Neural Networks
Lecture 2:Single Layer Classifiers

H.A Talebi
Farzaneh Abdollahi

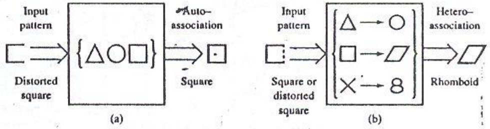
Department of Electrical Engineering

Amirkabir University of Technology

Winter 2011

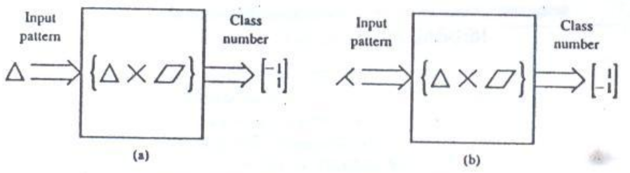
Neural Processing

- One of the most applications of NN is in mapping inputs to the corresponding outputs $o = f(wx)$
- The process of finding o for a given x is named **recall**.
- Assume that a set of patterns can be stored in the network.
- **Autoassociation:** The network presented with a pattern similar to a member of the stored set, it associates the input with the closest stored pattern.
  - A degraded input pattern serves as a cue for retrieval of its original
- **Hetroassociation:** The associations between pairs of patterns are stored.



Association response: (a) autoassociation and (b) heteroassociation.

- **Classification:** The set of input patterns is divided into a number of classes. The classifier recall the information regarding class membership of the input pattern. The outputs are usually binary.
  - Classification can be considered as a special class of hetroassociation.
- *Recognition:* If the desired response is numbers but input pattern does not fit any pattern.

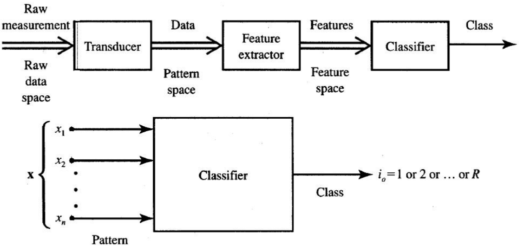


Classification response: (a) classification and (b) recognition.

- **Function Approximation:** Having I/O of a system, their corresponding function f is approximated.
  - This application is useful for control
- In all mentioned aspects of neural processing, it is assumed the data is already stored to be recalled
- Data are stored in a network in learning process

Pattern Classification

- The goal of pattern classification is to assign a physical object, event, or phenomenon to one of predefined classes.
- Pattern is quantified description of the physical object or event.
- Pattern can be based on time (sensors output signals, acoustic signals) or place (pictures, fingertips):
- Example of classifiers: disease diagnosis, fingertip identification, radar and signal detection, speech recognition
- Fig. shows the block diagram of pattern recognition and classification

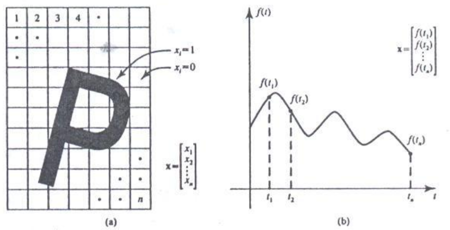


Pattern Classification

- Input of feature extractors are sets of data vectors belonging to a certain category.
- Feature extractor compress the dimensionality as much as does not ruin the probability of correct classification
- Any pattern is represented as a point in n-dimensional Euclidean space E^n, called pattern space.
- The points in the space are n-tuple vectors $X = [x_1 \; ... \; x_n]^T$.
- A pattern classifier maps sets of points in E^n space into one of the numbers $i_0 = 1, ..., R$ based on decision function $i_0 = i_0(x)$.
- The set containing patterns of classes 1,..., R are denoted by $\S_i, ...\S_n$

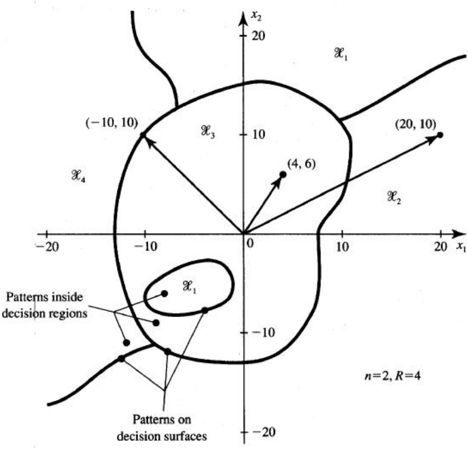
Pattern Classification

- The fig. depicts two simple method to generate the pattern vector
  - Fig. a: x_i of vector $X = [x_1 \ \dots \ x_n]^T$ is 1 if ith cell contains a portion of a spatial object, otherwise is 0
  - Fig b: when the object is continuous function of time, the pattern vector is obtained at discrete time instance t_i, by letting $x_i = f(t_i)$ for $i = 1, \dots, n$



Two simple ways of coding patterns into pattern vectors: (a) spatial object and (b) temporal object (waveform).

- Example: for $n = 2$ and $R = 4$
- $X = [20\ 10] \in \S_2$, $X = [4\ 6] \in \S_3$
- The regions denoted by $\S_i$ are called decision regions.
- Regions are seperated by decision surface
  - The patterns on decision surface does not belong to any class
  - decision surface inE^2 is curve, for general case, E^n is $(n-1)-$dimentional hepersurface.



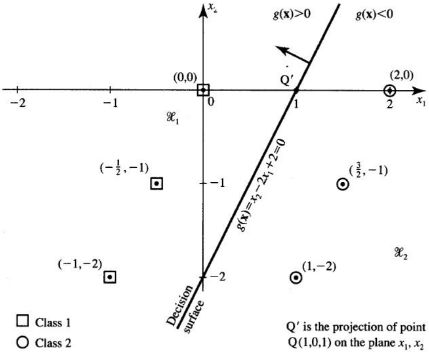
Discriminant Functions

- During the classification, the membership in a category should be determined by classifier based on discriminant functions $g_1(X), ..., g_R(X)$
- Assume $g_i(X)$ is scalar.
- The pattern belongs to the ith category iff $g_i(X) > g_j(X)$ for $i, j = 1, ..., R, i \neq j$.
- $\therefore$ within the region $\S_i$, ith discriminant function have the largest value.
- Decision surface contain patterns X without membership in any classes
- The decision surface is defined as:

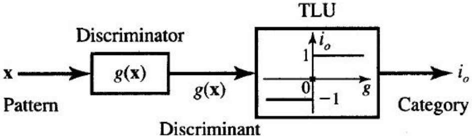
$$g_i(X) - g_j(X) = 0$$

- **Example:** Consider six patterns, in two dimensional pattern space to be classified in two classes:
  $\{[0\ 0]', [-0.5\ \ -1]', [-1\ \ -2]'\}$: class 1
  $\{[2\ 0]', [1.5\ \ -1]', [1\ \ -2]'\}$: class 2
- Inspection of the patterns shows that the $g(X)$ can be arbitrarily chosen

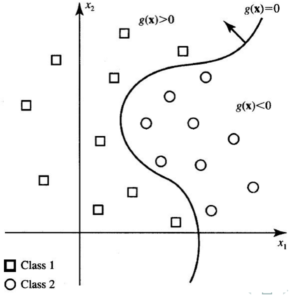
$$g(X) = -2x_1 + x_2 + 2$$
$$g(X) > 0 : \text{class } 1$$
$$g(X) < 0 : \text{class } 2$$
$$g(X) = 0 : \text{on the surface}$$



- The classifier can be implemented as shown in Fig. below (TLU is threshold logic



- **Example 2:** Consider a classification problem as shown in fig. below
- the discriminant surface can not be estimated easily.
- It may result in a nonlinear function of x_1 and x_2.



Pattern Classification

- In pattern classification we assume
  - The sets of classes and their members are known
- Having the patterns, We are looking to find the discriminant surface by using NN,
- The only condition is that the patterns are separable
- The patterns like first example are linearly separable and in second example are nonlinearly separable
- In first step, simple separable systems are considered.

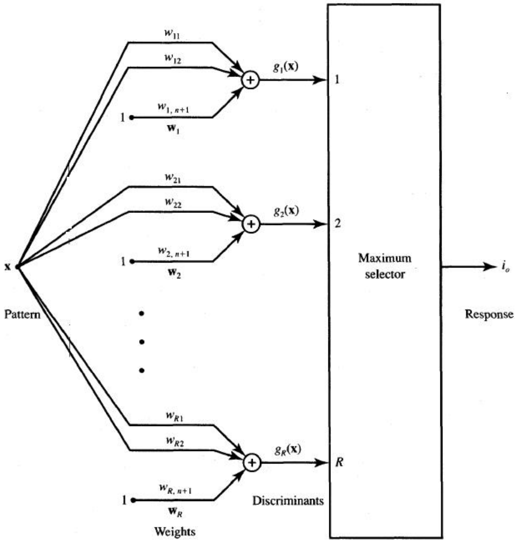
Linear Machine

- Linear discernment functions are the simplest discriminant functions:
$$g(x) = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{n+1} \tag{1}$$
- Consider $\mathbf{x} = [x_1, \ldots, x_n]^T$, and $\mathbf{w} = [w_1, \ldots, w_n]^T$, (1) can be redefined as $g(x) = \mathbf{w}^T \mathbf{x} + w_{n+1}$
- Now we are looking for $\mathbf{w}$ and w_{n+1} for classification
- The classifier using the discriminant function (1) is called Linear Machine.
- Minimum distance classifier (MDC) or nearest neighborhood are employed to classify the patterns and find w's:
  - E^n is the n-dimensional Euclidean pattern space $\rightsquigarrow$ Euclidean distance between two point are $\|x_i - x_j\| = [(x_i - x_j)^T (x_i - x_j)]^{1/2}$.
  - P_i is center of gravity of cluster i.
  - A MDC computes the distance from pattern x of unknown to each prototype ($\|x - p_i\|$).
  - The prototype with smallest distance is assigned to the pattern.

Linear Machine

- Calculating the squared distance:
  $\|x - p_i\|^2 = (x - p_i)^T (x - p_i) = x^T x - 2x^T p_i + p_i^T p_i$
- $x^T x$ is independent of i
- $\therefore$ min the equation above is obtained by max the discernment function: $g_i(x) = x^T p_i - \frac{1}{2} p_i^T p_i$
- We had $g_i(x) = \mathbf{w}_i^T \mathbf{x} + w_{in+1}$
- $\therefore$ Considering $p_i = (p_{i1}, p_{i2}, ..., p_{in})^T$,
- The weights are defined as

$$\begin{aligned} w_{ij} &= p_{ij} \\ w_{in+1} &= -\frac{1}{2} p_i^T p_i, \\ & \quad i = 1, ..., R, j = 1, ..., n \end{aligned} \tag{2}$$



A linear classifier.

Example

- In this example a linear classifier is designed
- Center of gravity of the prototypes are known a priori

$$p_1 = \begin{bmatrix} 10 \\ 2 \end{bmatrix}, p_2 = \begin{bmatrix} 2 \\ -5 \end{bmatrix}, p_3 = \begin{bmatrix} -5 \\ 5 \end{bmatrix}$$

- Using (2) for $R = 3$, the weights are

$$w_1 = \begin{bmatrix} 10 \\ 2 \\ -52 \end{bmatrix}, w_2 = \begin{bmatrix} 2 \\ -5 \\ -14.5 \end{bmatrix}, w_3 = \begin{bmatrix} -5 \\ 5 \\ -25 \end{bmatrix}$$

- Discriminant functions are:

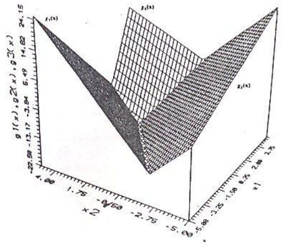
$$\begin{aligned} g_1(x) &= 10x_1 + 2x_2 - 52 \\ g_2(x) &= 2x_1 - 5x_2 - 14.5 \\ g_3(x) &= -5x_1 + 5x_2 - 25 \end{aligned}$$

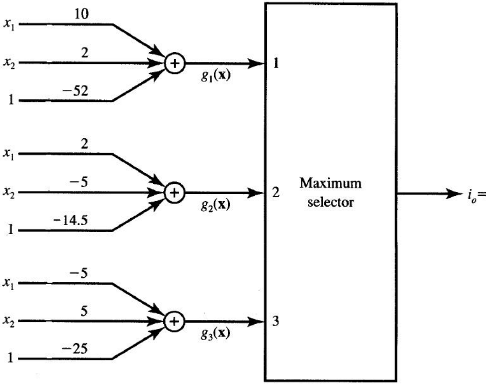
The decision lines are:

$$S_{12} \quad : 8x_1 + 7x_2 - 37.5 = 0$$

$$S_{13} \quad : -15x_1 + 3x_2 + 27 = 0$$

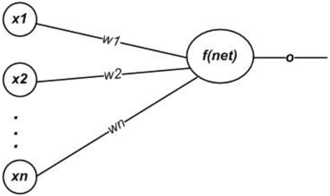
$$S_{23} \quad : -7x_1 + 10x_2 - 10.5 = 0$$





Bias or Threshold?

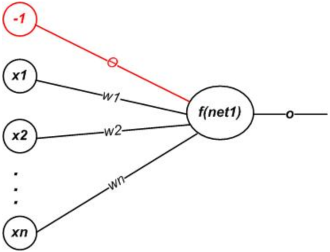
- Revisit the structure of a single layer network



- Considering the threshold (θ) the activation function is defined as

$$o = \begin{cases} 1 & net \geq \theta \\ -1 & net < \theta \end{cases} \tag{3}$$

- Now define $net_1 = net - \theta$:



- $\therefore$ The activation function can be considered as

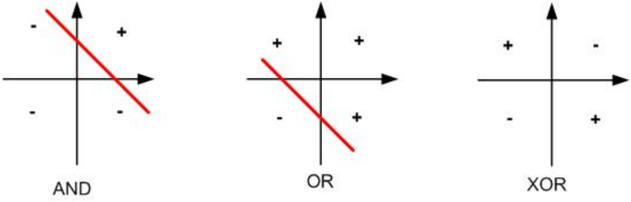
$$o = \begin{cases} 1 & net_1 \geq 0 \\ -1 & net_1 < 0 \end{cases} \tag{4}$$

- $\therefore$Bias can be play as a threshold in activation function.
- Considering neither threshold nor bias implies that discriminant function always intersects the origin which is not always correct.

- If R linear functions $g_i(x) = w_1x_1 + w_2x_2 + \ldots + w_nx_n + w_{n+1}, i = 1, \ldots, R$ exists s.t

$$g_i(x) > g_j(x) \ \forall x \in \S_i, i, j = 1, \ldots, R, i \neq j$$

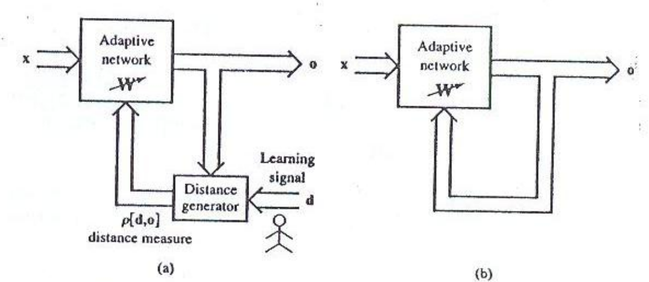
  the pattern set is linearly separable
- Single layer networks can only classify linearly separable patterns
- Nonlinearly separable patterns are classified by multiple layer networks
- **Example:** AND: $x_2 = -x_1 + 1 \ \ (b = -1, w_1 = 1, \ w_2 = 1)$
- **Example:** OR: $x_2 = -x_1 - 1 \ \ (b = 1, w_1 = 1, \ w_2 = 1)$



Learning

- When there is no a priori knowledge of p_i's, a method should be found to adjust weights (w_i).
- We should learn the network to behave as we wish
- Learning task is finding w based on the set of training examples x to provide the best possible approximation of $h(x)$.
    - In classification problem $h(x)$ is discriminant function $g(x)$.
- Two types of learning is defined
    1. Supervised learning: At each instant of time when the input is applied the desired response d is provided by teacher
        - The error between actual and desired response is used to correct and adjust the weights.
        - It rewards accurate classifications/ associations and punishes those yields inaccurate response.

2. Unsupervised learning: desired response is not known to improve the network behavior.
   - A proper self-adoption mechanism have to be embedded.



Block diagram for explanation of basic learning modes: (a) supervised learning and (b) unsupervised learning.

- General learning rule is the weight vector $w_i = [w_{i1} \; w_{i2} \; ... \; w_{in}]^T$ increases in proportion to the product of input x and learning signal r

$$\begin{aligned} w_i^{k+1} &= w_i^k + \Delta w_i^k \\ \Delta w_i^k &= c r^k(w_i^k, x^k) x^k \end{aligned}$$

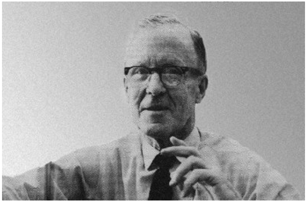
c is pos. const.: learning constant.

- For supervised learning $r = r(w_i, x, d_i)$
- For continuous learning

$$\frac{dw_i}{dt} = crx$$

Hebb Learning

- Reference: Hebb, D.O. (1949), The organization of behavior, New York: John Wiley and Sons
- Wikipedia:”Donald Olding Hebb (July 22, 1904 August 20, 1985) was a Canadian psychologist who was influential in the area of neuropsychology, where he sought to understand how the function of neurons contributed to psychological processes such as learning. ”
- ”He has been described as the father of neuropsychology and neural networks.”



http://www.scholarpedia.org/article/Donald_Hebb

- Hebbean rule is the oldest and simplest learning rule.
- "The general idea is an old one, that any two cells or systems of cells that are repeatedly active at the same time will tend to become 'associated', so that activity in one facilitates activity in the other." (Hebb 1949, p. 70)
- Hebb's principle is a method of learning, i.e., adjust the weights between model neurons.
  - The weight between two neurons increases if the two neurons activate simultaneously and reduces if they activate separately.
- Mathematically the Hebbian learning can be expressed:

$$\begin{aligned} r^k &= y^k \\ \Delta w_i^k &= c x_i^k y^k \end{aligned} \tag{5}$$

- $\therefore$ Larger input yields more effect on its corresponding weights.
- For supervised learning y is replaced by d in (5).
  - This rule is known as Correlation rule

Learning Algorithm for Supervised Hebbian Rule

- Assume m I/O pairs of (s, d) are available for training
  1. Consider a random initial values for weights
  2. Set $k = 1$
  3. $x_i^k = s_i^k, \; y^k = d^k, \; i = 1, ..., n$
  4. Update the weights as follows

$$\begin{aligned} w_i^{k+1} &= w_i^k + x_i^k y^k, i = 1, ..., n \\ b^{k+1} &= b^k + y^k \end{aligned}$$

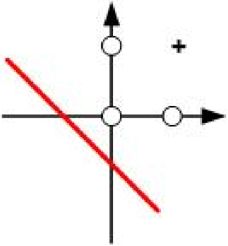
  5. $k = k + 1$ if $k \leq m$ return to step 3 (Repeat steps 3 and 4 for another pattern), otherwise end
- In this algorithm bias is also updated.
- Note: If the I/O data are binary, there is no difference between $x = 1, y = 0$ and $x = 0, y = 0$ in Hebbian learning.

Example: And

- The desired I/O pairs are given in the table

x_1	x_2	b (bias)	d (target)
1	1	1	1
1	0	1	0
0	1	1	0
0	0	1	0

- $\Delta w_1 = x_1 d, \ \Delta w_2 = x_2 d, \ \Delta b = d$
- Consider $w_0 = [0 \ 0 \ 0]$
- Using first pattern: $\Delta w^1 = [1 \ 1 \ 1] \rightsquigarrow w^1 = [1 \ 1 \ 1]$
- $\therefore x_2 = -x_1 - 1$
- Using pattern 2,3, and 4 does not change weights.
- training is stopped but the weights cannot represent pattern 1!!



- Now assume the target is bipolar

x_1	x_2	b (bias)	d (target)
1	1	1	1
1	0	1	-1
0	1	1	-1
0	0	1	-1

- Pattern 1:$\Delta w^1 = [1\ 1\ 1] \rightsquigarrow w^1 = [1\ 1\ 1], x_2 = -x_1 - 1$
- Pattern 2:$\Delta w^2 = [-1\ 0\ \ -1] \rightsquigarrow w^2 = [0\ 1\ 0]$
- Pattern 3:$\Delta w^3 = [0\ \ -1\ \ -1] \rightsquigarrow w^3 = [0\ 0\ \ -1]$
- Pattern 4:$\Delta w^4 = [0\ 0\ \ -1] \rightsquigarrow w^4 = [0\ 0\ \ -2]$
- The training is finished but it does not provide correct response to pattern 1:(

- Now assume both input and target are bipolar

x_1	x_2	b (bias)	d (target)
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

- Pattern 1:$\Delta w^1 = [1\ 1\ 1] \rightsquigarrow w^1 = [1\ 1\ 1], x_2 = -x_1 - 1$ (correct for p_1 and p_4)
- Pattern 2:$\Delta w^2 = [-1\ 1\ -1] \rightsquigarrow w^2 = [0\ 2\ 0], x_2 = 0$ (correct for p_1, p_2 and p_4)
- Pattern 3:$\Delta w^3 = [1\ -1\ -1] \rightsquigarrow w^3 = [1\ 1\ -1], x_2 = -x_1 + 1$ (correct for all patterns)
- Pattern 4:$\Delta w^4 = [1\ 1\ -1] \rightsquigarrow w^4 = [2\ 2\ -2] x_2 = -x_1 + 1$
- Correct for all patterns:)

Perceptron Learning Rule

- Perceptron learning rule was first time proposed by Rosenblatt in 1960.
- Learning is supervised.
- The weights are updated based the error between the system output and desired output

$$\begin{aligned} r^k &= d^k - o^k \\ \Delta W_i^k &= c(d_i^k - o_i^k)x^k \end{aligned} \quad (6)$$

- Based on this rule weights are adjusted iff output o_i^k is incorrect.
- The learning is repeated until the output error is zero for every training pattern
- It is proven that by using Perceptron rule the network can learn what it can present
  - If there are desired weights to solve the problem, the network weights converge to them.

- In classification, consider $R = 2$
  - The Decision lines are

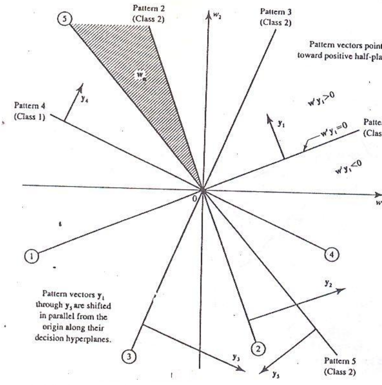
$$\begin{aligned} \bar{W}^T x &- b = 0 \\ \bar{W} &= [w_1,\ w_2,\ \ldots,\ w_n]^T, x = [x_1,\ x_2, \ldots, x_n]^T \end{aligned} \quad (7)$$

  - Considering bias, the augmented weight and input vectors are $y = [x_1,\ x_2, \ldots, x_n,\ \ -1]^T,\ W = [w_1,\ w_2,\ \ldots,\ w_n,\ b]^T$
  - Therefore, (7) can be written

$$W^T y = 0 \quad (8)$$

  - Since in learning, we are focusing on updating weights, the decision lines are presented on weights plane.
  - $\therefore$ The decision line always intersects the origin $(w = 0)$.
  - Its normal vector is y which is perpendicular to the plane

- The normal vector always points toward $W^T y > 0$.
- Positive decision region, $W^T y > 0$ is class 1
- Negative decision region, $W^T y < 0$ is class 2
- Using geometrical analysis, we are looking for a guideline for developing weight vector adjustment procedure.



Decision hyperplane in augmented weight space for a five pattern set from two classes.

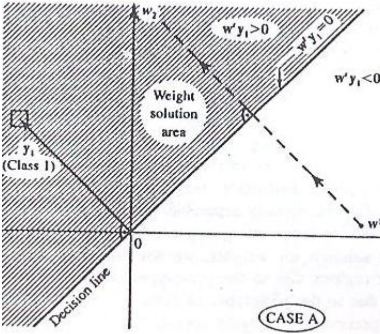
- **Case A:** W^1 is in neg. half-plane, y_1 belongs to class 1
  - $W^1 y_1 < 0 \rightsquigarrow W^T$ should be moved toward gray section.
  - The fast solution is moving W^1 in the direction of steepest increase which is the gradient

$$\nabla_w(W^T y_1) = y_1$$

  - $\therefore$ The adjusted weights become

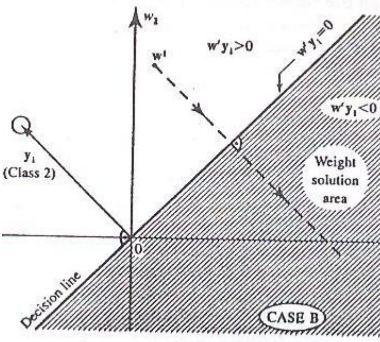
$$W' = W^1 + c y_1$$

  - $c > 0$ is called correction increment or learning rate, it is the size of adjustment.
  - W' is the weights after correction.

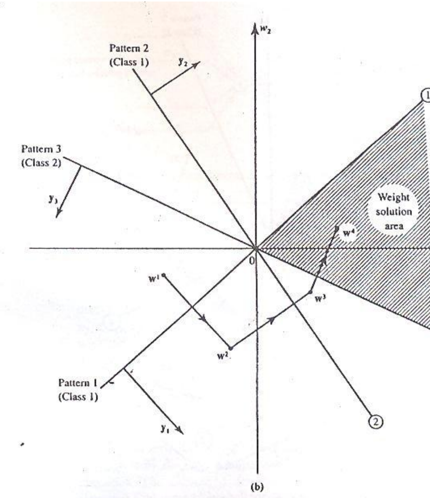


- **Case B:** W^1 is in pos. half-plane, y_1 belongs to class 2
  - $W^1 y_1 > 0 \rightsquigarrow W^T$ should be moved toward gray section.
  - The fast solution is moving W^1 in the direction of steepest decrease which is the neg. gradient
  - $\therefore$ The adjusted weights become

$$W' = W^1 - c y_1$$



- **Case C:** Consider three augmented patterns $y_1,\ y_2$ and y_3, given in sequence
- The response to $y_1,\ y_2$ should be 1 and for y_3 should be -1
- The lines 1, 2, and 3 are fixed for variable weights.
  - Starting with W^1 and input y_1, $W^T y_1 < 0 \rightsquigarrow W^2 = W^1 + cy_1$
  - For y_2, $W^T y_2 < 0 \rightsquigarrow W^3 = W^2 + cy_2$
  - For y_3, $W^T y_3 > 0 \rightsquigarrow W^4 = W^3 - cy_3$
  - W^4 is in the gray area (solution)



Weight adjustments of learning dichotomizer *(continued)*:

- To summarize, the updating rule is

$$W' = W^1 \pm cy$$

where

  - Pos. sign is for undetected pattern of class 1
  - Neg. sign is for undetected pattern of class 2
  - For correct classification, no adjustment is made.
- The updating rule is, indeed, (6)

Single Discrete Perceptron Training Algorithm

- Given P training pairs $\{x_1, d_1,\ x_2, d_2,\ \ldots, x_p, d_p\}$ where x_i is $(n \times 1)$, d_i is (1×1), $i = 1, \ldots, P$
- The augmented input vectors are $y_i = \begin{bmatrix} x_i \\ -1 \end{bmatrix}$, for $i = 1, \ldots, P$
- In the following, k is training step and p is step counter within training cycle.
  1. Choose $c > 0$
  2. Initialized weights at small random values, w is $(n+1) \times 1$
  3. Initialize counters and error: $k \longleftarrow 1,\ p \longleftarrow 1, E \longleftarrow 0$
  4. Training cycle begins here. Set $y \longleftarrow y_p, d \longleftarrow d_p,\ o = sgn(w^T y)$ (sgn is sign function)
  5. Update weights $w \longleftarrow w + \frac{1}{2}c(d - o)y$
  6. Find error: $E \longleftarrow \frac{1}{2}(d - o)^2 + E$
  7. If $p < P$ then $p \longleftarrow p + 1, k \longleftarrow k + 1$, go to step 4, otherwise, go to step 8.
  8. If $E = 0$ the training is terminated, otherwise $E \longleftarrow 0, p \longleftarrow 1$ go to step 4 for new training cycle

Convergence of Perceptron Learning Rule

- Learning is finding optimum weights W^* s.t.

$$\begin{cases} {W^*}^T y > 0 & \text{for} x \in \S_1 \\ {W^*}^T y < 0 & \text{for} x \in \S_2 \end{cases}$$

- Training is terminated when there is no error in classification, $(w^* = w^n = w^{n+1})$.
- Assume after n steps learning is terminated.
- Objective: Show n is bounded, i.e., after limited number of updating, the weights converge to their optimum values
- Assume $W^0 = 0$
- For $\delta = \min\{abs({W^*}^T y)\}$:

$$\begin{cases} {W^*}^T y \geq \delta > 0 & \text{for} x \in \S_1 \\ {W^*}^T y \leq -\delta < 0 \ \ (-{W^*}^T y \geq \delta) & \text{for} x \in \S_2 \end{cases}$$

- If at each step the error is nonzero, the weights are updated:

$$W^{k+1} = W^k \pm y \tag{9}$$

- Multiply (9) by W^{*T}

$$W^{*T} W^{k+1} = W^{*T} W^k \pm W^{*T} y \Longrightarrow W^{*T} W^{k+1} \geq W^{*T} W^k + \delta$$

- For n times updating rule, and considering $W^0 = 0$

$$W^{*T} W^n \geq W^{*T} W^0 + n\delta = n\delta \tag{10}$$

- Using Schwartz inequality $\|W^n\|^2 \geq \frac{(W^{*T} W^n)^2}{\|W^*\|^2}$

$$\|W^n\|^2 \geq \frac{n^2\delta^2}{B} \tag{11}$$

where $\|W^*\|^2 = B$

- On the other hand

$$\|W^{k+1}\|^2 = (W^k \pm y)^T (W^k \pm y) = W^{kT} W^k + y^T y \pm 2W^{kT} y \quad (12)$$

- Weights are updated when there is an error, i.e., $-sgn(W^{kT} y)$ appears in last term of (12)

$$\begin{cases} W^{kT} y < 0 & \text{error for class 1} \\ W^{kT} y > 0 & \text{error for class 2} \end{cases}$$

- $\therefore$ The last term of (12) is neg. and

$$\|W^{k+1}\|^2 \leq \|W^k\|^2 + M \quad (13)$$

where $\|y\|^2 \leq M$

- After n times updating and considering $W^0 = 0$

$$\|W^n\|^2 \leq nM \tag{14}$$

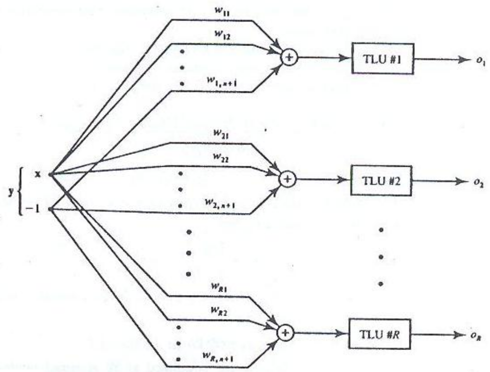
- Considering (11) and (14)

$$\frac{n^2\delta^2}{B} \leq \|W^n\|^2 \leq nM \rightsquigarrow \frac{n^2\delta^2}{B} \leq nM$$
$$\therefore \quad n \leq \frac{MB}{\delta^2}$$

- So n is bounded

Multi-category Single Layer Perceptron

- The perceptron learning rule so far was limited for two category classification
- We want to extend it for multigategory classification
- The weight of each neuron (TLU) is updated independent of other weights.
- The k's TLU reponses +1 and other TLU's -1 to indicate class k



R-category linear classifier using R discrete perceptrons.

R-Category Discrete Perceptron Training Algorithm

- Given P training pairs $\{x_1, d_1, \ x_2, d_2, \ \ldots, x_p, d_p\}$ where x_i is $(n \times 1)$, d_i is $(R \times 1)$, $i = 1, \ldots, P$
- The augmented input vectors are $y_i = \begin{bmatrix} x_i \\ -1 \end{bmatrix}$, for $i = 1, \ldots, P$
- In the following, k is training step and p is step counter within training cycle.
  1. Choose $c > 0$
  2. Initialized weights at small random values, $W = [w_{ij}]$ is $R \times (n+1)$
  3. Initialize counters and error: $k \longleftarrow 1$, $p \longleftarrow 1, E \longleftarrow 0$
  4. Training cycle begins here. Set $y \longleftarrow y_p, d \longleftarrow d_p$, $o_i = sgn(w_i^T y)$ for $i = 1, \ldots, R$ (sgn is sign function)
  5. Update weights $w_i \longleftarrow w_i + \frac{1}{2}c(d_i - o_i)y$ for $i = 1, \ldots, R$
  6. Find error: $E \longleftarrow \frac{1}{2}(d_i - o_i)^2 + E$ for $i = 1, \ldots, R$
  7. If $p < P$ then $p \longleftarrow p + 1, k \longleftarrow k + 1$, go to step 4, otherwise, go to step 8.
  8. If $E = 0$ the training is terminated, otherwise $E \longleftarrow 0, p \longleftarrow 1$ go to step 4 for new training cycle

Example

- Revisit the three classes example
- The discriminant values are

Discriminant	Class 1 $[10\ 2]'$	Class 2 $[2\ \ -5]'$	Class 3 $[-5\ 5]'$
$g_1(x)$	52	-42	-92
$g_2(x)$	-4.5	14.5	-49.5
$g_3(x)$	-65	-60	25

- So the thresholds values:w_{13}, w_{23}, and w_{33} are 52, 14.5, and 25, respectively.

- Now use perceptron learning rule:
- Consider randomly chosen initial values:
  $w_1^1 = [1 \ -2 \ 0]', \ w_2^1 = [0 \ -1 \ 2]', \ w_3^1 = [1 \ 3 \ -1]'$
- Use the patterns in sequence to update the weights:
  - y_1 is input:

$$sgn([1 \ -2 \ 0] \begin{bmatrix} 10 \\ 2 \\ -1 \end{bmatrix}) = 1$$

$$sgn([0 \ -1 \ 2] \begin{bmatrix} 10 \\ 2 \\ -1 \end{bmatrix}) = -1$$

$$sgn([1 \ 3 \ -1] \begin{bmatrix} 10 \\ 2 \\ -1 \end{bmatrix}) = 1^*$$

  - TLU # 3 has incorrect response. So
    $w_1^2 = w_1^1, \ w_2^2 = w_2^1, \ w_3^2 = [1 \ 3 \ -1]' - [10 \ 2 \ -1]' = [-9 \ 1 \ 0]'$

- y_2 is input:

$$sgn([1\ \ -2\ 0]\begin{bmatrix} 2 \\ -5 \\ -1 \end{bmatrix}) \ = \ 1^*$$

$$sgn([0\ \ -1\ 2]\begin{bmatrix} 2 \\ -5 \\ -1 \end{bmatrix}) \ = \ 1$$

$$sgn([-9\ 1\ 0]\begin{bmatrix} 2 \\ -5 \\ -1 \end{bmatrix}) \ = \ -1$$

  - TLU # 1 has incorrect response. So $w_1^3 = [1\ 2\ 0]' - [2`-5\ \ -1]' = [-1\ 3\ 1]',\ w_2^3 = w_2^2,\ w_3^3 = w_3^2$

- y_3 is input:

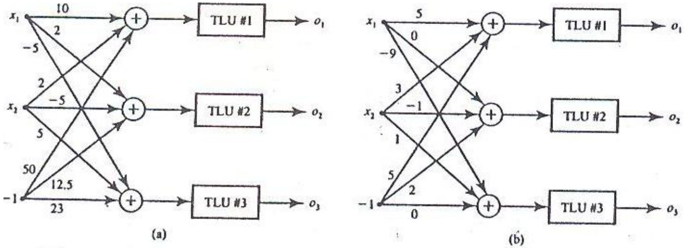
$$sgn([-1\ 3\ 1]\begin{bmatrix}-5\\5\\-1\end{bmatrix}) = 1^*$$

$$sgn([0\ -1\ 2]\begin{bmatrix}-5\\5\\-1\end{bmatrix}) = -1$$

$$sgn([-9\ 1\ 0]\begin{bmatrix}-5\\5\\-1\end{bmatrix}) = 1$$

  - TLU # 1 has incorrect response. So $w_1^4 = [4\ -2\ 2]',\ w_2^4 = w_2^3,\ w_3^4 = w_3^3$
- First learning cycle is finished but the error is not zero, so the training is not terminated

- In next training cycles TLU # 2 and 3 are correct.
- TLU # 1 is changed as follows
  $w_1^5 = w_1^4,\ w_1^6 = [2\ 3\ 3]',\ w_1^7 = [7\ \ -2\ 4]',\ w_1^8 = w_1^7,\ w_1^9 = [5\ 3\ 5]$
- The trained network is
$$\begin{aligned} o_1 &= sgn(5x_1 + 3x_2 - 5) \\ o_2 &= sgn(-x_2 - 2) \\ o_3 &= sgn(-9x_1 + x_2) \end{aligned}$$
- The discriminant functions for classification are not unique.



(a) three-perceptron classifier from maxi- mum selector, (b) three-perceptron trained classifier,

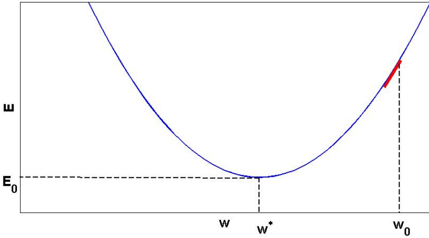
Continuous Perceptron

- In many cases the output is not necessarily limited to two values (± 1)
- Therefore, the activation function of NN should be continuous
- The training is indeed defined as adjusting the optimum values of the weights, s.t. minimize a criterion function
- This criterion function can be defined based on error between the network output and desired output.
- Sum of square root error is a popular error function
- The optimum weights are achieved using gradient or steepest decent procedure.

- Consider the error function: $E = E_0 + \lambda(w - w^*)^2 \rightsquigarrow \frac{dE}{dw} = 2\lambda(w - w^*)$, $\frac{d^2E}{dw^2} = 2\lambda$
- The problem is finding w^* s.t min E
- To achieve min error at $w = w^*$ from initial weight w_0, the weights should move in direction of negative gradient of the curve.
- $\therefore$ The updating rule is

$$w^{k+1} = w^k - \eta \nabla E(w^k)$$

where η is pos. const. called learning constant.



- The error to be minimized is

$$E^k = \frac{1}{2}(d^k - o^k)^2$$
$$o^k = f(net^k)$$

- For simplicity superscript k is skipped. But remember the weights updates is doing at kth training step.
- The gradient vector is

$$\nabla E(w) = -(d-o)f'(net)\begin{bmatrix} \frac{\partial(net)}{\partial w_1} \\ \frac{\partial(net)}{\partial w_2} \\ \vdots \\ \frac{\partial(net)}{\partial w_{n+1}} \end{bmatrix}$$

- $net = w^T y \rightsquigarrow \frac{\partial(net)}{\partial w_i} = y_i$ for $i = 1, ..., n+1$
- $\therefore \nabla E = -(d-o)f'(net)y$

- The TLU activation function is not useful, since its time derivative is always zero and indefinite at $net = 0$
- Use sigmoid activation function

$$f(net) = \frac{2}{1 + exp(-net)} - 1$$

- Time derivative of sigmoid function can be expressed based on the function itself

$$f'(net) = \frac{2exp(-net)}{(1 + exp(-net))^2} = \frac{1}{2}(1 - f(net)^2)$$

- $o = f(net)$, therefore,

$$\nabla E(w) = -\frac{1}{2}(d - o)(1 - o^2)y$$

- Finally the updating rule is

$$w^{k+1} = w^k + \frac{1}{2}\eta(d^k - o^k)(1 - o^{k2})y^k \tag{15}$$

- Comparing the updating rule of continuous perceprton (15) with the discrete perceptron learning ($w^{k+1} = w^k + \frac{c}{2}(d^k - o^k)y^k$)
  - The correction weights are in the same direction
  - Both involve adding/subtracting a fraction of the pattern vector y
  - The essential difference is scaling factor $1 - o^{k2}$ which is always positive and smaller than 1.
  - In continuous learning, for a weaker committed perceptron (*net* close to zero) the correction scaling factor is larger than the more close responses with large magnitude.

Single Continuous Perceptron Training Algorithm

- Given P training pairs $\{x_1, d_1,\ x_2, d_2,\ \ldots, x_p, d_p\}$ where x_i is $(n \times 1)$, d_i is (1×1), $i = 1, \ldots, P$
- The augmented input vectors are $y_i = [x_i \ \ -1]^T$, for $i = 1, \ldots, P$
- In the following, k is training step and p is step counter within training cycle.
  1. Choose $\eta > 0,\ \lambda = 1, E_{max} > 0$
  2. Initialized weights at small random values, w is $(n \times 1) \times 1$
  3. Initialize counters and error: $k \longleftarrow 1,\ p \longleftarrow 1, E \longleftarrow 0$
  4. Training cycle begins here. Set $y \longleftarrow y_p, d \longleftarrow d_p,\ o = f(w^T y)$ (f(net) is sigmoid function)
  5. Update weights $w \longleftarrow w + \frac{1}{2}\eta(d - o)(1 - o^2)y$
  6. Find error: $E \longleftarrow \frac{1}{2}(d - o)^2 + E$
  7. If $p < P$ then $p \longleftarrow p + 1, k \longleftarrow k + 1$, go to step 4, otherwise, go to step 8.
  8. If $E < E_{max}$ the training is terminated, otherwise $E \longleftarrow 0, p \longleftarrow 1$ go to step 4 for new training cycle.

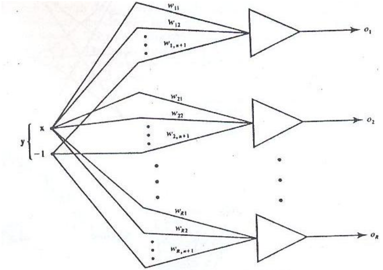
R-Category Continues Perceptron

- Gradient training rule derived for $R = 2$ is also applicable for multi-category classifier
- The training rule with be changed to

$$w_i^{k+1} = w_i^k + \frac{1}{2}\eta(d_i^k - o_i^k)(1 - o_i^{k2})y^k, \quad for \;\; i = 1, ..., R$$

- It is equivalent to individual weight adjustment

$$w_{ij}^{k+1} = w_{ij}^k + \frac{1}{2}\eta(d_i^k - o_i^k)(1 - o_i^{k2})y_j^k, \quad for \;\; j = 1, ..., n+1, \;\; i = 1, ..., R$$



R-category linear classifier using continuous perceptrons.

ADAptive LInear NEuron (ADALINE)

- Similar to Perceptron with different activation function.
- Activation function is linear $x^T w = d$ (16)

  where $x = [x_1 \ x_2 \ ...x_n \ 1]^T$, $w = [w_1 \ w_2 \ ...w_n \ b]^T$, $d = [d_1 \ d_2 \ ...d_n]^T$ is desired output.
- For m patterns, Eq. (16) will be

$$
\begin{aligned}
x_{11}w_1 + x_{12}w_2 + \ldots + x_{1n}w_n + b &= d_1 \\
x_{21}w_1 + x_{22}w_2 + \ldots + x_{2n}w_n + b &= d_2 \\
&\vdots \\
x_{m1}w_1 + x_{m2}w_2 + \ldots + x_{mn}w_n + b &= d_m
\end{aligned}
$$

$$
= \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} & 1 \\ x_{21} & x_{22} & \ldots & x_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 \\ x_{m1} & x_{m2} & \ldots & x_{mn} & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ b \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix}
$$

$$x^T x w = x^T d \Rightarrow w = x^* d$$

- x^* is pseudo inverse matrix of x

$$x^* = (x^T x)^{-1} x^T$$

- This method is not useful in practice
  - The weights are obtained from fixed patterns
  - All the patterns are applied in one shot.
- In most practical applications, patterns encountered sequentially one at a time.

$$w^{k+1} = w^k + \alpha(d^k - w^k x^k)x^k \tag{17}$$

- This method is named Widraw-Hoff Procedure
- This learning method is also based in min least square error method between output and desired signal