Linear Classification

1. Recalling Bayes Classifier
2. What is Linear Classification
3. Why is Linear Classification required?
4. Linear Classifier Equation
5. Perceptron Algorithm

Our major concern in Bayes Classifier was to design classifiers based on probability density or probability functions. Today, we will focus on the design of linear classifiers.

The major advantage of linear classifiers is their simplicity and computational attractiveness.

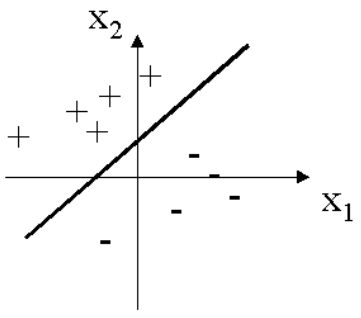
The lecture starts with the assumption that **all feature vectors from the available classes can be classified correctly using a linear classifier**

- We will learn about linear classification.
- We will develop techniques for the computation of the corresponding linear functions.
- In the sequel, **we will focus on to train the perceptron to classify inputs correctly by adjusting the connecting weights**.

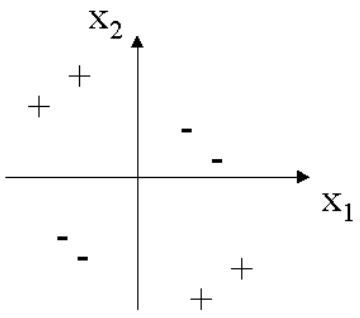
Linear Classification:

The goal of statistical classification is to use an object's characteristics to identify which class (or group) it belongs to. A **linear classifier** achieves this by making a classification decision based on the value of a linear combination of the characteristics. An object's characteristics are also known as **feature values** and are typically presented to the machine in a vector called a **feature vector**.

We can use linear classifier where things are linearly separable.



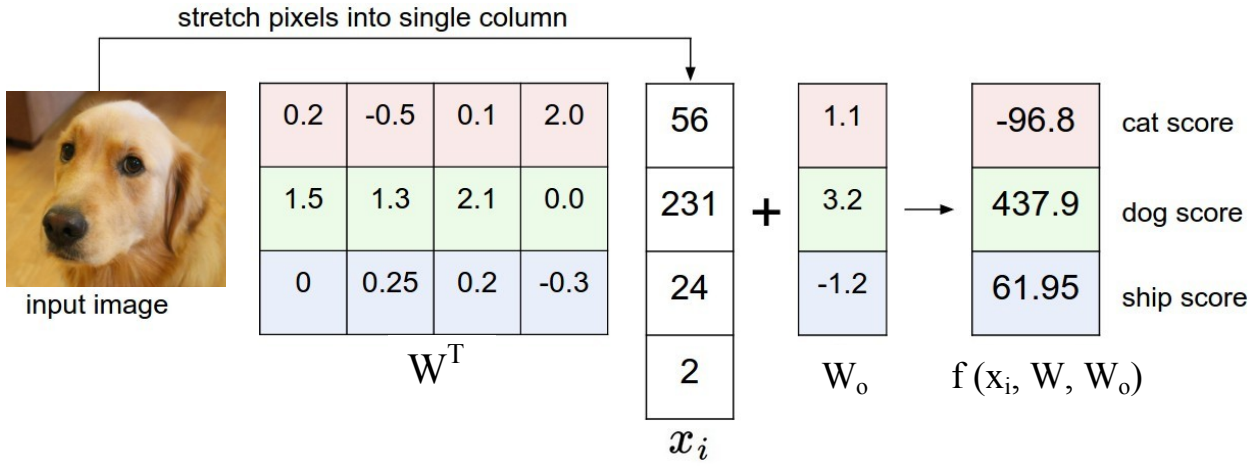
Linearly Separable



Not Linearly Separable

The simplest possible function, a linear mapping:

$$\mathbf{f(x, W, W_o) = W^T x + W_o}$$



Two class discriminator:

Let assume, we have two classes C_1 and C_2.

$$\mathbf{f(x, W, W_o) = y(x) = W^T x + W_o = \sum_{i=1}^{L} W_i\ x_i + W_o}$$

$y(x) \geq 0 \rightarrow$ x assigned to C1
$y(x) < 0 \rightarrow$ x assigned to C2

Consider the things as point, so we have two features: x value and y value.

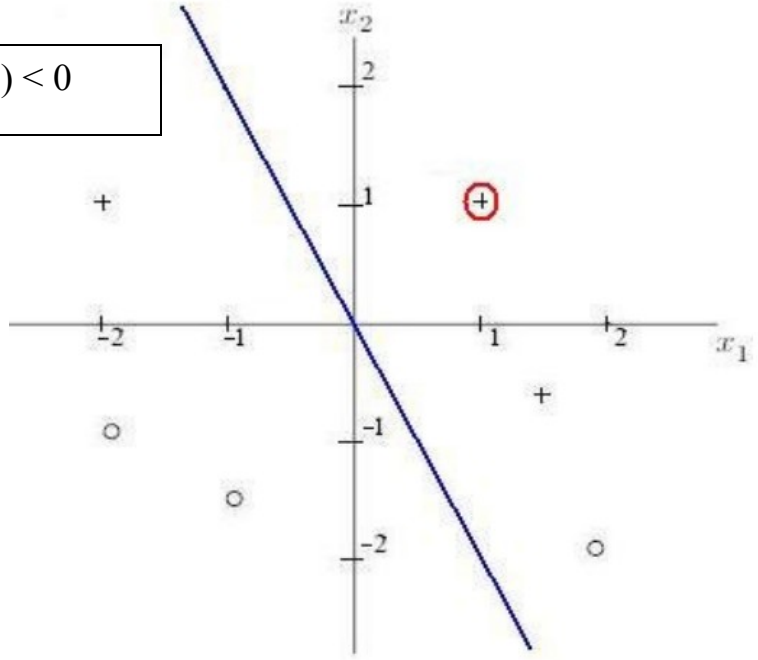
X = [1, X1, X2]
W = [W_o, W_1, W_2] = [0, 1, 0.5]

The linear equation will be:

$W_1x_1 + W_2 x_2 + W_o = 0$
Or, $x_1 + 0.5\, x_2 + 0 = 0$
Or, $x_2 = -2x_1$

y(x) < 0

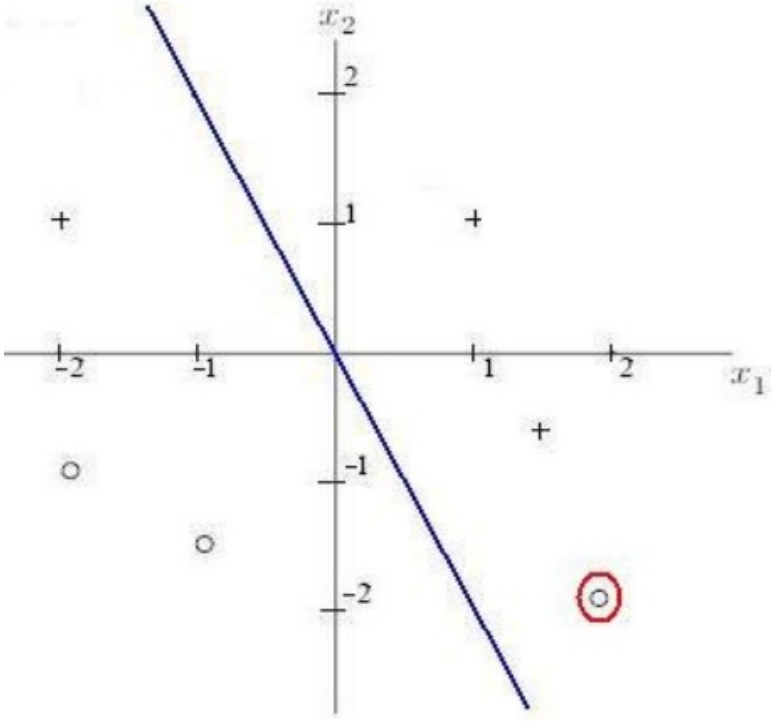
y(x) ≥ 0



$x_1 = 1$
$x_2 = 1$

From $W^T x + W_o$
1.5 >= 0, So, this will be classified in class C_1

Do the same thing for the following point:
$x_1 = 2$
$x_2 = -2$



Perceptron Learning Algorithm

In this section, we assume that the two classes C1, C2 are linearly separable. In other words, we assume that there exists a hyperplane, defined by $\mathbf{W^T x = 0}$ such that

$\mathbf{W^T x > 0} \in \mathbf{C_1}$
$\mathbf{W^T x < 0} \in \mathbf{C_2}$

The formulation above also covers the case of a hyperplane not crossing the origin, that is, $\mathbf{W^T x + W_o = 0}$, since this can be brought into the previous formulation by defining the extended (l + 1)-dimensional vectors $x' \equiv [x^T , 1]^T$, $W' \equiv [W^T , W_o]^T$

So, $\mathbf{W^T x + W_o = W'^T x'}$

Thus we need to adopt
(a) an appropriate cost function and
(b) an algorithmic scheme to optimize it.

To this end,we choose the perceptron cost defined as,

$$J(w) = \sum_{x \in Y} (\delta_x w^T x)$$

where Y is the subset of the training vectors, which are misclassified by the hyperplane defined by the weight vector w. The variable δ_x is chosen so that $\delta_x = -1$ if $x \in$ $\mathbf{C_1}$ $\delta_x = +1$ if $x \in$ $\mathbf{C_2}$
positive, and it becomes zero when Y becomes the empty set, that is, if there are not misclassified vectors x. Indeed, if $x \in \omega_1$ and it is misclassified, then $w^T x < 0$ and $\delta_x < 0$, and the product is positive.

The perceptron cost function is continuous and piecewise linear. Indeed, if we change the weight vector smoothly, the cost J(w) changes linearly until the point at which there is a change in the number of misclassified vectors.

We will adopt an iterative scheme in the spirit of the gradient descent method,

$$w(t+1) = w(t) - \rho_t \frac{\partial J(w)}{\partial w}\Big|_{w=w(t)}$$

$$\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_{\boldsymbol{x} \in Y} \delta_{\boldsymbol{x}} \boldsymbol{x}$$

Substituting,

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \rho_t \sum_{\boldsymbol{x} \in Y} \delta_{\boldsymbol{x}} \boldsymbol{x}$$

The algorithm is known as the perceptron algorithm and is quite simple in its structure. The algorithm is initialized from an arbitrary weight vector w(0), and the correction vector is formed using the misclassified features. The weight vector is then corrected according to the preceding rule. This is repeated until the algorithm converges to a solution, that is, all features are correctly classified.
A pseudocode for the perceptron algorithm is given below.

The Perceptron Algorithm

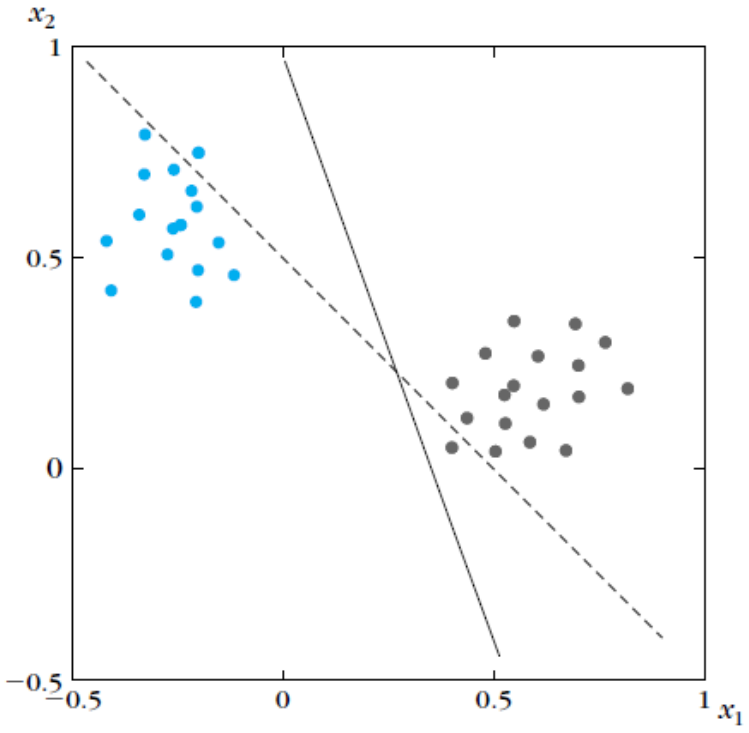
- Choose $\boldsymbol{w}(0)$ randomly
- Choose ρ_0
- $t = 0$
- Repeat
  - $Y = \emptyset$
  - For $i = 1$ to N
    - If $\delta_{\boldsymbol{x}_i} \boldsymbol{w}(t)^T \boldsymbol{x}_i \geq 0$ then $Y = Y \cup \{\boldsymbol{x}_i\}$
  - End {For}
  - $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \rho_t \sum_{\boldsymbol{x} \in Y} \delta_{\boldsymbol{x}} \boldsymbol{x}$
  - Adjust ρ_t
  - $t = t + 1$
- Until $Y = \emptyset$

Example 3.1

Figure 3.3 shows the dashed line

$$x_1 + x_2 - 0.5 = 0$$

corresponding to the weight vector $[1, 1, -0.5]^T$, which has been computed from the latest iteration step of the perceptron algorithm (3.9), with $\rho_t = \rho = 0.7$. The line classifies correctly



all the vectors except $[0.4, 0.05]^T$ and $[-0.20, 0.75]^T$. According to the algorithm, the next weight vector will be

$$\boldsymbol{w}(t+1) = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7(-1) \begin{bmatrix} 0.4 \\ 0.05 \\ 1 \end{bmatrix} - 0.7(+1) \begin{bmatrix} -0.2 \\ 0.75 \\ 1 \end{bmatrix}$$

or

$$\boldsymbol{w}(t+1) = \begin{bmatrix} 1.42 \\ 0.51 \\ -0.5 \end{bmatrix}$$

The resulting new (solid) line $1.42x_1 + 0.51x_2 - 0.5 = 0$ classifies all vectors correctly, and the algorithm is terminated.