SINGLE-LAYER PERCEPTRONS

The perceptron is the simplest form of a neural network used for the classification of patterns said to be linearly separable. A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect business intelligence in the input data.

Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule.

A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.

There are two types of Perceptrons: Single layer and Multilayer.

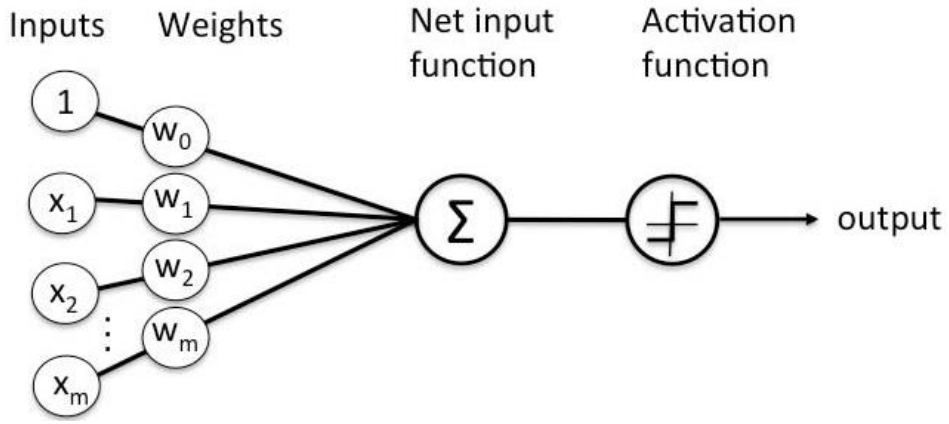
Single layer Perceptrons can learn only linearly separable patterns.

Multilayer Perceptrons or feedforward neural networks with two or more layers have the greater processing power.

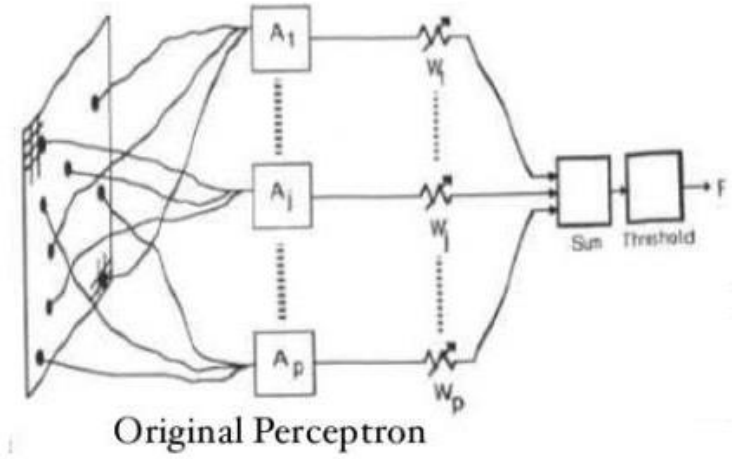
The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.

This enables you to distinguish between the two linearly separable classes +1 and -1.

The neuronproduces an output equal to + 1 if the hard limiter input is positive, and - 1 if it isnegative.



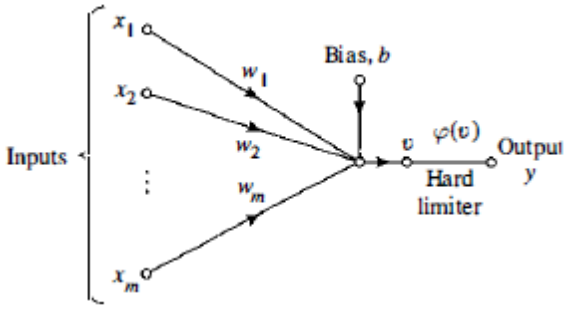
The original Perceptron is as follows.



STRUCTURE AND LEARNING OF PERCEPTRON

In the following signal-flow graph model, the synaptic weights of the perceptron are denoted by w1,w2,…..,wm. Correspondingly, the inputs applied to the perceptron are denoted by x1,x2,…..xm. The externally applied bias is denoted by b. From themodel we find that the hard limiter input or induced local field of the neuron is

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



The goal of the perceptron is to correctly classify the set of externally applied stimuli x1,x2,…..xminto one of two classes, $\mathscr{C}_1$ or $\mathscr{C}_2$ The decision rule for the classification isto assign the point represented by the inputs x1,x2,…..xm to class $\mathscr{C}_1$ if the perceptronoutput y is + 1 and to class $\mathscr{C}_2$ if it is -1.

To develop insight into the behavior of a pattern classifier, it is customary to plota map of the decision regions in the m-dimensional signal space spanned by the minput variables x1,x2,…..xm. In the simplest form of the perceptron there are two decisionregions separated by a hyperplane defined by

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

This is illustrated in following Fig.for the case of two input variables Xl and X2, for which thedecision boundary takes the form of a straight line. A point (Xl' x2) that lies above theboundary line is assigned to class $\mathscr{C}_1$ and a point (Xl' x2) that lies below the boundaryline is assigned to class $\mathscr{C}_2$.

The synaptic weights w1,w2,…..,wmof the perceptron can be adapted on aniteration-by-iteration basis. For the adaptation we may use an error-correction ruleknown as the perceptron convergence algorithm

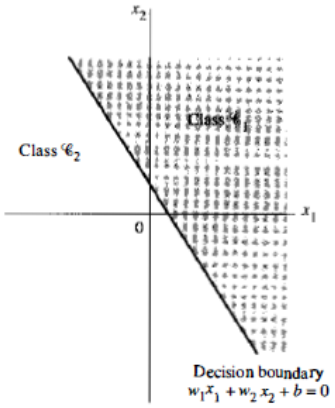
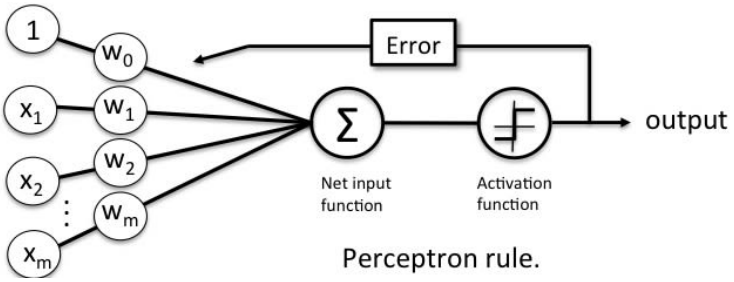


Illustration of the hyperplane (in this example, a straight line) as decision boundary for a two-dimensional, two-class pattern-classification problem.

Perceptron Learning Rule

Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients. The input features are then multiplied with these weights to determine if a neuron fires or not.

The Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output. In the context of supervised learning and classification, this can then be used to predict the class of a sample..



Perceptron rule.

Perceptron Function

Perceptron is a function that maps its input "x," which is multiplied with the learned weight coefficient; an output value "f(x)"is generated.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

In the equation given:

"w" = vector of real-valued weights

"b" = bias (an element that adjusts the boundary away from origin without any dependence on the input value)

"x" = vector of input x values

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

"m" = number of inputs to the Perceptron

The output can be represented as "1" or "0." It can also be represented as "1" or "-1" depending on which activation function is used.

Inputs of a Perceptron

A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result. The above diagram shows a Perceptron with a Boolean output.

A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values: Yes and No or True and False. The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

W0+w1x1+w2x2+……wnxn

Output of Perceptron

Perceptron with a Boolean output:

Inputs: x1…xn

Output: o(x1….xn)

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

Weights: wi=> contribution of input xi to the Perceptron output;w0=> bias or threshold

If ∑w.x> 0, output is +1, else -1. The neuron gets triggered only when weighted input reaches a certain threshold value.

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x}) \qquad sgn(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

An output of +1 specifies that the neuron is triggered. An output of -1 specifies that the neuron did not get triggered.

"sgn" stands for sign function with output +1 or -1.

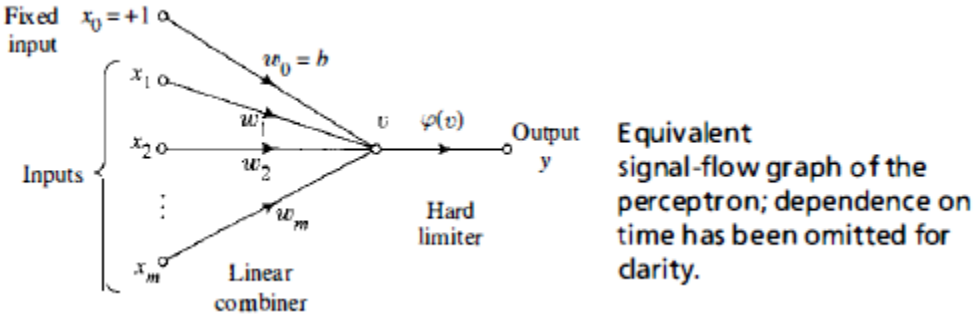
Error in Perceptron

In the Perceptron Learning Rule, the predicted output is compared with the known output. If it does not match, the error is propagated backward to allow weight adjustment to happen.

PERCEPTRON CONVERGENCE

The synaptic weights w1,w2,…..,wm of the perceptron can be adapted on aniteration-by-iteration basis. For the adaptation we may use an error-correction ruleknown as the perceptron convergence algorithm

To derive the error-correction learning algorithm for the perceptron, we find it moreconvenient to work with the modified signal-flow graph model in following Figure.



In this model, the bias b(n) is treated as a synapticweight driven by a fixed input equal to + 1 .

We may thus define the (m + 1)-by-1 input vector

$$\mathbf{x}(n) = [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$$

where n denotes the iteration step in applying the algorithm.

Correspondingly wedefine the (m + 1)-by-1 weight vector as

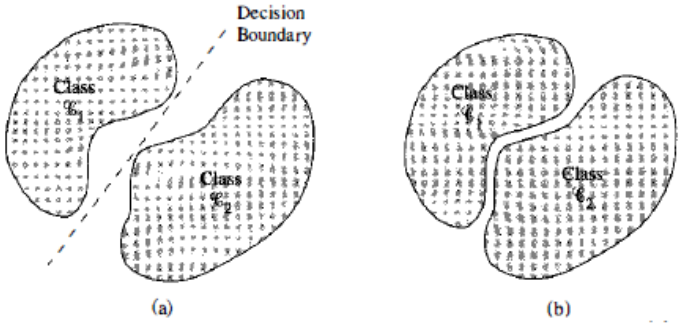
$$\mathbf{w}(n) = [b(n), w_1(n), w_2(n), \ldots, w_m(n)]^T$$

Accordingly, the linear combiner output is written in the compact form

$$v(n) = \sum_{i=0}^{m} w_i(n)x_i(n) \quad = \mathbf{w}^T(n)\mathbf{x}(n)$$

wherew(n) represents the bias b(n). For fixed n, the equation $W^T = 0$, plotted in an m-dimensional space with coordinates x1, x2,…$x_m$defines a hyper plane as the decision surface between two different classes of inputs.

For the perceptron to function properly, the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$must be linearlyseparable. This means that the patterns to be classified must be sufficientlyseparated from each other to ensure that the decision surface consists of ahyperplane. This requirement is illustrated in following Figure.for the case of a two-dimensionalperceptron.



(a) A pair of linearly separable patterns. (b) A pair of non- linearly separable patterns.

In Fig. a, the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$are sufficiently separated from eachother for us to draw a hyperplane (in this case a straight line) as the decision boundary.If the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$ are allowed to move too close to each other, as inFig. b, they become nonlinearly separable, a situation that is beyond the computingcapability of the perceptron.

Suppose that the input variables of the perceptron originate from two linearly separable classes.

Let *X1* be the subset of training vectors x1(l), xl(2), . . . thatbelong to class $\mathscr{C}_1$

Let *X2*, be the subset of training vectors x2(l), x2(2), . . . thatbelong to class $\mathscr{C}_2$.

The union of $\mathscr{C}_1$ and $\mathscr{C}_2$is the complete training set *X*.Given the setsof vectors *X1*, and *X2*to train the classifier, the training process involves the adjustmentof the weight vector w in such a way that the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$are linearly separable.i.e, there exists a weight vector w such that we may state

$$\mathbf{w}^T\mathbf{x} > 0 \text{ for every input vector } \mathbf{x} \text{ belonging to class } \mathscr{C}_1$$

$$\mathbf{w}^T\mathbf{x} \leq 0 \text{ for every input vector } \mathbf{x} \text{ belonging to class } \mathscr{C}_2$$

→The algorithm for adapting the weight vector of the elementary perceptron maynow be formulated as follows:

1. If the n^{th} member of the training set, x(n), is correctly classified by the weightvector w(n) computed at the n^{th} iteration of the algorithm, no correction is made tothe weight vector of the perceptron in accordance with the rule:

$$\mathbf{w}(n+1) = \mathbf{w}(n) \quad \text{if } \mathbf{w}^T\mathbf{x}(n) > 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) \quad \text{if } \mathbf{w}^T\mathbf{x}(n) \leq 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2$$

2. Otherwise, the weight vector of the perceptron is updated in accordance withthe rule

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n) \quad \text{if } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n) \quad \text{if } \mathbf{w}^T(n)\mathbf{x}(n) \leq 0 \text{ and } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1$$

Where the learning-rate parameter $\eta(n)$: controls the adjustment applied to the weightvector at iteration n.

Summary of the Perceptron Convergence Algorithm is as follows.

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

$= [b(n), w_1(n), w_2(n), \ldots, w_m(n)]^T$

$b(n)$ = 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:

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

where $\text{sgn}(\cdot)$ is the signum function.

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

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

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.

PATTERN CLASSIFIER – INTRODUCTION

Classification involves prediction which class an item belongs to. Some classifiers are primarily resulting in yes/no decision. Others are multi classable to classify and categorize an item into one of several categories.

In this context a neural network is one of several machine learning algorithms that can help to solve classification problems. Its unique strength is its ability to dynamically create complex prediction function and emulate human thinking in a way that no algorithm can. There are many classification problems for which neural network yield the best result.

Some of them are as follows.

	Classifier type
1. Logistic regression	Binary classifier
2. Decision tress algorithm	Multi classifier
3. Random forest algorithm	Multi classifier
4. Naïve Bayes classifier	Multi classifier
5. K-Nearest neighbor	Multi classifier

There are several properties that pattern classifier should possess. If one wants to create his own classifier then that classifier should possess following properties in order to sustain along with other classifiers. Artificial Neural Networks are one of the best pattern classifiers of all time.

Following are the some of the properties that pattern classifier should follow.

1. On-Line Adaptation

A pattern classifier should be able to learn new classes simultaneously along with refining of old classes without destroying the old class patterns. This property is referred to as an online adaptation or online learning. Each time when new class information is added to the neural network it should be added along with old class information otherwise older class information get lost during the addition of newer class information.

2. Nonlinear Separability

A pattern classifier should be able to create a decision region that able to separate non linearly separable classes of any shape and in any dimension. This property is referred to as non-linear separability.

3. Overlapping Classes

A good pattern classifier should have the ability to form a decision boundary that minimizes the amount of misclassification of all the overlapping classes. Bayes classifier is the most prevalent method of minimizing overlapping classes.

4. Training Time

The success of the algorithm is determined on the basis of the time required to learn objective function for creating decision boundaries for classes. Most of the algorithms take hundreds of millions of passes to maximize or minimize the objective function. Example of such algorithms is Backpropagation, Boltzmann machine, cascade-correlation etc.

5. Soft and Hard Decision

Pattern classifier should provide both soft and hard decision. The hard or Crisp decision is eighter 0 or 1 while soft decision provides value that provides the degree to which pattern fits in the class or not. This soft and hard decision provides by pattern classifier should be inter convertible also.

6. Verification and Validation

It is important that pattern classifier have a mechanism for verification and validation of its performance. Counterplots, scatter plots and closed-form solutions are the mechanism used to perform this function.

7. Tuning Parameters

Tuning parameters are those parameters that are used to control the learning of pattern classifiers such as learning rate in Backpropagation. A pattern classifier should have lesser tuning parameters during the learning process or if such tuning parameters are there then the effect of these parameters on learning must be lesser. For ideal classifier, there are no tuning parameters.

8. NonParametric classification

Parametric classification requires a priori knowledge about the underlying probability density function of each class. With this information, it is possible to construct a reliable classifier. If this information is not present then classification is termed as nonparametric classification.

Bayes classifiers

The perceptron bears a certain relationship to a classical pattern classifier known as the Bayes classifier. When the environment is Gaussian, the Bayes classifier reduces to a linear classifier.

In the Bayes classifier or Bayes hypothesis testing procedure, we minimize the average risk, denoted by $\mathcal{R}$. For a two-class problem, represented by classes $\mathscr{C}_1$ and $\mathscr{C}_2$the average risk is defined by Van Trees (1968):

$$\mathcal{R} = c_{11}p_1 \int_{\mathcal{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathscr{C}_1)d\mathbf{x} + c_{22}p_2 \int_{\mathcal{X}_2} f_{\mathbf{X}}(\mathbf{x}|\mathscr{C}_2)d\mathbf{x}$$
$$+ c_{21}p_1 \int_{\mathcal{X}_2} f_{\mathbf{X}}(\mathbf{x}|\mathscr{C}_1)d\mathbf{x} + c_{12}p_2 \int_{\mathcal{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathscr{C}_2)d\mathbf{x}$$

where the various terms are defined as follows:

p_i = *a priori probability* that the observation vector $\mathbf{x}$ (representing a realization of the random vector $\mathbf{X}$) is drawn from subspace $\mathcal{X}_i$, with $i = 1, 2$, and $p_1 + p_2 = 1$.

c_{ij} = cost of deciding in favor of class $\mathscr{C}_i$ represented by subspace $\mathcal{X}_i$ when class $\mathscr{C}_j$ is true (i.e., observation vector $\mathbf{x}$ is drawn from subspace $\mathcal{X}_j$), with $(i, j) = 1, 2$.

$f_{\mathbf{X}}(\mathbf{x}|\mathscr{C}_i)$ = conditional probability density function of the random vector $\mathbf{X}$, given that the observation vector $\mathbf{x}$ is drawn from subspace $\mathcal{X}_i$, with $i = 1, 2$.

The first two terms on the right-hand side of above Eq. represent correct decisions (i.e.,correct classifications), whereas the last two terms represent incorrect decisions (i.e.,misclassifications). Each decision is weighted by the product of two factors: the cost involved in making the decision, and the relative frequency (i.e., a priori probability) with which it occurs.

The intention is to determine a strategy for the minimum average risk. Because we require that a decision be made, each observation vector x must be assigned in the overall observation space $\mathcal{X}$ to either $\mathcal{X}_1$ or $\mathcal{X}_2$,. Thus, $\mathcal{X} = \mathcal{X}_1 + \mathcal{X}_2$

Accordingly we may rewrite the above equation as follows

$$\mathcal{R} = c_{11}p_1 \int_{\mathcal{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)d\mathbf{x} + c_{22}p_2 \int_{\mathcal{X}-\mathcal{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2)d\mathbf{x}$$
$$+ c_{21}p_1 \int_{\mathcal{X}-\mathcal{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)\mathbf{x} + c_{12}p_2 \int_{\mathcal{X}_1} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2)d\mathbf{x}$$

where $c_{11} < c_{21}$ and $c_{22} < c_{12}$. We now observe the fact that

$$\int_{\mathcal{X}} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)d\mathbf{x} = \int_{\mathcal{X}} f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2)d\mathbf{x} = 1$$

Hence the equation is reduced to

$$\mathcal{R} = c_{21}p_1 + c_{22}p_2 + \int_{\mathcal{X}_1} [p_2(c_{12} - c_{22}) f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2) - p_1(c_{21} - c_{11}) f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)]d\mathbf{x}$$

The first two terms on the right-hand side of above represent a fixed cost. Since the requirement is to minimize the average risk $\mathcal{R}$, we may therefore deduce the following strategy from above Equation optimum classification:

1. All values of the observation vector $\mathbf{x}$ for which the integrand (i.e., the expression inside the square brackets) is negative should be assigned to subspace $\mathcal{X}_1$ (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 $\mathbf{x}$ for which the integrand is positive should be excluded from subspace $\mathcal{X}_1$ (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 $\mathbf{x}$ for which the integrand is zero have no effect on the average risk $\mathcal{R}$ and may be assigned arbitrarily. We shall assume that these points are assigned to subspace $\mathcal{X}_2$ (i.e., class $\mathcal{C}_2$).

On this basis, we may formulate the Bayes classifier as follows:

If the condition

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

holds, assign the observation vector $\mathbf{x}$ *to subspace* $\mathcal{X}_1$ *(i.e., class* $\mathcal{C}_1$*). Otherwise assign* $\mathbf{x}$ *to* $\mathcal{X}_2$ *(i.e., class* $\mathcal{C}_2$*).*

To simplify matters, define

$$\Lambda(\mathbf{x}) = \frac{f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)}{f_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2)} \quad \text{and} \quad \xi = \frac{p_2(c_{12} - c_{22})}{p_1(c_{21} - c_{11})}$$

The quantity $\Lambda(\mathbf{x})$, the ratio of two conditional probability density functions, is called the *likelihood ratio*. The quantity ξ is called the *threshold* of the test. Note that both $\Lambda(\mathbf{x})$ and ξ are always positive. In terms of these two quantities, we may now reformulate the Bayes classifier by stating:

If, for an observation vector $\mathbf{x}$*, the likelihood ratio* $\Lambda(\mathbf{x})$ *is greater than the threshold* ξ*, assign* $\mathbf{x}$ *to class* $\mathcal{C}_1$*. Otherwise, assign it to class* $\mathcal{C}_2$*.*

The following block diagram is a representation of Bayes classifier.

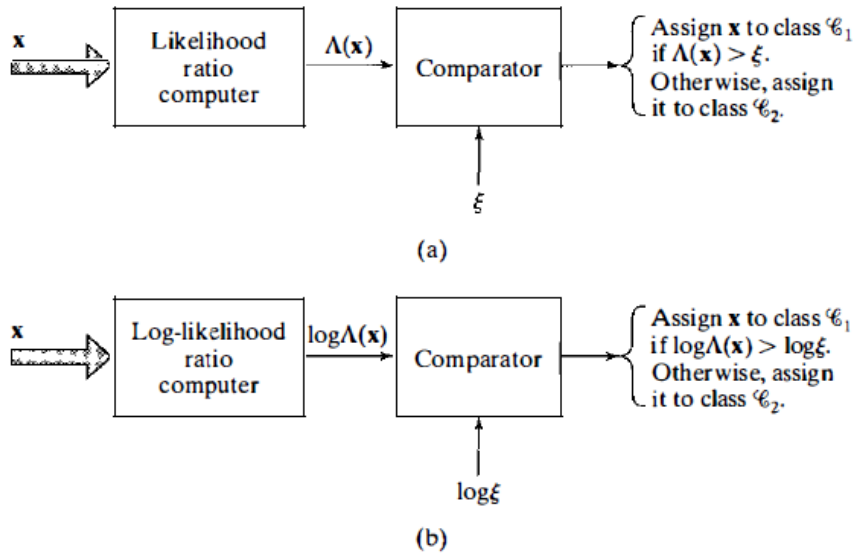


FIGURE Two equivalent implementations of the Bayes classifier: (a) Likelihood ratio test, (b) Log-likelihood ratio test.

The important points in this block diagram are

1. **The data processing involved in designing the Bayes classifier is confined entirely to the computation of the likelihood ratio $\Lambda(\mathbf{x})$.**
2. **This computation is completely invariant to the values assigned to the *a priori* probabilities and costs involved in the decision-making process. These quantities merely affect the value of the threshold ξ.**

Perceptron as a pattern classifier

The perceptron is simply separating the input into 2 categories.

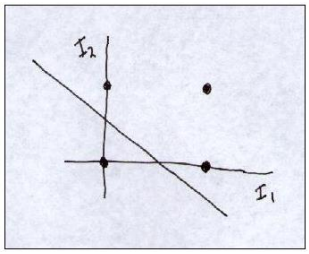
1. Those that cause fire
2. Those that don't.

It does this by looking at (in the 2-dimensional case):

$w_1I_1 + w_2I_2 < t$

If the LHS is < t, it doesn't fire, otherwise it fires. That is, it is drawing the line:

$w_1I_1 + w_2I_2 = t$



and looking at where the input point lies. Points on one side of the line fall into 1 category, points on the other side fall into the other category. And because the weights and thresholds can be anything, this is just *any line* across the 2 dimensional input space.

So what the perceptron is doing is simply drawing a line across the 2-d input space. Inputs to one side of the line are classified into one category, inputs on the other side are classified into another. e.g. the OR perceptron, w_1=1, w_2=1, t=0.5, draws the line:

$I_1 + I_2 = 0.5$
across the input space, thus separating the points (0,1),(1,0),(1,1) from the point (0,0):
As you might imagine, not every set of points can be divided by a line like this. Those that can be, are called *linearly separable*.
In 2 input dimensions, we draw a 1 dimensional line. In n dimensions, we are drawing the (n-1) dimensional *hyperplane*:
$w_1I_1 + .. + w_nI_n = t$

Limitations of a perceptron

Perceptron networks have several limitations.

1. The output values of a perceptron can take on only one of two values (0 or 1) due to the hard-limit transfer function.
2. Perceptrons can only classify linearly separable sets of vectors. If a straight line or a plane can be drawn to separate the input vectors into their correct categories, the input vectors are linearly separable. Note that it has been proven that if the vectors are linearly separable, perceptrons trained adaptively will always find a solution in finite time.
3. The perceptron can only model linearly separable functions such as following functions:
   a) AND
   b) OR
   c) COMPLEMENT
   d) It cannot model XOR(Non linearly separable)

4. XOR is not linearly separable. Its +ve and –ve instance cannot be separated by a line or hyperplane.
5. When two classes are not linearly separable, it may be desirable to obtain a linear separator that minimizes the mean squared error.
6. Perceptron training always converge with training data x+ and x- are linearly separable sets.