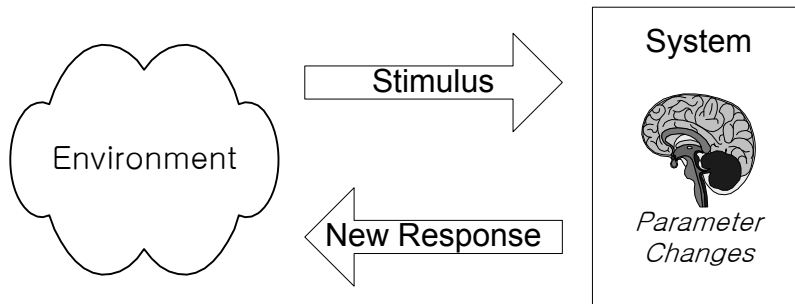


Learning Processes

○ What is learning?



○ Learning paradigm

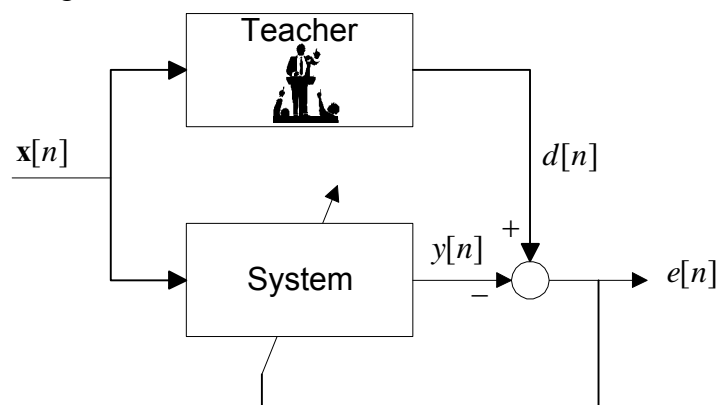
- Supervised learning: learning with a teacher
- Unsupervised learning or self-organized learning: learning without a teacher

○ Learning algorithm

- Error-correction learning
- Memory-based learning
- Hebbian learning
- Competitive learning
- Stochastic learning

(1) Error-correction learning

• Supervised learning



• Delta rule or Widrow-Hoff rule: LMS (least mean square) algorithm

□ Let the system parameter be $\mathbf{w}[n]$ and $y[n] = \mathbf{w}^T[n]\mathbf{x}[n]$. Then,

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} E[n] = \arg \min_{\mathbf{w}} \frac{1}{2} e^2[n].$$

□ From LMS algorithm, $\mathbf{w}[n+1] = \mathbf{w}[n] + \eta e[n]\mathbf{x}[n]$.

- Step size parameter η is called as the "learning rate parameter"
- Batch mode algorithm: LS (least square) algorithm

(2) Memory-based learning

- Supervised learning
- Past experiences are explicitly stored in a large memory as $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$
- Given a new input \mathbf{x}_{test} ,
 - Define a local neighborhood of \mathbf{x}_{test}
 - Classification rule applied to the local neighborhood
 - Nearest neighbor rule
 - k-nearest neighbor classifier

(3) Hebbian learning

- Unsupervised learning
- Hebbian synapse
 - If two neurons on either side of a synapse (connection) are activated simultaneously (or synchronously), then the strength of that synapse is selectively strengthened.
 - If two neurons on either side of a synapse (connection) are activated asynchronously, then that synapse is selectively weakened or eliminated.
- Properties of Hebbian synapse
 - Time-dependent mechanism
 - Local mechanism
 - Interactive mechanism
 - Conjunctival or corelational mechanism
- Mathematical models: $\Delta w_{kj}[n] = F(y_k[n], x_j[n])$
 - Hebb's hypothesis: $\Delta w_{kj}[n] = \eta y_k[n] x_j[n]$, exponential growth, may be saturated
 - Covariance hypothesis: $\Delta w_{kj}[n] = \eta (y_k[n] - \overline{y_k})(x_j[n] - \overline{x_j})$ with time averages $\overline{y_k}$ and $\overline{x_j}$
 - Modification using a forgetting factor:

$$\Delta w_{kj}[n] = \eta y_k[n] x_j[n] - \alpha y_k[n] w_{kj}[n] = \alpha y_k[n] (c x_j[n] - w_{kj}[n])$$

(4) Competitive learning

- Unsupervised learning
- Competitive learning rule
 - Competition among neurons with the same structure but with different weights
 - Strength (output) of each neuron has a certain limit
 - The winner(s) is one (i.e., competitive) or more neurons (i.e., cooperative) with the biggest strength and called winner-takes-all neuron

- Mathematical models:
$$\Delta w_{kj}[n] = \begin{cases} \eta(x_j[n] - w_{kj}[n]) & \text{if winner} \\ 0 & \text{otherwise} \end{cases}$$

- Clustering

Clustering and Classification

(1) Clustering

- Unsupervised learning
 - Labeling could be too cost
 - Understand internal structure of data distribution from clusters
 - Preprocessing for classification since features within the same cluster are similar
- Clustering problem definition. Given a set of vectors $\{\mathbf{x}_k\}_{k=1}^K$, find a set of C clustering centers $\{\mathbf{w}_i\}_{i=1}^C$ such that each \mathbf{x}_k is assigned to a cluster \mathbf{w}_i so that the average distortion

$$D = \frac{1}{K} \sum_{i=1}^C \sum_{k=1}^K I(\mathbf{x}_k, i) d(\mathbf{x}_k, \mathbf{w}_i)$$

where $d(\mathbf{x}_k, \mathbf{w}_i)$ is a distance measure and the indicator function is

$$I(\mathbf{x}_k, i) = \begin{cases} 1 & \text{if } d(\mathbf{x}_k, \mathbf{w}_i) < d(\mathbf{x}_k, \mathbf{w}_j), \forall j \neq i \\ 0 & \text{otherwise} \end{cases}$$

- K-means clustering algorithm.

- Initialization.

Randomize $\{\mathbf{w}_i\}_{i=1}^C$, $I(\mathbf{x}_k, i) = 0$ for $1 \leq k \leq K$ and $1 \leq i \leq C$, $D(0) = 0$, $n = 1$

- Repeat.

Compute $d(\mathbf{x}_k, \mathbf{w}_i)$ for $1 \leq k \leq K$ and $1 \leq i \leq C$

Evaluate $I(\mathbf{x}_k, i) = \begin{cases} 1 & \text{if } d(\mathbf{x}_k, \mathbf{w}_i) < d(\mathbf{x}_k, \mathbf{w}_j), \forall j \neq i \\ 0 & \text{otherwise} \end{cases}$ for $1 \leq k \leq K$

Compute $D[n] = \frac{1}{K} \sum_{i=1}^C \sum_{k=1}^K I(\mathbf{x}_k, i) d(\mathbf{x}_k, \mathbf{w}_i)$

Update $\mathbf{w}_i = \frac{1}{N_i} \sum_{k=1}^K I(\mathbf{x}_k, i) \mathbf{x}_k$ with $N_i = \sum_{k=1}^K I(\mathbf{x}_k, i)$ for $1 \leq i \leq C$

If $1 - \frac{D[n-1]}{D[n]} < \varepsilon$, stop

- Distance measure.

- Norm, $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p$
- Mahalanobis distance, $d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \mathbf{S}_{xy}^{-1} (\mathbf{x} - \mathbf{y})$
- Angle, $d(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$
- Tanimoto coefficient, $d(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\mathbf{x}^T \mathbf{x} + \mathbf{y}^T \mathbf{y} - \mathbf{x}^T \mathbf{y}}$

- Distortion measure.

- Mean square error, $D = \frac{1}{K} \sum_{i=1}^C \sum_{k=1}^K I(\mathbf{x}_k, i) \|\mathbf{x}_k - \mathbf{w}_i\|_2^2 = \sum_{i=1}^C \frac{1}{N_i} \sum_{\mathbf{x}, \mathbf{y} \in C(i)} \|\mathbf{x} - \mathbf{y}\|_2^2$
- In general, $D = \sum_{i=1}^C \frac{1}{N_i} \sum_{\mathbf{x}, \mathbf{y} \in C(i)} d(\mathbf{x}, \mathbf{y})$ or $D = \sum_{i=1}^C \left\{ \min_{\mathbf{x}, \mathbf{y} \in C(i)} d(\mathbf{x}, \mathbf{y}) \right\}$

- Scattering criteria.

- Mean of cluster i , $\mathbf{m}_i = \frac{1}{N_i} \sum_{k=1}^K I(\mathbf{x}_k, i) \mathbf{x}_k$
- Total mean, $\mathbf{m} = \frac{1}{K} \sum_{i=1}^C N_i \mathbf{m}_i$
- Scatter matrix of cluster i , $\mathbf{S}_i = \sum_{k=1}^K I(\mathbf{x}_k, i) (\mathbf{x}_k - \mathbf{m}_i)(\mathbf{x}_k - \mathbf{m}_i)^T$
- Within cluster scatter matrix, $\mathbf{S}_W = \sum_{i=1}^C \mathbf{S}_i$
- Between clusters scatter matrix, $\mathbf{S}_B = \sum_{i=1}^C N_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$
- Total scatter matrix, $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B = \sum_{k=1}^K (\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T$
- Note that $D = \text{tr}(\mathbf{S}_B^{-1} \mathbf{S}_W)$
- Distance between clusters,

$$d_{\min} \{C(i), C(j)\} = \min_{\mathbf{x} \in C(i), \mathbf{y} \in C(j)} d(\mathbf{x}, \mathbf{y})$$

$$d_{\max} \{C(i), C(j)\} = \max_{\mathbf{x} \in C(i), \mathbf{y} \in C(j)} d(\mathbf{x}, \mathbf{y})$$

$$d_{\text{avg}} \{C(i), C(j)\} = \frac{1}{N_i N_j} \sum_{\mathbf{x} \in C(i)} \sum_{\mathbf{y} \in C(j)} d(\mathbf{x}, \mathbf{y})$$

$$d_{mean} \{C(i), C(j)\} = |\mathbf{m}_i - \mathbf{m}_j|$$

- Hierarchical clustering.

- Merge. Initially, each \mathbf{x}_k is a cluster. During iterations, nearest pair of distinct clusters are merged.

- Split. Initially, all $\{\mathbf{x}_k\}_{k=1}^K$ belong to one cluster. During iterations, one cluster is spitted into two or more clusters if within cluster scattering is large.

(2) Classification

- Classification problem definition. Assume data samples $X = \{\mathbf{x}_k\}_{k=1}^K$ are drawn from

M classes $C = \{C(i)\}_{i=1}^M$. Given an observation \mathbf{x} , find a decision rule $g(\mathbf{x}) \in C$

such that the probability of classification $\sum_{i=1}^M \Pr\{g(\mathbf{x}) = C(i) | \mathbf{x} \in C(i)\}$ is maximized.

- Nearest neighbor classifier. Assume that the already classified set of data or mapping

or ANN is available, i.e., we have $\{(\mathbf{y}_i, C(j))\}$ for $1 \leq i \leq N$ and $1 \leq j \leq M$. For a new

sample \mathbf{x} , the decision rule choose $g(\mathbf{x}) = C(k)$ if $\mathbf{y}^* = \arg \min_{\mathbf{y}_i} \|\mathbf{y}_i - \mathbf{x}\|$ is paired

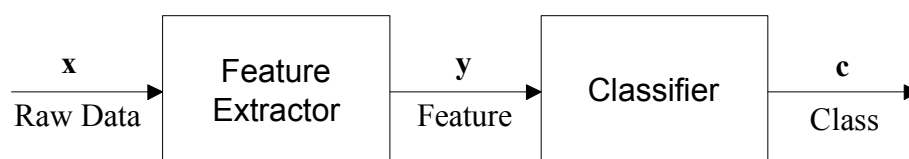
with $C(k)$.

- k-nearest neighbor classifier. Examine k nearest classes and classify \mathbf{x} into the majority of them.

- Statistical decision rules.

- Maximum posterior probability (MAP) classifier
 - Maximum Likelihood (ML) classifier
 - Neyman-Pearson (NP) detector
 - Bayes detector

(3) Features



- Feature representation.

- Symbolic vs. numeric.
- Higher dimensional features.
- Feature selection.
 - Select a subset of available features can improve classification.
 - Selection of subspace or subspace approximation.
 - Hidden neurons in MPL are feature detectors. Hidden neuron pruning is a kind of feature selection.
- Feature transformation.
 - Affine transformation. $y = Tx + b$
 - Rotation
 - Linear filtering
 - Fourier transform (DFT)
 - Discrete cosine transform
 - Karhunen-Loeve expansion (principal component analysis)
 - Eigendecomposition
 - Edge or line detection
 - Other linear or nonlinear operation

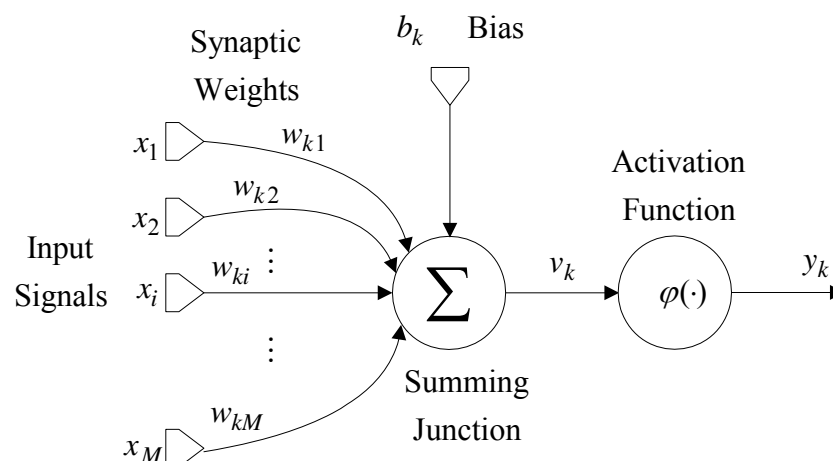
(4) Data sampling

- Sample data independently from the underlying population.
- Use resampling with randomization.
- Use M -fold cross-validation or leave-one-out cross-validation.

Artificial Neural Network (ANN)

- An (artificial) neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:
 - Knowledge is acquired by the network from its environment through a learning process.
 - Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.
- Properties of artificial neural networks
 - Nonlinearity
 - Input-output mapping
 - Adaptivity
 - Evidential response
 - Contextual information
 - Fault tolerance
 - VLSI implementability
 - Uniformity of analysis and design
 - Neurobiological analogy

(1) Models of a neuron



- Neuron is an information processing unit.
 - A set of synapses or connecting links with a weight or strength
 - Adder or linear combiner
 - Activation function or squashing function

$$v_k = \sum_{j=0}^M w_{kj} x_j \quad \text{with } w_{k0} = b_k, \quad x_0 = 1 \quad \text{and } y_k = \varphi(v_k)$$

- Activation function.

- Threshold function or Heaviside function \Rightarrow McCulloch-Pitts model, all-or-none

$$y = \varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \quad \text{or}$$

the signum function

$$y = \varphi(v) = \text{sgn}(v) = \begin{cases} 1 & \text{if } v > 0 \\ 0 & \text{if } v = 0 \\ -1 & \text{if } v < 0 \end{cases}$$

- Piecewise linear function (can have a gain)

$$y = \varphi(v) = \begin{cases} 1 & \text{if } v \geq +\frac{1}{2} \\ v_k & \text{if } -\frac{1}{2} < v < +\frac{1}{2} \\ 0 & \text{if } v \leq -\frac{1}{2} \end{cases} \quad \text{or} \quad y = \varphi(v) = \begin{cases} 1 & \text{if } v \geq +1 \\ v_k & \text{if } -1 < v < +1 \\ -1 & \text{if } v \leq -1 \end{cases}$$

- Sigmoid function: strictly increasing function with a graceful balance between linear and nonlinear behavior, for example logistic function

$$y = \varphi(v) = \frac{1}{1 + \exp(-av)} \quad \text{or}$$

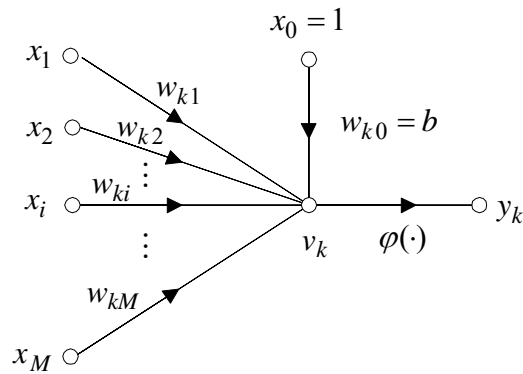
the hyperbolic tangent function

$$y = \varphi(v) = \tanh(v)$$

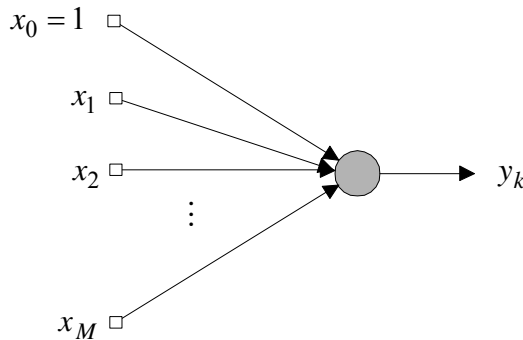
- Stochastic model.

(2) Signal flow graph, architectural graph, and Matlab representation

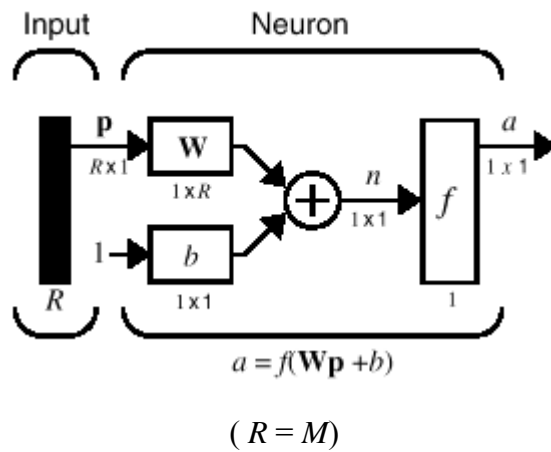
- Signal flow graph.



• Architectural graph.

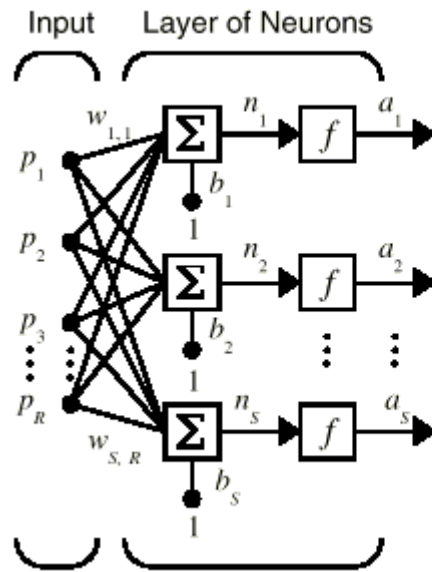


• Matlab representation.



(3) Network architecture

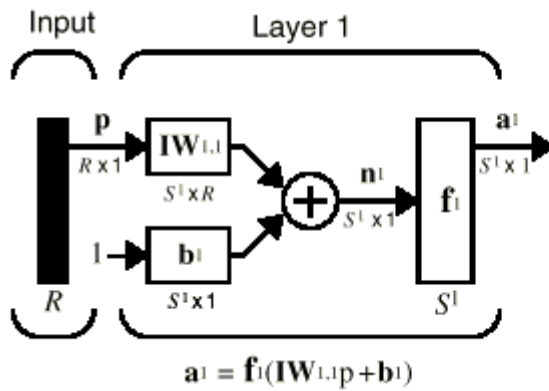
• Single-layer feedforward network.



Where...

R = # of elements in input vector

S = # Neurons in Layer

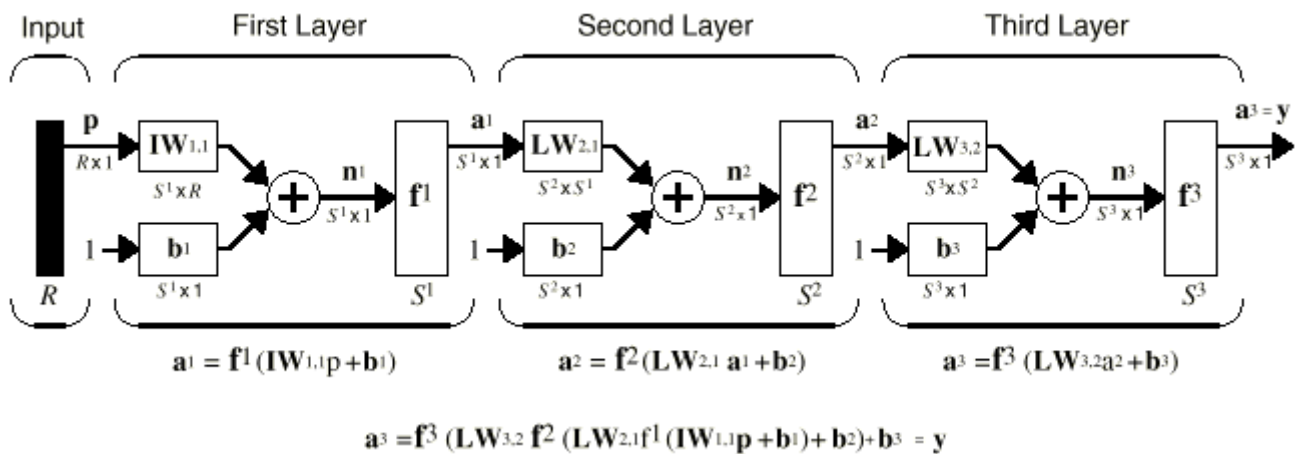
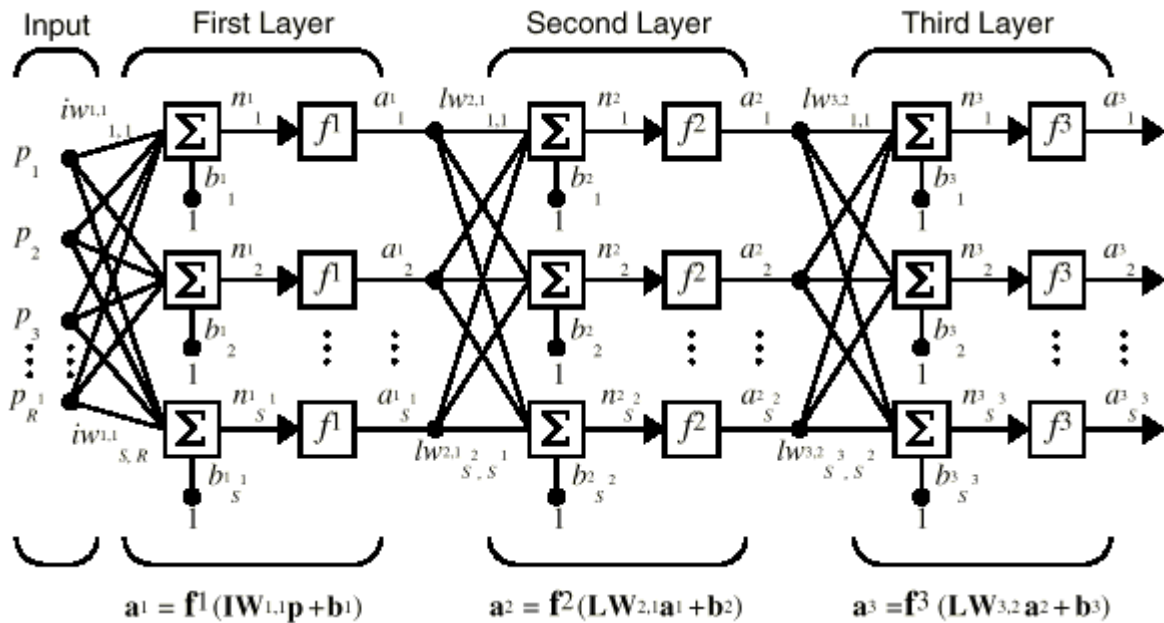


Where...

R = # of elements in input

S^1 = # Neurons in layer 1

- Multilayer feedforward network.



- Recurrent network or dynamic network.

(4) Knowledge representation

- Knowledge refers to stored information or model used by a person or machine to interpret, predict, and appropriately respond to the outside world.
- Information.
 - Prior information
 - Observations or measurements provide a pool of information from which the examples are drawn to train the ANN
- Examples ⇒ a set of training data or training samples

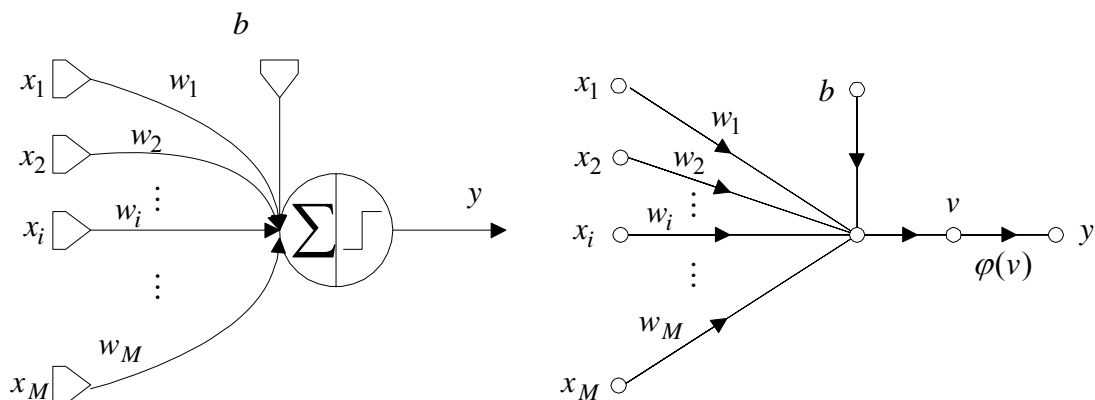
- Labeled \Rightarrow supervised learning
- Unlabelled \Rightarrow unsupervised learning
- Four rules of knowledge representation for ANN.
 - Rule 1. Similar inputs from similar class should usually produce similar representation inside the network, and should therefore be classified as belonging to the same category.
 - Rule 2. Items to be categorized as separate classes should be given widely different representations in the network.
 - Rule 3. If a particular feature is important, then there should be a large number of neurons involved in the representation of that item in the network.
 - Rule 4. Prior information and variances should be built into the design of a neural network, thereby simplifying the network design by not having to learn them.
- \Rightarrow "In general, use your common sense."
- Training and generalization.

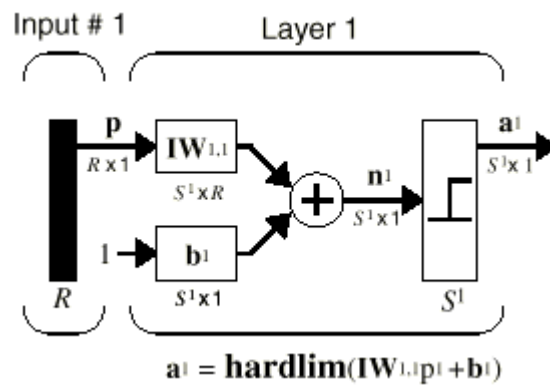
Single-Layer Perceptron

BACKGROUND MATERIALS

- Unconstrained optimization techniques
 - Steepest descent
 - Newton's method
 - Gauss-Newton method
- Wiener filter
- Adaptive filter using LMS (least mean square) algorithm
- LS (least square) method

(1) Perceptron





- Decision boundary is a hyperplane, $v = \sum_{i=1}^M w_i x_i + b = \mathbf{w}^T \mathbf{x} + b = 0$. And

$$y = \begin{cases} 1 & \Leftrightarrow \mathbf{x} \text{ belong to class } C_1 \Leftrightarrow \mathbf{w}^T \mathbf{x} > 0 \\ 0 & \Leftrightarrow \mathbf{x} \text{ belong to class } C_2 \Leftrightarrow \mathbf{w}^T \mathbf{x} \leq 0 \end{cases}$$

- Perceptron convergence algorithm

▫ Let $\mathbf{x}[n] = [+1, x_1[n], \dots, x_M[n]]^T$ and $\mathbf{w}[n] = [b[n], w_1[n], \dots, w_M[n]]^T$.

▫ Initialization. $n = 0$ and $\mathbf{w}[0] = \mathbf{0}$.

▫ Activation. Apply $\mathbf{x}[n]$ and get $d[n]$.

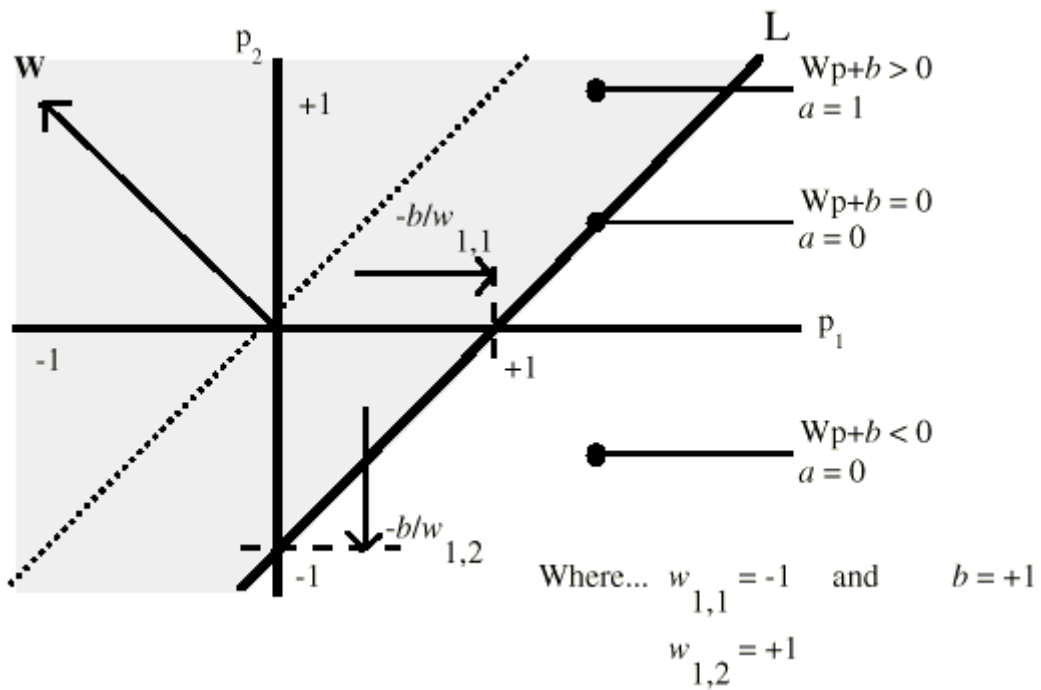
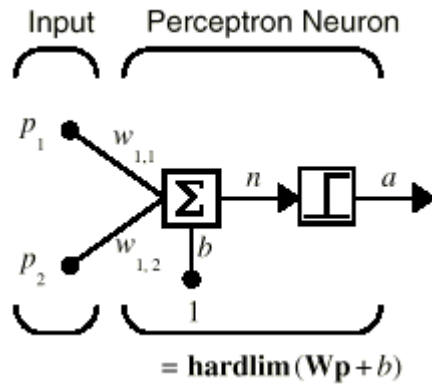
▫ Response. $y[n] = \text{sgn}(\mathbf{w}^T[n]\mathbf{x}[n])$

▫ Weight adaptation (LMS). $\mathbf{w}[n+1] = \mathbf{w}[n] + \eta(d[n] - y[n])\mathbf{x}[n]$ where

$$d[n] = \begin{cases} 1, & \text{if } \mathbf{x} \text{ belong to class } C_1 \\ -1, & \text{if } \mathbf{x} \text{ belong to class } C_2 \end{cases}$$

(2) Perceptron as a linear classifier (Matlab)

- newp
- sim
- init
- learnp
- adapt



(3) Limitations of perceptron

- XOR problem

Multilayer Perceptron

- Multilayer perceptron (MLP)
 - Input layer
 - Hidden layer
 - Output layer
- Feed forward
- Nonlinear activation function
- Backpropagation learning algorithm

(1) Structure of MLP

(2) Backpropagation learning algorithm

- Epoch: one complete presentation of the complete training samples
- At the output layer, at iteration n (i.e., n th training example)

$$\square e_j[n] = d_j[n] - y_j[n] \text{ at } j\text{th neuron}$$

$$\square \mathcal{E}[n] = \frac{1}{2} \sum_{j \in C} e_j^2[n] \text{ where } C \text{ is a set of all neurons at the output layer}$$

$$\square v_j[n] = \sum_{i=0}^M w_{ji} y_i[n] \text{ and } y_j[n] = \varphi_j(v_j[n])$$

□ Chain rule.

$$\begin{aligned} \frac{\partial \mathcal{E}[n]}{\partial w_{ji}[n]} &= \frac{\partial \mathcal{E}[n]}{\partial e_j[n]} \frac{\partial e_j[n]}{\partial y_j[n]} \frac{\partial y_j[n]}{\partial v_j[n]} \frac{\partial v_j[n]}{\partial w_{ji}[n]} \\ &= e_j[n] (-1) \varphi_j^n(v_j[n]) y_i[n] \end{aligned}$$

$$\square \text{LMS algorithm. } \Delta w_{ji} = -\eta \frac{\partial \mathcal{E}[n]}{\partial w_{ji}[n]} = \eta e_j[n] \varphi_j^n(v_j[n]) y_i[n] = \eta \delta_j[n] y_i[n] \text{ with}$$

$$\delta_j[n] = -\frac{\partial \mathcal{E}[n]}{\partial v_j[n]} = -\frac{\partial \mathcal{E}[n]}{\partial e_j[n]} \frac{\partial e_j[n]}{\partial y_j[n]} \frac{\partial y_j[n]}{\partial v_j[n]} = e_j[n] \varphi_j^n(v_j[n])$$

- At a hidden layer, at iteration n (i.e., n th training example)

$$\square \text{At } j\text{th neuron, } \delta_j[n] = -\frac{\partial \mathcal{E}[n]}{\partial v_j[n]} = -\frac{\partial \mathcal{E}[n]}{\partial y_j[n]} \frac{\partial y_j[n]}{\partial v_j[n]} = -\frac{\partial \mathcal{E}[n]}{\partial y_j[n]} \varphi_j^n(v_j[n])$$

$$\square \text{From } \mathcal{E}[n] = \frac{1}{2} \sum_{k \in C} e_k^2[n], \frac{\partial \mathcal{E}[n]}{\partial y_j[n]} = \sum_k e_k[n] \frac{\partial e_k[n]}{\partial y_j[n]} = \sum_k e_k[n] \frac{\partial e_k[n]}{\partial v_k[n]} \frac{\partial v_k[n]}{\partial y_j[n]}$$

▫ Since $e_k[n] = d_k[n] - y_k[n] = d_k[n] - \varphi_k(v_k[n])$ for the output layer neuron k ,

$$\frac{\partial e_k[n]}{\partial v_k[n]} = -\varphi_k^n(v_k[n]).$$

▫ Since $v_k[n] = \sum_{j=0}^M w_{kj} y_j[n]$, $\frac{\partial v_k[n]}{\partial y_j[n]} = w_{kj}$.

▫ Therefore, $\frac{\partial \mathcal{E}[n]}{\partial y_j[n]} = -\sum_k e_k[n] \varphi_k^n(v_k[n]) w_{kj} = -\sum_k \delta_k[n] w_{kj}$.

▫ Finally, at j th neuron of the hidden layer, $\delta_j[n] = \varphi_j^n(v_j[n]) \sum_k \delta_k[n] w_{kj}$.

▫ LMS algorithm. $\Delta w_{ji} = \eta \delta_j[n] y_i[n]$.

• Activation functions

▫ Logistic function.

$$y_j[n] = \varphi_j(v_j[n]) = \frac{1}{1 + \exp(-av_j[n])}, a > 0 \text{ and } -\infty < v_j[n] < \infty,$$

$$\varphi_j^n(v_j[n]) = ay_j[n](1 - y_j[n]), \text{ and}$$

$$\begin{aligned} \delta_j[n] &= e_j[n] \varphi_j^n(v_j[n]) \\ &= \begin{cases} a(d_j[n] - y_j[n]) y_j[n] (1 - y_j[n]) & \text{for output layer} \\ ay_j[n] (1 - y_j[n]) \sum_k \delta_k[n] w_{kj} & \text{for hidden layer} \end{cases} \end{aligned}$$

▫ Hyperbolic tangent function.

$$y_j[n] = \varphi_j(v_j[n]) = a \tanh(bv_j[n]), a, b > 0 \text{ and } -\infty < v_j[n] < \infty,$$

$$\varphi_j^n(v_j[n]) = \frac{b}{a} (a - y_j[n]) (a + y_j[n]), \text{ and}$$

$$\begin{aligned} \delta_j[n] &= e_j[n] \varphi_j^n(v_j[n]) \\ &= \begin{cases} \frac{b}{a} (d_j[n] - y_j[n]) (a - y_j[n]) (a + y_j[n]) & \text{for output layer} \\ \frac{b}{a} (a - y_j[n]) (a + y_j[n]) \sum_k \delta_k[n] w_{kj} & \text{for hidden layer} \end{cases} \end{aligned}$$

• Momentum. $\Delta w_{ji}[n] = \alpha \Delta w_{ji}[n-1] + \eta \delta_j[n] y_i[n] \Rightarrow$ stabilizing effect

- Modes of training.

- A set of training examples, $\{(\mathbf{x}[i], d[i])\}_{i=1}^N \Rightarrow$ epoch
- Randomize samples at each epoch
- Sequential mode (on-line, pattern, or stochastic mode): update weight sample by sample
- Batch mode (on-line, pattern, or stochastic mode): update weight at the end of epoch

$$\mathcal{E}_{av} = \frac{1}{N} \sum_{n=1}^N \mathcal{E}[n] \quad \text{and} \quad \Delta w_{ji} = -\eta \frac{\partial \mathcal{E}_{av}}{\partial w_{ji}} = -\frac{\eta}{N} \sum_{n=1}^N e_j[n] \frac{\partial e_j[n]}{\partial w_{ji}}$$

- Stopping criterion.

- Small norm of the gradient vector
- Small absolute value of change in the average squared error per epoch

(3) Heuristics

- When the training data set is large and redundant, sequential mode is usually faster and better.
- When the training data set is not large, there are several batch mode algorithms that are faster.
- Information content of a training example
 - Use an example that results in the largest training error
 - Use an example that is different from all those previously used
 - Distribution of training examples should not be distorted
 - Avoid any outlier in the training data set
- Activation function
- Target value
- Input normalization
- Initialization
- Use any prior information
- Learning rate and momentum
 - Every adjustable network parameter should have its own individual learning rate parameter.
 - Every learning rate parameter should be allowed to vary from one iteration to the next.
 - When the derivative of the cost function w.r.t. a synaptic weight has the same algebraic sign for several consecutive iterations, the corresponding learning rate

parameter should be increased.

▫ When the algebraic sign of the derivative of the cost function w.r.t. a synaptic weight alternates for several consecutive iterations, the corresponding learning rate parameter should be decreased.

(4) Output representation and decision rule

- ANN is already trained
- Consider M -class classification problem
 - Let \mathbf{x}_j denote j th sample (prototype) to be classified and ANN produces output

$$\mathbf{y}_j = [y_{1,j}, \dots, y_{M,j}]^T = [F_1(\mathbf{x}_j), \dots, F_M(\mathbf{x}_j)]^T = \mathbf{F}(\mathbf{x}_j).$$

- Note that the function \mathbf{F} depends on the training data set $\{(\mathbf{x}[i], d[i])\}_{i=1}^N$.
- What is the optimal decision rule for classifying the M outputs of ANN?
- Assigning single class from M distinct classes:

$$\mathbf{x}_j \in C_k \text{ if } F_k(\mathbf{x}_j) > F_l(\mathbf{x}_j) \text{ for all } l \neq k$$

- Assigning multiple class from M distinct classes:

$$\mathbf{x}_j \in C_k \text{ if } F_k(\mathbf{x}_j) > \text{threshold (ex, 0.5)}$$

(5) Generalization

- A network is said to *generalize* well when the input-output mapping is correct for test data never used in creating or training the network.
- *Overtraining* or *overfitting* problem
- *Bias-variance trade-off*
- Factors influencing generalization
 - Size of training set and how representative it is of the environment of interest
 - Architecture of ANN
 - Physical complexity of the problem at hand

(6) Cross-validation

- Backpropagation learning algorithm encodes an input-output mapping into the synaptic weights and thresholds of a MLP.
- For better generalization, partition the training set into two subsets.

- Estimation subset, used to train or select the model
- Validation subset, used to test or validate the model
- Early stopping rule, stop training when the error using validation subset starts increasing

(7) Network growing and pruning techniques

- Network growing
- Network pruning

(8) Supervised learning viewed as an optimization problem

- Conjugate gradient method
- Quasi-Newton method

(9) Matlab experiments

newff

init

sim

adapt: leardgd, leardgdm

train: traingd, traingdm, traingda, traingdx, trainrp, traincgf, traincgb, trainscg, trainbfg,
trainoss, trainlm, trainbr

premmx, postmmx, trammx, prestd, poststd, trastd

prepca, prapca

postreg