University of Crete- Computer Science Department

CS-473: Pattern Recognition

Instructor: A. Mouchtaris

TA Class: Non-parametric techniques & Linear discriminant functions

Problem 1 Consider a two-classes single dimensional classification problem. We have collected the following samples for each class: $D_1 = \{-1, -2, 3, 3, 6, 7\}$ and $D_2 = \{-3, -2, 3, 5, 8\}$.

For the classification we are going to use the Parzen windows technique with a window size equal to $h = 2$ and window function as:

$$\phi(u) = \begin{cases} -\frac{3}{4}u^2 + \frac{3}{4}, & -1 \leq u \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

(a) Classify the observation $x = 4$.

(b) Show that using the Parzen windows we end up with an estimate which is a valid probability distribution function.

Solution

(a) For class 1 it holds that:

$$p_1(4) = \frac{1}{6}\sum_{i=1}^{6}\frac{1}{2}\phi\left(\frac{4-x_i}{2}\right) = \frac{1}{12}\left(\phi\left(\frac{1}{2}\right) + \phi\left(\frac{1}{2}\right)\right) = \frac{1}{6}\phi\left(\frac{1}{2}\right)$$

For class 2 it holds that:

$$p_2(4) = \frac{1}{5}\sum_{i=1}^{5}\frac{1}{2}\phi\left(\frac{4-x_i}{2}\right) = \frac{1}{10}\left(\phi\left(\frac{1}{2}\right) + \phi\left(-\frac{1}{2}\right)\right) = \frac{1}{5}\phi\left(\frac{1}{2}\right)$$

where we have used $\phi\left(\frac{1}{2}\right) = \phi\left(-\frac{1}{2}\right)$ at the last step.

Then since $p_2(4) > p_1(4)$, $x = 4$ is classified in class 2.

(b) In order to show that the estimates resulting from the use of Parzen windows described herein result from a real probability distribution function, it suffices to show that the window function $\phi(u)$ is a non-negative one and that its integral equals 1. That is

$$\phi(u) \geq 0$$

and

$$\int_{-\infty}^{\infty} \phi(u)du = 1.$$

From the given window function $-\frac{3}{4}u^2 + \frac{3}{4}$ for $-1 \leq u \leq 1$ it follows directly that $-\frac{3}{4}u^2 + \frac{3}{4} \geq 0$, so the first condition is satisfied.

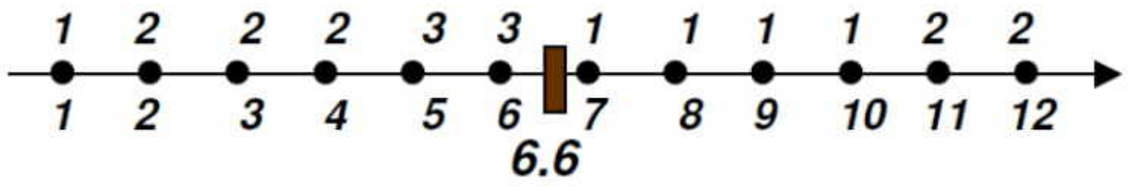
For the integral it holds:

$$\int_{-\infty}^{\infty} \phi(u)du = \int_{-1}^{1} \left(-\frac{3}{4}u^2 + \frac{3}{4}\right) du = -\frac{3}{12}u^3 + \frac{3}{4}u\Big|_{-1}^{1} = -\frac{3}{12} + \frac{3}{4} - \frac{3}{12} + \frac{3}{4} = 1,$$

so the second condition is also satisfied.

Problem 2 Consider the following classes with the corresponding single-dimensional data : $D_1 = \{1, 7, 8, 9, 10\}$, $D_2 = \{2, 3, 4, 11, 12\}$ and $D_3 = \{5, 6\}$. Using the k-NN classification algorithm with $k = 6$, classify the point 6.6.

Solution



First we estimate the distance of the point 6.6 from each of the provided sample values:

Point	Distance	Class
1	$\|6.6 - 1\| = 5.6$	1
2	$\|6.6 - 2\| = 4.6$	2
3	$\|6.6 - 3\| = 3.6$	2
4	$\|6.6 - 4\| = 2.6$	2
5	$\|6.6 - 5\| = 1.6$	3
6	$\|6.6 - 6\| = 0.6$	3
7	$\|6.6 - 7\| = 0.4$	1
8	$\|6.6 - 8\| = 1.4$	1
9	$\|6.6 - 9\| = 2.4$	1
10	$\|6.6 - 10\| = 3.4$	1
11	$\|6.6 - 11\| = 4.4$	2
12	$\|6.6 - 12\| = 5.4$	2

In the preceding table one sees in red the six closest neighbors of point 6.6. As we can observe, three out of six neighbors belong to class 1, two out of six to class 2 and two out of six to class 3. Since the biggest amount of the six closest neighbors belong to class 1, 6.6 is also classified to class 1.

Problem 3 You are going to use the single-sample Perceptron algorithm in order to estimate the linear discrimination function for the problem described in the following table:

	FEATURES				GRADE
name	good attendance?	tall?	sleeps in class?	chews gum?	
Jane	yes (1)	yes (1)	no (-1)	no (-1)	A
Steve	yes (1)	yes (1)	yes (1)	yes (1)	F
Mary	no (-1)	no (-1)	no (-1)	yes (1)	F
Peter	yes (1)	no (-1)	no (-1)	yes (1)	A

Class c_1: students getting grade equal to A.

Class c_2: students getting grade equal to F.

Solution

We define four features (one for each student) as:

$$\begin{aligned}
\boldsymbol{x}_1 &= \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}^T \\
\boldsymbol{x}_2 &= \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T \\
\boldsymbol{x}_3 &= \begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix}^T \\
\boldsymbol{x}_4 &= \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}^T
\end{aligned}$$

We transform the feature sample $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ to the augmented feature samples. We form matrix $\boldsymbol{Y}$ having the feature samples at its rows. We add one extra column with all values equal to 1 and replace samples belonging to class c_2 with their negative values. This way we have:

$$\boldsymbol{Y} = \begin{bmatrix} 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 \end{bmatrix}$$

We are looking for the weight vector $\boldsymbol{a}$ such that $\boldsymbol{a}^T\boldsymbol{y}_i > 0, \quad \forall\, \boldsymbol{y}_i$

We are going to use the single-sample Perceptron algorithm:

- A sample is misclassified if $\boldsymbol{a}^T\boldsymbol{y}_i < 0$.
- Update rule: $\boldsymbol{a}(k+1) = \boldsymbol{a}(k) + \eta(k)\boldsymbol{y}^k$, where $\boldsymbol{y}^k$ denotes the misclassified sample at k iteration.
- We use a fixed learning rate: $\eta(k) = 1, \quad \forall k$.

We initialize $\boldsymbol{a}(1) = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}^T$. We test samples $\boldsymbol{y}_i$ one after the other and update the weight vector as soon as a sample gets misclassified:

- Jane: $0.25 \times 1 + 0.25 \times 1 + 0.25 \times 1 + 0.25 \times (-1) + 0.25 \times (-1) > 0 \longrightarrow$ classification:CORRECT
- Steve: $0.25 \times (-1) + 0.5 \times (-1) + 0.25 \times (-1) + 0.25 \times (-1) + 0.25 \times (-1) < 0 \longrightarrow$ classification:WRONG

  $\Rightarrow$ Weight vector update:

$$\begin{aligned} \boldsymbol{a}_{\text{new}} &= \boldsymbol{a}_{\text{old}} + \boldsymbol{y}_{\text{misclassified}} \\ &= \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}^T + \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \end{bmatrix}^T \\ &= \begin{bmatrix} -0.75 & -0.75 & -0.75 & -0.75 & 0.75 \end{bmatrix}^T \end{aligned} \tag{1}$$

- Mary: $-0.75 \times (-1) - 0.75 \times 1 - 0.75 \times 1 - 0.75 \times 1 - 0.75 \times (-1) < 0 \longrightarrow$ classification:WRONG

  $\Rightarrow$ Weight vector update:

$$\begin{aligned} \boldsymbol{a}_{\text{new}} &= \boldsymbol{a}_{\text{old}} + \boldsymbol{y}_{\text{misclassified}} \\ &= \begin{bmatrix} -0.75 & -0.75 & -0.75 & -0.75 & -0.75 \end{bmatrix}^T + \begin{bmatrix} -1 & 1 & 1 & 1 & -1 \end{bmatrix}^T \\ &= \begin{bmatrix} -1.75 & 0.25 & 0.25 & 0.25 & -1.75 \end{bmatrix}^T \end{aligned} \tag{2}$$

- Peter: $-1.75 \times 1 + 0.25 \times 1 + 0.25 \times (-1) + 0.25 \times (-1) - 1.75 \times 1 < 0 \longrightarrow$ classification:WRONG

$\Rightarrow$ Weight vector update:

$$\begin{aligned} \boldsymbol{a}_{\text{new}} &= \boldsymbol{a}_{\text{old}} + \boldsymbol{y}_{\text{misclassified}} \\ &= \begin{bmatrix} -1.75 & 0.25 & 0.25 & 0.25 & -1.75 \end{bmatrix}^T + \begin{bmatrix} 1 & 1 & -1 & -1 & 1 \end{bmatrix}^T \\ &= \begin{bmatrix} -0.75 & 1.25 & -0.75 & -0.75 & -0.75 \end{bmatrix} \end{aligned} \tag{3}$$

- Jane: $-0.75 \times 1 + 1.25 \times 1 - 0.75 \times 1 - 0.75 \times (-1) - 0.75 \times (-1) > 0 \longrightarrow$ classification:CORRECT
- Steve: $-0.75 \times (-1) + 1.25 \times (-1) - 0.75 \times (-1) - 0.75 \times (-1) - 0.75 \times (-1) > 0 \longrightarrow$classification:CORRECT
- Mary: $-0.75 \times (-1) + 1.25 \times 1 - 0.75 \times 1 - 0.75 \times 1 - 0.75 \times (-1) > 0 \longrightarrow$ classification:CORRECT
- Peter: $-0.75 \times 1 + 1.25 \times 1 - 0.75 \times (-1) - 0.75 \times (-1) - 0.75 \times 1 > 0 \longrightarrow$ classification:CORRECT

Since all training samples are correctly classified the discrimination function is:

$$g(\boldsymbol{y}) = \boldsymbol{a}^T \boldsymbol{y} \text{ where } \boldsymbol{a} = \begin{bmatrix} -0.75 & 1.25 & -0.75 & -0.75 & -0.75 \end{bmatrix}^T$$

Going back to the initial feature space:

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 \text{ where } \boldsymbol{w} = \begin{bmatrix} 1.25 & -0.75 & -0.75 & -0.75 \end{bmatrix}^T \text{ and } w_0 = -0.75$$

Note

The above presented solution is ONE of potential solutions but it is by no means the UNIQUE solution. There may be more than one discrimination functions, classifying all training samples correctly. If, for example, the initialization of the weight vector was $\boldsymbol{a}(1) = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 & 0 \end{bmatrix}^T$ by applying the single-sample Perceptron algorithm we would end up with $\boldsymbol{a} = \begin{bmatrix} -1 & 1.5 & -0.5 & -1 & -1 \end{bmatrix}^T$ (you can verify the solution by applying all the steps of the algorithm!).

All correct solutions are accepted, since they all classify correctly the training sample vectors. However, each solution could experience different performance in classifying new (therefore unknown during training) classification vectors.

Problem 4

(a) Provide pseudocode for minimizing the following cost function using the gradient descent principle

$$J(\boldsymbol{v}) = \frac{\boldsymbol{v}^T \boldsymbol{A}\boldsymbol{v} + \boldsymbol{v}^T \boldsymbol{C}\boldsymbol{v}}{\boldsymbol{v}^T \boldsymbol{B}\boldsymbol{v}}$$

where $\boldsymbol{v}$ is the $n \times 1$ unknown vector and the $n \times n$ matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ are considered to be known.

(b) Describe what the learning rate could be in order to make sure that the algorithm will eventually terminate.

Solution

(a) We have:

$$J(\boldsymbol{v}) = \frac{\boldsymbol{v}^T \boldsymbol{A}\boldsymbol{v} + \boldsymbol{v}^T \boldsymbol{C}\boldsymbol{v}}{\boldsymbol{v}^T \boldsymbol{B}\boldsymbol{v}} = \frac{\boldsymbol{v}^T (\boldsymbol{A} + \boldsymbol{C})\boldsymbol{v}}{\boldsymbol{v}^T \boldsymbol{B}\boldsymbol{v}}$$

$$\nabla J(\boldsymbol{v}) = \frac{\left(2\left(\boldsymbol{A} + \boldsymbol{C}\right)\boldsymbol{v}\right)\left(\boldsymbol{v}^T \boldsymbol{B}\boldsymbol{v}\right) - \left(\boldsymbol{v}^T\left(\boldsymbol{A} + \boldsymbol{C}\right)\boldsymbol{v}\right)\left(2\boldsymbol{B}\boldsymbol{v}\right)}{\left(\boldsymbol{v}^T \boldsymbol{B}\boldsymbol{v}\right)^2}$$

According to gradient descent :

begin **initialize** $\boldsymbol{v}$ arbitrary, $\epsilon, \eta(\cdot), k = 0$

do $k \leftarrow k + 1$

$\boldsymbol{v} \leftarrow \boldsymbol{v} - \eta(k)\nabla J(\boldsymbol{v})$

until $|\eta(k)\nabla J(\boldsymbol{v})| < \epsilon$

return $\boldsymbol{v}$

end

(b) We could choose $\eta(k) = \frac{1}{k}$. This way the value at the **until** condition would be very small at a point thus the condition will hold and the algorithm will terminate.