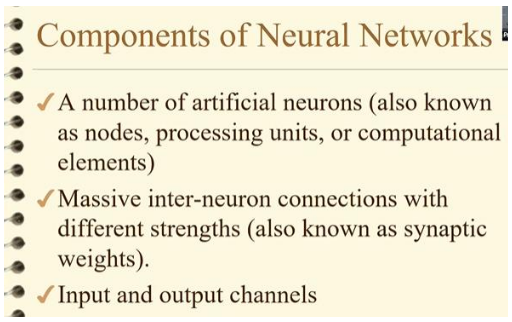
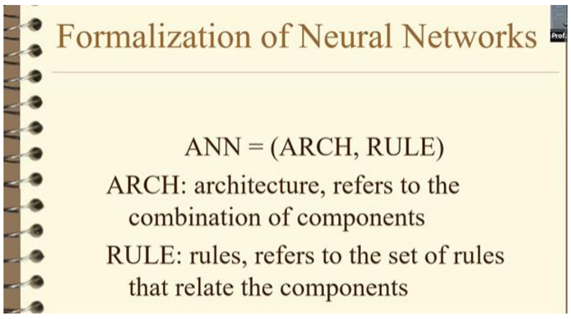


Components of Neural Networks

- ✓ A number of artificial neurons (also known as nodes, processing units, or computational elements)
- ✓ Massive inter-neuron connections with different strengths (also known as synaptic weights).
- ✓ Input and output channels

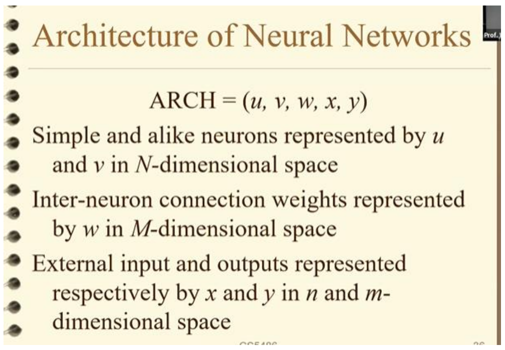


Formalization of Neural Networks

ANN = (ARCH, RULE)

ARCH: architecture, refers to the combination of components

RULE: rules, refers to the set of rules that relate the components



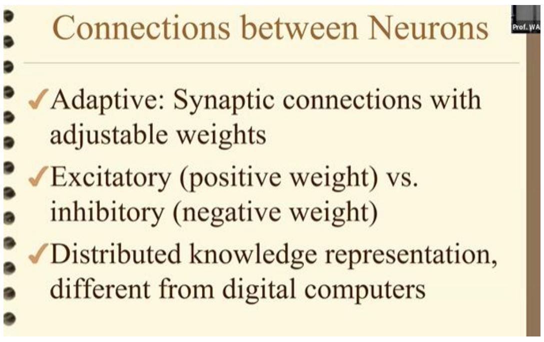
Architecture of Neural Networks

ARCH = (u, v, w, x, y)

Simple and alike neurons represented by u and v in N-dimensional space

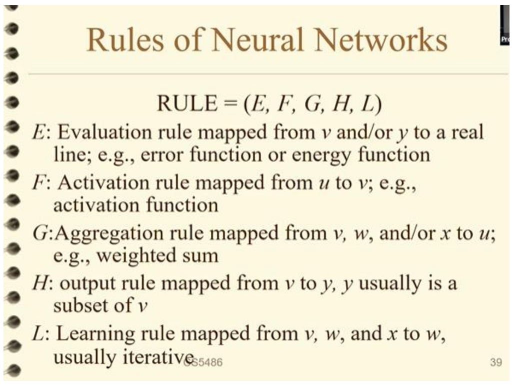
Inter-neuron connection weights represented by w in M-dimensional space

External input and outputs represented respectively by x and y in n and m-dimensional space



Connections between Neurons

- ✓ Adaptive: Synaptic connections with adjustable weights
- ✓ Excitatory (positive weight) vs. inhibitory (negative weight)
- ✓ Distributed knowledge representation, different from digital computers



Rules of Neural Networks

RULE = (E, F, G, H, L)

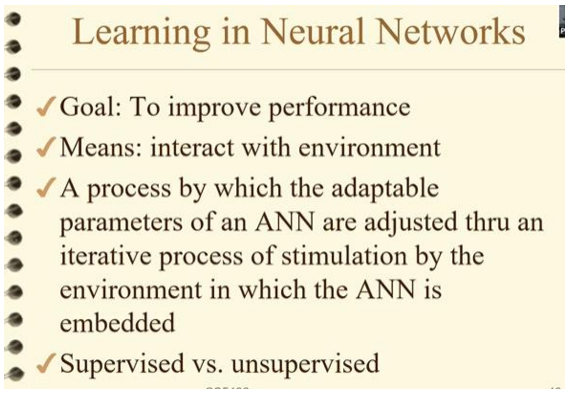
E: Evaluation rule mapped from v and/or y to a real line; e.g., error function or energy function

F: Activation rule mapped from u to v; e.g., activation function

G:Aggregation rule mapped from v, w, and/or x to u; e.g., weighted sum

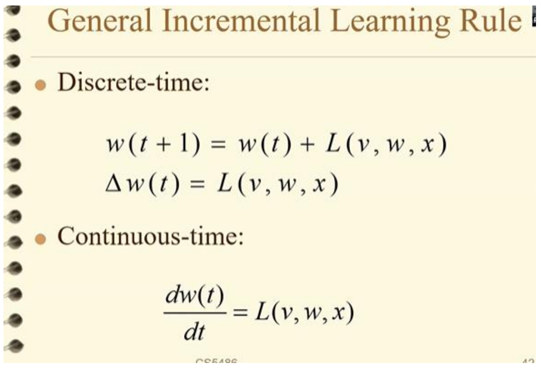
H: output rule mapped from v to y, y usually is a subset of v

L: Learning rule mapped from v, w, and x to w, usually iterative



Learning in Neural Networks

- ✓ Goal: To improve performance
- ✓ Means: interact with environment
- ✓ A process by which the adaptable parameters of an ANN are adjusted thru an iterative process of stimulation by the environment in which the ANN is embedded
- ✓ Supervised vs. unsupervised



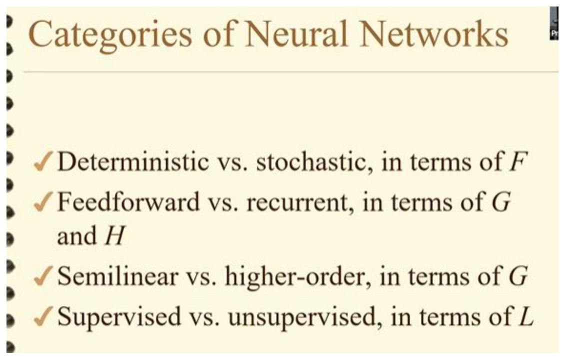
General Incremental Learning Rule

- Discrete-time:

$$w(t+1) = w(t) + L(v, w, x)$$
$$\Delta w(t) = L(v, w, x)$$

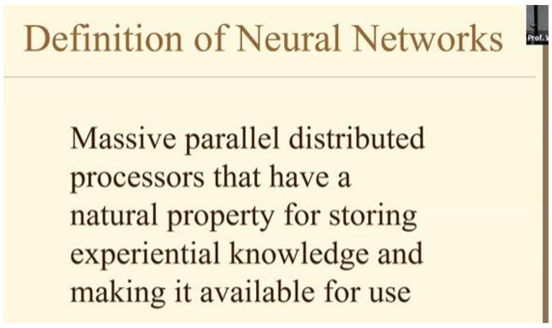
- Continuous-time:

$$\frac{dw(t)}{dt} = L(v, w, x)$$



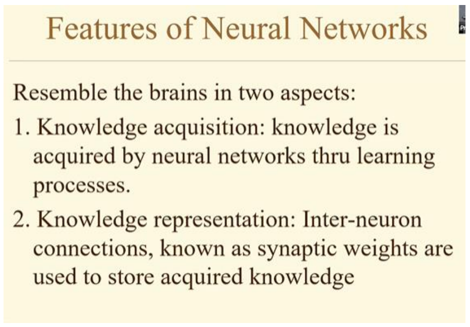
Categories of Neural Networks

- ✓ Deterministic vs. stochastic, in terms of F
- ✓ Feedforward vs. recurrent, in terms of G and H
- ✓ Semilinear vs. higher-order, in terms of G
- ✓ Supervised vs. unsupervised, in terms of L



Definition of Neural Networks

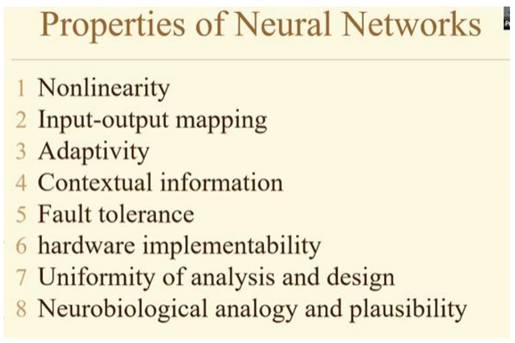
Massive parallel distributed processors that have a natural property for storing experiential knowledge and making it available for use



Features of Neural Networks

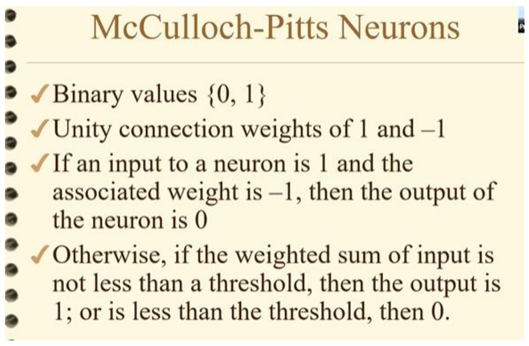
Resemble the brains in two aspects:

1. Knowledge acquisition: knowledge is acquired by neural networks thru learning processes.
2. Knowledge representation: Inter-neuron connections, known as synaptic weights are used to store acquired knowledge



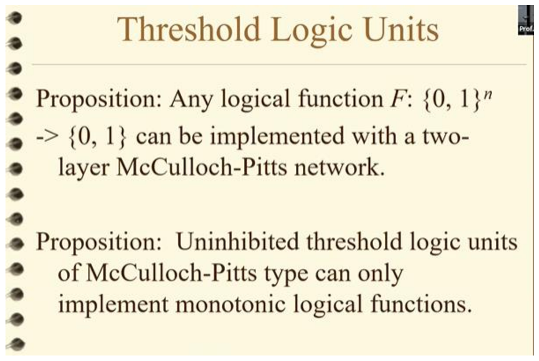
Properties of Neural Networks

1 Nonlinearity
2 Input-output mapping
3 Adaptivity
4 Contextual information
5 Fault tolerance
6 hardware implementability
7 Uniformity of analysis and design
8 Neurobiological analogy and plausibility



McCulloch-Pitts Neurons

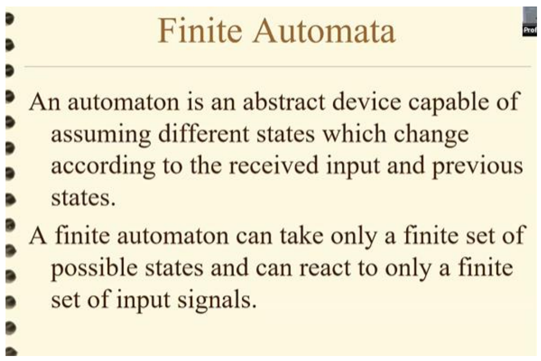
- ✓ Binary values {0, 1}
- ✓ Unity connection weights of 1 and –1
- ✓ If an input to a neuron is 1 and the associated weight is –1, then the output of the neuron is 0
- ✓ Otherwise, if the weighted sum of input is not less than a threshold, then the output is 1; or is less than the threshold, then 0.



Threshold Logic Units

Proposition: Any logical function F: $\{0, 1\}^n$ -> {0, 1} can be implemented with a two-layer McCulloch-Pitts network.

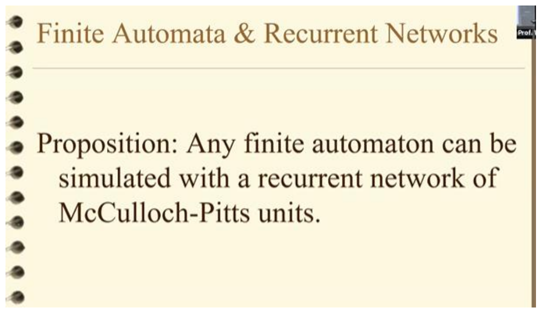
Proposition: Uninhibited threshold logic units of McCulloch-Pitts type can only implement monotonic logical functions.



Finite Automata

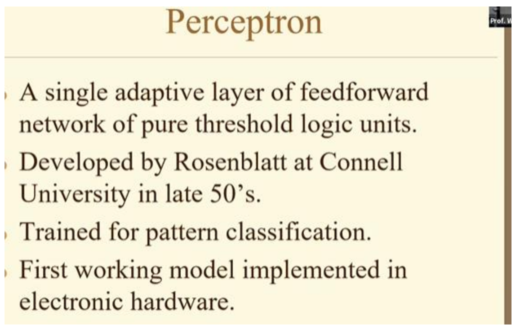
An automaton is an abstract device capable of assuming different states which change according to the received input and previous states.

A finite automaton can take only a finite set of possible states and can react to only a finite set of input signals.



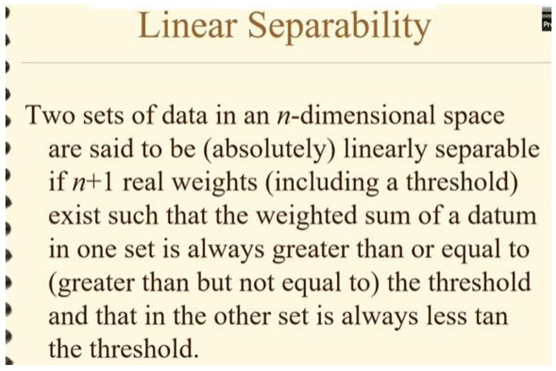
Finite Automata & Recurrent Networks

Proposition: Any finite automaton can be simulated with a recurrent network of McCulloch-Pitts units.



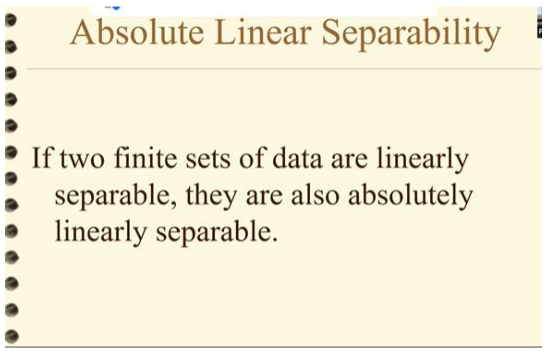
Perceptron

- A single adaptive layer of feedforward network of pure threshold logic units.
- Developed by Rosenblatt at Connell University in late 50's.
- Trained for pattern classification.
- First working model implemented in electronic hardware.



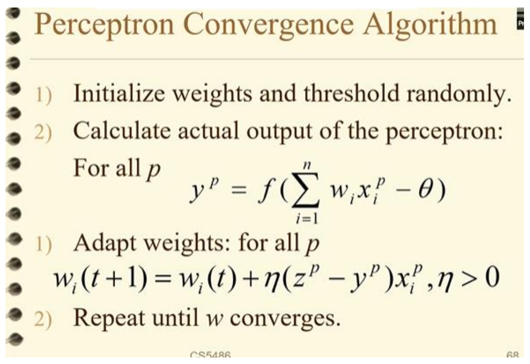
Linear Separability

Two sets of data in an n-dimensional space are said to be (absolutely) linearly separable if n+1 real weights (including a threshold) exist such that the weighted sum of a datum in one set is always greater than or equal to (greater than but not equal to) the threshold and that in the other set is always less tan the threshold.



Absolute Linear Separability

If two finite sets of data are linearly separable, they are also absolutely linearly separable.



Perceptron Convergence Algorithm

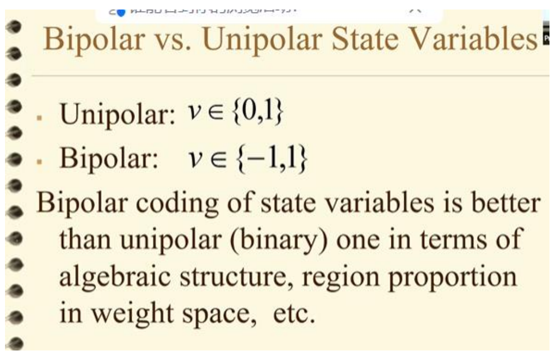
1) Initialize weights and threshold randomly.
2) Calculate actual output of the perceptron:
   For all p

$$y^p = f(\sum_{i=1}^{n} w_i x_i^p - \theta)$$

1) Adapt weights: for all p

$$w_i(t+1) = w_i(t) + \eta(z^p - y^p)x_i^p, \eta > 0$$

2) Repeat until w converges.



Bipolar vs. Unipolar State Variables

- Unipolar: $v \in \{0,1\}$
- Bipolar: $v \in \{-1,1\}$

Bipolar coding of state variables is better than unipolar (binary) one in terms of algebraic structure, region proportion in weight space, etc.

ADALINE

- A single adaptive layer of feedforward network of linear elements.
- Full name: Adaptive linear elements.
- Developed by Widrow and Hoff at Stanford University in early 60's.
- Trained using a learning algorithm called Delta Rule or Least Mean Squares (LMS) Algorithm.

LMS Learning Algorithm

1) Initialize weights and threshold randomly.
2) Calculate actual output of the ADALINE:

$$y = \alpha \left(\sum_{i=1}^{n} w_i x_i - \theta \right), \quad \alpha > 0$$

3) Adapt weights:

$$w_i(t+1) = w_i(t) + \sum_p \eta (z^p - y^p) x_i^p, \eta > 0$$

4) Repeat until w converges

Gradient Descent Learning Algorithms

$$E = \sum_p E_p = \sum_p (z^p - y^p)^2$$

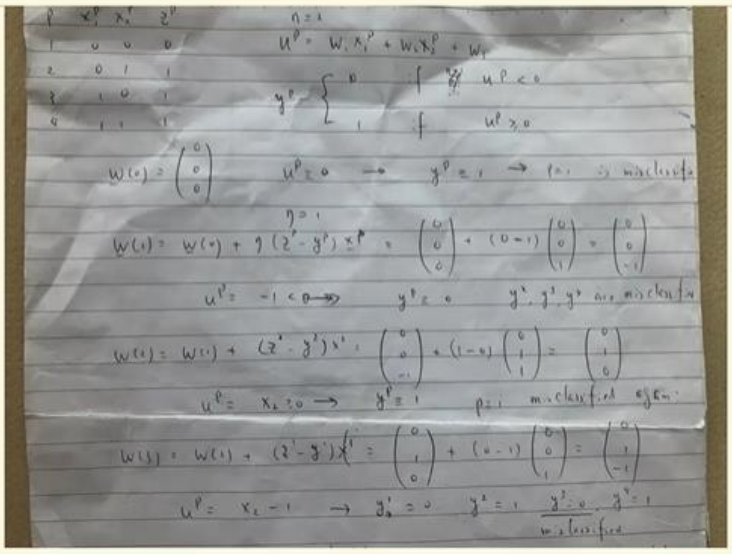
$$w(t+1) = w(t) - \eta \nabla E_w$$

$$\Delta w(t) = -\eta \nabla E_w$$

$$\nabla E_w = (\partial E / \partial w_1, \partial E / \partial w_2, \ldots, \partial E / \partial w_M)^T.$$

Training Modes

- ✓ Series mode: input training sample pairs one by one orderly or randomly.
- ✓ Batch mode: input training sample pairs in the whole training set at each iteration.
- ✓ Perceptron learning: either sequential or batch mode.
- ✓ ADALINE training: batch mode only.



p	x_1^p	x_2^p	z^p
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

$\eta = 1$

$u^p = w_1 x_1^p + w_2 x_2^p + w_3$

$$y^p = \begin{cases} 0 & \text{if } u^p < 0 \\ 1 & \text{if } u^p \geq 0 \end{cases}$$

$W(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ $\quad u^p = 0 \rightarrow y^p = 1 \rightarrow$ $p=1$ is misclassified

$\eta = 1$

$W(1) = W(0) + \eta(z^p - y^p) x^p = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + (0-1) \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$

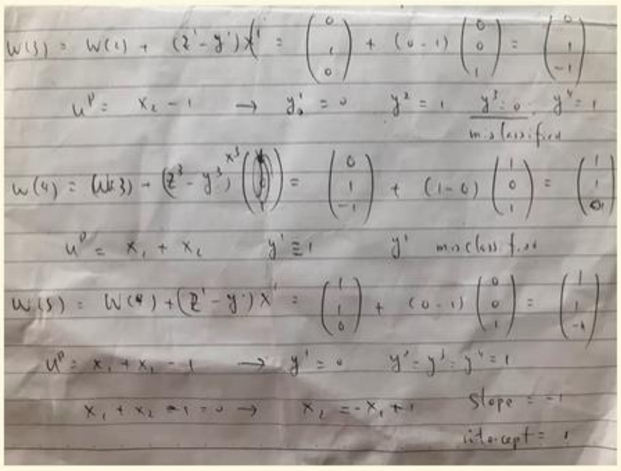
$u^p = -1 < 0 \rightarrow y^p = 0$ $\quad y^2, y^3, y^4$ are misclassified

$W(2) = W(1) + (z^2 - y^2) x^2 = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} + (1-0) \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$

$u^p = x_2 \geq 0 \rightarrow y^p = 1$ $\quad p=1$ misclassified again

$W(3) = W(2) + (z^1 - y^1) x^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + (0-1) \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$

$u^p = x_2 - 1 \rightarrow y_0^1 = 0 \quad y^2 = 1 \quad \underbrace{y^3 = 0}_{\text{misclassified}}, \; y^4 = 1$



$W(3) = W(2) + (z^1 - y^1) x^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + (0-1) \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}$

$u^p = x_2 - 1 \rightarrow y_0^1 = 0 \quad y^2 = 1 \quad \underbrace{y^3 = 0}_{\text{misclassified}}, \; y^4 = 1$

$W(4) = W(3) + (z^3 - y^3) x^3 = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} + (1-0) \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$

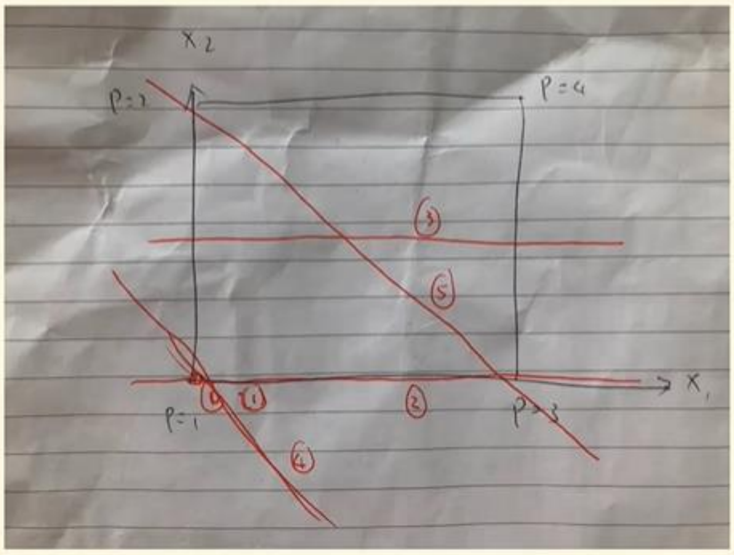
$u^p = x_1 + x_2$ $\quad y^1 = 1$ $\quad y^1$ misclassified

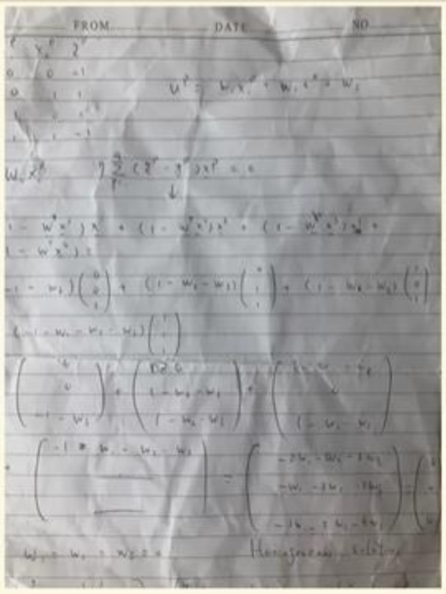
$W(5) = W(4) + (z^1 - y^1) x^1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} + (0-1) \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}$

$u^p = x_1 + x_2 - 1 \rightarrow y^1 = 0 \quad y^2 = y^3 = y^4 = 1$

$x_1 + x_2 - 1 = 0 \rightarrow x_2 = -x_1 + 1$ $\quad$ Slope $= -1$

intercept $= 1$




FROM DATE NO

x_1^p	x_2^p	ζ^p
0	0	-1
0	1	1
1	0	1
1	1	-1

$u^p = w_1 x_1^p + w_2 x_2^p + w_3$

$w_i x_i^p$

$\eta \sum_{p=1}^{4} (\zeta^p - y^p) x^p = 0$

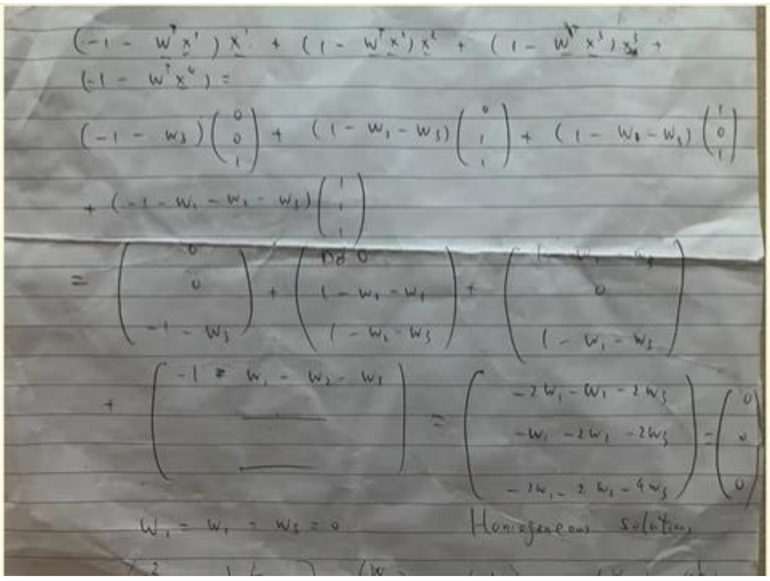
$\downarrow$

$(-1 - \underline{w}^T\underline{x}^1)\underline{x}^1 + (1 - \underline{w}^T\underline{x}^2)\underline{x}^2 + (1 - \underline{w}^T\underline{x}^3)\underline{x}^3 +$
$(-1 - \underline{w}^T\underline{x}^4) =$

$(-1 - w_3)\begin{pmatrix}0\\0\\1\end{pmatrix} + (1 - w_2 - w_3)\begin{pmatrix}0\\1\\1\end{pmatrix} + (1 - w_1 - w_3)\begin{pmatrix}1\\0\\1\end{pmatrix}$

$(-1 - w_1 - w_2 - w_3)\begin{pmatrix}1\\1\\1\end{pmatrix}$

$w_1 = w_2 = w_3 = 0$ Homogeneous solution


$(-1 - \underline{w}^T\underline{x}^1)\underline{x}^1 + (1 - \underline{w}^T\underline{x}^2)\underline{x}^2 + (1 - \underline{w}^T\underline{x}^3)\underline{x}^3 +$
$(-1 - \underline{w}^T\underline{x}^4) =$

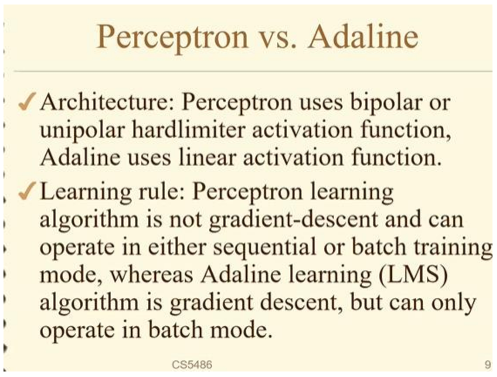
$$(-1 - w_3)\begin{pmatrix}0\\0\\1\end{pmatrix} + (1 - w_2 - w_3)\begin{pmatrix}0\\1\\1\end{pmatrix} + (1 - w_1 - w_3)\begin{pmatrix}1\\0\\1\end{pmatrix}$$

$$+ (-1 - w_1 - w_2 - w_3)\begin{pmatrix}1\\1\\1\end{pmatrix}$$

$$= \begin{pmatrix}0\\0\\-1-w_3\end{pmatrix} + \begin{pmatrix}0\\1-w_2-w_3\\1-w_2-w_3\end{pmatrix} + \begin{pmatrix}1-w_1-w_3\\0\\1-w_1-w_3\end{pmatrix}$$

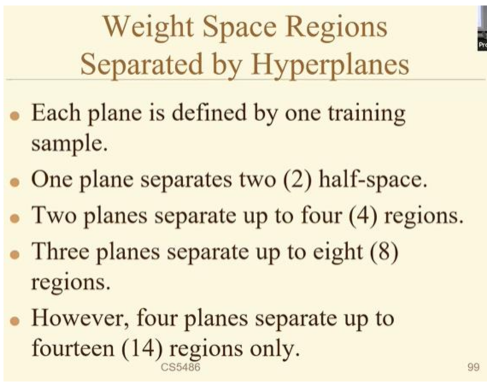
$$+ \begin{pmatrix}-1 - w_1 - w_2 - w_3\\ \_\_\_\_ \\ \_\_\_\_\end{pmatrix} = \begin{pmatrix}-2w_1 - w_2 - 2w_3\\ -w_1 - 2w_2 - 2w_3\\ -2w_1 - 2w_2 - 4w_3\end{pmatrix} = \begin{pmatrix}0\\0\\0\end{pmatrix}$$

$w_1 = w_2 = w_3 = 0$ Homogeneous solution



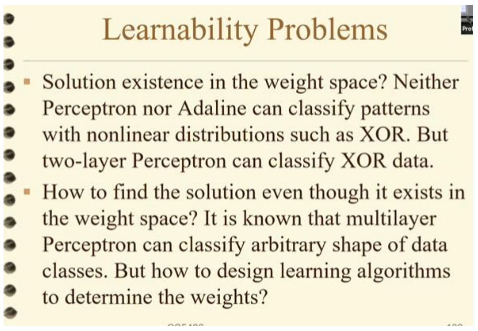
Perceptron vs. Adaline

- ✓ Architecture: Perceptron uses bipolar or unipolar hardlimiter activation function, Adaline uses linear activation function.
- ✓ Learning rule: Perceptron learning algorithm is not gradient-descent and can operate in either sequential or batch training mode, whereas Adaline learning (LMS) algorithm is gradient descent, but can only operate in batch mode.



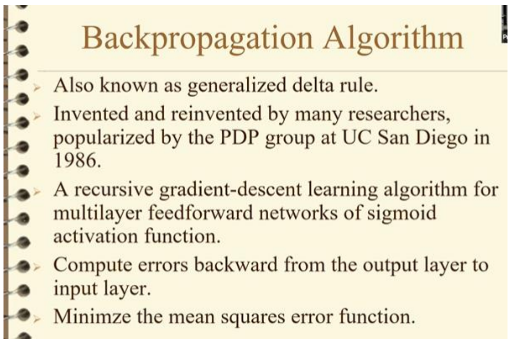
Weight Space Regions Separated by Hyperplanes

- Each plane is defined by one training sample.
- One plane separates two (2) half-space.
- Two planes separate up to four (4) regions.
- Three planes separate up to eight (8) regions.
- However, four planes separate up to fourteen (14) regions only.



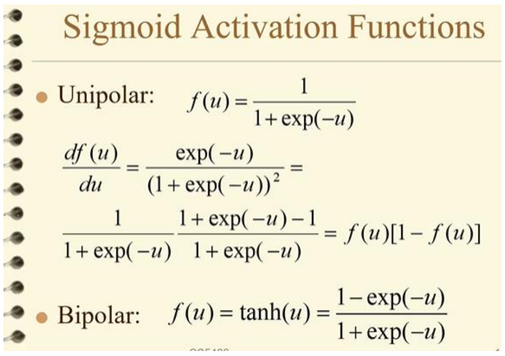
Learnability Problems

- Solution existence in the weight space? Neither Perceptron nor Adaline can classify patterns with nonlinear distributions such as XOR. But two-layer Perceptron can classify XOR data.
- How to find the solution even though it exists in the weight space? It is known that multilayer Perceptron can classify arbitrary shape of data classes. But how to design learning algorithms to determine the weights?



Backpropagation Algorithm

- Also known as generalized delta rule.
- Invented and reinvented by many researchers, popularized by the PDP group at UC San Diego in 1986.
- A recursive gradient-descent learning algorithm for multilayer feedforward networks of sigmoid activation function.
- Compute errors backward from the output layer to input layer.
- Minimze the mean squares error function.



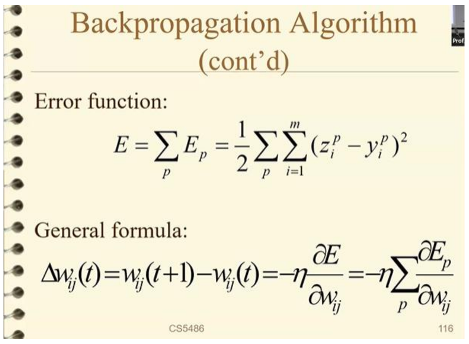
Sigmoid Activation Functions

- Unipolar: $f(u)=\frac{1}{1+\exp(-u)}$

$$\frac{df(u)}{du}=\frac{\exp(-u)}{(1+\exp(-u))^2}=$$

$$\frac{1}{1+\exp(-u)}\frac{1+\exp(-u)-1}{1+\exp(-u)}=f(u)[1-f(u)]$$

- Bipolar: $f(u)=\tanh(u)=\frac{1-\exp(-u)}{1+\exp(-u)}$



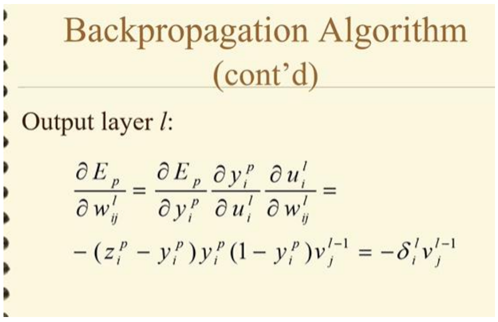
Backpropagation Algorithm (cont'd)

Error function:

$$E=\sum_p E_p=\frac{1}{2}\sum_p\sum_{i=1}^{m}(z_i^p-y_i^p)^2$$

General formula:

$$\Delta w_{ij}(t)=w_{ij}(t+1)-w_{ij}(t)=-\eta\frac{\partial E}{\partial w_{ij}}=-\eta\sum_p\frac{\partial E_p}{\partial w_{ij}}$$

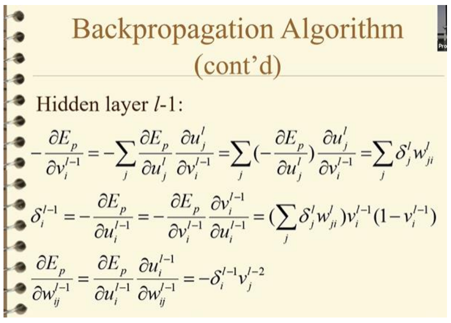


Backpropagation Algorithm (cont'd)

Output layer l:

$$\frac{\partial E_p}{\partial w_{ij}^l}=\frac{\partial E_p}{\partial y_i^p}\frac{\partial y_i^p}{\partial u_i^l}\frac{\partial u_i^l}{\partial w_{ij}^l}=$$

$$-(z_i^p-y_i^p)y_i^p(1-y_i^p)v_j^{l-1}=-\delta_i^l v_j^{l-1}$$



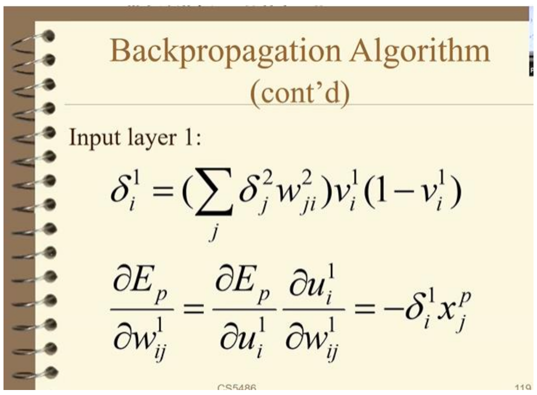
Backpropagation Algorithm (cont'd)

Hidden layer l-1:

$$-\frac{\partial E_p}{\partial v_i^{l-1}}=-\sum_j\frac{\partial E_p}{\partial u_j^l}\frac{\partial u_j^l}{\partial v_i^{l-1}}=\sum_j(-\frac{\partial E_p}{\partial u_j^l})\frac{\partial u_j^l}{\partial v_i^{l-1}}=\sum_j\delta_j^l w_{ji}^l$$

$$\delta_i^{l-1}=-\frac{\partial E_p}{\partial u_i^{l-1}}=-\frac{\partial E_p}{\partial v_i^{l-1}}\frac{\partial v_i^{l-1}}{\partial u_i^{l-1}}=(\sum_j\delta_j^l w_{ji}^l)v_i^{l-1}(1-v_i^{l-1})$$

$$\frac{\partial E_p}{\partial w_{ij}^{l-1}}=\frac{\partial E_p}{\partial u_i^{l-1}}\frac{\partial u_i^{l-1}}{\partial w_{ij}^{l-1}}=-\delta_i^{l-1}v_j^{l-2}$$

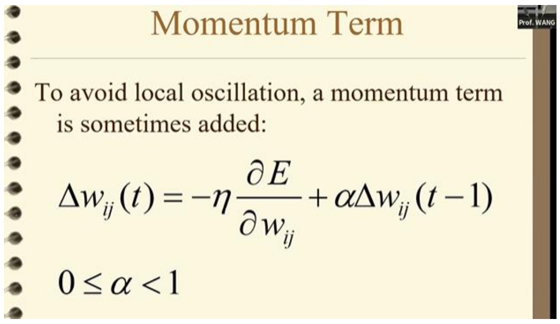


Backpropagation Algorithm (cont'd)

Input layer 1:

$$\delta_i^1 = (\sum_j \delta_j^2 w_{ji}^2) v_i^1 (1 - v_i^1)$$

$$\frac{\partial E_p}{\partial w_{ij}^1} = \frac{\partial E_p}{\partial u_i^1} \frac{\partial u_i^1}{\partial w_{ij}^1} = -\delta_i^1 x_j^p$$

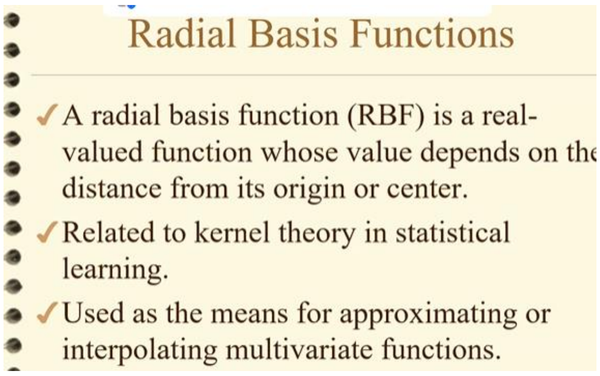


Momentum Term

To avoid local oscillation, a momentum term is sometimes added:

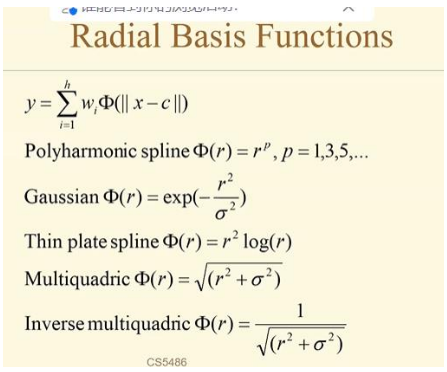
$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t-1)$$

$$0 \le \alpha < 1$$



Radial Basis Functions

- ✓ A radial basis function (RBF) is a real-valued function whose value depends on the distance from its origin or center.
- ✓ Related to kernel theory in statistical learning.
- ✓ Used as the means for approximating or interpolating multivariate functions.



Radial Basis Functions

$$y = \sum_{i=1}^{h} w_i \Phi(\| x - c \|)$$

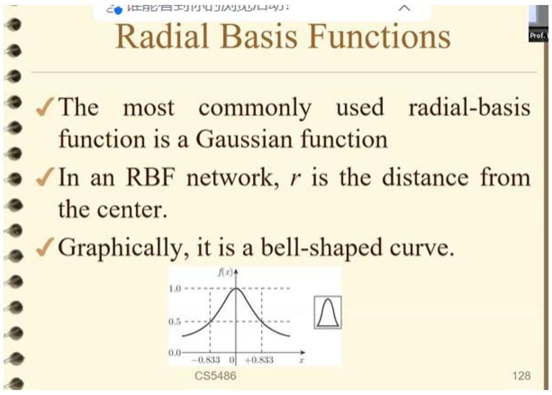
Polyharmonic spline $\Phi(r) = r^p,\ p = 1,3,5,\ldots$

Gaussian $\Phi(r) = \exp(-\frac{r^2}{\sigma^2})$

Thin plate spline $\Phi(r) = r^2 \log(r)$

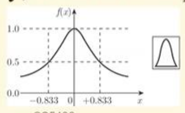
Multiquadric $\Phi(r) = \sqrt{(r^2 + \sigma^2)}$

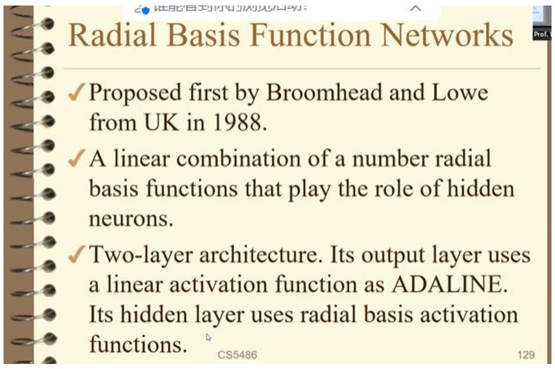
Inverse multiquadric $\Phi(r) = \frac{1}{\sqrt{(r^2 + \sigma^2)}}$



Radial Basis Functions

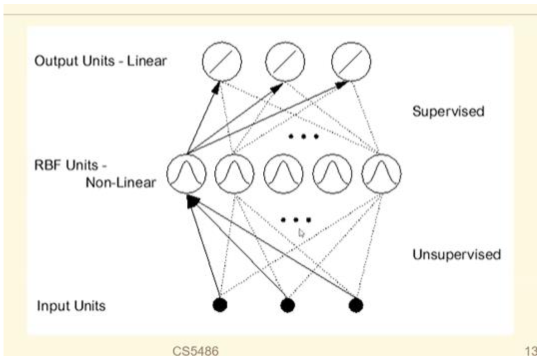
- ✓ The most commonly used radial-basis function is a Gaussian function
- ✓ In an RBF network, r is the distance from the center.
- ✓ Graphically, it is a bell-shaped curve.

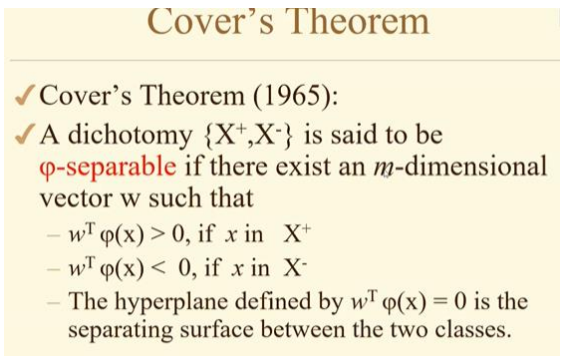




Radial Basis Function Networks

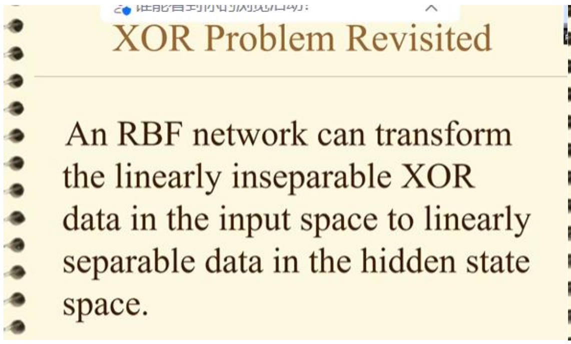
- ✓ Proposed first by Broomhead and Lowe from UK in 1988.
- ✓ A linear combination of a number radial basis functions that play the role of hidden neurons.
- ✓ Two-layer architecture. Its output layer uses a linear activation function as ADALINE. Its hidden layer uses radial basis activation functions.





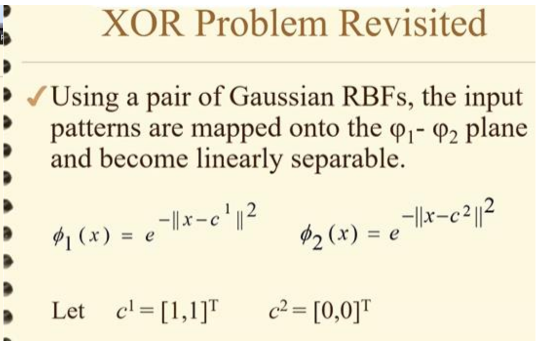
Cover's Theorem

- ✓ Cover's Theorem (1965):
- ✓ A dichotomy $\{X^+, X^-\}$ is said to be φ-separable if there exist an m-dimensional vector w such that
  - $w^T \varphi(x) > 0$, if x in X^+
  - $w^T \varphi(x) < 0$, if x in X^-
  - The hyperplane defined by $w^T \varphi(x) = 0$ is the separating surface between the two classes.



XOR Problem Revisited

An RBF network can transform the linearly inseparable XOR data in the input space to linearly separable data in the hidden state space.

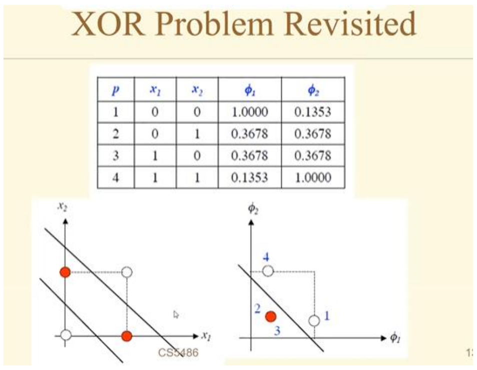


XOR Problem Revisited

✓ Using a pair of Gaussian RBFs, the input patterns are mapped onto the φ_1- φ_2 plane and become linearly separable.

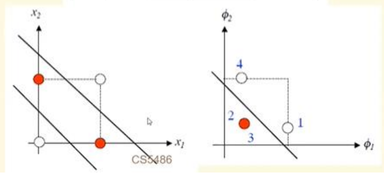
$$\phi_1(x) = e^{-\|x-c^1\|^2} \qquad \phi_2(x) = e^{-\|x-c^2\|^2}$$

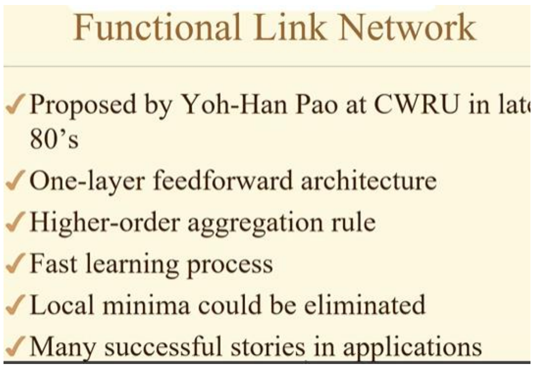
Let $c^1 = [1,1]^T$ $c^2 = [0,0]^T$



XOR Problem Revisited

p	x_1	x_2	ϕ_1	ϕ_2
1	0	0	1.0000	0.1353
2	0	1	0.3678	0.3678
3	1	0	0.3678	0.3678
4	1	1	0.1353	1.0000





Functional Link Network

✓ Proposed by Yoh-Han Pao at CWRU in lat
80's

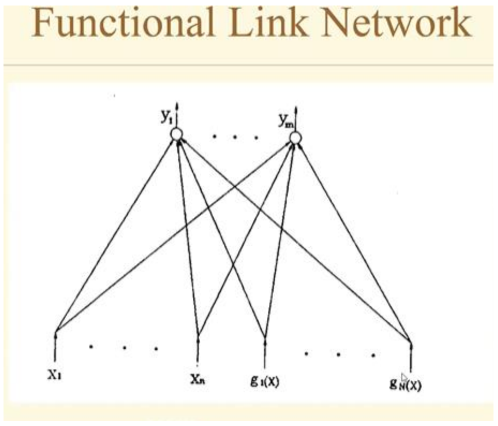
✓ One-layer feedforward architecture

✓ Higher-order aggregation rule

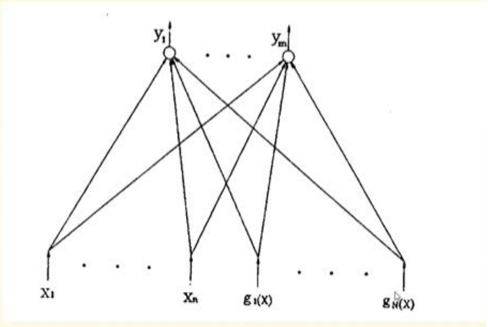
✓ Fast learning process

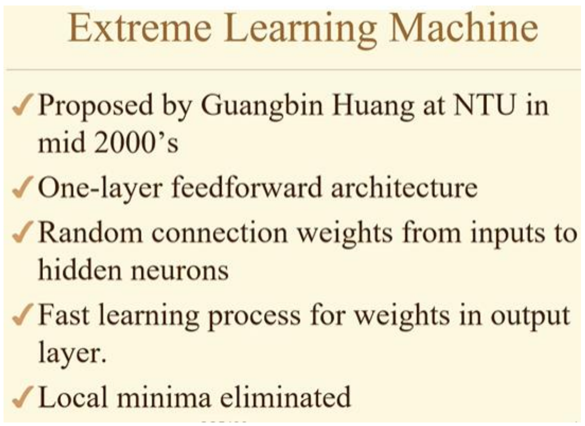
✓ Local minima could be eliminated

✓ Many successful stories in applications



Functional Link Network





Extreme Learning Machine

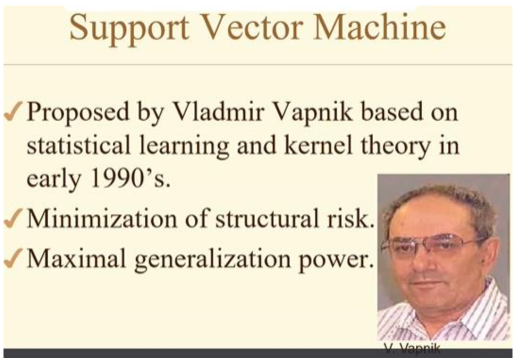
✓ Proposed by Guangbin Huang at NTU in mid 2000's

✓ One-layer feedforward architecture

✓ Random connection weights from inputs to hidden neurons

✓ Fast learning process for weights in output layer.

✓ Local minima eliminated



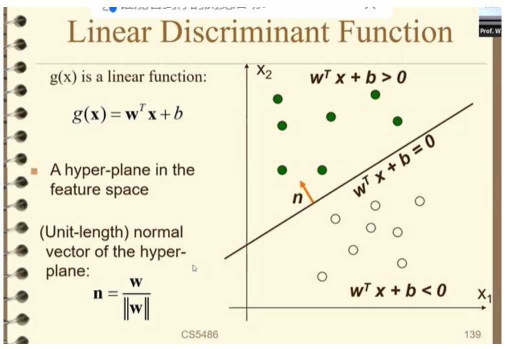
Support Vector Machine

✓ Proposed by Vladmir Vapnik based on statistical learning and kernel theory in early 1990's.

✓ Minimization of structural risk.

✓ Maximal generalization power.

V. Vapnik



Linear Discriminant Function

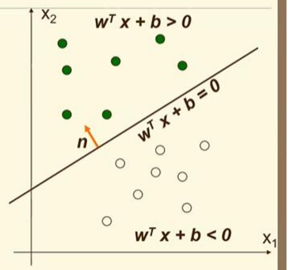
g(x) is a linear function:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

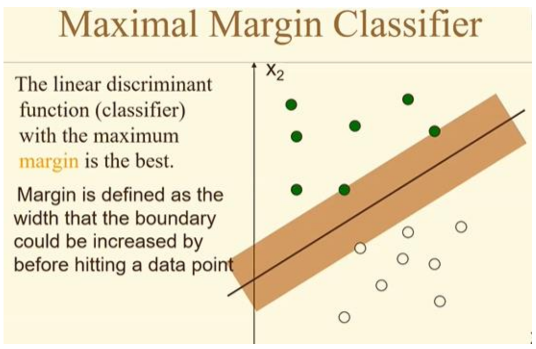
- A hyper-plane in the feature space

(Unit-length) normal vector of the hyper-plane:

$$\mathbf{n} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$



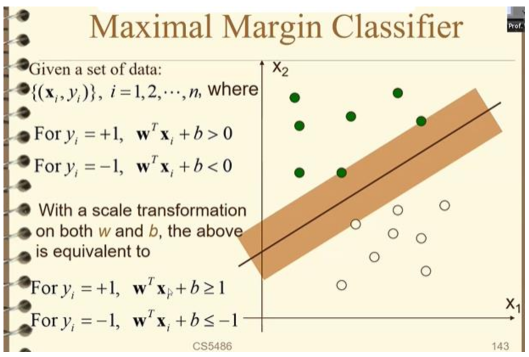
CS5486 139



Maximal Margin Classifier

The linear discriminant function (classifier) with the maximum margin is the best.

Margin is defined as the width that the boundary could be increased by before hitting a data point



Maximal Margin Classifier

Given a set of data: $\{(\mathbf{x}_i, y_i)\},\ i = 1, 2, \cdots, n$, where

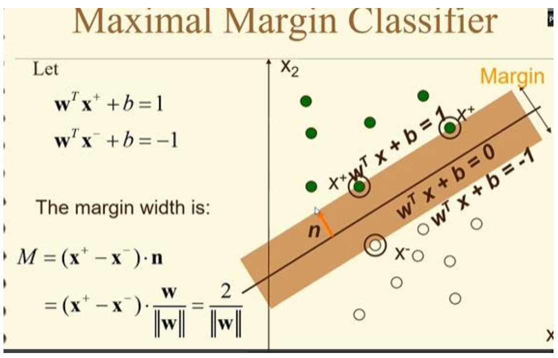
For $y_i = +1,\ \ \mathbf{w}^T\mathbf{x}_i + b > 0$

For $y_i = -1,\ \ \mathbf{w}^T\mathbf{x}_i + b < 0$

With a scale transformation on both w and b, the above is equivalent to

For $y_i = +1,\ \ \mathbf{w}^T\mathbf{x}_i + b \geq 1$

For $y_i = -1,\ \ \mathbf{w}^T\mathbf{x}_i + b \leq -1$



Maximal Margin Classifier

Let

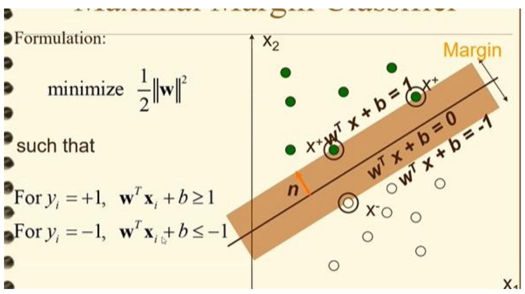
$\mathbf{w}^T\mathbf{x}^+ + b = 1$

$\mathbf{w}^T\mathbf{x}^- + b = -1$

The margin width is:

$$M = (\mathbf{x}^+ - \mathbf{x}^-)\cdot\mathbf{n}$$

$$= (\mathbf{x}^+ - \mathbf{x}^-)\cdot\frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$



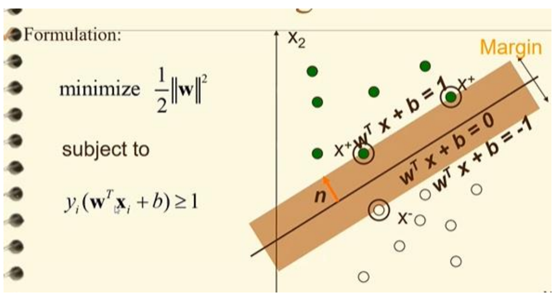
Formulation:

minimize $\frac{1}{2}\|\mathbf{w}\|^2$

such that

For $y_i = +1,\ \ \mathbf{w}^T\mathbf{x}_i + b \geq 1$

For $y_i = -1,\ \ \mathbf{w}^T\mathbf{x}_i + b \leq -1$

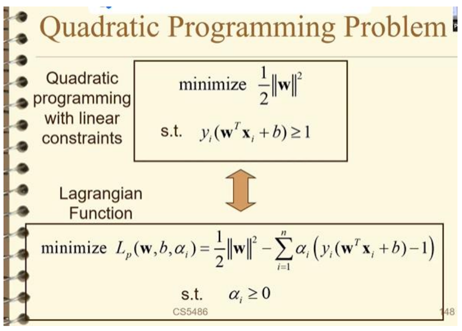


Formulation:

minimize $\frac{1}{2}\|\mathbf{w}\|^2$

subject to

$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$



Quadratic Programming Problem

Quadratic programming with linear constraints

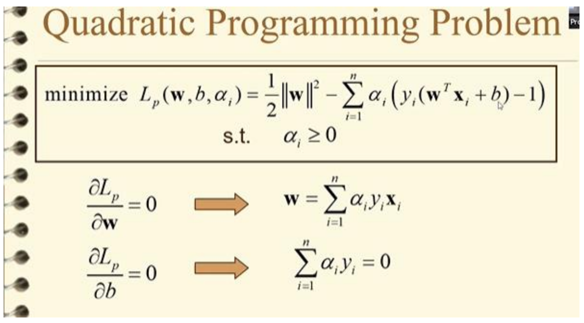
minimize $\frac{1}{2}\|\mathbf{w}\|^2$

s.t. $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Lagrangian Function

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n}\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right)$$

s.t. $\alpha_i \geq 0$



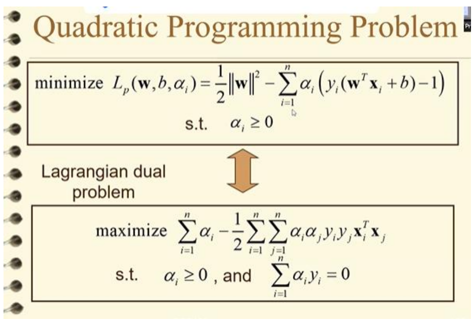
Quadratic Programming Problem

$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n}\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right)$$

s.t. $\alpha_i \geq 0$

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \quad\Rightarrow\quad \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L_p}{\partial b} = 0 \quad\Rightarrow\quad \sum_{i=1}^{n}\alpha_i y_i = 0$$



Quadratic Programming Problem

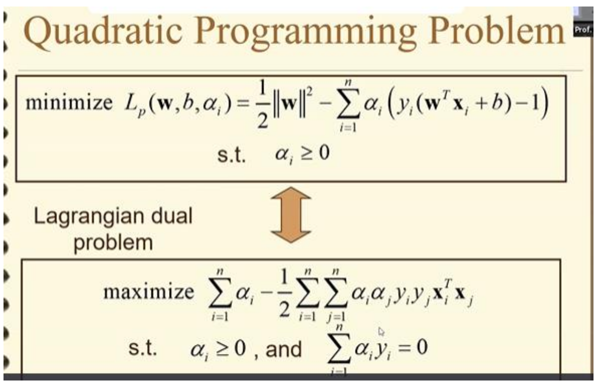
$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n}\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right)$$

s.t. $\alpha_i \geq 0$

Lagrangian dual problem

$$\text{maximize } \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

s.t. $\alpha_i \geq 0$, and $\sum_{i=1}^{n}\alpha_i y_i = 0$



Quadratic Programming Problem

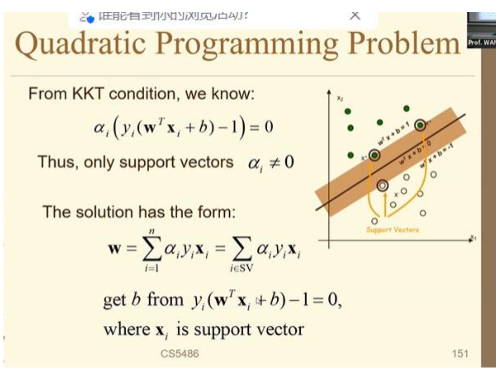
$$\text{minimize } L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n}\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right)$$

s.t. $\alpha_i \geq 0$

Lagrangian dual problem

$$\text{maximize } \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

s.t. $\alpha_i \geq 0$, and $\sum_{i=1}^{n}\alpha_i y_i = 0$



Quadratic Programming Problem

From KKT condition, we know:

$$\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\right) = 0$$

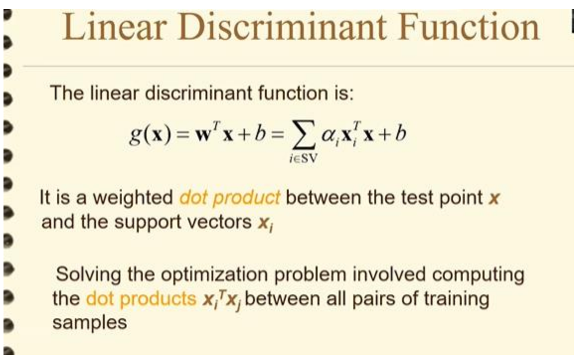
Thus, only support vectors $\alpha_i \neq 0$

The solution has the form:

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i = \sum_{i\in SV}\alpha_i y_i \mathbf{x}_i$$

get b from $y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 = 0$,

where $\mathbf{x}_i$ is support vector



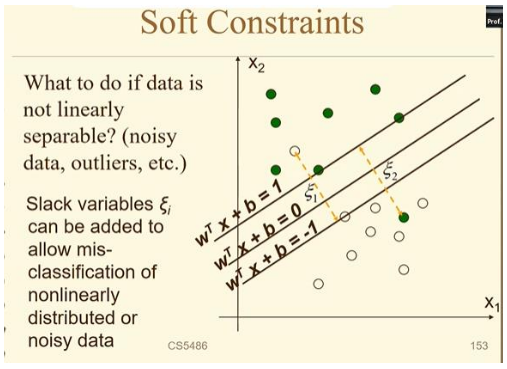
Linear Discriminant Function

The linear discriminant function is:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = \sum_{i\in SV} \alpha_i \mathbf{x}_i^T\mathbf{x} + b$$

It is a weighted *dot product* between the test point ***x*** and the support vectors ***x**$_i$*

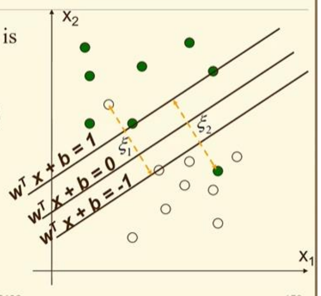
Solving the optimization problem involved computing the dot products $\mathbf{x}_i^T\mathbf{x}_j$ between all pairs of training samples

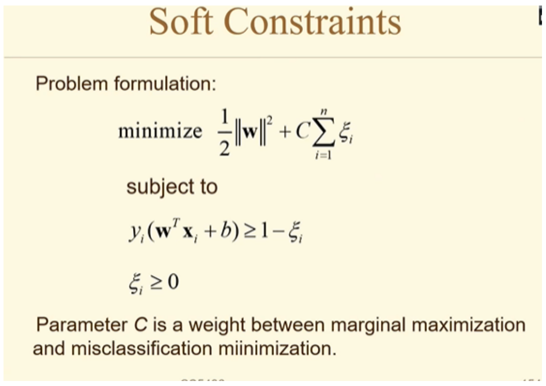


Soft Constraints

What to do if data is not linearly separable? (noisy data, outliers, etc.)

Slack variables ξ_i can be added to allow mis-classification of nonlinearly distributed or noisy data





Soft Constraints

Problem formulation:

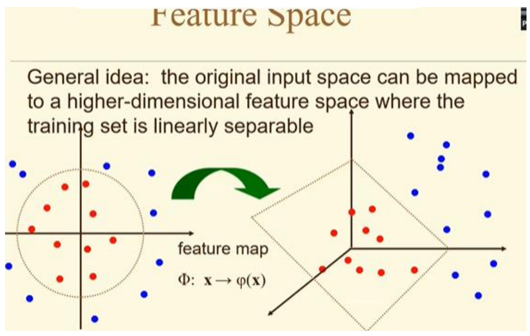
$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

subject to

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

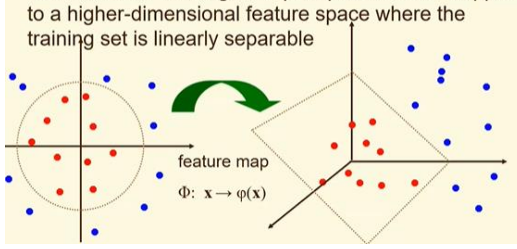
$$\xi_i \geq 0$$

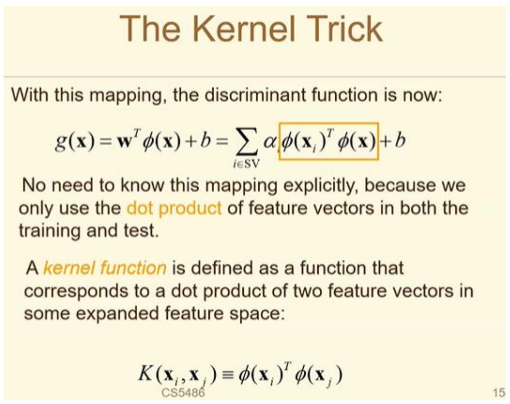
Parameter *C* is a weight between marginal maximization and misclassification miinimization.



Feature Space

General idea: the original input space can be mapped to a higher-dimensional feature space where the training set is linearly separable





The Kernel Trick

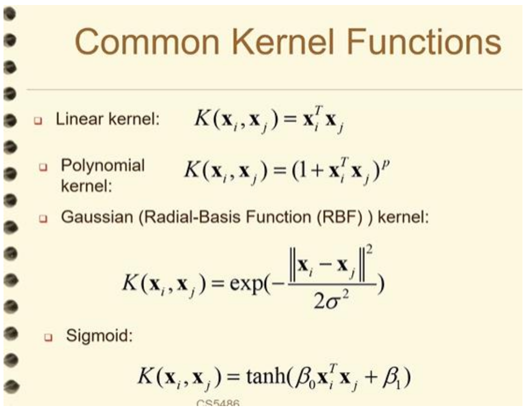
With this mapping, the discriminant function is now:

$$g(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b = \sum_{i\in SV} \alpha_i \boxed{\phi(\mathbf{x}_i)^T\phi(\mathbf{x})} + b$$

No need to know this mapping explicitly, because we only use the dot product of feature vectors in both the training and test.

A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$$



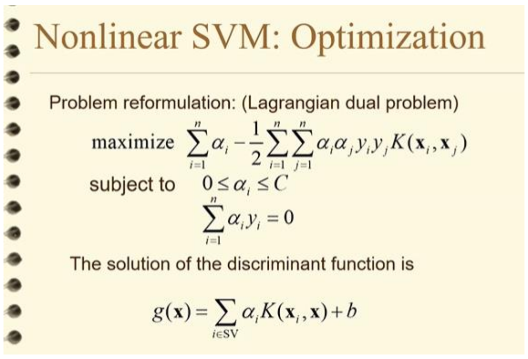
Common Kernel Functions

- Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$
- Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T\mathbf{x}_j)^p$
- Gaussian (Radial-Basis Function (RBF)) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

- Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0\mathbf{x}_i^T\mathbf{x}_j + \beta_1)$$



Nonlinear SVM: Optimization

Problem reformulation: (Lagrangian dual problem)

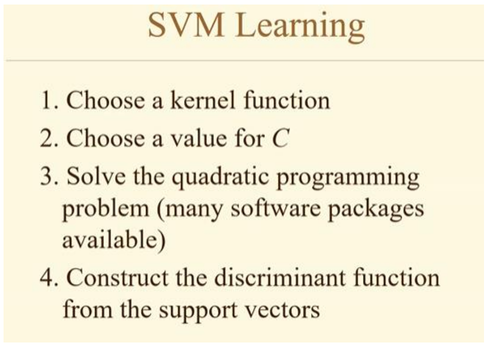
$$\text{maximize } \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to $0 \leq \alpha_i \leq C$

$$\sum_{i=1}^{n}\alpha_i y_i = 0$$

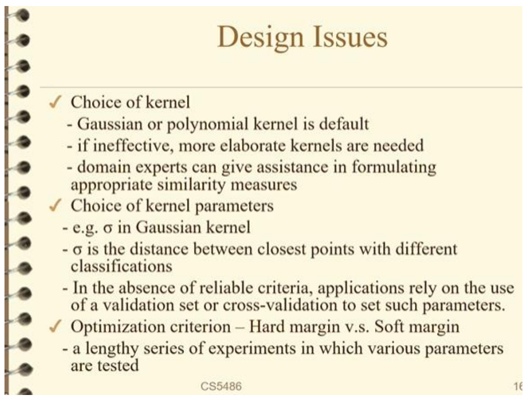
The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i\in SV}\alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$



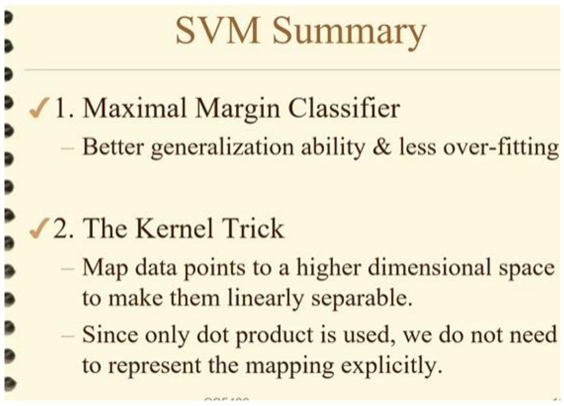
SVM Learning

1. Choose a kernel function
2. Choose a value for *C*
3. Solve the quadratic programming problem (many software packages available)
4. Construct the discriminant function from the support vectors



Design Issues

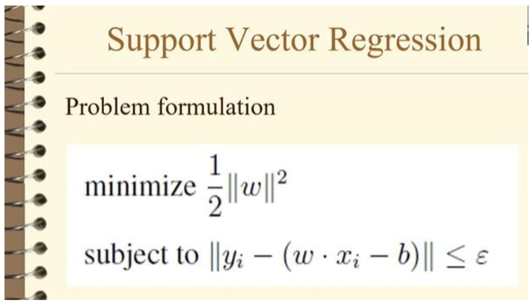
- ✓ Choice of kernel
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures
- ✓ Choice of kernel parameters
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.
- ✓ Optimization criterion – Hard margin v.s. Soft margin
  - a lengthy series of experiments in which various parameters are tested



SVM Summary

✓1. Maximal Margin Classifier
- Better generalization ability & less over-fitting

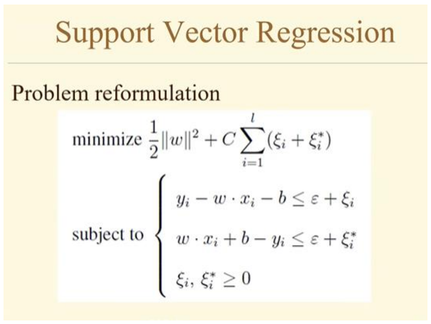
✓2. The Kernel Trick
- Map data points to a higher dimensional space to make them linearly separable.
- Since only dot product is used, we do not need to represent the mapping explicitly.



Support Vector Regression

Problem formulation

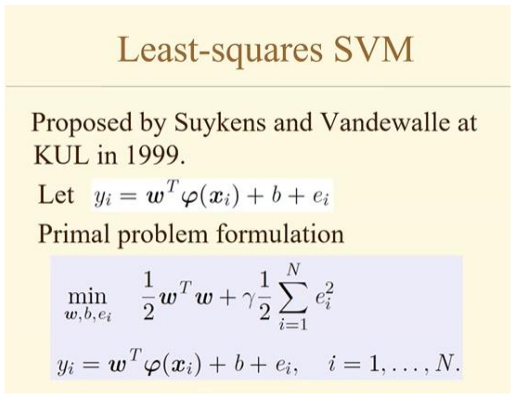
$$\text{minimize } \frac{1}{2}\|w\|^2$$
$$\text{subject to } \|y_i - (w \cdot x_i - b)\| \le \varepsilon$$



Support Vector Regression

Problem reformulation

$$\text{minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$
$$\text{subject to } \begin{cases} y_i - w\cdot x_i - b \le \varepsilon + \xi_i \\ w\cdot x_i + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i,\ \xi_i^* \ge 0 \end{cases}$$



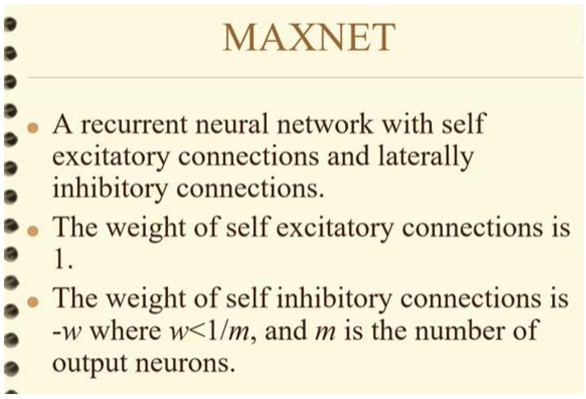
Least-squares SVM

Proposed by Suykens and Vandewalle at KUL in 1999.

Let $y_i = \boldsymbol{w}^T\varphi(\boldsymbol{x}_i) + b + e_i$

Primal problem formulation

$$\min_{\boldsymbol{w},b,e_i} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \gamma\frac{1}{2}\sum_{i=1}^{N} e_i^2$$
$$y_i = \boldsymbol{w}^T\varphi(\boldsymbol{x}_i) + b + e_i, \quad i = 1, \ldots, N.$$

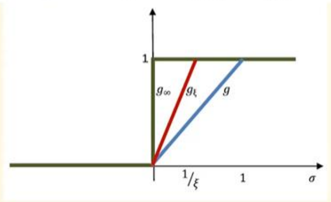


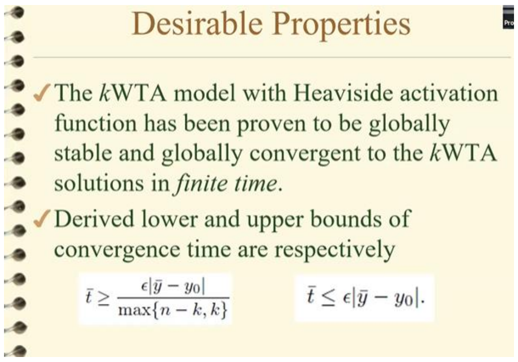
MAXNET

- A recurrent neural network with self excitatory connections and laterally inhibitory connections.
- The weight of self excitatory connections is 1.
- The weight of self inhibitory connections is -*w* where *w*<1/*m*, and *m* is the number of output neurons.

kWTA Model

$$\text{state equation } \epsilon\frac{dy}{dt} = \sum_{i=1}^{n} x_i - k,$$
$$\text{output equation } x_i = g_\infty(u_i - y),\ i = 1, \cdots, n;$$

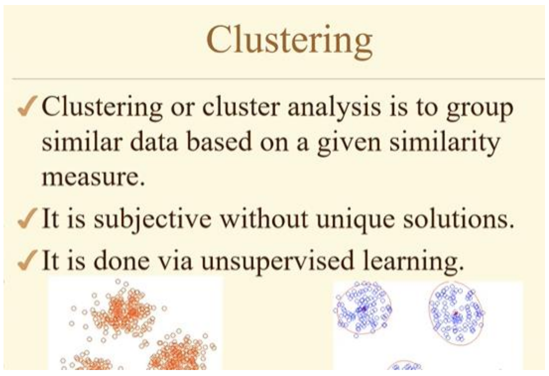




Desirable Properties

- ✓The *k*WTA model with Heaviside activation function has been proven to be globally stable and globally convergent to the *k*WTA solutions in *finite time*.
- ✓Derived lower and upper bounds of convergence time are respectively

$$\bar{t} \ge \frac{\epsilon|\bar{y} - y_0|}{\max\{n-k, k\}} \qquad \bar{t} \le \epsilon|\bar{y} - y_0|.$$

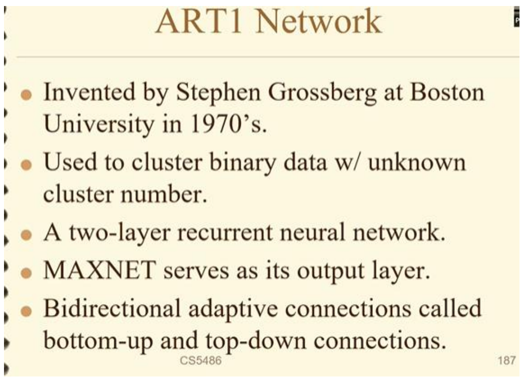


Clustering

✓ Clustering or cluster analysis is to group similar data based on a given similarity measure.

✓ It is subjective without unique solutions.

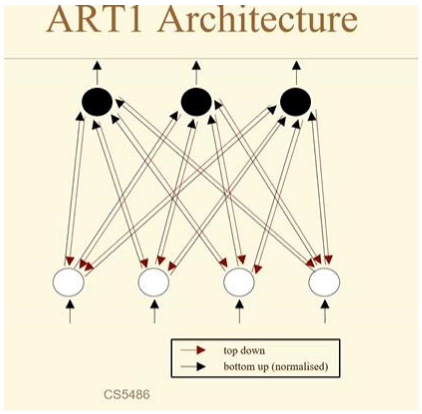
✓ It is done via unsupervised learning.



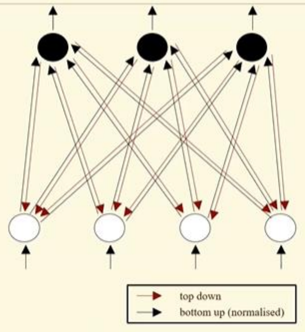
ART1 Network

- Invented by Stephen Grossberg at Boston University in 1970's.
- Used to cluster binary data w/ unknown cluster number.
- A two-layer recurrent neural network.
- MAXNET serves as its output layer.
- Bidirectional adaptive connections called bottom-up and top-down connections.

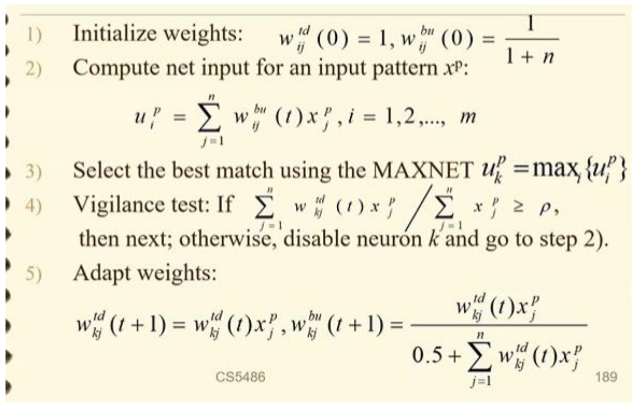
CS5486 187



ART1 Architecture



CS5486



1) Initialize weights: $w_{ij}^{td}(0) = 1, w_{ij}^{bu}(0) = \frac{1}{1+n}$

2) Compute net input for an input pattern x^p:

$$u_i^p = \sum_{j=1}^{n} w_{ij}^{bu}(t) x_j^p, i = 1,2,\ldots, m$$

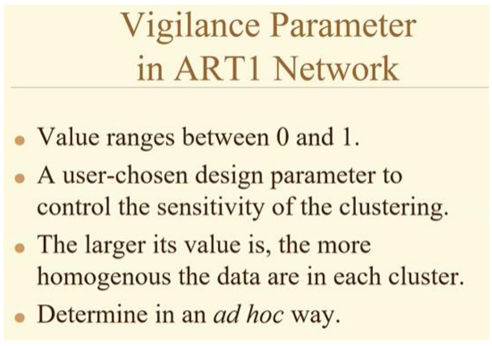
3) Select the best match using the MAXNET $u_k^p = \max_i \{u_i^p\}$

4) Vigilance test: If $\sum_{j=1}^{n} w_{kj}^{td}(t) x_j^p \Big/ \sum_{j=1}^{n} x_j^p \geq \rho$, then next; otherwise, disable neuron k and go to step 2).

5) Adapt weights:

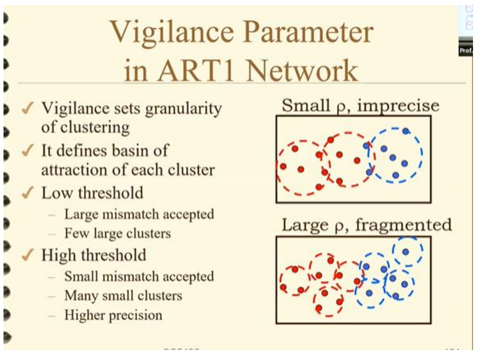
$$w_{kj}^{td}(t+1) = w_{kj}^{td}(t) x_j^p, w_{kj}^{bu}(t+1) = \frac{w_{kj}^{td}(t) x_j^p}{0.5 + \sum_{j=1}^{n} w_{kj}^{td}(t) x_j^p}$$

CS5486 189



Vigilance Parameter in ART1 Network

- Value ranges between 0 and 1.
- A user-chosen design parameter to control the sensitivity of the clustering.
- The larger its value is, the more homogenous the data are in each cluster.
- Determine in an *ad hoc* way.



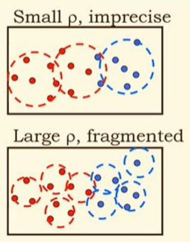
Vigilance Parameter in ART1 Network

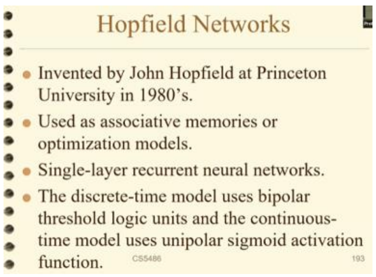
✓ Vigilance sets granularity of clustering

✓ It defines basin of attraction of each cluster

✓ Low threshold
  - Large mismatch accepted
  - Few large clusters

✓ High threshold
  - Small mismatch accepted
  - Many small clusters
  - Higher precision

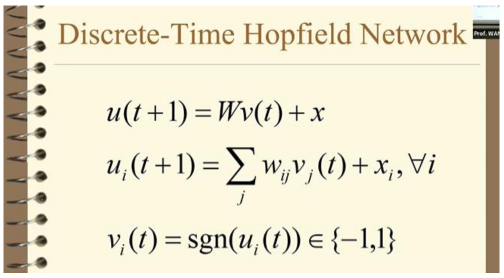




Hopfield Networks

- Invented by John Hopfield at Princeton University in 1980's.
- Used as associative memories or optimization models.
- Single-layer recurrent neural networks.
- The discrete-time model uses bipolar threshold logic units and the continuous-time model uses unipolar sigmoid activation function.

CS5486 193

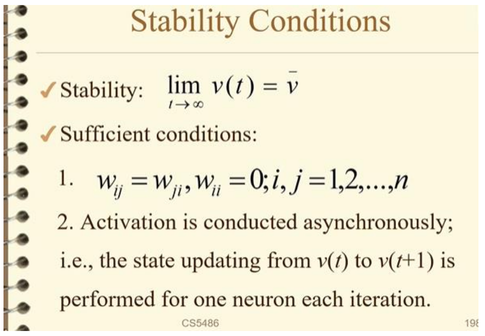


Discrete-Time Hopfield Network

$$u(t+1) = Wv(t) + x$$

$$u_i(t+1) = \sum_j w_{ij} v_j(t) + x_i, \forall i$$

$$v_i(t) = \text{sgn}(u_i(t)) \in \{-1,1\}$$



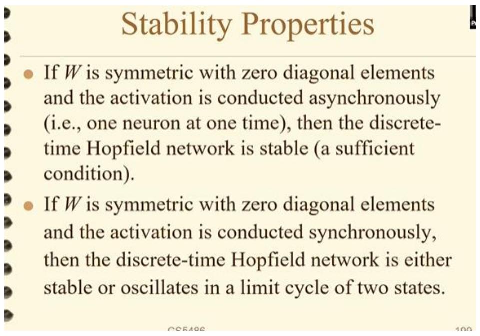
Stability Conditions

✓ Stability: $\lim_{t\to\infty} v(t) = \bar{v}$

✓ Sufficient conditions:

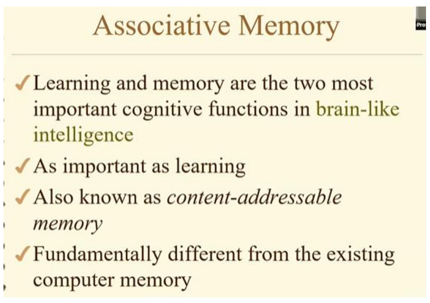
1. $w_{ij} = w_{ji}, w_{ii} = 0; i, j = 1,2,\ldots,n$
2. Activation is conducted asynchronously; i.e., the state updating from $v(t)$ to $v(t+1)$ is performed for one neuron each iteration.

CS5486 198



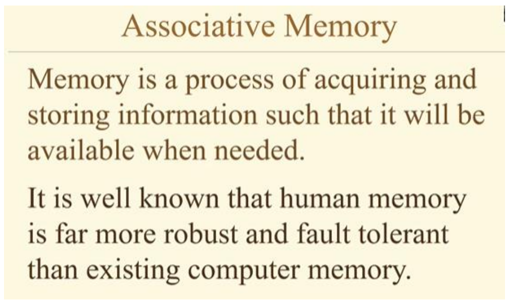
Stability Properties

- If W is symmetric with zero diagonal elements and the activation is conducted asynchronously (i.e., one neuron at one time), then the discrete-time Hopfield network is stable (a sufficient condition).
- If W is symmetric with zero diagonal elements and the activation is conducted synchronously, then the discrete-time Hopfield network is either stable or oscillates in a limit cycle of two states.



Associative Memory

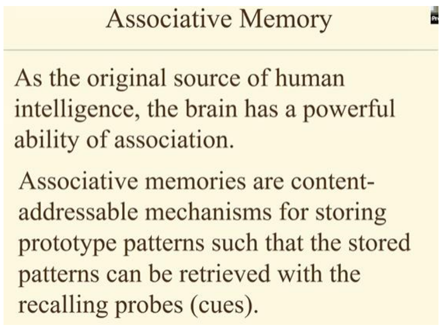
- ✓ Learning and memory are the two most important cognitive functions in brain-like intelligence
- ✓ As important as learning
- ✓ Also known as *content-addressable memory*
- ✓ Fundamentally different from the existing computer memory



Associative Memory

Memory is a process of acquiring and storing information such that it will be available when needed.

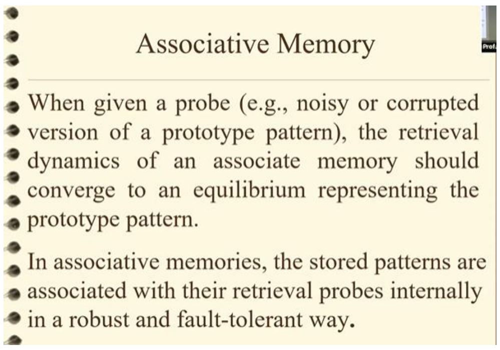
It is well known that human memory is far more robust and fault tolerant than existing computer memory.



Associative Memory

As the original source of human intelligence, the brain has a powerful ability of association.

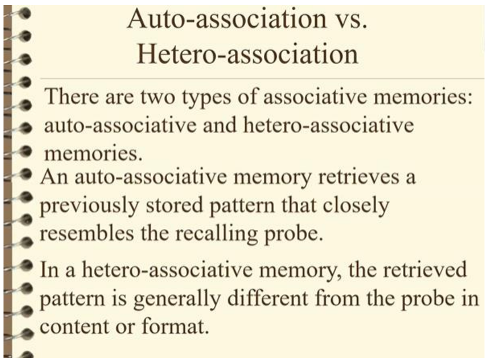
Associative memories are content-addressable mechanisms for storing prototype patterns such that the stored patterns can be retrieved with the recalling probes (cues).



Associative Memory

When given a probe (e.g., noisy or corrupted version of a prototype pattern), the retrieval dynamics of an associate memory should converge to an equilibrium representing the prototype pattern.

In associative memories, the stored patterns are associated with their retrieval probes internally in a robust and fault-tolerant way.

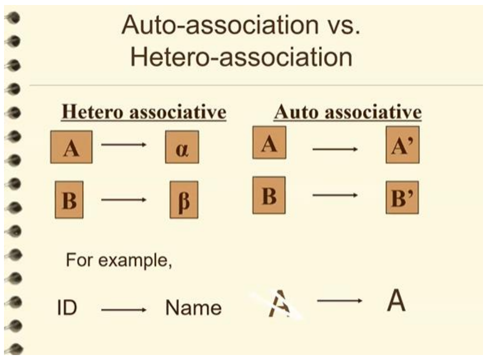


Auto-association vs. Hetero-association

There are two types of associative memories: auto-associative and hetero-associative memories.

An auto-associative memory retrieves a previously stored pattern that closely resembles the recalling probe.

In a hetero-associative memory, the retrieved pattern is generally different from the probe in content or format.



Auto-association vs. Hetero-association

Hetero associative

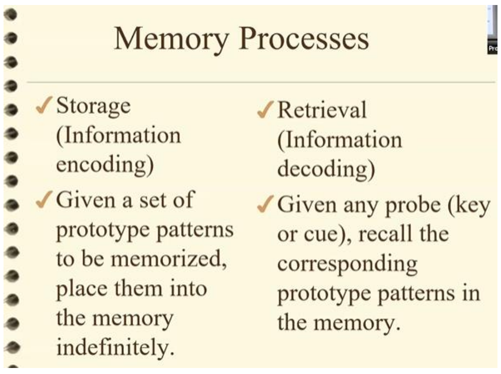
$$\begin{tikzcd} \boxed{A} \arrow[r] & \boxed{\alpha} \\ \boxed{B} \arrow[r] & \boxed{\beta} \end{tikzcd}$$

Auto associative

$$\begin{tikzcd} \boxed{A} \arrow[r] & \boxed{A'} \\ \boxed{B} \arrow[r] & \boxed{B'} \end{tikzcd}$$

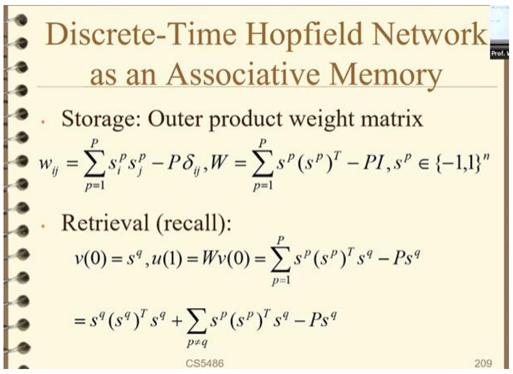
For example,

$$\begin{tikzcd} \text{ID} \arrow[r] & \text{Name} & A \arrow[r] & A \end{tikzcd}$$



Memory Processes

- ✓ Storage (Information encoding)
- ✓ Given a set of prototype patterns to be memorized, place them into the memory indefinitely.
- ✓ Retrieval (Information decoding)
- ✓ Given any probe (key or cue), recall the corresponding prototype patterns in the memory.



Discrete-Time Hopfield Network as an Associative Memory

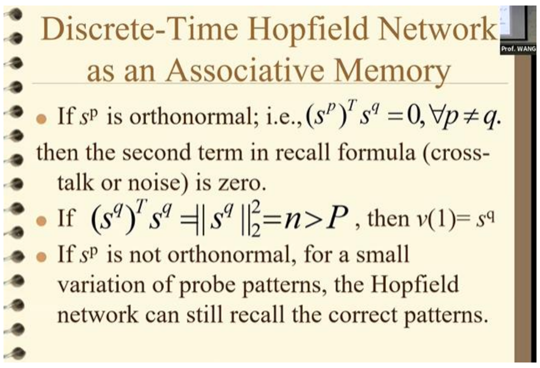
- Storage: Outer product weight matrix

$$w_{ij} = \sum_{p=1}^{P} s_i^p s_j^p - P\delta_{ij}, W = \sum_{p=1}^{P} s^p (s^p)^T - PI, s^p \in \{-1,1\}^n$$

- Retrieval (recall):

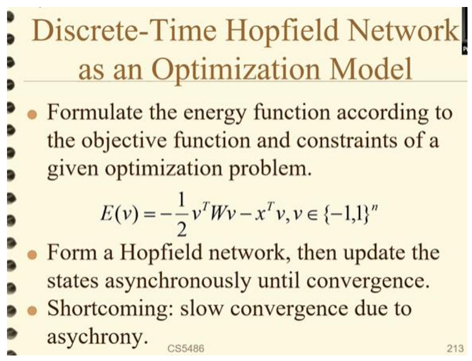
$$v(0) = s^q, u(1) = Wv(0) = \sum_{p=1}^{P} s^p (s^p)^T s^q - Ps^q$$

$$= s^q (s^q)^T s^q + \sum_{p \neq q} s^p (s^p)^T s^q - Ps^q$$



Discrete-Time Hopfield Network as an Associative Memory

- If s^p is orthonormal; i.e., $(s^p)^T s^q = 0, \forall p \neq q.$ then the second term in recall formula (cross-talk or noise) is zero.
- If $(s^q)^T s^q = \| s^q \|_2^2 = n > P$, then $v(1) = s^q$
- If s^p is not orthonormal, for a small variation of probe patterns, the Hopfield network can still recall the correct patterns.

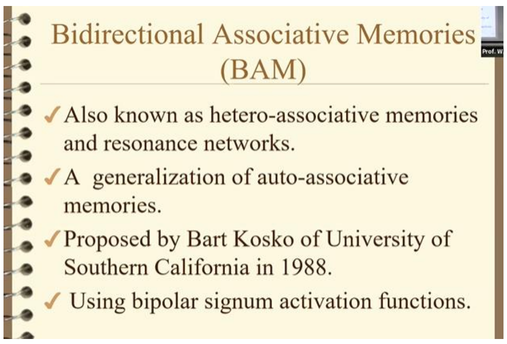


Discrete-Time Hopfield Network as an Optimization Model

- Formulate the energy function according to the objective function and constraints of a given optimization problem.

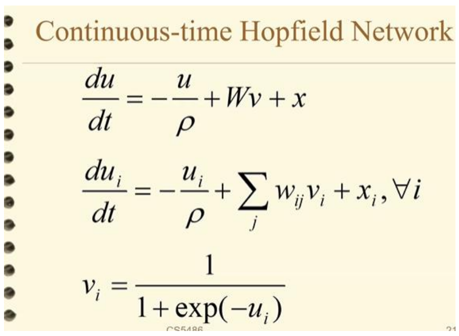
$$E(v) = -\frac{1}{2} v^T W v - x^T v, v \in \{-1,1\}^n$$

- Form a Hopfield network, then update the states asynchronously until convergence.
- Shortcoming: slow convergence due to asychrony.



Bidirectional Associative Memories (BAM)

- ✓ Also known as hetero-associative memories and resonance networks.
- ✓ A generalization of auto-associative memories.
- ✓ Proposed by Bart Kosko of University of Southern California in 1988.
- ✓ Using bipolar signum activation functions.

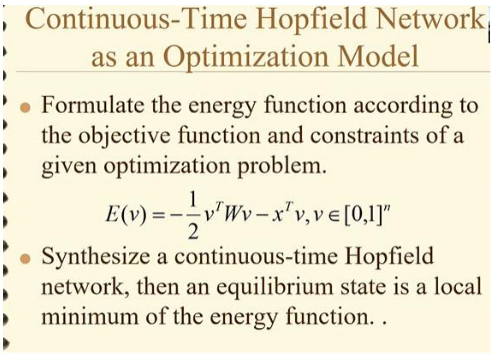


Continuous-time Hopfield Network

$$\frac{du}{dt} = -\frac{u}{\rho} + Wv + x$$

$$\frac{du_i}{dt} = -\frac{u_i}{\rho} + \sum_j w_{ij} v_i + x_i, \forall i$$

$$v_i = \frac{1}{1 + \exp(-u_i)}$$



Continuous-Time Hopfield Network as an Optimization Model

- Formulate the energy function according to the objective function and constraints of a given optimization problem.

$$E(v) = -\frac{1}{2} v^T W v - x^T v, v \in [0,1]^n$$

- Synthesize a continuous-time Hopfield network, then an equilibrium state is a local minimum of the energy function. .