Perceptron

Sanjay Singh[⋆†]

[⋆]Department of Information and Communication Technology
Manipal Institute of Technology, MAHE
Manipal-576104, INDIA
sanjay.singh@manipal.edu

[†]Centre for Artificial and Machine Intelligence (CAMI)
MAHE, Manipal-576104, INDIA

February 1, 2019

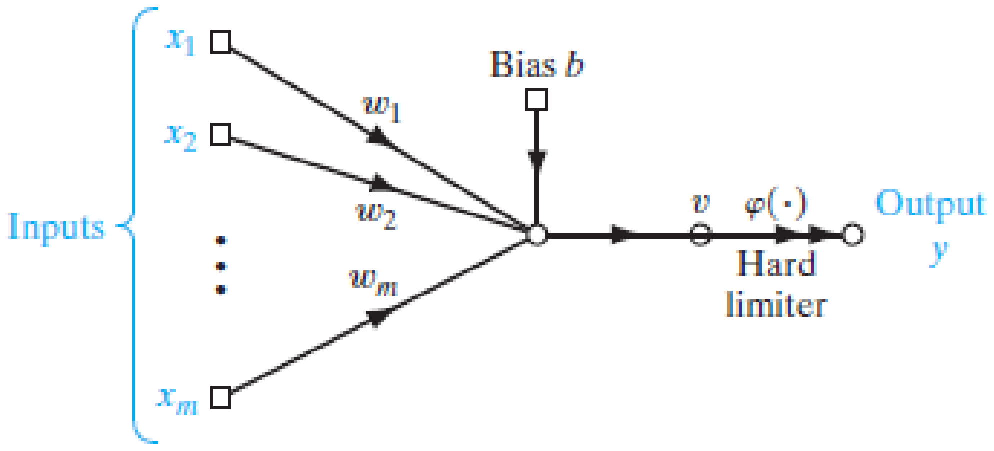
Historical Overview (1943-1958)

- McCulloch and Pitts (1943): neural networks as computing machine
- Hebb (1949): postulated the first rule for self-organizing learning
- Rosenblatt (1958): perceptron as a first model of supervised learning
- Widrow and Hoff (1960): adaptive filters using least-mean-square (LMS) algorithm (delta rule)

Introduction

- Perceptron is the somplest form of a neural network used for classification of linearly separable patterns
- It consist of a single neuron with adjustable synaptic weight and bias
- Perceptron built around a single neuron is limited to performing pattern classification with only two classes, but can be extended to more than two classes
- Perceptron uses McCulloch-Pitts model of a neuron

Perceptron



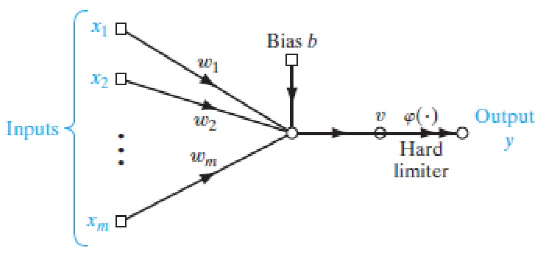
- Perceptron uses a non-linear model

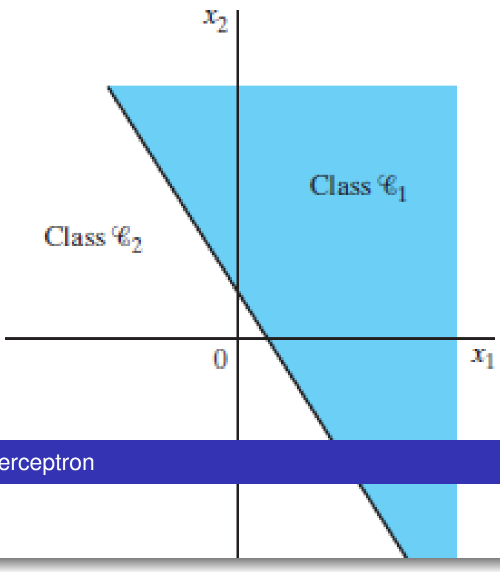
$$v = \sum_{i=1}^{m} w_i x_i + b, \quad y = \varphi(v) = \begin{cases} +1 & v > 0 \\ -1 & v \leq 0 \end{cases}$$

- Goal: classify input vectors into two classes

- To get the insight into the behavior of a pattern classifier, we need to plot a map of the decision regions in m-dimensional signal space spanned by the m input variables $x_1, x_2, \ldots, x_m$
- In the simplest form of perceptron, there are two decision regions separated by a hyperplane, which is defined by

$$\sum_{i=1}^{m} w_i x_i + b = 0$$

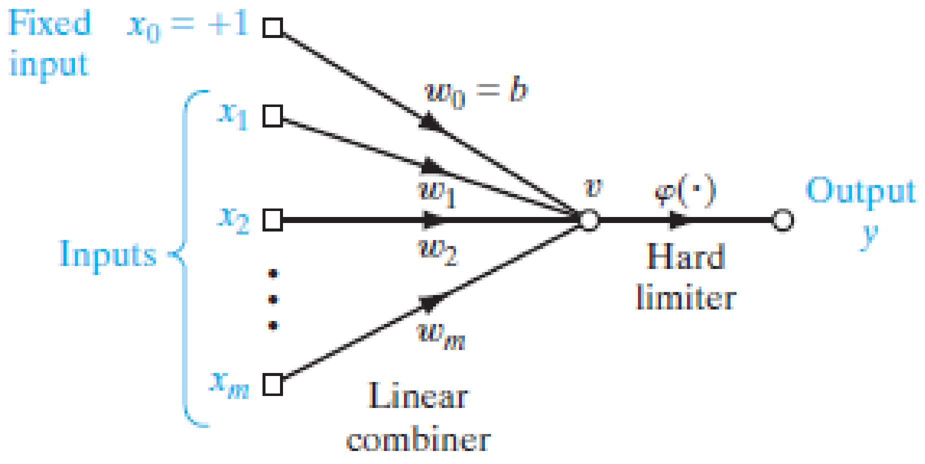




Perceptron Convergence Theorem

If the patterns (vectors) used to train the perceptron are drawn from two linearly separable classes, then the perceptron algorithm converges and positions the decision surface in the form of a hyperplane between the two classes. The proof of convergence of the algorithm is known as perceptron convergence theorem.

Perceptron Convergence Theorem



- $x(n) = [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$,
$$w(n) = [b(n), w_1(n), w_2(n), \ldots, w_m(n)]^T$$
- Liner combiner output is written as
$$v(n) = \sum_{i=}^{m} w_i(n)x_i(n) = w^T(n)x(n)$$

Fixed-Increment Convergence Theorem

Theorem

Let the subsets of training vector $\mathcal{L}_1$ and $\mathcal{L}_2$ be linearly separable. Let the inputs presented to the perceptron originate from these two subsets. The perceptron converges after some n_0 iteration, in the sense that
$$w(n_0) = w(n_0+1) = w(n_0+2) = \cdots$$
is a solution vector for $n_0 \le n_{max}$.

Perceptron Convergence Algorithm

Variable and Parameters:

- $x(n) = [+1, x_1(n), x_2(n), \dots, x_m(n)]^T$
- $w(n) = [b(n), w_1(n), w_2(n), \dots, w_m(n)]^T$
- $b(n) = bias$
- $y(n) =$ actual response
- $d(n) =$ desired response
- $\eta =$ learning rate parameter

Perceptron Convergence Algorithm

1. **Initialization**: set $w(0) = 0$, and perform following computations for $n = 1, 2, \dots$
2. **Activation**: at time step n, activate the perceptron by applying continuous-valued input vector $x(n)$ and desired response $d(n)$
3. **Computation of Actual Response**: compute the actual response of the perceptron:

$$y(n) = sgn[w^T(n)x(n)]$$

where $sgn(.)$ is the signum function, which is defined as

$$sgn(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ +1 & x > 0 \end{cases}$$

Perceptron Convergence Algorithm

4. **Adaptation of Weight Vector**: Update the weight vector of the perceptron:

$$w(n+1) = w(n) + \eta[d(n) - y(n)]x(n)$$

where

$$d(n) = \begin{cases} +1 & x(n) \in \mathcal{C}_1 \\ -1 & x(n) \in \mathcal{C}_2 \end{cases}$$

5. **Continuation**: increment time step n by one and go back to step 2.

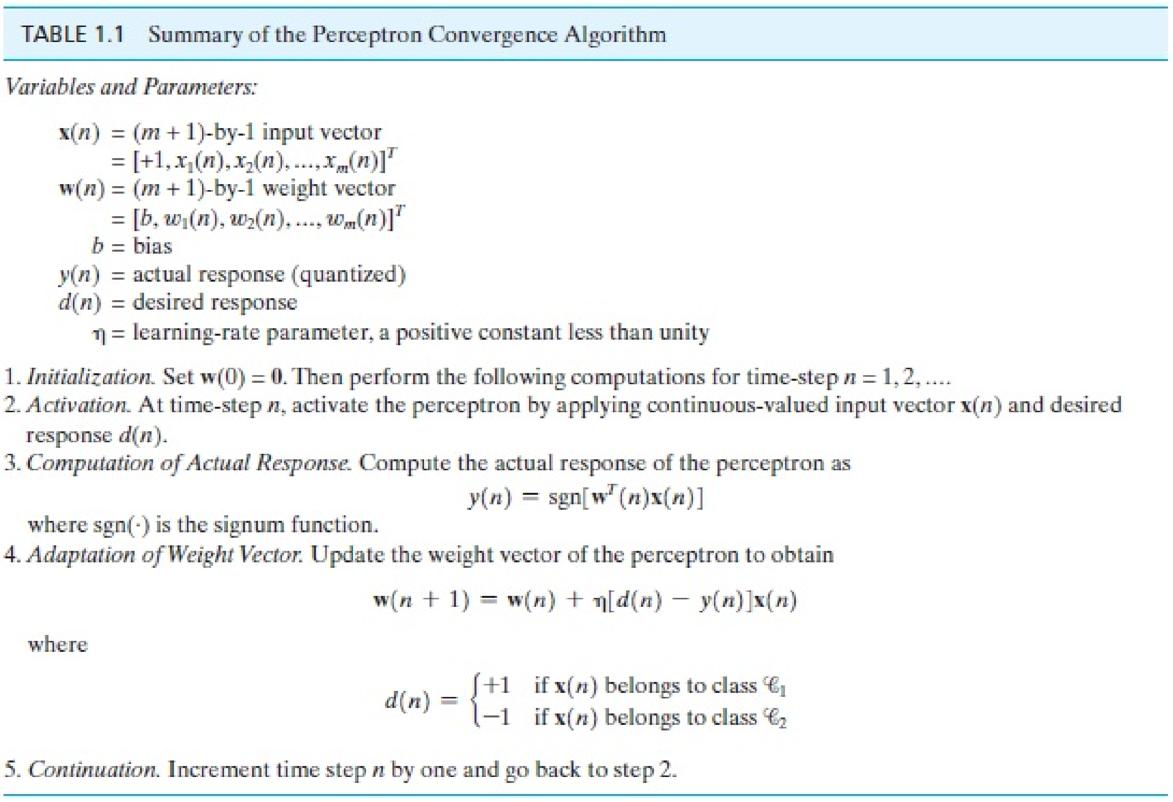


TABLE 1.1 Summary of the Perceptron Convergence Algorithm

Variables and Parameters:

$\mathbf{x}(n) = (m+1)$-by-1 input vector
$= [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$
$\mathbf{w}(n) = (m+1)$-by-1 weight vector
$= [b, w_1(n), w_2(n), \ldots, w_m(n)]^T$
b = bias
$y(n)$ = actual response (quantized)
$d(n)$ = desired response
η = learning-rate parameter, a positive constant less than unity

1. *Initialization.* Set $\mathbf{w}(0) = \mathbf{0}$. Then perform the following computations for time-step $n = 1, 2, \ldots$.
2. *Activation.* At time-step n, activate the perceptron by applying continuous-valued input vector $\mathbf{x}(n)$ and desired response $d(n)$.
3. *Computation of Actual Response.* Compute the actual response of the perceptron as
$$y(n) = \operatorname{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$
where $\operatorname{sgn}(\cdot)$ is the signum function.
4. *Adaptation of Weight Vector.* Update the weight vector of the perceptron to obtain
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$
where
$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{cases}$$
5. *Continuation.* Increment time step n by one and go back to step 2.

Perceptron and Bayes Classifier for Gaussian Environment

- Perceptron bears a certain relationship with classical pattern classifier known as Bayes classifier
- When the environment is Gaussian, the Bayes classifier reduces to a linear classifier
- Linear nature of the perceptron is not contingent on the assumption of Gaussianity

Bayes Classifier

- Bayes classifier or Bayes hypothesis testing procedure, we minimize the avg. risk, $\mathcal{R}$
- For two class problem, say $\mathcal{C}_1$ and $\mathcal{C}_2$, the avg.risk is defined by[1] Van Trees as

$$\mathcal{R} = c_{11}p_1 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_1)dx + c_{22}p_2 \int_{\mathcal{L}_2} f_x(x|\mathcal{C}_2)dx \\ + c_{21}p_1 \int_{\mathcal{L}_2} f_x(x|\mathcal{C}_1)dx + c_{12}p_2 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_2)dx \quad (1)$$

[1] H.L. Van Trees. *Detection, Estimation, and Modulation Theory, Set (Volumes: I,II, III,IV)*. . Detection, Estimation, and Modulation Theory. Wiley, 2003. ISBN: 978-0-471-44967-6.

Various terms are defined as:

- p_i: a priori probability that the observation vector x is drawn from subspace $\mathcal{L}_i$ and $p_1 + p_2 = 1$
- c_{ij}: cost of deciding in favor of class $\mathcal{C}_i$ represented by subspace $\mathcal{L}_i$ when class $\mathcal{C}_j$ is true
- $f_x(x|\mathcal{C}_i)$: conditional probability density function of the random variable X, given that the observation vector x is drawn from subspace $\mathcal{L}_i$

First two terms in eqn(1) represent correct decisions, whereas the last two terms represents incorrect decision

- To determine a strategy for the minimum avg.risk
- Overall observation space $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$
- Eqn(1) can be rewritten in the equivalent form as

$$\mathcal{R} = c_{11}p_1 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_1)dx + c_{22}p_2 \int_{\mathcal{L}-\mathcal{L}_1} f_x(x|\mathcal{C}_2)dx + c_{21}p_1 \int_{\mathcal{L}-\mathcal{L}_1} f_x(x|\mathcal{C}_1)dx + c_{12}p_2 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_2)dx \quad (2)$$

where $c_{11} < c_{21}$ and $c_{22} < c_{12}$

- $\int_{\mathcal{L}} f_x(x|\mathcal{C}_1)dx = \int_{\mathcal{L}} f_x(x|\mathcal{C}_2)dx = 1$

- Using following property of integral

$$\int_{VecSpace-SubSpace} f(x)dx = \int_{VecSpace} f(x)dx - \int_{SubSpace} f(x)dx$$

to eqn(2), we get

- $$\begin{aligned}\mathcal{R} = c_{11}p_1 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_1)dx + c_{22}p_2 \int_{\mathcal{L}} f_x(x|\mathcal{C}_2)dx \\ - c_{22}p_2 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_2)dx + c_{21}p_1 \int_{\mathcal{L}} f_x(x|\mathcal{C}_1)dx \\ - c_{21}p_1 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_1)dx + c_{12}p_2 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_2)dx \end{aligned} \quad (3)$$

- $$\begin{aligned}\mathcal{R} = c_{11}p_1 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_1)dx + c_{22}p_2 - c_{22}p_2 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_2)dx \\ + c_{21}p_1 - c_{21}p_1 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_1)dx + c_{12}p_2 \int_{\mathcal{L}_1} f_x(x|\mathcal{C}_2)dx \end{aligned} \quad (4)$$

- $$\begin{aligned}\mathcal{R} = c_{21}p_1 + c_{22}p_2 + \\ \int_{\mathcal{L}_1} \Big[p_2(c_{12} - c_{22})f_x(x|\mathcal{C}_2) - p_1(c_{21} - c_{11})f_x(x|\mathcal{C}_1)\Big]dx \end{aligned} \quad (5)$$

- First two terms in eqn(5) represent a fixed cost

- $$\mathcal{R} = c_{21}p_1 + c_{22}p_2 + \int_{\mathcal{L}_1} \Big[p_2(c_{12} - c_{22})f_x(x|\mathcal{C}_2) - p_1(c_{21} - c_{11})f_x(x|\mathcal{C}_1)\Big] dx$$
- First two terms in eqn(5) represent a fixed cost
- To minimize the average risk $\mathcal{R}$, we may deduce following strategy from eqn(5) for optimum classification:
  1. All the values of the observation vector x for which the integrand is negative should be assigned to subspace $\mathcal{L}_1$ (i.e., class $\mathcal{C}_1$) for the integral would then make a negative contribution to the risk $\mathcal{R}$
  2. All values of the observation vector x for which the integrand is positive should be excluded from subspace $\mathcal{L}_1$ (i.e., assigned to class $\mathcal{C}_2$) for the integral would then make a positive contribution to the risk $\mathcal{R}$
  3. Values of x for which the integrand is zero have no effect on the average risk $\mathcal{R}$ and may be assigned arbitrarily. We assume that such points are assigned to subspace $\mathcal{L}_2$

- Now we may formulate Bayes classifier as follows: if the condition

$$p_1(c_{21} - c_{11})f_x(x|\mathcal{C}_1) > p_2(c_{12} - c_{22})f_x(x|\mathcal{C}_2)$$

  holds, assign the observation vector x to subspace $\mathcal{L}_1$ (i.e., class $\mathcal{C}_1$). Otherwise assign x to $\mathcal{L}_2$
- Define $\Lambda(x) = \dfrac{f_x(x|\mathcal{C}_1)}{f_x(x|\mathcal{C}_2)}$ and

$$\xi = \frac{p_2(c_{12} - c_{22})}{p_1(c_{21} - c_{11})}$$

- Where $\Lambda(x)$ is called the likelihood ratio and ξ is called the threshold of the test. Both quantities are always positive
- *If, for an observation vector x, $\Lambda(x) > \xi$, assign x to class $\mathcal{C}_1$. Otherwise, assign it to class $\mathcal{C}_2$*

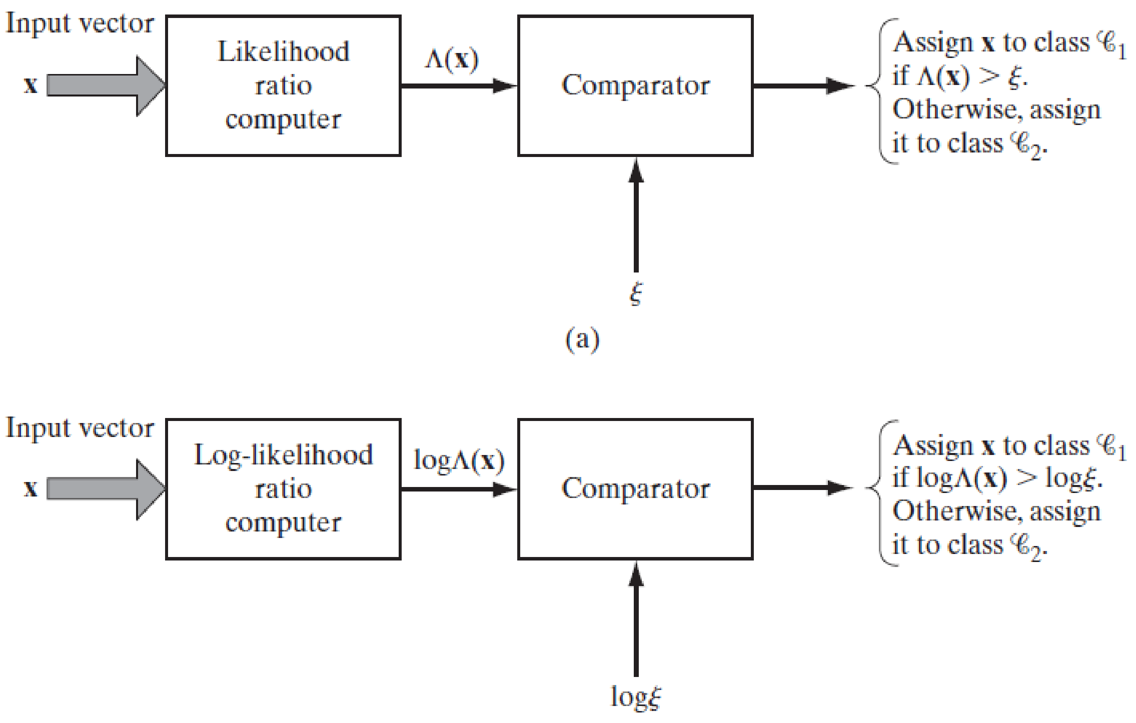

Input vector
x
Likelihood ratio computer
Λ(x)
Comparator
Assign x to class 𝒞1 if Λ(x) > ξ. Otherwise, assign it to class 𝒞2.
ξ
(a)
Input vector
x
Log-likelihood ratio computer
logΛ(x)
Comparator
Assign x to class 𝒞1 if logΛ(x) > logξ. Otherwise, assign it to class 𝒞2.
logξ

Figure 1: Two equivalent implementations of the Bayes classifier: (a) Likelihood ratio test, (b) Log-likelihood ratio test.

Bayes Classifier for a Gaussian Environment

- Consider a two class problem, with Gaussian is underlying distribution
- Random vector X has a mean value that depend on whether it belongs to class $\mathcal{C}_1$ or $\mathcal{C}_2$, but the covariance matrix of X is the same for both the classes
- That is to say

$$\text{Class } \mathcal{C}_1: \begin{aligned} E[X] &= \mu_1 \\ E[(X-\mu_1)(X-\mu_1)^T] &= C \end{aligned}$$

$$\text{Class } \mathcal{C}_2: \begin{aligned} E[X] &= \mu_2 \\ E[(X-\mu_2)(X-\mu_2)^T] &= C \end{aligned}$$

- C is non-diagonal, which means that samples drawn from classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are correlated
- It is assumed that C is non-singular, so that its inverse exists

- Conditional probability density function of $X \in \mathbb{R}^m$ is defined as

$$f_x(x|\mathcal{C}_i) = \frac{1}{(2\pi)^{m/2}(det(C))^{1/2}} exp\left(-\frac{1}{2}(x-\mu_i)^T C^{-1}(x-\mu_i)\right) \; i = 1,2 \tag{6}$$

- It is assumed that
  1. Two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ are equiprobable:

$$p_1 = p_2 = \frac{1}{2}$$

  2. Misclassification carry the same cost, and no cost is incurred on correct classifications:

$$c_{21} = c_{12} \qquad c_{11} = c_{22} = 0$$

- We have information to design the Bayes classifier for two class problem

Substituting eqn(6) in $\Lambda(x) = \dfrac{f_x(x|\mathcal{C}_1)}{f_x(x|\mathcal{C}_2)}$ and taking natural log we get

$$\begin{aligned}
\Lambda(x) &= \frac{\frac{1}{(2\pi)^{m/2}|C|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_1)^T C^{-1}(X-\mu_1)\right)}{\frac{1}{(2\pi)^{m/2}|C|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_2)^T C^{-1}(X-\mu_2)\right)} \\
&= \frac{\exp\left(-\frac{1}{2}(X-\mu_1)^T C^{-1}(X-\mu_1)\right)}{\exp\left(-\frac{1}{2}(X-\mu_2)^T C^{-1}(X-\mu_2)\right)} \\
\log \Lambda(x) &= \log \frac{\exp\left(-\frac{1}{2}(X-\mu_1)^T C^{-1}(X-\mu_1)\right)}{\exp\left(-\frac{1}{2}(X-\mu_2)^T C^{-1}(X-\mu_2)\right)} \\
\log \Lambda(x) &= \log \exp\left(-\frac{1}{2}(X-\mu_1)^T C^{-1}(X-\mu_1)\right) - \log \exp\left(-\frac{1}{2}(X-\mu_2)^T C^{-1}(X-\mu_2)\right) \\
&= -\frac{1}{2}(X-\mu_1)^T C^{-1}(X-\mu_1) + \frac{1}{2}(X-\mu_2)^T C^{-1}(X-\mu_2)
\end{aligned}$$

$$
\begin{aligned}
\log \Lambda(X) &= \frac{-1}{2}(X-\mu_1)^T C^{-1}(X-\mu_1) + \frac{1}{2}(X-\mu_2)^T C^{-1}(X-\mu_2) \\
&= \frac{1}{2}[-(X-\mu_1)^T C^{-1}(X-\mu_1) + (X-\mu_2)^T C^{-1}(X-\mu_2)] \\
&= \frac{1}{2}[-(X^T-\mu_1^T) C^{-1}(X-\mu_1) + (X^T-\mu_2^T) C^{-1}(X-\mu_2)] \\
&= \frac{1}{2}[-X^T C^{-1}(X-\mu_1) + \mu_1^T C^{-1}(X-\mu_1) + X^T C^{-1}(X-\mu_2) - \mu_2^T C^{-1}(X-\mu_2)] \\
&= \frac{1}{2}[-X^T C^{-1}X + X^T C^{-1}\mu_1 + \mu_1^T C^{-1}X - \mu_1^T C^{-1}\mu_1 + X^T C^{-1}X - X^T C^{-1}\mu_2 - \mu_2^T C^{-1}X \\
&= \frac{1}{2}[(\mu_1-\mu_2)^T C^{-1}X + (\mu_2^T C^{-1}\mu_2 - \mu_1^T C^{-1}\mu_1) + X^T C^{-1}(\mu_1-\mu_2)] \\
&= \frac{1}{2}[(\mu_1-\mu_2)^T C^{-1}X + (\mu_2^T C^{-1}\mu_2 - \mu_1^T C^{-1}\mu_1) + (\mu_1-\mu_2)^T C^{-1}X] \\
&= \frac{1}{2}[2(\mu_1-\mu_2)^T C^{-1}X + (\mu_2^T C^{-1}\mu_2 - \mu_1^T C^{-1}\mu_1)] \\
&= (\mu_1-\mu_2)^T C^{-1}X + \frac{1}{2}(\mu_2^T C^{-1}\mu_2 - \mu_1^T C^{-1}\mu_1)
\end{aligned} \tag{7}
$$

- Substituting for $p_1, p_2, c_{11}, c_{22}, c_{21}$,and c_{12} for ξ, and taking natural logarithm, we get

$$\log \xi = 0 \tag{8}$$

- Eqn(7) and (8) state that the Bayes classifier for the problem at hand is a linear classifier, as described by the relation

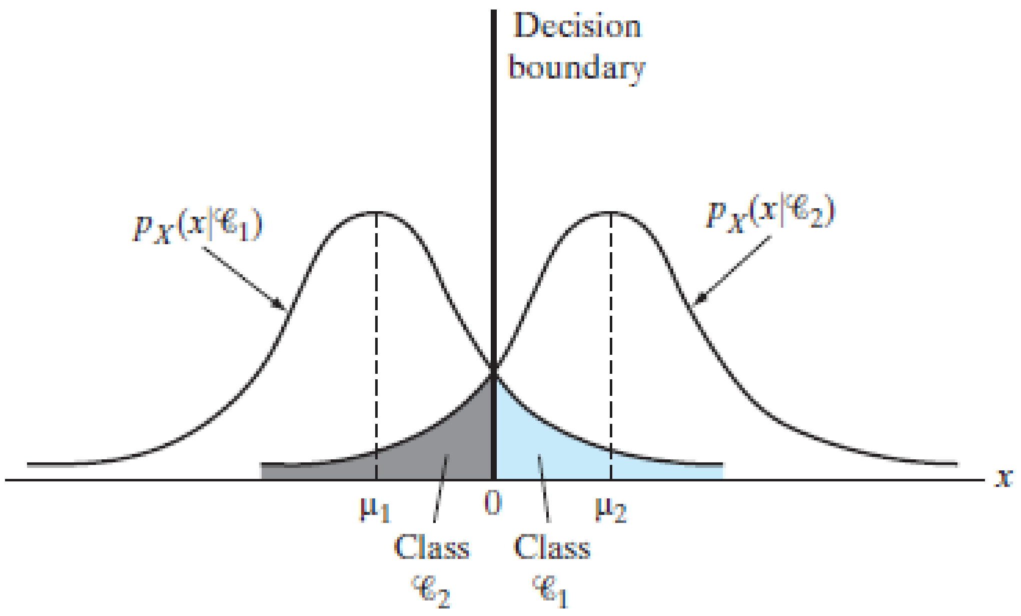
$$y = w^T x + b \tag{9}$$

where $y = \log \Lambda(x)$, $w = C^{-1}(\mu_1 - \mu_2)$, and

$$b = \frac{1}{2}(\mu_2^T C^{-1}\mu_2 - \mu_1^T C^{-1}\mu_1)$$

- *If the output y of the linear combiner is positive, assign the observation vector x to class $\mathcal{C}_1$. Otherwise, assign it to class $\mathcal{C}_2$.*

1. Perceptron operates on the premise that the pattern to be classified are linearly separable. The Gaussian distribution of the two patterns assumed in the derivation of the Bayes classifier certainly do overlap each other and are not separable



1. When the inputs are non-separable and their distribution overlap, the perceptron convergence algorithm develops a problem because decision boundaries between different classes may oscillate continuously
2. Bayes classifier minimizes the probability of classification error, and it is independent of overlap between the distributions
3. Perceptron convergence algorithm is nonparametric in the sense that it makes no assumption about the form of underlying distribution. In contrast, Bayes classifier is parametric
4. Perceptron convergence algorithm is both adaptive and simple to implement; its storage requirement is confined to the set of synaptic weights and bias.
5. Design of the Bayes classifier is fixed; it can be made adaptive, but at the expense of increased storage requirements and more complex computations.

Batch Perceptron Algorithm

- Cost function that permits the application of gradient search, we define the perceptron cost function as

$$J(\mathbf{w}) = \sum_{\mathbf{x}(n)\in\mathcal{L}} (-\mathbf{w}^T\mathbf{x}(n)d(n))$$

- Here $\mathcal{L}$ is the set of samples $\mathbf{x}$ misclassified by a perceptron using $\mathbf{w}$
- If all samples are classified correctly, then the set $\mathcal{L}$ is empty, and the cost $J(\mathbf{w})$ is zero
- Differentiating $J(\mathbf{w})$ wrt $\mathbf{w}$ yields the gradient vector

$$\nabla J(\mathbf{w}) = \sum_{\mathbf{x}(n)\in\mathcal{L}} (-\mathbf{x}(n)d(n))$$

- Where the gradient operator is defined as

$$\nabla = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \dots, \frac{\partial}{\partial w_m}\right]^T$$

- In the method of steepest descent, the adjustment to the weight vector $\mathbf{w}$ at each time-step is applied in a direction opposite to the gradient vector $\nabla\mathbf{J}(\mathbf{w})$
- Accordingly, the algorithm takes the form

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) - \eta\nabla\mathbf{J}(\mathbf{w})\\ &= \mathbf{w}(n) + \eta(n)\sum_{\mathbf{x}(n)\in\mathcal{L}} \mathbf{x}(n)d(n)\end{aligned}$$

- The algorithm is called batch because at each time-step of the algorithm, a batch of misclassified samples is used to compute the adjustment