and choose C_1 if $g(\boldsymbol{x}) > 0$, or
2. Calculate $y = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x} + w_0)$ and choose C_1 if $y > 0.5$

Gradient-Descent

- E($\boldsymbol{w}|\mathcal{X}$) is error with parameters $\boldsymbol{w}$ on sample $\mathcal{X}$

  $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w} \mid \mathcal{X})$

- Gradient

$$\nabla_w E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_d} \right]^T$$

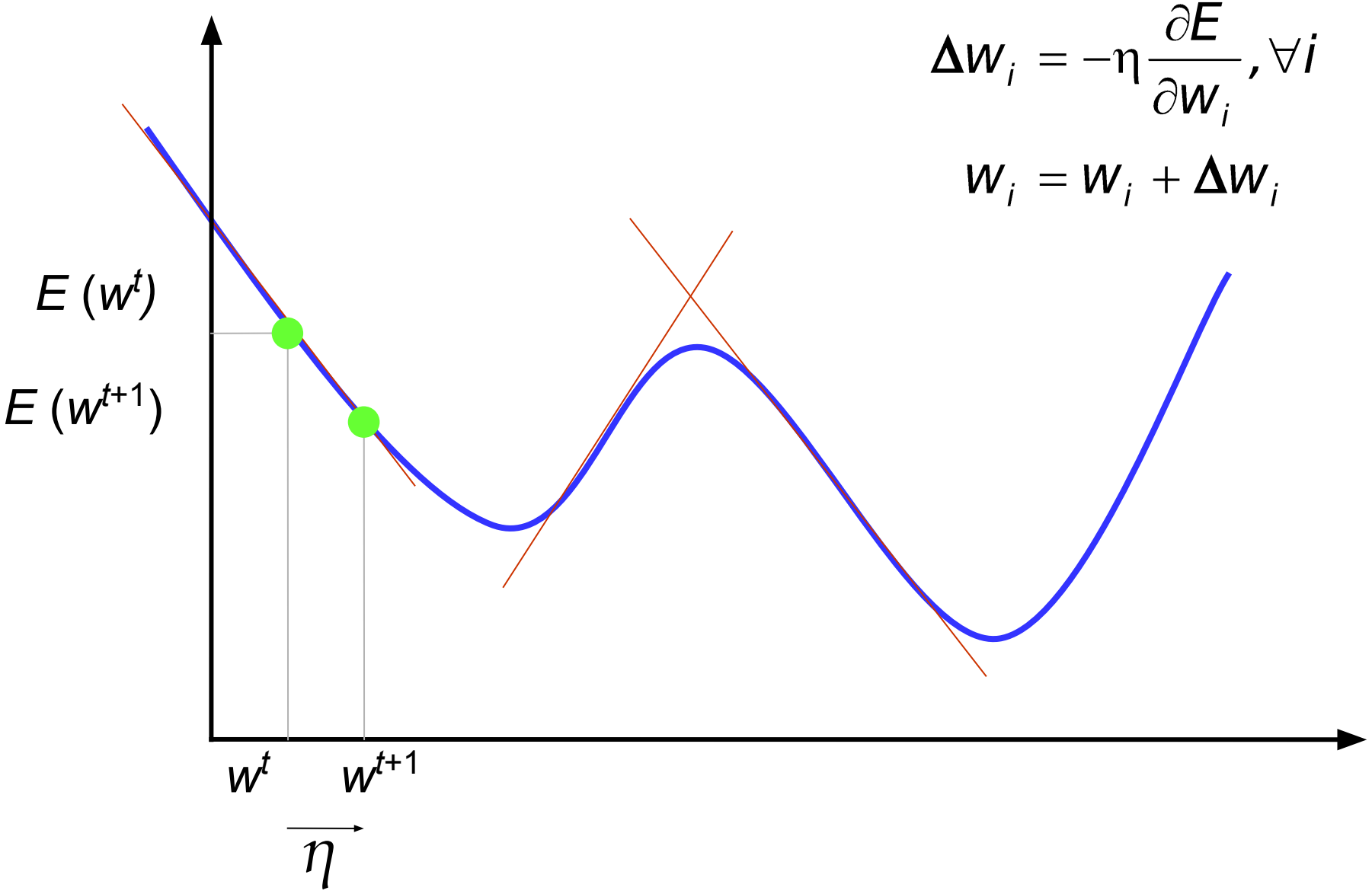
- Gradient-descent:

  Starts from random $\boldsymbol{w}$ and updates $\boldsymbol{w}$ iteratively in the negative direction of gradient

Gradient-Descent

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \forall i$$

$$w_i = w_i + \Delta w_i$$

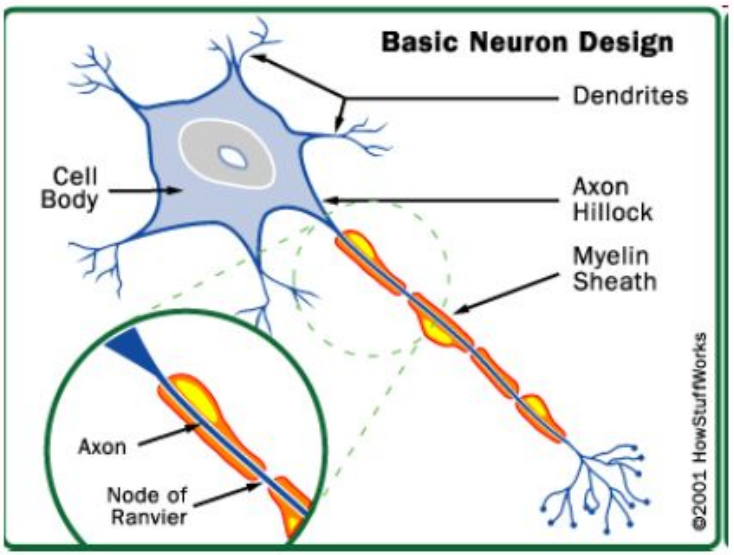


Brain

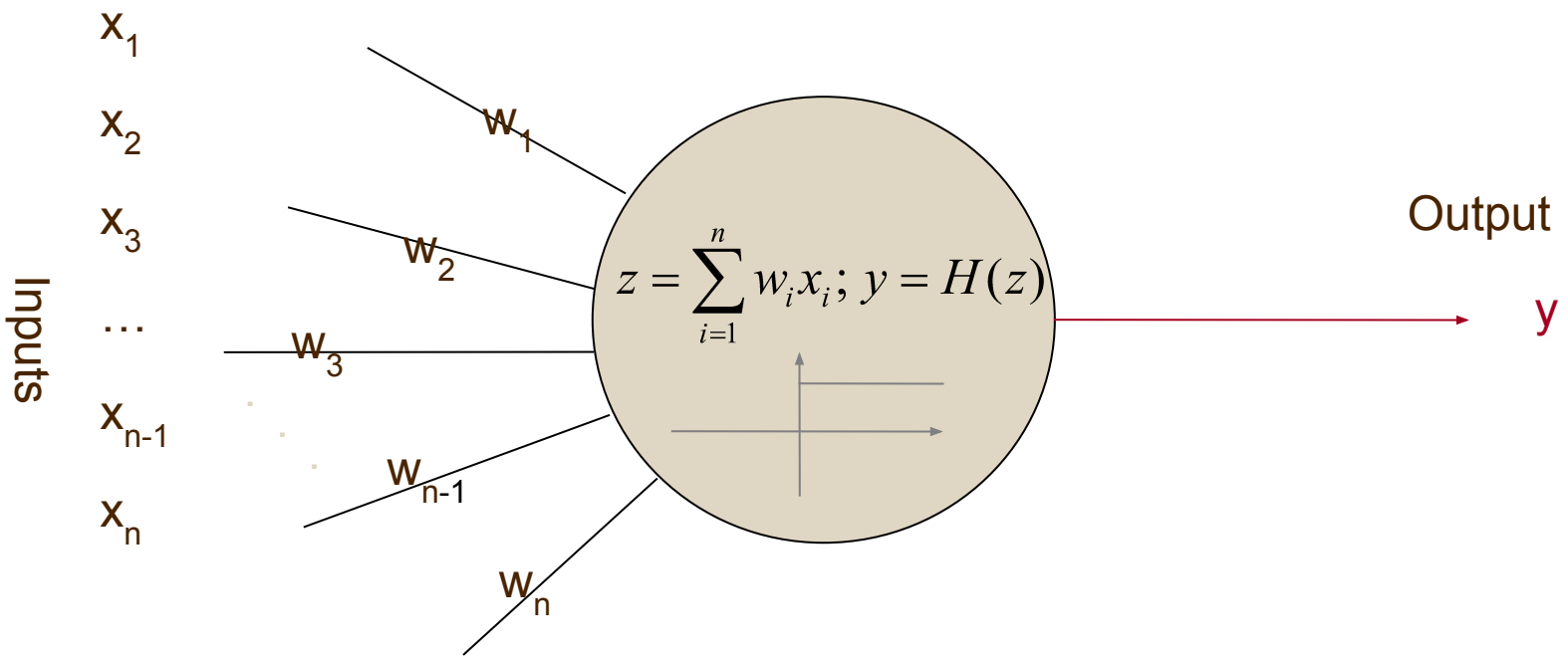
- Large number of neurons: 1010
- Large connectitivity: 105
- Parallel processing
- Distributed computation/memory
- Robust to noise, failures

Biological inspiration

- The spikes travelling along the axon of the pre-synaptic neuron trigger the release of neurotransmitter substances at the synapse.
- The neurotransmitters cause excitation or inhibition in the dendrite of the post-synaptic neuron.
- The integration of the excitatory and inhibitory signals may produce spikes in the post-synaptic neuron.
- The contribution of the signals depends on the strength of the synaptic connection.



Artificial neurons



The McCullogh-Pitts model

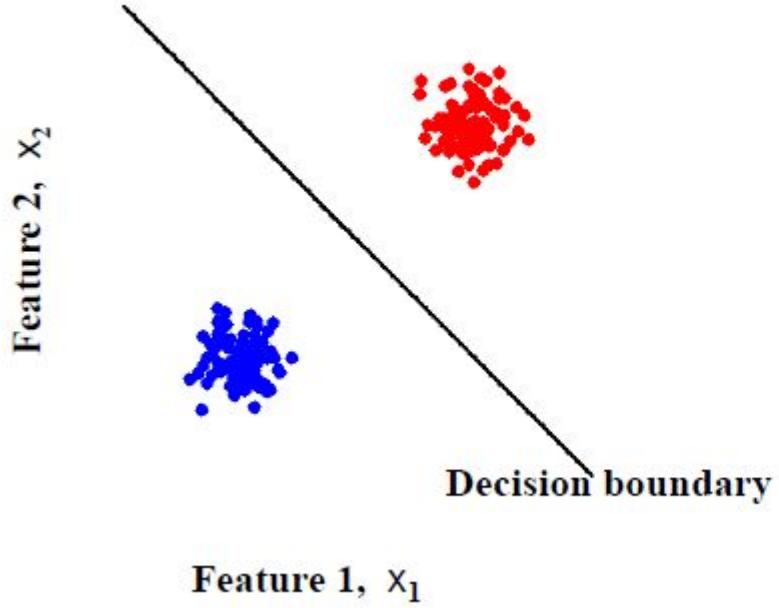
History

- 1943: McCulloch–Pitts “neuron”
  - Started the field
- 1962: Rosenblatt’s perceptron
  - Learned its own weight values; convergence proof
- 1969: Minsky & Papert book on perceptrons
  - Proved limitations of single-layer perceptron networks
- 1982: Hopfield and convergence in symmetric networks
  - Introduced energy-function concept
- 1986: Backpropagation of errors
  - Method for training multilayer networks
- Present: Probabilistic interpretations, Bayesian and spiking networks,
- Deep-Networks

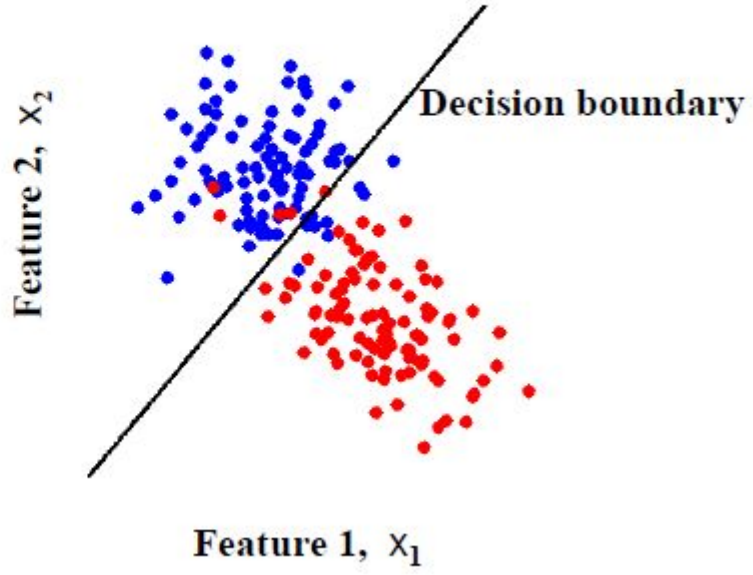
Linear Classifiers (Perceptrons)

- Linear Classifiers
  - a linear classifier is a mapping which partitions feature space using
  - a linear function (a straight line, or a hyperplane)
  - separates the two classes using a straight line in feature space

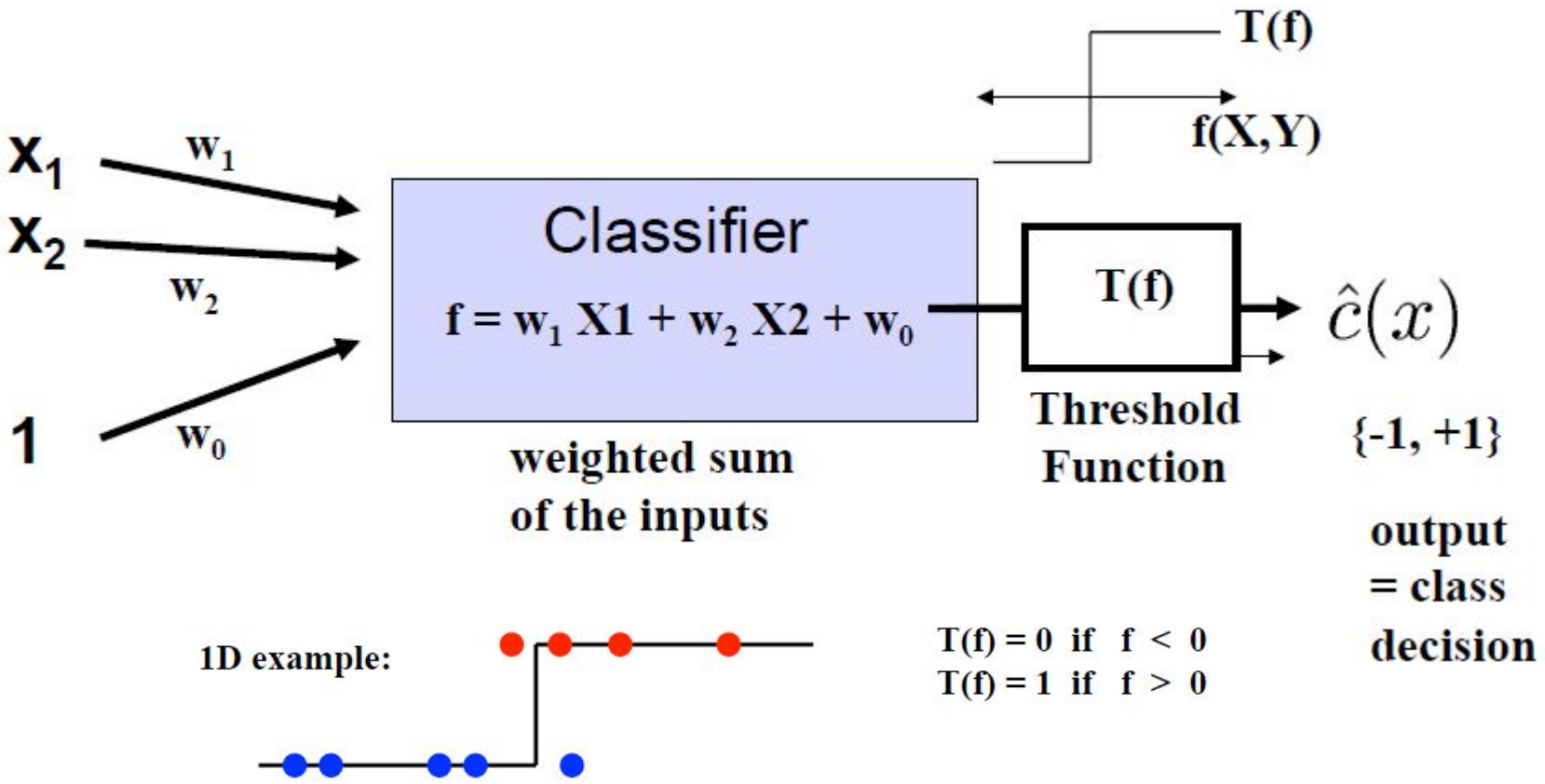
Linearly separable data



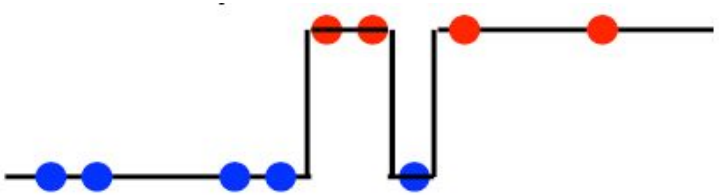
Linearly non-separable data



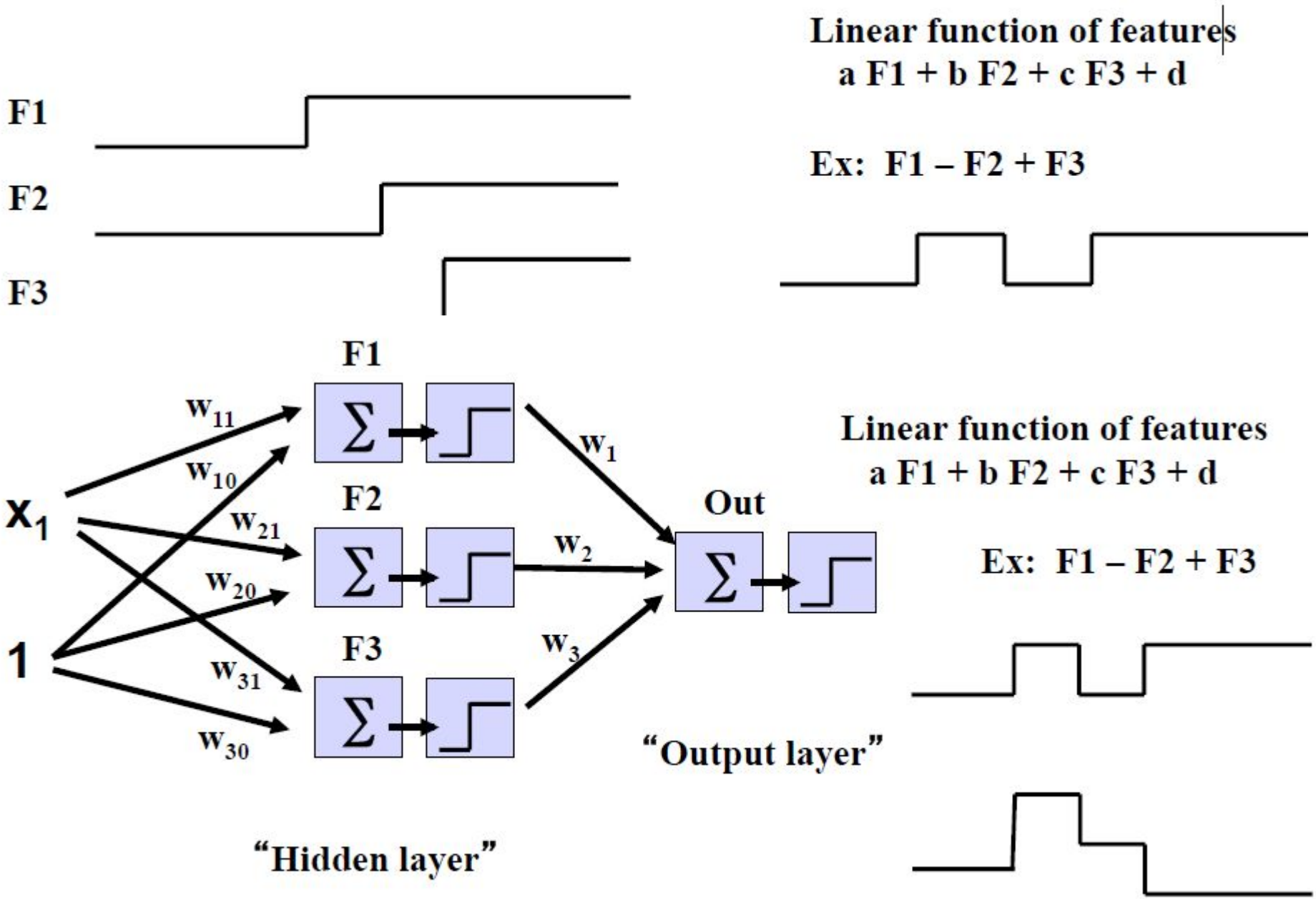
Perceptron Classifier (2 features)



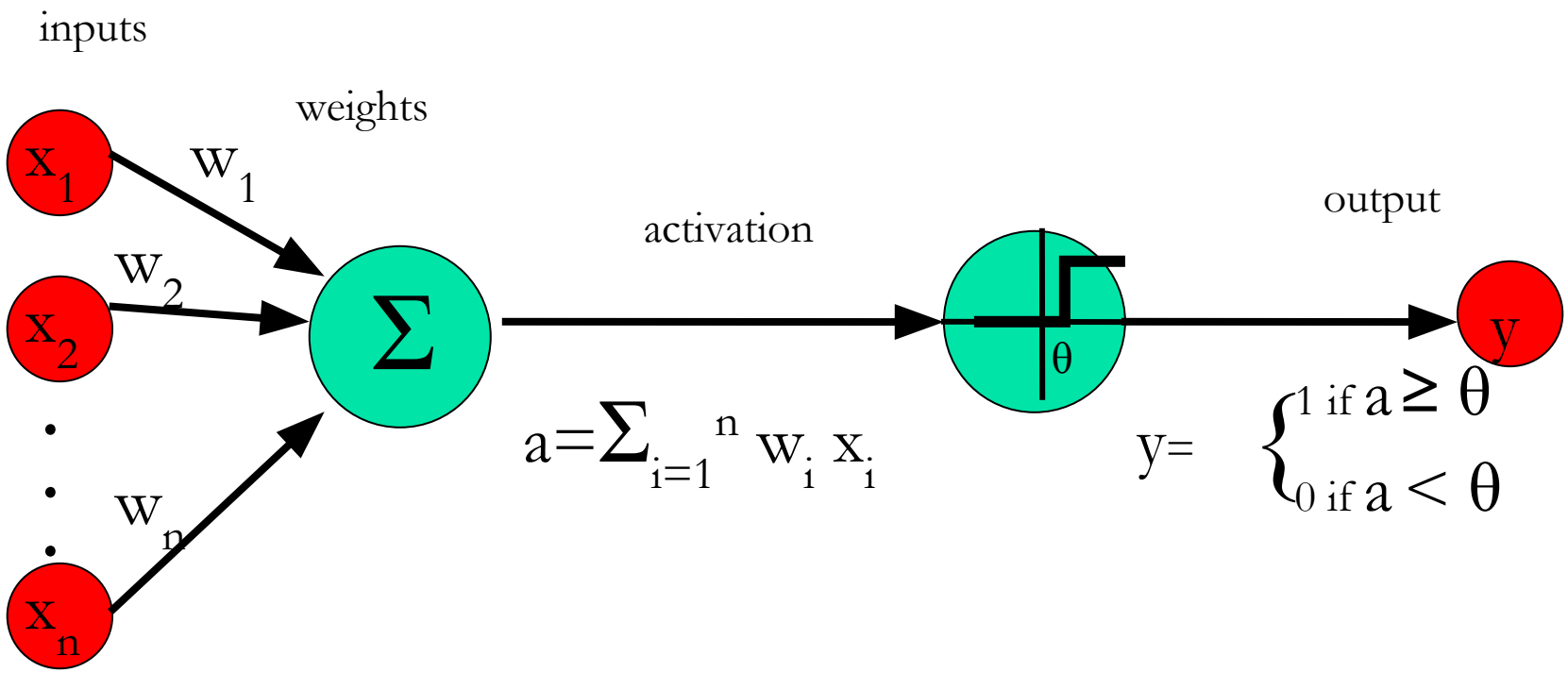
- If features change, it will find non-linear boundaries.
- What features can produce the following decision rule?



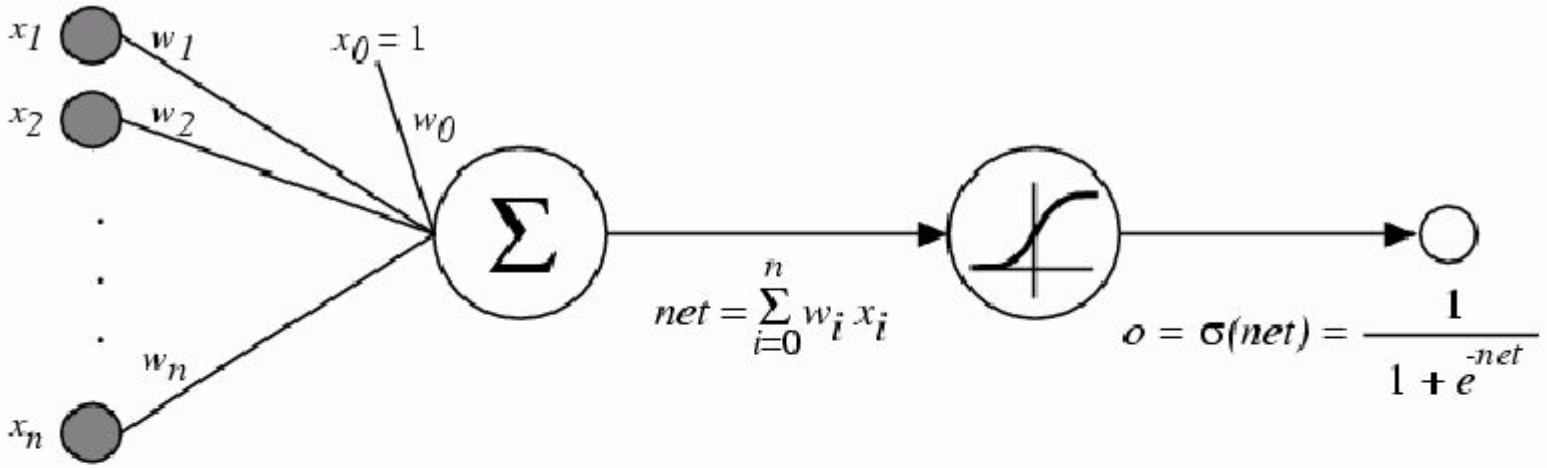
Features and perceptrons



Threshold Unit



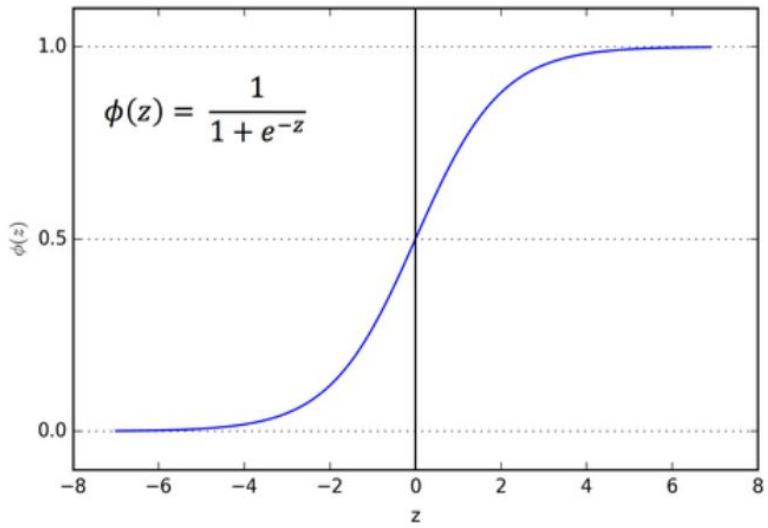
Sigmoid Unit



$\sigma(x)$ is the sigmoid function

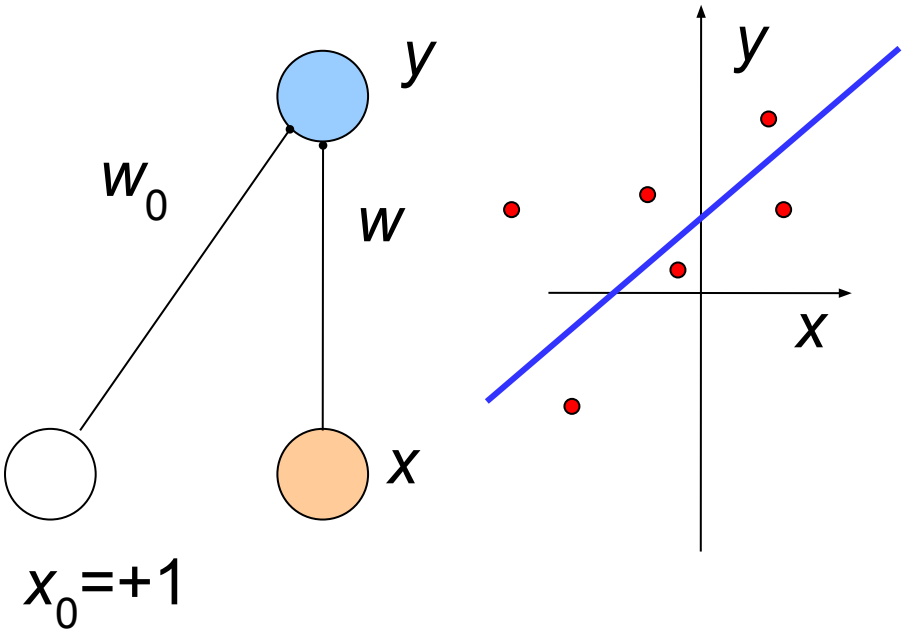
$$\frac{1}{1+e^{-x}}$$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$

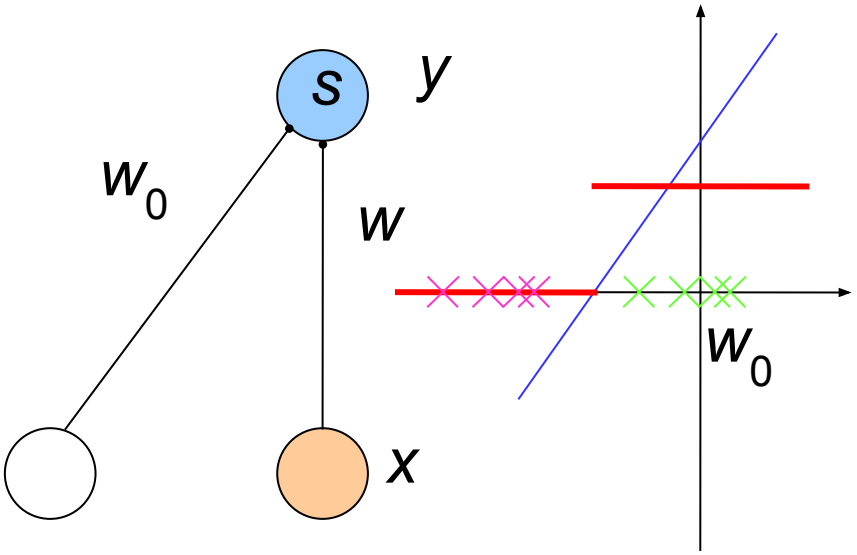


What a perceptron does

- Regression: $y=wx+w_0$



- Classification: $y=1(wx+w_0>0)$



$$y = \text{sigmoid}(o) = \frac{1}{1+\exp\left[-\boldsymbol{w}^T\boldsymbol{x}\right]}$$

- If we need posterior probability

K outputs

- Regression

$$y_i = \sum_{j=1}^{d} w_{ij} x_j + w_{i0} = \mathbf{w}_i^T \mathbf{x}$$

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

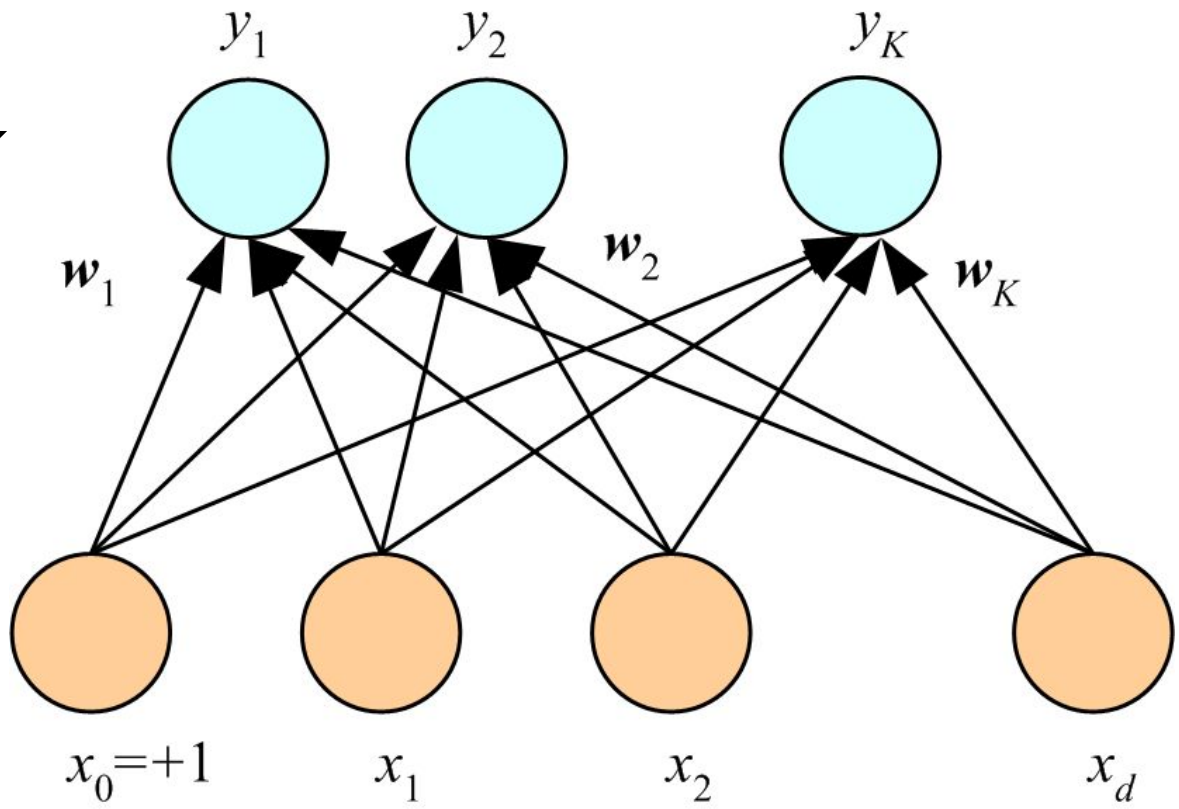
- Classification

$$o_i = \mathbf{w}_i^T \mathbf{x}$$

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

$$\text{choose } C_i$$

$$\text{if } y_i = \max_k y_k$$



Training

- Online (instances seen one by one) vs batch (whole sample) learning:
  - No need to store the whole sample
  - Problem may change in time
  - Wear and degradation in system components
- Stochastic gradient-descent: Update after a single pattern
- Generic update rule (LMS rule):

$$\Delta w_{ij}^t = \eta\left(r_i^t - y_i^t\right)x_j^t$$

$$\text{Update} = \text{LearningFactor} \cdot \left(\text{DesiredOutput} - \text{ActualOutput}\right) \cdot \text{Input}$$

Training a Perceptron

- Regression (linear output)

$$E^t(\boldsymbol{w} \mid \boldsymbol{x}^t, r^t) = \frac{1}{2}(r^t - y^t)^2 = \frac{1}{2}\left[r^t - (\boldsymbol{w}^T \boldsymbol{x}^t)\right]^2$$

$$\Delta w_j^t = \eta(r^t - y^t)x_j^t$$

- Classification with single sigmoid output

$$y^t = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x}^t)$$

$$E^t(\boldsymbol{w} \mid \boldsymbol{x}^t, \boldsymbol{r}^t) = -r^t \log y^t - (1 - r^t) \log (1 - y^t)$$

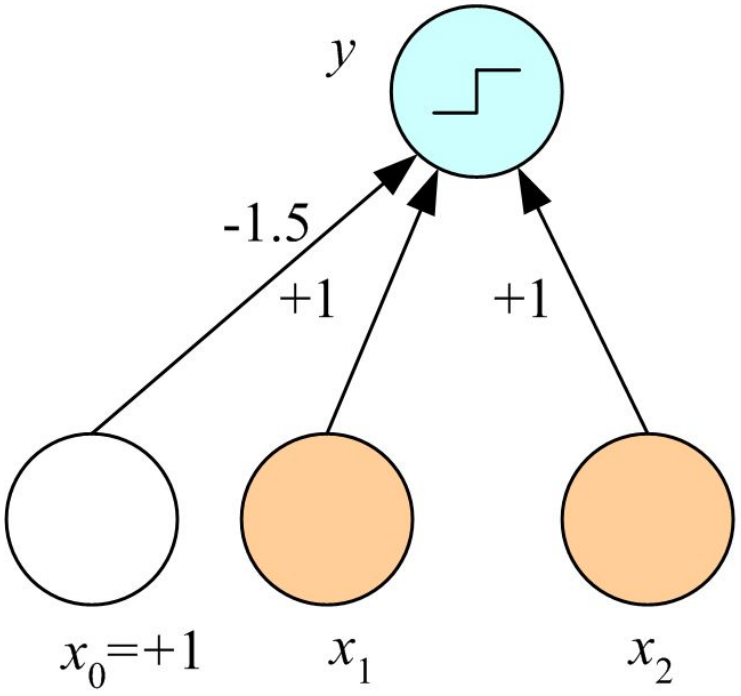
$$\Delta w_j^t = \eta(r^t - y^t)x_j^t$$

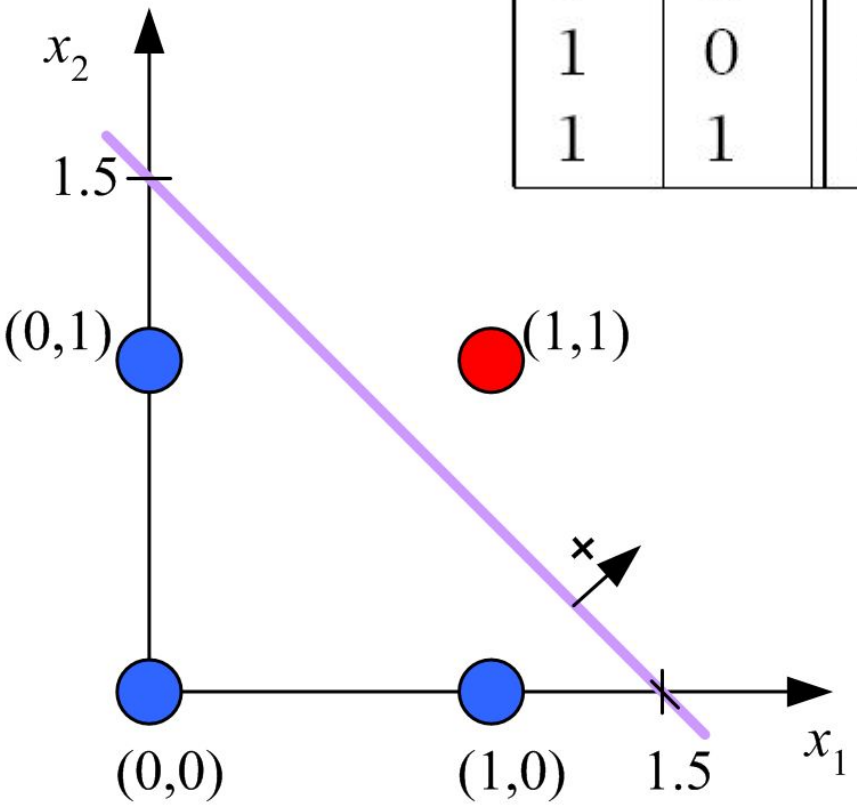
Convergence

- The algorithm converges to the correct classification
  - if the training data is linearly separable, and
  - learning rate is sufficiently small

Learning Boolean AND

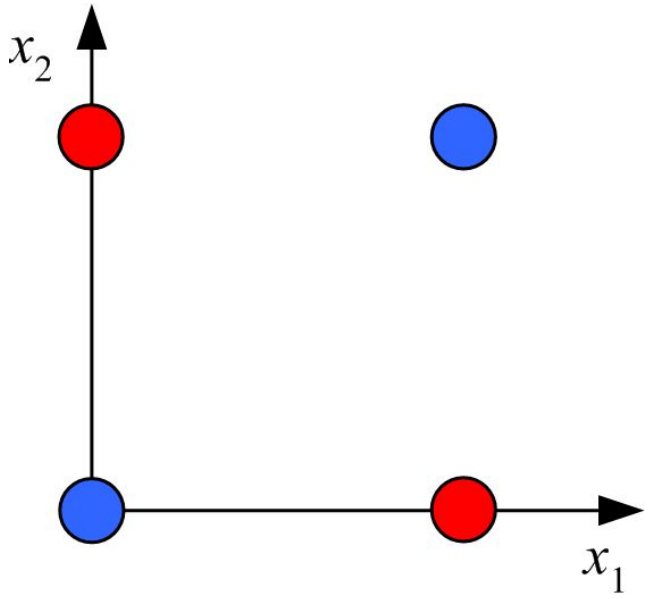
x_1	x_2	r
0	0	0
0	1	0
1	0	0
1	1	1





Learning Boolean XOR

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0



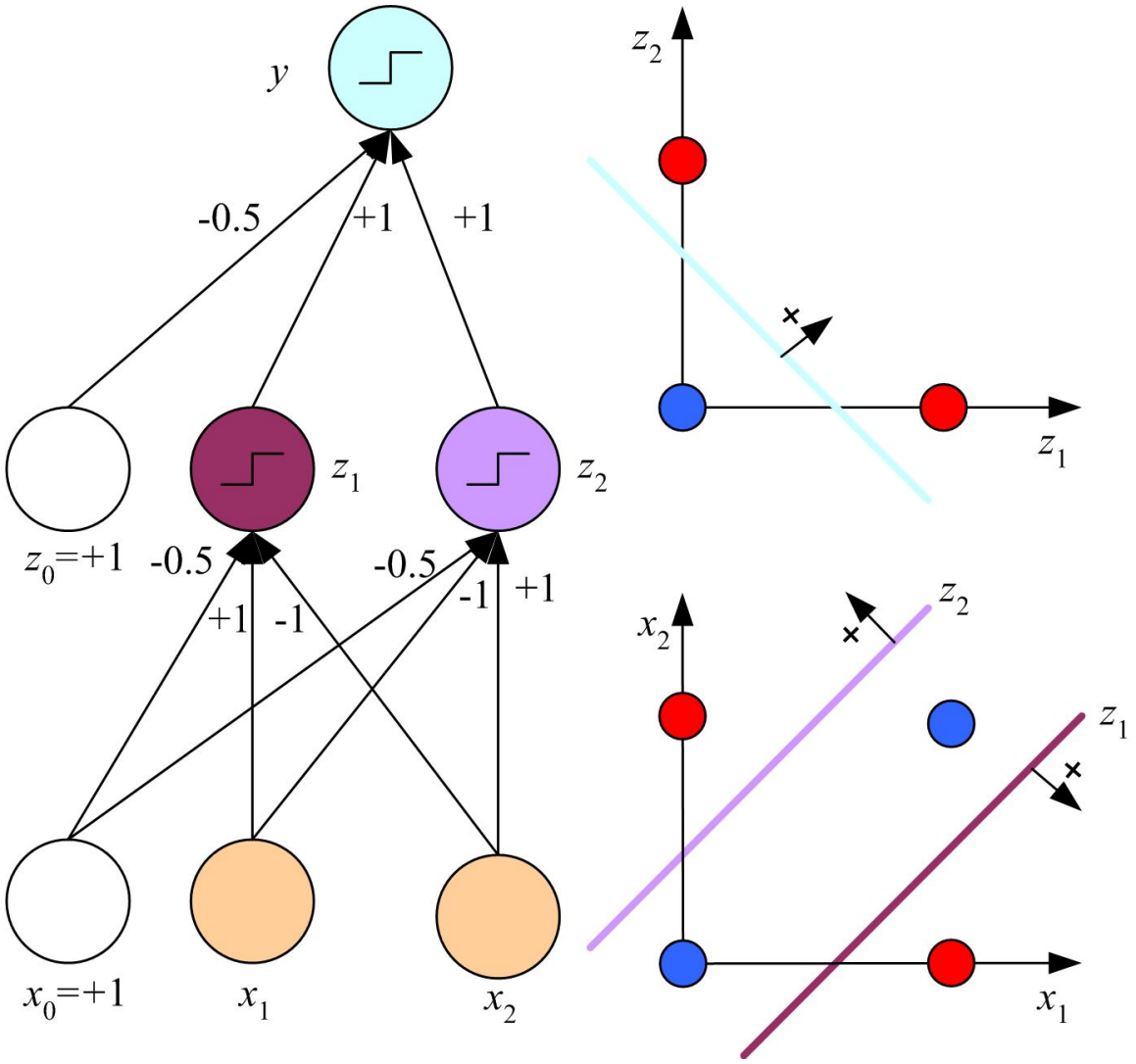
No w_0, w_1, w_2 satisfy:

$$
\begin{aligned}
w_0 &\le 0 \\
w_2 + w_0 &> 0 \\
w_1 + w_0 &> 0 \\
w_1 + w_2 + w_0 &\le 0
\end{aligned}
$$

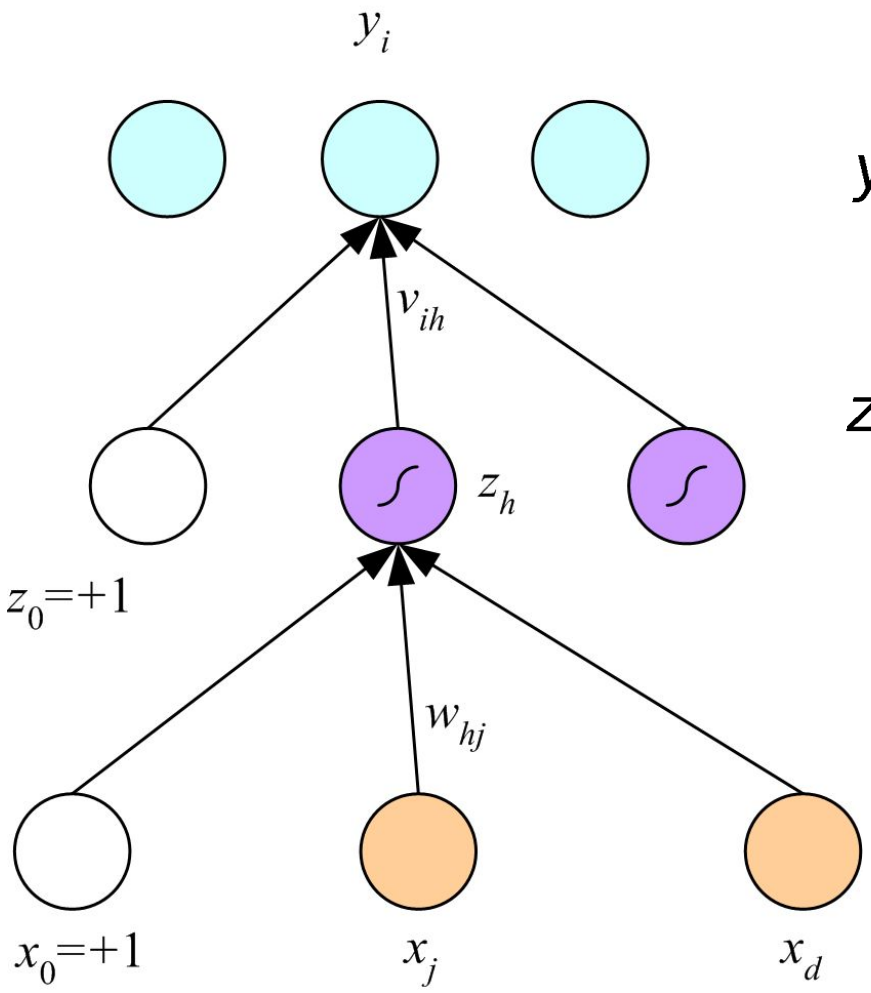
(Minsky and Papert, 1969)

Multilayer perceptrons

x_1 XOR x_2 = (x_1 AND ~x_2) OR (~x_1 AND x_2)



Multilayer Perceptrons

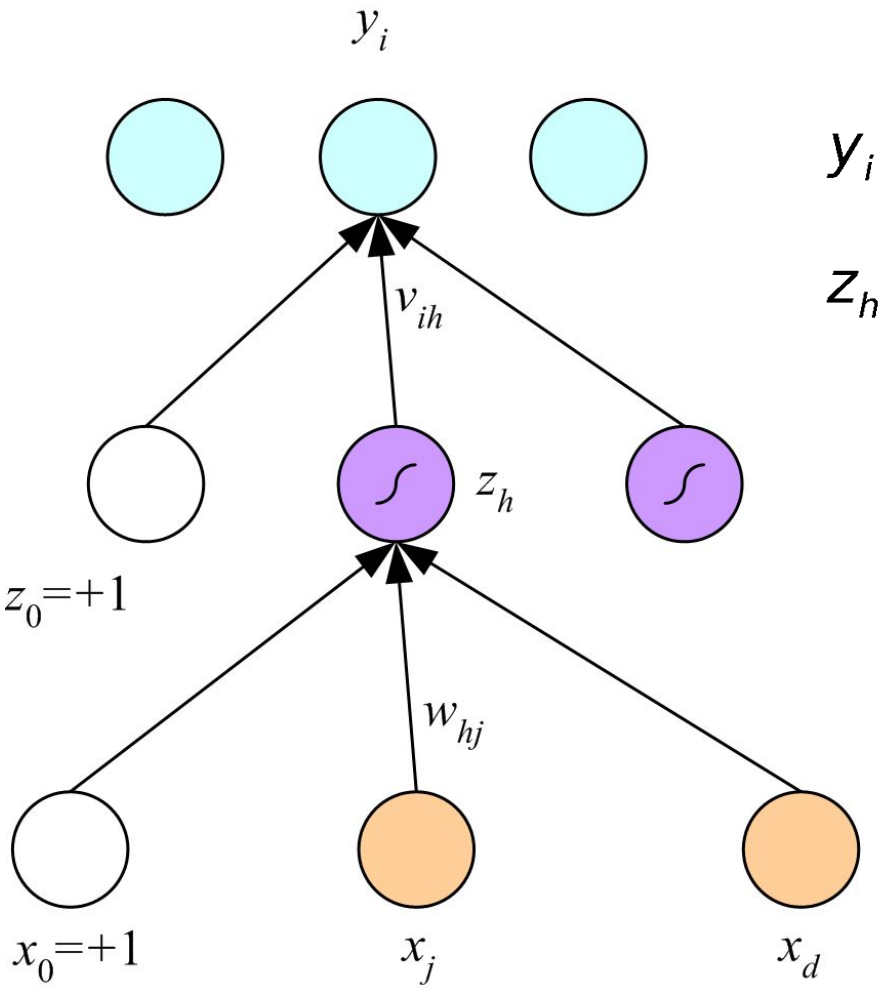


$$y_i = \boldsymbol{v}_i^T \boldsymbol{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}\left(\boldsymbol{w}_h^T \boldsymbol{x}\right) = \frac{1}{1+\exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

(Rumelhart et al., 1986)

Training multilayer perceptron

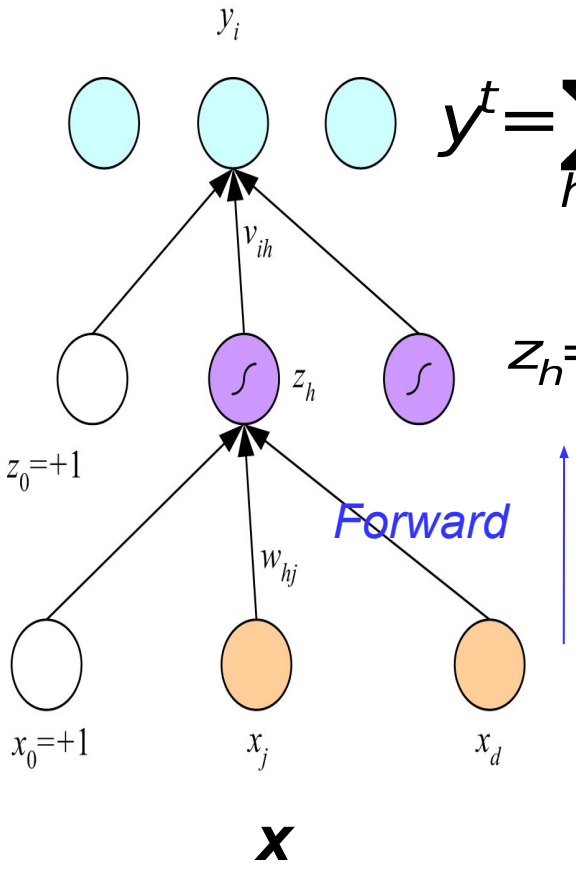


$$y_i = \boldsymbol{v}_i^T \boldsymbol{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}\left(\boldsymbol{w}_h^T \boldsymbol{x}\right) = \frac{1}{1+\exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial z_h}\frac{\partial z_h}{\partial w_{hj}}$$

Training multilayer perceptron



$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_0$$

$$z_h = \text{sigmoid}(w_h^T x)$$

Forward

$$E(W,v|X) = \frac{1}{2}\sum_t (r^t - y^t)^2$$

$$\Delta v_h = \sum_t (r^t - y^t) z_h^t$$

Backward

$$\Delta w_{hj} = -\eta \frac{\partial E}{\partial w_{hj}}$$

$$-\eta \sum_t \frac{\partial E}{\partial y^t} \frac{\partial y^t}{\partial z_h^t} \frac{\partial z_h^t}{\partial w_{hj}}$$

$$-\eta \sum_t -(r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

$$\eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t$$

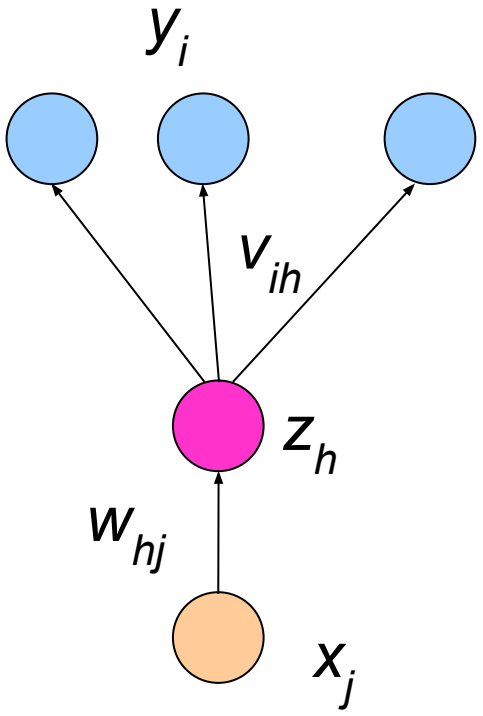
Regression with Multiple Outputs

$$E(\mathbf{W},\mathbf{V}\mid X)=\frac{1}{2}\sum_t\sum_i\left(r_i^t-y_i^t\right)^2$$

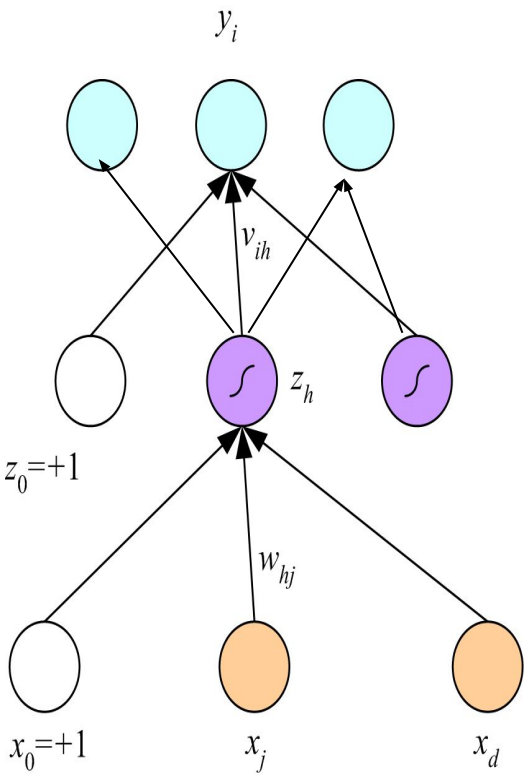
$$y_i^t=\sum_{h=1}^{H}v_{ih}z_h^t+v_{i0}$$

$$\Delta v_{ih}=\eta\sum_t\left(r_i^t-y_i^t\right)z_h^t$$

$$\Delta w_{hj}=\eta\sum_t\left[\sum_i\left(r_i^t-y_i^t\right)v_{ih}\right]z_h^t\left(1-z_h^t\right)x_j^t$$



Algorithm for backpropagation



Initialize all v_{ih} and w_{hj} to rand$(-0.01, 0.01)$

Repeat

For all $(\boldsymbol{x}^t, r^t) \in \mathcal{X}$ in random order

For $h = 1, \ldots, H$

$z_h \leftarrow \text{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x}^t)$

For $i = 1, \ldots, K$

$y_i = \boldsymbol{v}_i^T \boldsymbol{z}$

For $i = 1, \ldots, K$

$\Delta \boldsymbol{v}_i = \eta(r_i^t - y_i^t)\boldsymbol{z}$

For $h = 1, \ldots, H$

$\Delta \boldsymbol{w}_h = \eta(\sum_i (r_i^t - y_i^t) v_{ih}) z_h (1 - z_h) \boldsymbol{x}^t$

For $i = 1, \ldots, K$

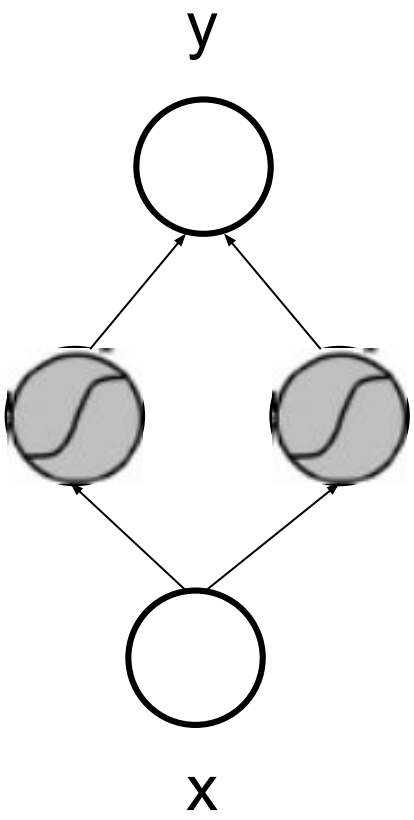
$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \Delta \boldsymbol{v}_i$

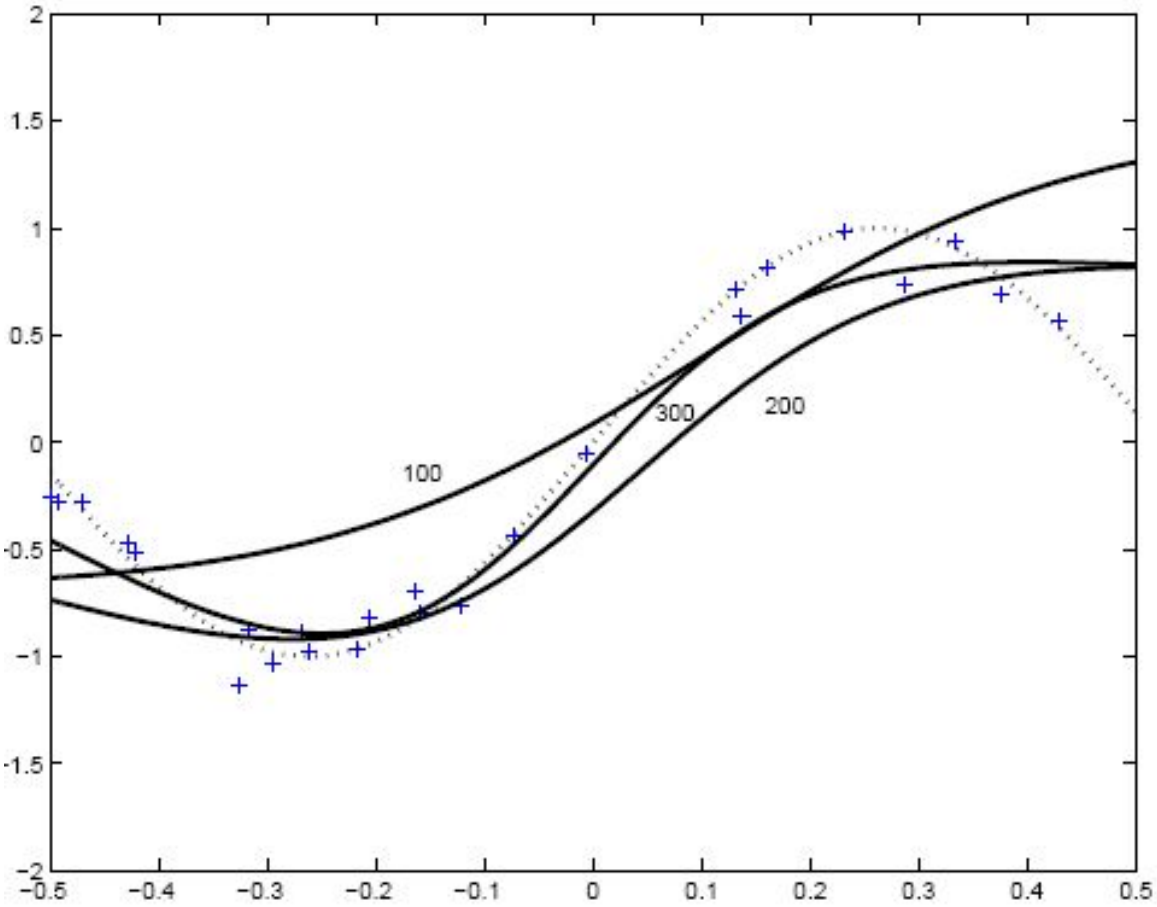
For $h = 1, \ldots, H$

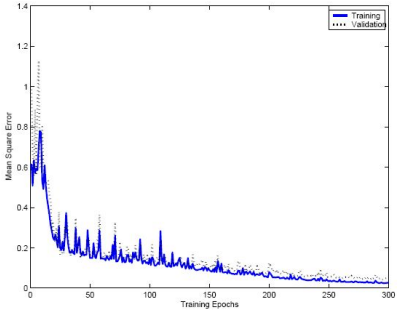
$\boldsymbol{w}_h \leftarrow \boldsymbol{w}_h + \Delta \boldsymbol{w}_h$

Until convergence

Sample training data, $y^t = \sin(6x)+N(0,0.1)$

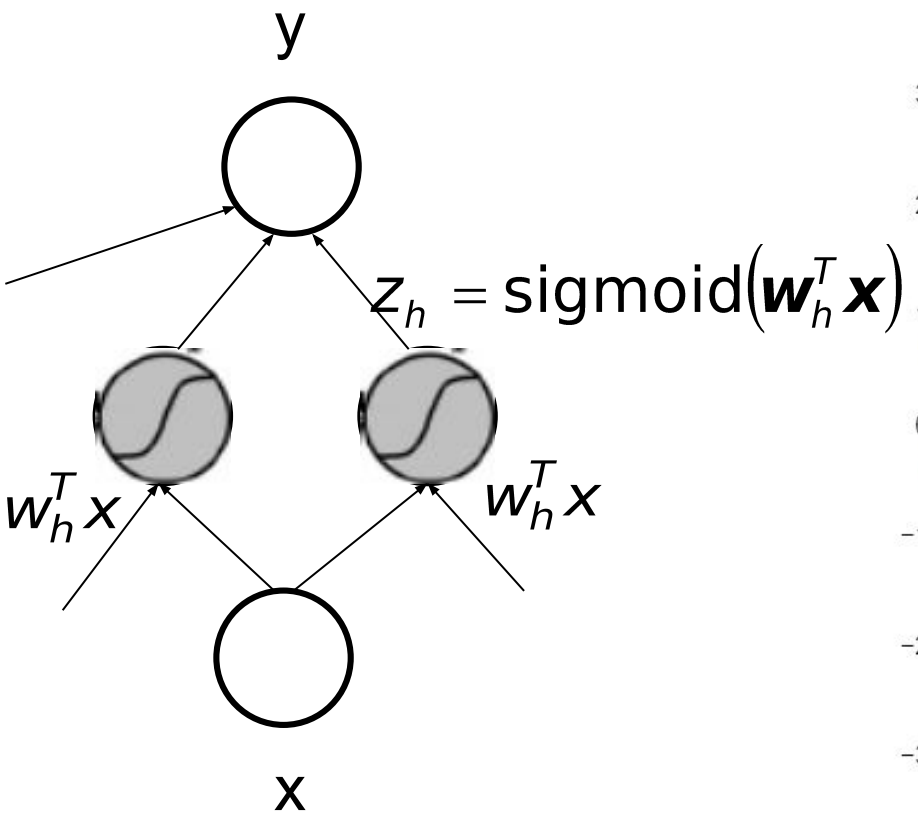


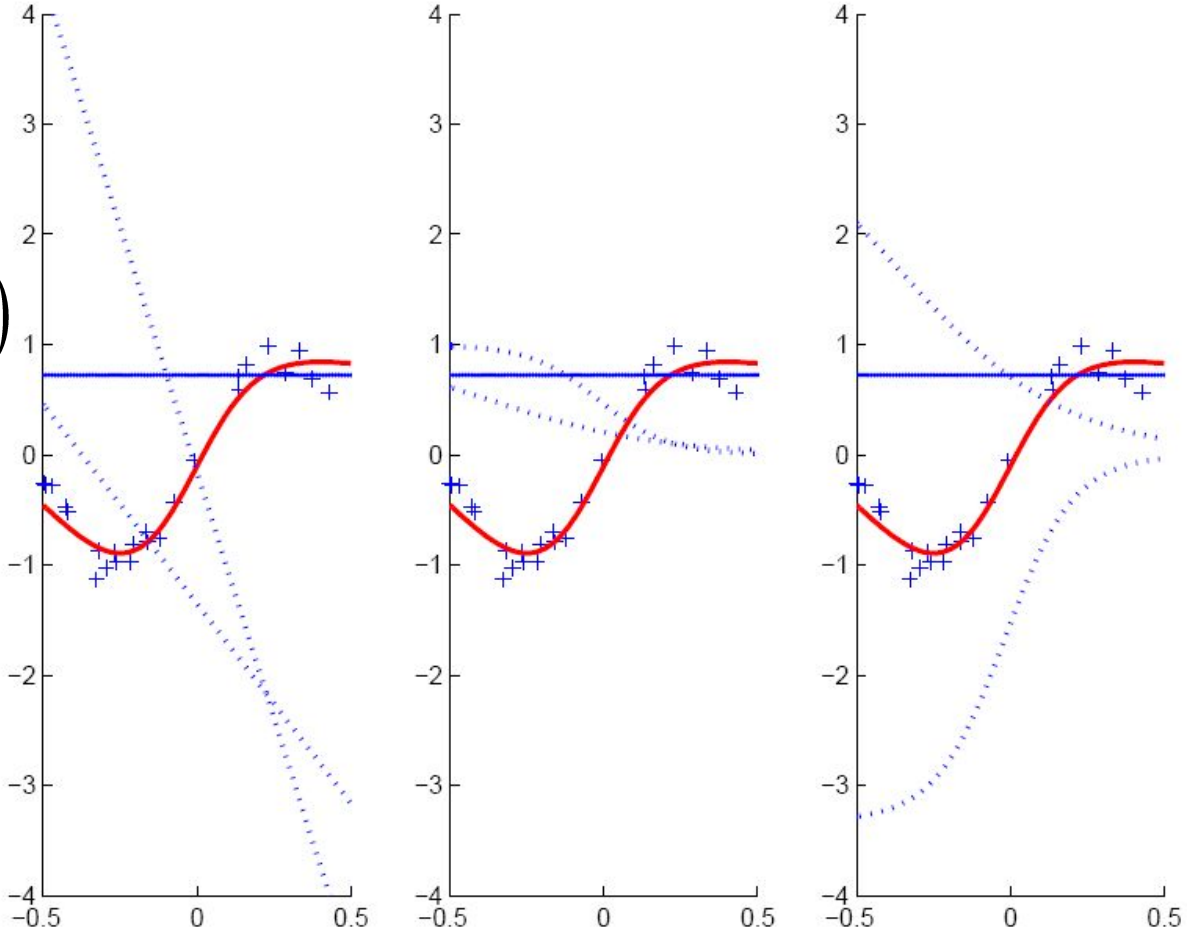




Sample training data, $y^t = \sin(6x)+N(0,0.1)$

$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_0$$





Two-Class Discrimination

- One sigmoid output

$$y^t = \text{sigmoid}\left(\sum_{h=1}^{H} v_h z_h^t + v_0\right)$$

$$E(\mathbf{W},\mathbf{v} \mid \mathcal{X}) = -\sum_t r^t \log y^t + (1-r^t)\log(1-y^t)$$

$$\Delta v_h = \eta \sum_t (r^t - y^t) z_h^t$$

$$\Delta w_{hj} = \eta \sum_t (r^t - y^t) v_h z_h^t (1-z_h^t) x_j^t$$

Parametric Discrimination Revisited

In chapter 5, we saw that if the class densities, $p(\mathbf{x}|C_i)$, are Gaussian and share a common covariance matrix, the discriminant function is linear

$$g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

where the parameters can be analytically calculated

Let us again see the special case where there are two classes: We define $y \equiv P(C_1|\mathbf{x})$ and $p(C_2|\mathbf{x}) = 1 - y$. Then in classification, we

$$\text{choose } C_1 \text{ if } \begin{cases} y > 0.5 \\ \frac{y}{1-y} > 1 \\ \log \frac{y}{1-y} > 0 \end{cases} \quad \text{and } C_2 \text{ otherwise}$$

$\log y/(1-y)$ is known as the *logit* transformation or *log odds* of y. ...

$$\text{logit}(P(C_1|\mathbf{x})) = \log \frac{P(C_1|\mathbf{x})}{1 - P(C_1|\mathbf{x})} = \mathbf{w}^T \mathbf{x} + w_0$$

The inverse of logit

$$\log \frac{P(C_1|\mathbf{x})}{1 - P(C_1|\mathbf{x})} = \mathbf{w}^T \mathbf{x} + w_0$$

is the *logistic* function, also called the *sigmoid* function (see figure 10.5):

$$P(C_1|\mathbf{x}) = \text{sigmoid}(\mathbf{w}^T \mathbf{x} + w_0) = \frac{1}{1 + \exp[-(\mathbf{w}^T \mathbf{x} + w_0)]}$$

Logistic Discrimination / Logistic Regression

In *logistic discrimination*, we do not model the class-conditional densities, $p(\mathbf{x}|C_i)$, but rather their ratio. Let us again start with two classes and assume that the log likelihood ratio is linear:

$$\log \frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} = \mathbf{w}^T\mathbf{x} + w_0^o$$

Rearranging terms, we get the sigmoid function

$$y = \hat{P}(C_1|\mathbf{x}) = \frac{1}{1 + \exp[-(\mathbf{w}^T\mathbf{x} + w_0)]}$$

We assume r^t, given $\mathbf{x}^t$, is Bernoulli with probability $y^t \equiv P(C_1|\mathbf{x}^t)$ as calculated in equation 10.21:

$$r^t|\mathbf{x}^t \sim \text{Bernoulli}(y^t)$$

$$l(\mathbf{w}, w_0|\mathcal{X}) = \prod_t (y^t)^{(r^t)}(1 - y^t)^{(1-r^t)}$$

$$E(\mathbf{w}, w_o|\mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t)\log(1 - y^t)$$

$$\begin{aligned}
\Delta w_j &= -\eta \frac{\partial E}{\partial w_j} = \eta \sum_t \left(\frac{r^t}{y^t} - \frac{1 - r^t}{1 - y^t}\right) y^t(1 - y^t)x_j^t \\
&= \eta \sum_t (r^t - y^t)x_j^t, j = 1, \ldots, d \\
\Delta w_0 &= -\eta \frac{\partial E}{\partial w_0} = \eta \sum_t (r^t - y^t)
\end{aligned}$$

Improving Convergence: Momentum

- Successive Δw values may be so different that large oscillations may occur and slow convergence
- Take a running average by incorporating the previous update

$$\Delta w_i^t = -\eta \frac{\partial E^t}{\partial w_i} + \alpha \Delta w_i^{t-1}$$

t is time index (epoch number or iteration number

- get an effect of averaging and smooth the trajectory during convergence.

Improving Convergence: Adaptive Learning Rate

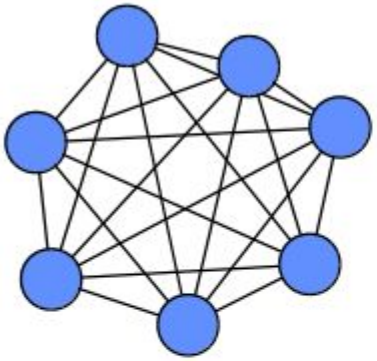
- adaptive for faster convergence, where it is kept large when learning takes place and is decreased when learning slows down
- Increase learning rate by a constant amount if the error on the training set decreases and decrease it geometrically if it increases

$$\Delta\eta = \begin{cases} +a & \text{if } E^{t+\tau} < E^t \\ -b\eta & \text{otherwise} \end{cases}$$

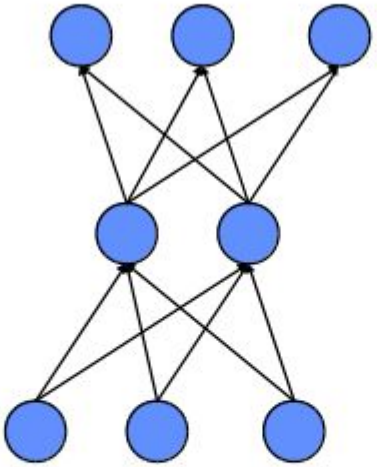
Practice

Scale inputs and outputs

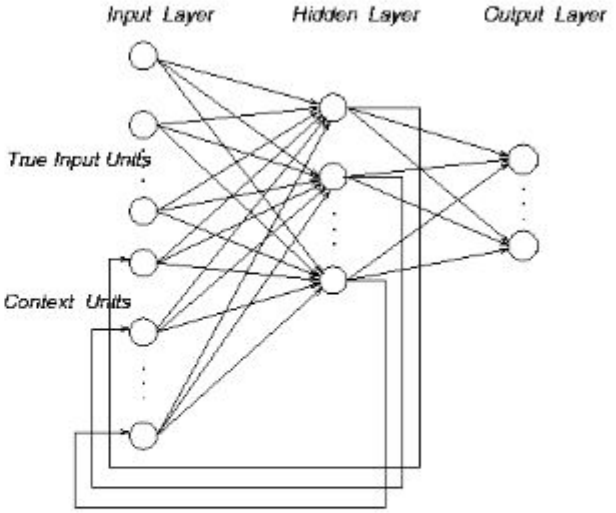
Topologies of Neural Networks



completely connected



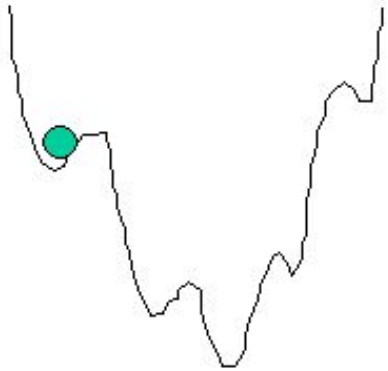
feedforward (directed, a-cyclic)

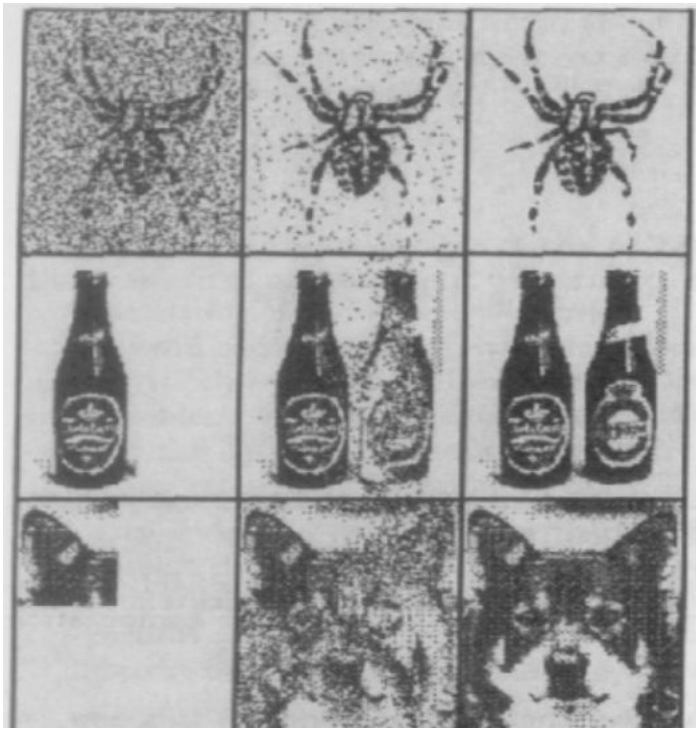


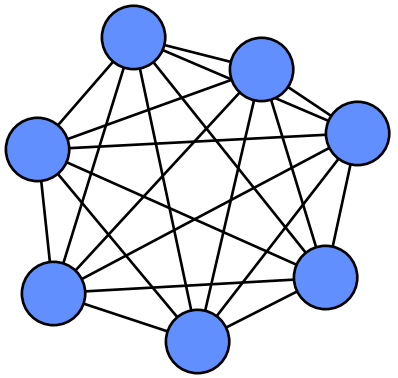
recurrent (feedback connections)

Hopfield Networks

- Act as “autoassociative” memories to store patterns
- Network converges to local minima which store different patterns.

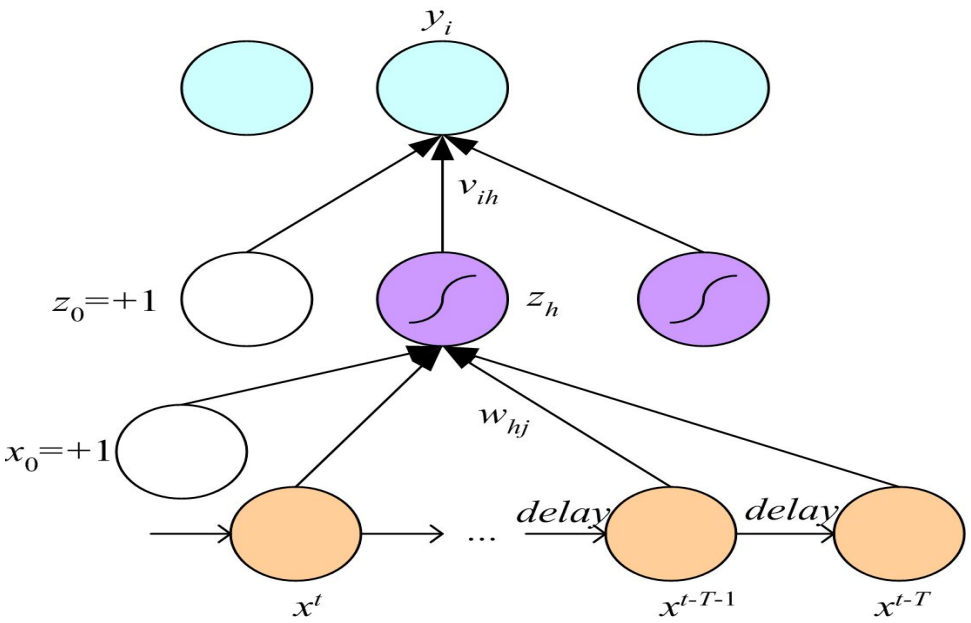


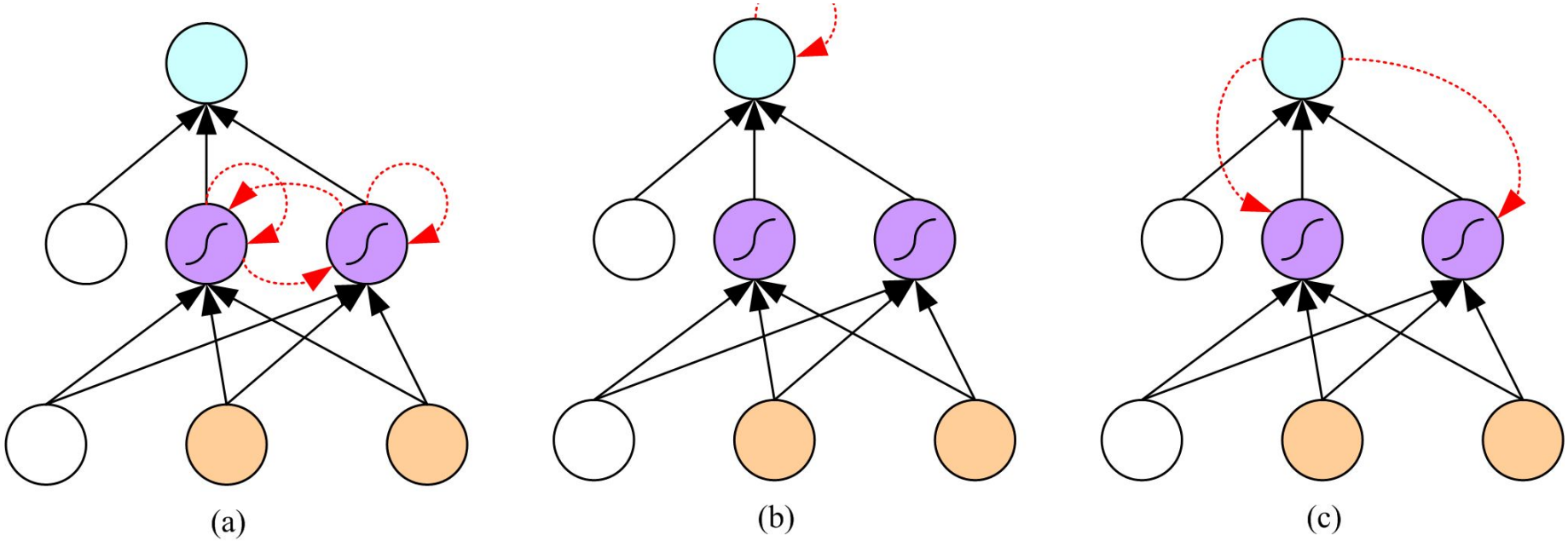




completely connected

Time-delay Neural Networks





Recurrent Networks