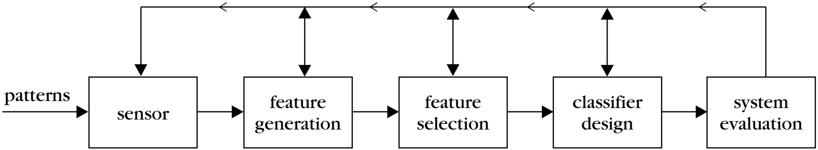
Supervised Classification Using Support Vector Machines

(Scenario: Activity Monitoring)

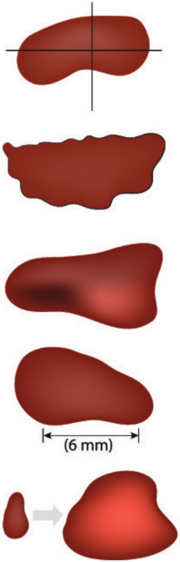
Dr. Muhammad Adeel Nisar
(Courtesy: Prof. Dr. Marcin Grzegorzek)

Department of Information Technology
University of the Punjab, Lahore

Basic Stages of Pattern Analysis

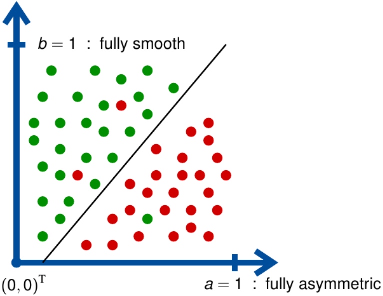


Linear Classification – Example



A **Asymmetry** $a \in [0, 1]$

B **Border** $b \in [0, 1]$



Confusing Notation

Weight Vector without Threshold	Weight Vector with Threshold
$\boldsymbol{w} = [w_1, \ldots, w_l]^{\mathrm{T}}$	$\boldsymbol{w} = [w_1, \ldots, w_l, w_0]^{\mathrm{T}}$
$\boldsymbol{x} = [x_1, \ldots, x_l]^{\mathrm{T}}$	$\boldsymbol{x} = [x_1, \ldots, x_l, 1]^{\mathrm{T}}$
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 = 0$	$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = 0$

Decision Hyperplanes

- Let us focus on a two-class problem and consider linear discriminant functions. The decision hypersurface in the l-dimensional feature space is then given by

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = 0 \quad .$$

- The dimensionality problem ($\boldsymbol{w} \in \mathbb{R}^{l+1}$, but feature vectors have l elements) is overcome by increasing the dimensionality of each feature vector, so that

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_l, 1]^{\mathrm{T}} \quad .$$

This does not change anything in the linear classification process.

Decision Hyperplanes

- If $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are two points on the decision hyperplane, then the following is valid

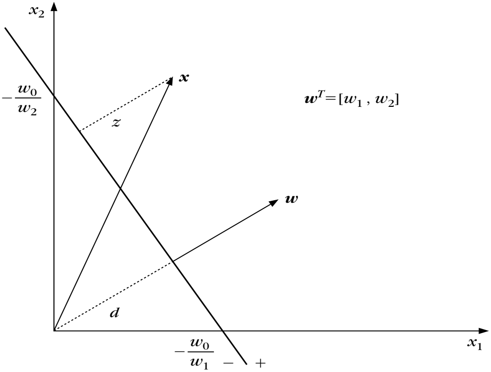
$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_1 = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_2 = 0$$

$$\Updownarrow$$

$$\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$$

- Since the difference vector $\boldsymbol{x} = \boldsymbol{x}_1 - \boldsymbol{x}_2$ obviously lies on the decision hyperplane, it is apparent that the weight vector $\boldsymbol{w}$ is orthogonal to the decision hyperplane.

Decision Hyperplanes



$$d = \frac{|w_0|}{\sqrt{w_1^2 + w_2^2}} \qquad z = \frac{|g(\boldsymbol{x})|}{\sqrt{w_1^2 + w_2^2}}$$

Problem Statement

Problem: How to compute the unknown parameters $w_1, \ldots, w_l, w_0$?

Assumptions: The two classes ω_1 and ω_2 are linearly separable, i. e., there exist a hyperplane $\widehat{\boldsymbol{w}}$ such that

$$\widehat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{x} > 0; \qquad \forall \boldsymbol{x} \in \omega_1$$

$$\widehat{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{x} < 0; \qquad \forall \boldsymbol{x} \in \omega_2$$

Approach: The problem will be solved as an optimisation task. Therefore, we need:

- an appropriate cost function,
- an algorithmic scheme to optimise it.

Perceptron Cost Function – Definition

- As a cost function, the perceptron cost is used:

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

- Y – subset of training vectors misclassified by the hyperplane $\boldsymbol{w}$.

- The variable δ_x is chosen so that:

$$\begin{cases} \boldsymbol{x} \in \omega_1 & \Rightarrow \quad \delta_x = -1 \\ \boldsymbol{x} \in \omega_2 & \Rightarrow \quad \delta_x = +1 \end{cases} \quad .$$

Perceptron Cost Function – Properties

- The perceptron cost is not negative. It becomes zero when $Y = \emptyset$, that is, if there are no misclassified vectors $\boldsymbol{x}$.
- Indeed, if $\boldsymbol{x} \in \omega_1$ and it is misclassified, then $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} < 0$ and $\delta_x < 0$. Thus, the product is positive.
- The perceptron cost function is continuous and piecewise linear.

Minimisation of the Perceptron Cost Function

- The iterative minimisation works according to:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \rho_t \left. \frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} \right|_{\boldsymbol{w}=\boldsymbol{w}(t)} \quad .$$

- $\boldsymbol{w}$ is the weight vector at the iteration step no. t .

- ρ_t is a positive real number chosen manually.

Minimisation of the Perceptron Cost Function

- From the perceptron definition and the points where this is valid, we get

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

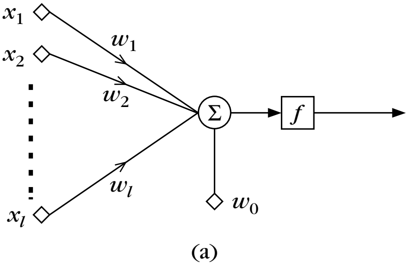
- Thus, the iterative minimisation of the cost function from the previous slide can be written as

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

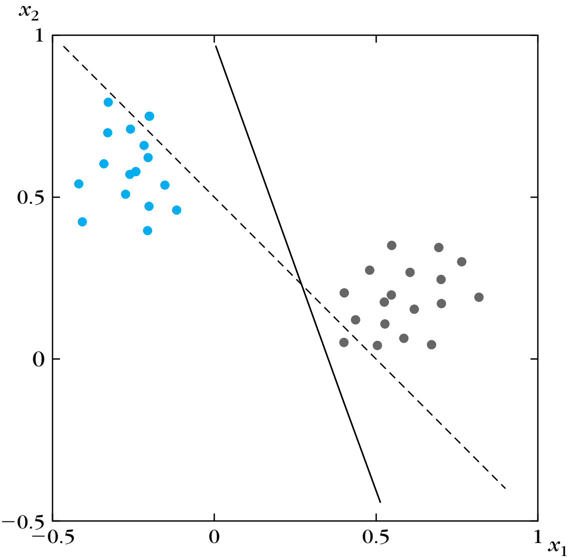
The Perceptron Algorithm – Pseudocode

- Choose $\boldsymbol{w}(0)$ randomly
- Choose ρ_0
- $t = 0$
- Repeat
    - Set $Y = \emptyset$
    - For $j = 1$ to K
        - If $\delta_{x_j} \boldsymbol{w}(j)^{\mathrm{T}} \boldsymbol{x}_j \geq 0$ then $Y = Y \cup \{\boldsymbol{x}_j\}$
    - End For
    - $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \rho_t \sum_{\boldsymbol{x} \in Y} \delta_x \boldsymbol{x}$
    - Adjust ρ_t
    - Iterate $t = t + 1$
- Until $Y = \emptyset$

Basic Perceptron Model



Perceptron Algorithm – Example



Perceptron Algorithm – Example

Known:

- Decision line after the iteration no. t is given by

$$x_1 + x_2 - 0.5 = 0 \quad \Leftrightarrow \quad \boldsymbol{w}(t) = [1, 1, -0.5]^{\mathrm{T}}$$

- $\rho_t = 0.7$.
- Misclassified vectors: $[0.4, 0.05]^{\mathrm{T}}$ and $[-0.2, 0.75]^{\mathrm{T}}$.

Unknown:

- The decision line after the iteration no. $t + 1$:

$$\boldsymbol{w}(t+1) = \begin{bmatrix} w_1(t+1) \\ w_2(t+1) \\ w_0(t+1) \end{bmatrix} = ?$$

Perceptron Algorithm – Example

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

$$\Updownarrow$$

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

Note that the dimensionality of the misclassified vectors has been increased by one!

SVMs for Linearly Separable Classes

- A two-class problem $\Omega = \{\omega_1, \omega_2\}$.
- $\boldsymbol{x}_{i=1,\dots,N}$ are all training feature vectors.
- The goal, once more, is to design a hyperplane[1]

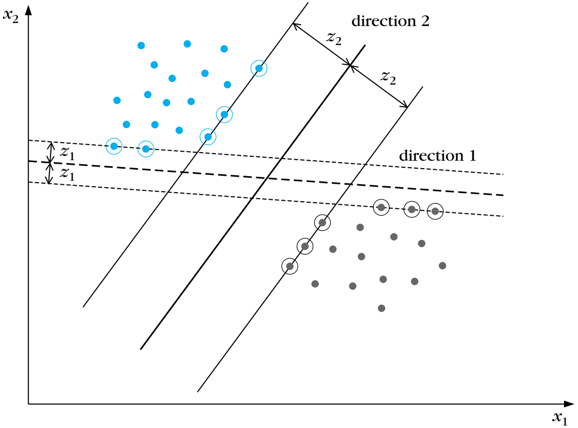
$$g(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 = 0$$

that classifies correctly all the training feature vectors.

[1] Note that $\boldsymbol{w} = [w_1, \dots, w_l]^{\mathrm{T}}$ and w_0 are treated separately here.

SVMs for Linearly Separable Classes

- The goal is to search for the direction that gives the maximum possible margin.



SVMs for Linearly Separable Classes

- The distance of a point from a hyperplane is given by

$$z = \frac{|g(\boldsymbol{x})|}{||\boldsymbol{w}||}$$

- $\boldsymbol{w}$ and w_0 are now scaled so that the value $|g(\boldsymbol{x})|$ at the nearest points in both classes is equal to 1:

$$\begin{cases} \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 \geq 1 & \forall \boldsymbol{x} \in \omega_1 \\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 \leq -1 & \forall \boldsymbol{x} \in \omega_2 \end{cases}$$

- In this case, the margin is equal to

$$\frac{1}{||\boldsymbol{w}||} + \frac{1}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||}$$

Final Statements

- For many applications, **linear classification** delivers satisfactory results.

- For a limited number of labelled training examples, **Support Vector Machines** perform often better than deep learning algorithms.