KECE470 Pattern Recognition

Chapter 3. Linear Classifiers

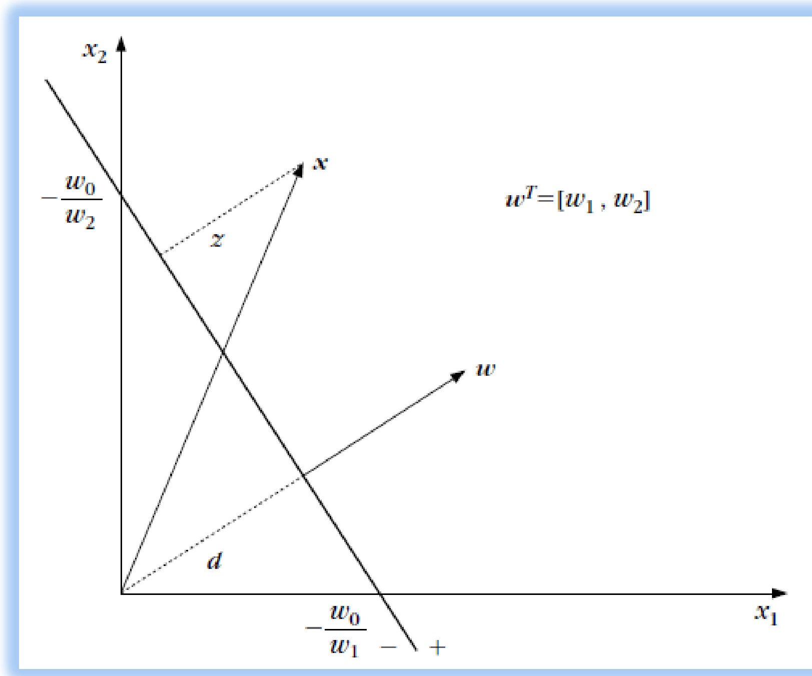
Chang-Su Kim

Many slides are modified from Serigos Theodoridis's own notes.

Linear Classifiers

- Linear classifiers are simple and computationally attractive
- Linear discriminant functions → linear decision surfaces (decision hyperplanes)

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0$$
$$\Leftrightarrow \quad \mathbf{w}^T(\mathbf{x} - \mathbf{x}_0) = 0$$



PERCEPTRON ALGORITHM

Linearly Separable Case

- There exists a hyperplane, $\mathbf{w}^{*T}\mathbf{x} = 0$, such that

$$\mathbf{w}^{*T}\mathbf{x} > 0 \;\; \forall \mathbf{x} \in \omega_1$$
$$\mathbf{w}^{*T}\mathbf{x} < 0 \;\; \forall \mathbf{x} \in \omega_2$$

  – This formulation also covers the case of a hyperplane not crossing the origin, *i.e.*, $\mathbf{w}^{*T}\mathbf{x} + w_0^*$=0, by defining the **extended** $(l+1)$-dimensional vectors

$$\mathbf{x}' \equiv [\mathbf{x}^T, 1]^T \text{ and } \mathbf{w}' \equiv [\mathbf{w}^{*T}, w_0^*]^T.$$

  – Then $\mathbf{w}^{*T}\mathbf{x} + w_0^* = \mathbf{w}'^T\mathbf{x}'$

Problem Formulation: Perceptron Cost

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

- Y is the set of vectors, misclassified by the hyperplane $\mathbf{w}$
- The variable δ_x

$$\delta_x = \begin{cases} -1 & \text{if } x \in \omega_1 \\ +1 & \text{if } x \in \omega_2 \end{cases}$$

- For separating $\mathbf{w}$, $J(\mathbf{w}) = 0$ because $Y = \emptyset$
- $J(\mathbf{w})$ is continuous and piecewise linear

Optimization: Perceptron Algorithm

- Inspired by gradient descent

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

- Note that $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_{\mathbf{x}\in Y} \delta_x \mathbf{x}$. Thus, we have

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

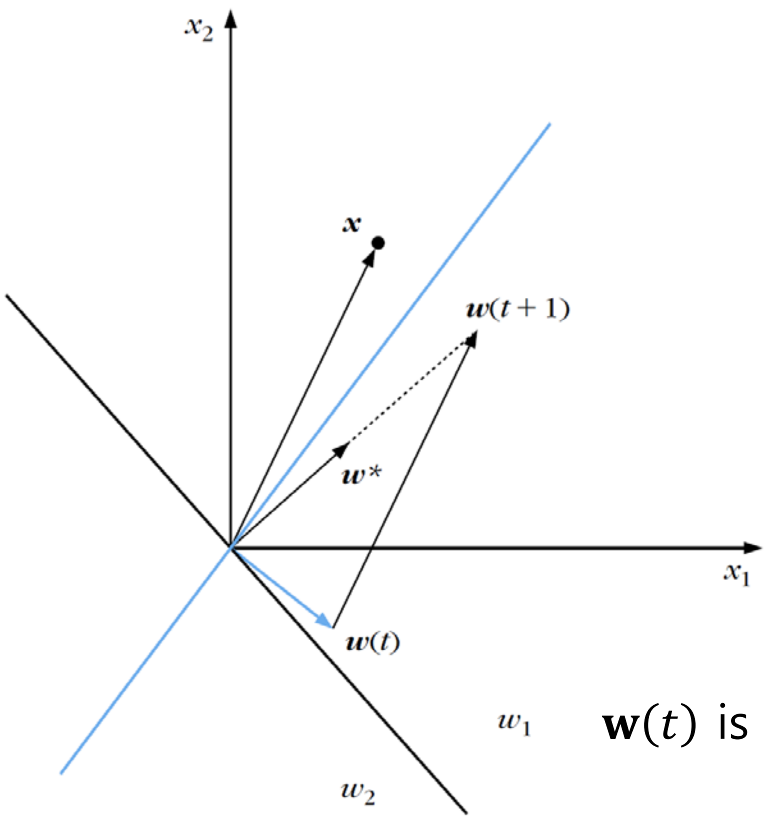
Perceptron Algorithm

- Choose $\mathbf{w}(0)$ randomly
- Choose ρ_0
- $t = 0$
- Repeat
  - $Y = \emptyset$
  - For $i = 1$ to N

    If $\delta_{\mathbf{x}_i}\mathbf{w}(t)^T\mathbf{x}_i \geq 0$ then $Y = Y \cup \{\mathbf{x}_i\}$
  - End For
  - $\mathbf{w}(t+1) = \mathbf{w}(t) - \rho_t \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}|_{\mathbf{w}=\mathbf{w}(t)}$
  - Adjust ρ_t
  - $t = t + 1$
- Until $Y = \emptyset$

Perceptron Algorithm

- Remark
  - It converges to a solution in a finite number of steps, provided that $\rho_t \propto \frac{1}{t}$ (proof skipped)



$\mathbf{w}(t)$ is updated to $\mathbf{w}(t+1)$ to include $\mathbf{x}$

Perceptron Algorithm

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

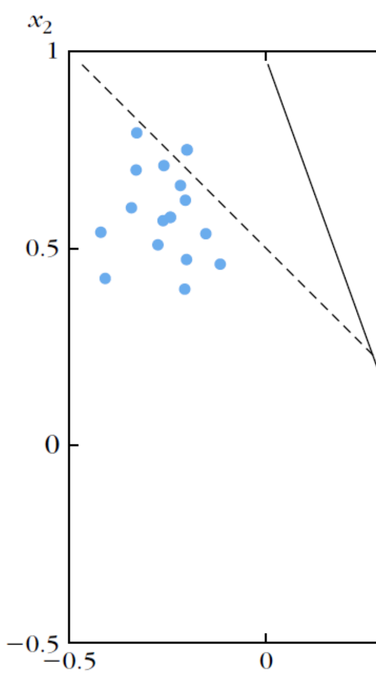


FIGURE 3.3

An example of the perceptron algorithm. After the
turned from its initial location (dotted line) to the r
classified.

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.

Terminology

If $\mathbf{w}^T\mathbf{x} + w_0 > 0$ assign $\mathbf{x}$ to ω_1

If $\mathbf{w}^T\mathbf{x} + w_0 < 0$ assign $\mathbf{x}$ to ω_2

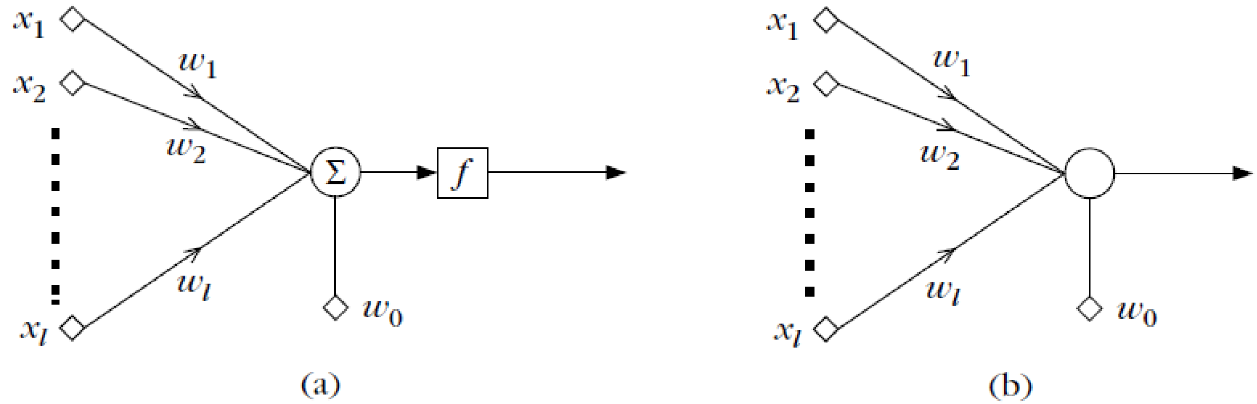


FIGURE 3.5

The basic perceptron model. (a) A linear combiner is followed by the activation function. (b) The combiner and the activation function are merged together.

- **Perceptron** or **neuron**
- **Synaptic weights** or **synapses**
- **Activation function**: *e.g.* $f(x) = 2\delta(x) - 1$

Variants

- Reward and punishment schemes
  - Training vectors enter the algorithm cyclically

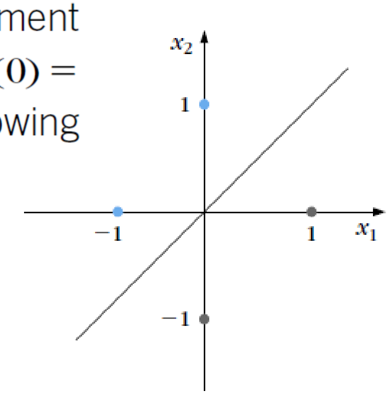
$$\mathbf{w}(t+1) = \mathbf{w}(t) + \rho\mathbf{x}_{(t)} \quad \text{if } \mathbf{x}_{(t)} \in \omega_1 \text{ and } \mathbf{w}^T(t)\mathbf{x}_{(t)} \leq 0$$

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \rho\mathbf{x}_{(t)} \quad \text{if } \mathbf{x}_{(t)} \in \omega_2 \text{ and } \mathbf{w}^T(t)\mathbf{x}_{(t)} \geq 0$$

$$\mathbf{w}(t+1) = \mathbf{w}(t) \quad \text{otherwise}$$

Example 3.2

Figure 3.4 shows four points in the two-dimensional space. Points $(-1,0), (0,1)$ belong to class ω_1, and points $(0,-1), (1,0)$ belong to class ω_2. The goal of this example is to design a linear classifier using the perceptron algorithm in its reward and punishment form. The parameter ρ is set equal to one, and the initial weight vector is chosen as $\boldsymbol{w}(0) = [0,0,0]^T$ in the extended three-dimensional space. According to (3.21)–(3.23), the following computations are in order:



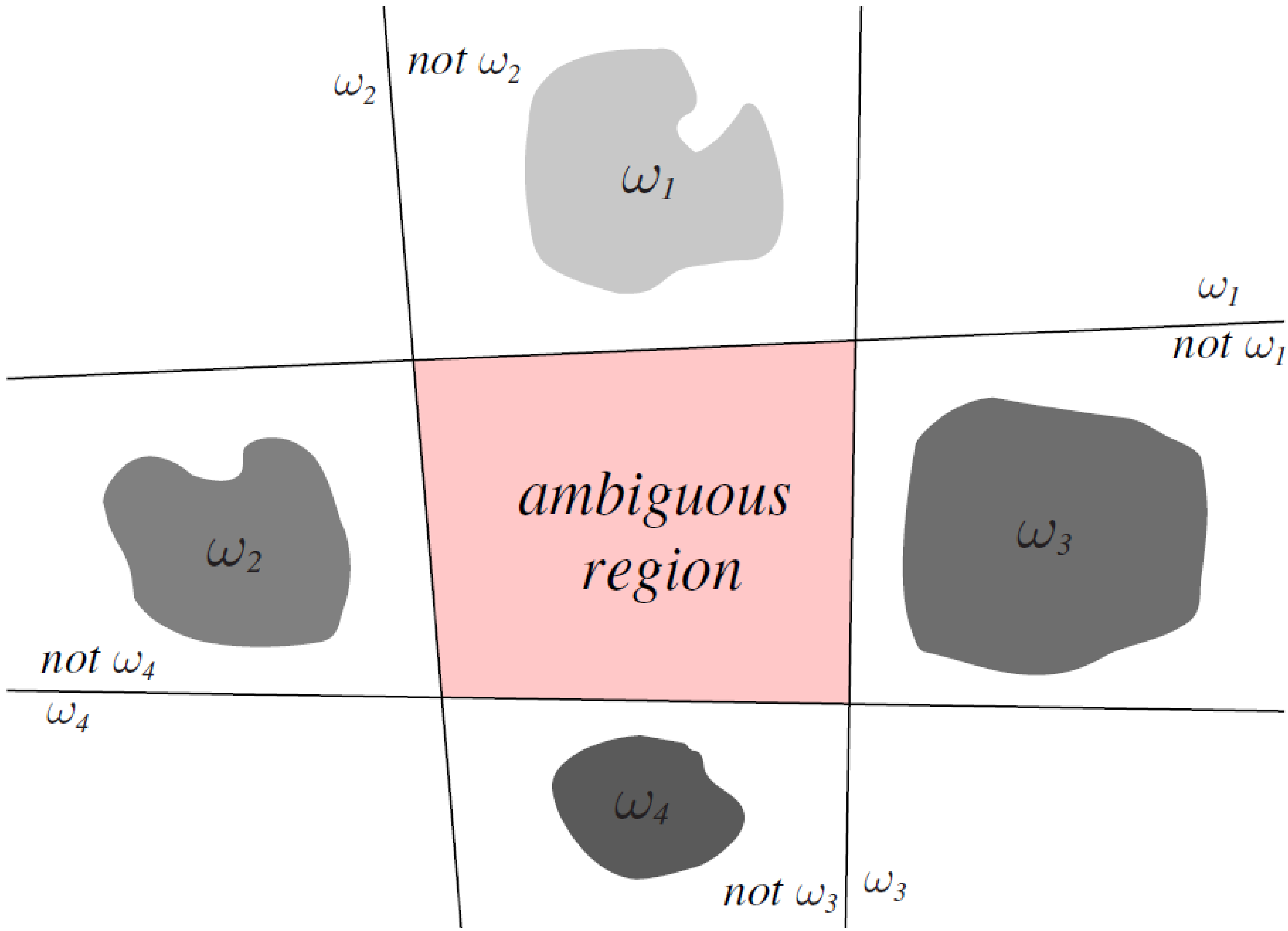
Variants

- Pocket Algorithm
  - Converges to an optimal solution, even if not linearly separable

    - Initialize the weight vector $\mathbf{w}(0)$ randomly. Define a stored (in the pocket!) vector $\mathbf{w}_s$. Set a history counter h_s of $\mathbf{w}_s$ to zero.
    - At the t-th iteration step, update $\mathbf{w}(t+1)$ according to the perceptron rule. Use $\mathbf{w}(t+1)$ to test the number h of training vectors correctly classified. If $h > h_s$ replace $\mathbf{w}_s$ with $\mathbf{w}(t+1)$ and h_s with h. Continue the iterations.

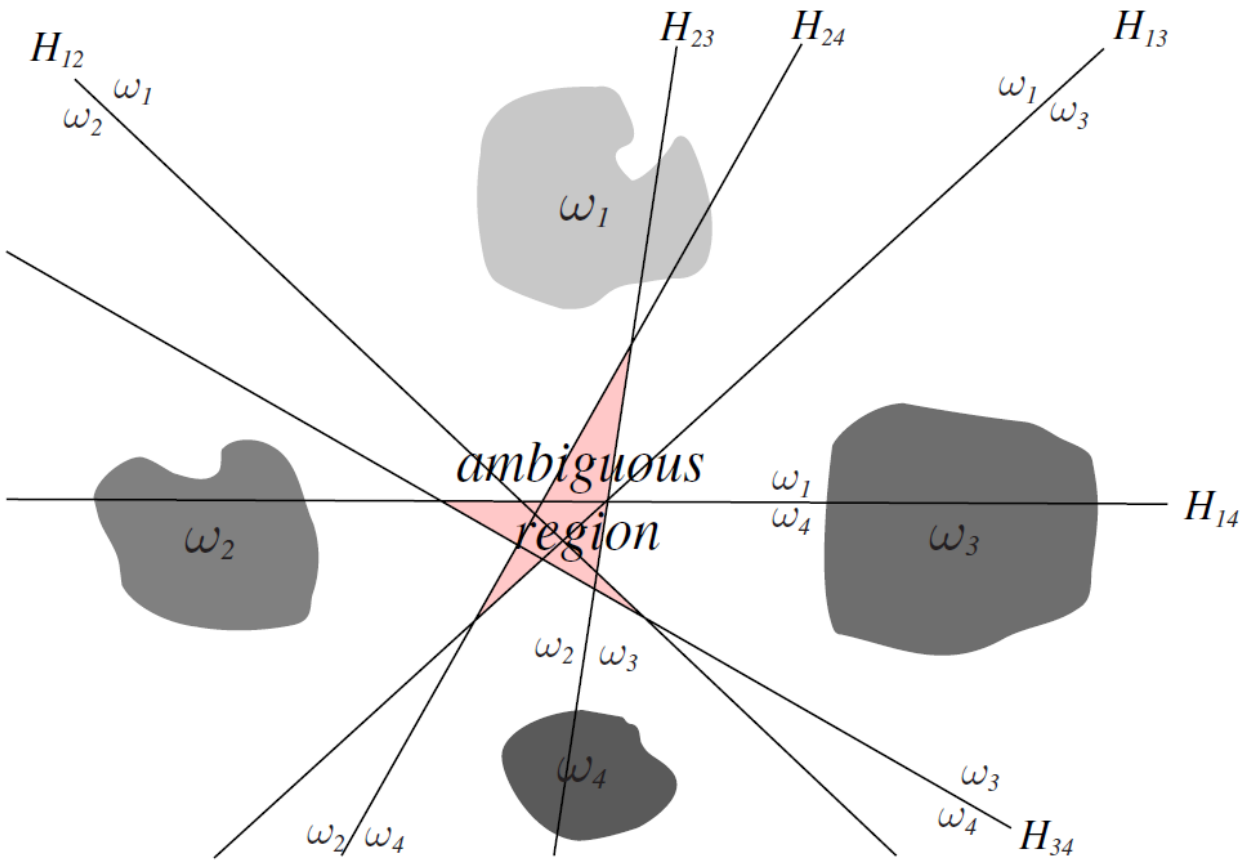
M-class Case

- Naïve approach I



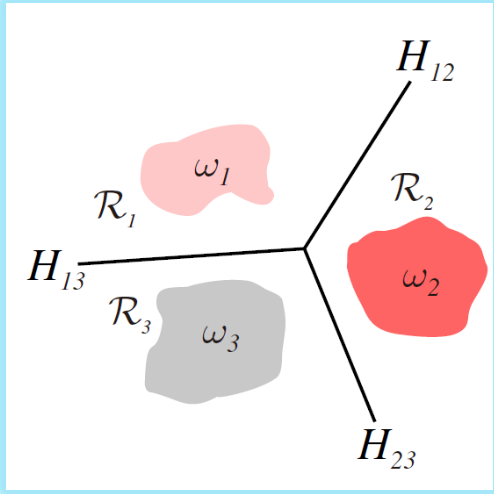
M-class Case

- Naïve approach II



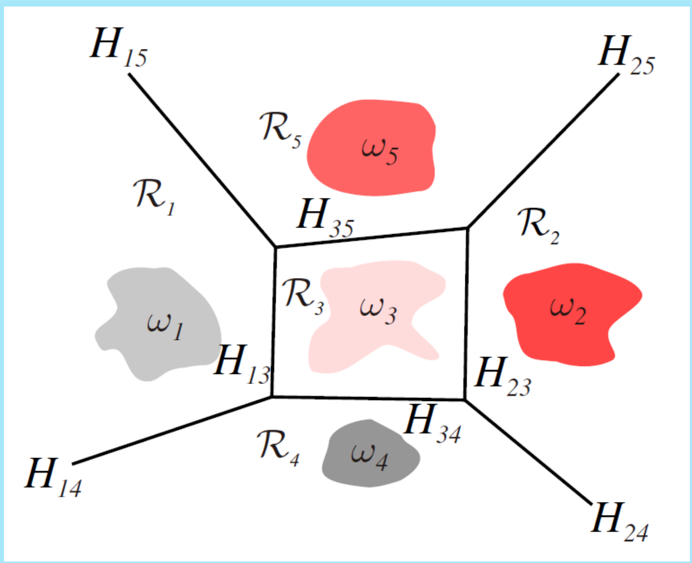
M-class Case

- Linear Machine
  - Define M linear discriminant functions $g_i(\mathbf{x})$
  - Assign $\mathbf{x}$ to ω_i if $g_i(\mathbf{x}) \geq g_j(\mathbf{x})$ for all j



- If R_i and R_j are contiguous, the boundary is given by the hyperplane

$$g_i(\mathbf{x}) = g_j(\mathbf{x})$$



Kesler's Construction

- Generalization to M-class task
  - Define a linear discriminant function $\mathbf{w}_i$, $i = 1,2,\dots,M$, for each class. Classify a feature vector $\mathbf{x}$ into class ω_i if

    $$\mathbf{w}_i^T\mathbf{x} > \mathbf{w}_j^T\mathbf{x},\ \forall j \neq i$$

  - For each training vector from class ω_i, construct $M-1$ vectors $\mathbf{x}_{ij} = [0^T, 0^T, \dots, \mathbf{x}^T, \dots, -\mathbf{x}^T, \dots, 0^T]^T$. It is a block vector, having zeros everywhere except at the ith and jth block positions, where it has $\mathbf{x}$ and $-\mathbf{x}$, respectively.
  - Also construct the block vector $\mathbf{w} = [\mathbf{w}_1^T, \dots, \mathbf{w}_M^T]^T$.
  - If $\mathbf{x} \in \omega_i$, this imposes the requirement that $\mathbf{w}^T\mathbf{x}_{ij} > 0, \forall j \neq i$.
  - The task now is to design a linear classifier, in the extended space, so that each extended training vector lies in its positive side.