T-61.231 Principles of Pattern Recognition

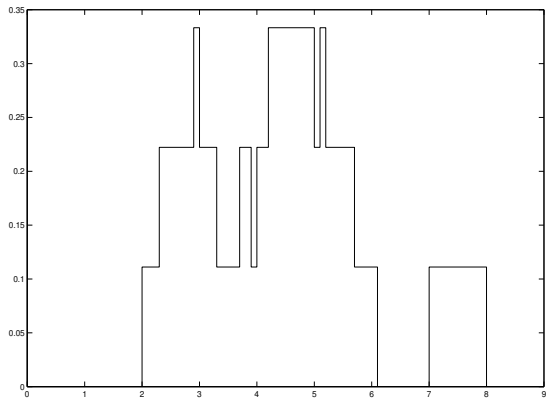
Answers to exercise 3: 15.10.2001

1. Parzen window $\Theta(x) = \Theta\left(\frac{\overline{x}-\overline{x^i}}{h_n}\right) = \begin{cases} 1, \text{ if } \overline{x} \text{ in the hypercuve center } \overline{x^i}, \text{ side length } h_n \\ 0, \text{ otherwise} \end{cases}$

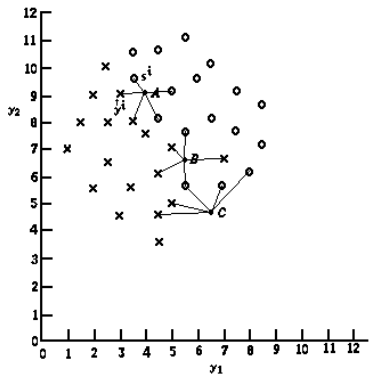
   $k_n = \sum_{i=1}^{n} \Theta\left(\frac{\overline{x}-\overline{x^i}}{h_n}\right)$ is the number of samples in the hypercube and n the total number of samples.

   $p_n(\overline{x}) = \frac{k_n/n}{V_n} = \frac{1}{nV_n}\sum_{i=1}^{n} \Theta\left(\frac{\overline{x}-\overline{x^i}}{h_n}\right)$, where V_n is the volume of the hypercube.

   Now $\overline{x^i} = y^{(i)}$, $h_n = 1$, $V_n = 1$ and $n = 9$. Thus



2. Point X is classified according to it's 5 nearest neighbors. The distance metric used is the Euclidean distance and the decision rule the voting result of the 5 nearest neighbors.



Point	Votes for O	Votes for X	Resulting class
A	3	2	O
B	2	3	X
C	3	2	O

3. **a)** $y^{(i)} = [y_1^{(i)}\ y_2^{(i)}]^T, s^{(i)} = [s_1^{(i)}\ s_2^{(i)}]^T$. A point $\hat{y} = [y_1\ y_2]$ is on the decision boundary, if

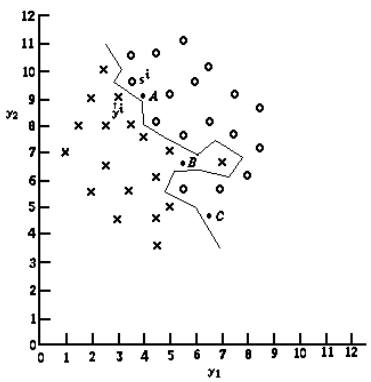
$$d(\hat{y}, y^{(i)}) = d(\hat{y}, s^{(j)})$$
$$\sqrt{(\hat{y}_1 - y_1^{(i)})^2 + (\hat{y}_2 - y_2^{(i)})^2} = \sqrt{(\hat{y}_1 - s_1^{(i)})^2 + (\hat{y}_2 - s_2^{(i)})^2}$$

$\Leftrightarrow \hat{y}_1 2(s_1^{(j)} - y_1^{(i)}) + \hat{y}_2 2(s_2^{(j)} - y_2^{(i)}) + (y_1^{(i)})^2 + (s_1^{(i)})^2 + (y_2^{(i)})^2 + (s_2^{(i)})^2 = 0$

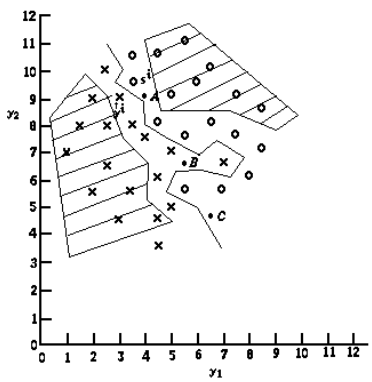
and by setting $a = 2(s_1^{(j)} - y_1^{(i)})$, $b = 2(s_2^{(j)} - y_2^{(i)})$ and $c = (y_1^{(i)})^2 + (s_1^{(i)})^2 + (y_2^{(i)})^2 + (s_2^{(i)})^2$ it is clear that the resulting equation $a\hat{y}_1 + b\hat{y}_2 + c = 0$ represents a line. By taking the points from the figure,

$$\begin{cases} s_1^{(j)} = 3.6 \\ s_2^{(j)} = 9.5 \\ y_1^{(j)} = 3.0 \\ y_2^{(j)} = 9.0 \end{cases} \Rightarrow \begin{cases} a = 1.2 \\ b = 1.0 \\ c = -13.12 \end{cases} \Rightarrow \hat{y}_2 = -1.2\hat{y}_1 + 13.12$$

b)



c)



4. A 1-NN Classifier is created by using a training set $H_n = \{\overline{x}_1, \ldots, \overline{x}_n\}$ and it is tested with the vector $\overline{x}$. $\overline{x}$ belongs to the class ω and its nearest neighbor $\overline{x}'$ belongs to class θ. The error probability for $\overline{x}$ is (Schalkoff page 79):

$$e_{1NNR}(\overline{x}, \overline{x}') = P(\omega \neq \theta | \overline{x}, \overline{x}') = \textstyle\sum_i P(\omega = \omega_i | \overline{x}) P(\theta \neq \omega_i | \overline{x}')$$

Now $\overline{x} = \overline{x}'$ as the training set is also used as a test set.

$\Rightarrow \forall i P(\theta \neq \omega_i | \overline{x}') = 0$, if $\omega = \omega_i \Rightarrow e_{1NNR} = 0$

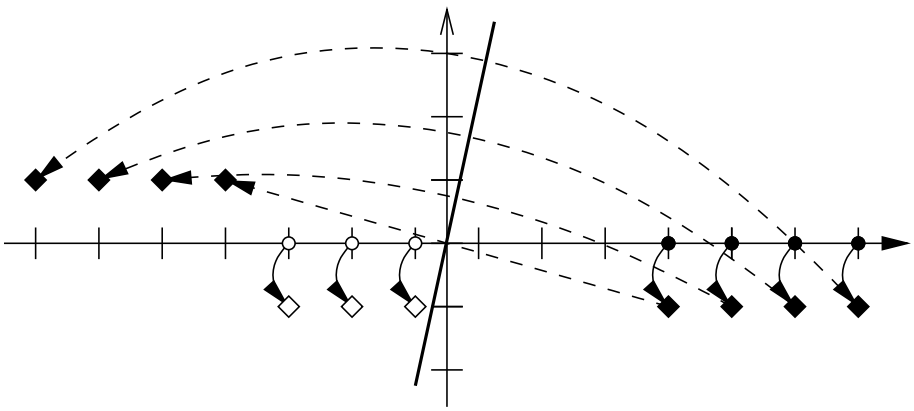
This is obviously an useless estimate for the recognition performance.

5. $\underline{\omega}^T \underline{x} - \omega_0 \begin{cases} > 0, \forall \underline{x} \in H_1 \\ < 0, \forall \underline{x} \in H_2 \end{cases}$.

a) $\underline{\omega'}^T \underline{x} - \omega_0 = \begin{bmatrix} \underline{\omega'} & \omega_0 \end{bmatrix} \begin{bmatrix} \underline{x} \\ -1 \end{bmatrix} = \underline{\omega}^T \underline{\hat{x}}$

If $\underline{\hat{x}} \in H_1 \Rightarrow \underline{\omega}^T \underline{\hat{x}} > 0$. If $\underline{\hat{x}} \in H_2 \Rightarrow \underline{\omega}^T \underline{\hat{x}} < 0 \Rightarrow \underline{\omega}^T (-1) \underline{\hat{x}} > 0$. So if the sign of $\underline{\hat{x}}$ is changed when $\underline{\hat{x}} \in H_2$, it holds that $\underline{\omega}^T \underline{\hat{x}} > 0 \; \forall \underline{\hat{x}}$

This is illustrated in the figure below, where the arrows represent the augmentation and dashed arrow the change in sign. After the augmentation and sign change all the samples are on the same side of the line.



6. $\underline{\omega}^{(n)^T} \underline{\hat{x}}_i < 0$ means that $\underline{\hat{x}}_i$ was not correctly classified in the nth iteration round.

In order to have $\underline{\hat{x}}_i$ correctly classified in the next iteration, $\underline{\omega}^{(n)}$ must be updated so that $\underline{\omega}^{(n+1)^T} \underline{\hat{x}}_i > 0$

The updating rule is $\underline{\omega}^{(n+1)} = \underline{\omega}^{(n)} + \alpha \underline{\hat{x}}_i$

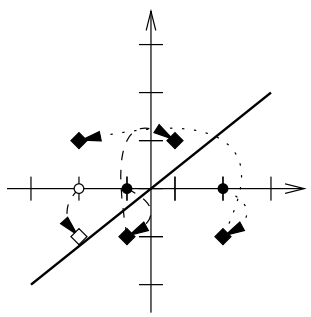
$$\Rightarrow \underline{\omega}^{(n+1)^T} \underline{\hat{x}}_i = (\underline{\omega}^{(n)} + \alpha \underline{\hat{x}}_i)^T \underline{\hat{x}}_i = \underline{\omega}^{(n)^T} \underline{\hat{x}}_i + \alpha_n ||\underline{\hat{x}}_i||^2 > 0 \Leftrightarrow \alpha_n > \frac{-\underline{\omega}^{(n)^T} \underline{\hat{x}}_i}{||\underline{\hat{x}}_i||^2}$$

7. Let's augment the vectors and choose to change the sign of the samples belonging to H_2. Thus the sample vectors are $\underline{x}_1 = \begin{bmatrix} -2 \\ -1 \end{bmatrix} \in H_1$ and $\underline{x}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \in H_2$, $\underline{x}_3 = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \in H_2$. The initial weight vector is $\underline{\omega}^0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \in H_1$ and $\alpha = 0.5$.

The iteration will proceed as follows: (Note that the update is made even when $\underline{\omega}^T \underline{x} = 0$, even though this is not really indicated by the rule. I think that it is better to update them too often as this should only improve convergence, and as the original decision rule is based on < 0 or > 0 the situation $\underline{\omega}^T \underline{x} = 0$ is "indecisive". Although, in real-life situations the values would probably never be so precise as for 0 to be the result and this slight dilemma would never be seen.)

Sample	$\underline{\omega}^T \underline{x}$	New $\underline{\omega}^T$
1	$-2-1=-3<0$	$[(1-1)\ \ (1-0.5)]=[0\ \ 0.5]$
2	$0+0.5=0.5>0$	no change
3	$0+0.5=0.5>0$	no change
1	$0-0.5=-0.5<0$	$[(0-1)\ \ (0.5-0.5)]=[-1\ \ 0]$
2	$-1+0=-1<0$	$[(-1+0.5)\ \ (0+0.5)]=[-0.5\ \ 0.5]$
3	$1+0.5=1.5>0$	no change
1	$1-0.5=0.5>0$	no change
2	$-0.5+0.5=0$	$[(-0.5+0.5)\ \ (0.5+0.5)]=[0\ \ 1]$
3	$0+1=1>0$	no change
1	$0-1=-1<0$	$[(0-1)\ \ (1-0.5)]=[-1\ \ 0.5]$
2	$-1+0.5=-0.5<0$	$[(-1+0.5)\ \ (0.5+0.5)]=[-0.5\ \ 1]$
3	$1+1=2>0$	no change
1	$1-1=0$	$[(-0.5-1)\ \ (1-0.5)]=[-1.5\ \ 0.5]$
2	$-1.5+0.5=-1<0$	$[(-1.5+0.5)\ \ (0.5+0.5)]=[-1\ \ 1]$
3	$2+1=3>0$	no change
1	$2-1=1>0$	no change
2	$-1+1=0$	$[(-1+0.5)\ \ (1+0.5)]=[0.5\ \ 1.5]$
3	$-1+1.5=0.5>0$	no change
1	$1-1.5=-0.5<0$	$[(-0.5-1)\ \ (1.5-0.5)]=[-1.5\ \ 1]$
2	$-1.5+1=0.5<0$	$[(-1.5+0.5)\ \ (1.5+0.5)]=[-1\ \ 1.5]$
3	$2+1.5=2.5>0$	no change
1	$2-1.5=0.5>0$	no change
2	$-1+1.5=0.5>0$	no change

So the algorithm converged at a weight vector of $\underline{\omega}=[-1\ \ 1.5]^T=[\omega'\ \ \omega_0]$, which is illustrated in the figure below.



The correctness of the result can also be verified by calculating with the original values:

$$x_1=-2 \Rightarrow \omega' x-\omega_0=-1*-2-1.5=0.5>0$$
$$x_2=-1 \Rightarrow \omega' x-\omega_0=-1*-1-1.5=-0.5<0$$
$$x_3=2 \Rightarrow \omega' x-\omega_0=-1*2-1.5=-3.5<0$$