4 Support Vector Machines

Separable Classes

This is a modified technique for designing linear classifiers. We start with the two-class linearly separable case.

The idea is to design a hyperplane $g(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + w_0 = 0$ that correctly classifies the data in the two training sets ω_1 and ω_2. As we have seen earlier, such a hyperplane is not unique and there are several possible solutions. Figure 4.1 illustrates the classification with two possible hyperplane solutions. While both hyperplanes do the job of classification of the training data, it is obvious that we would want to choose the hyperplane denoted by the solid line, since it leaves more room on either side.

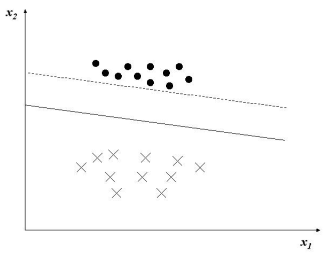

Fig. 4.1: Two possible linear classifiers for a linearly separable two-class problem. (Fig. from Theodoridis)

If we accept this rather "intuitive" idea, then our aim is now to design a hyperplane that maximizes the "margin" from both the classes. Further, if we don't give any preference to either of the two classes, then it is reasonable to select the hyperplane that has the same distance from the respective nearest points in ω_1 and ω_2. This is illustrated in Figure 4.2. In the respective directions, the hyperplanes shown with the solid lines should be selected as they have the maximum margin in those directions. Between the two solid lines, the margin for direction 1 is $2z_1$ and the margin for direction 2 is $2z_2$. Our aim then is to search for the hyperplane that provides the maximum margin. However, each hyperplane is determined within a scaling factor. We will free ourselves from it, by appropriately scaling all the candidate hyperplanes. From our earlier discussion we know that the distance of a point from a hyperplane is given by:

$$z = \frac{|g(\mathbf{x})|}{\|\mathbf{w}\|} \tag{4.1}$$

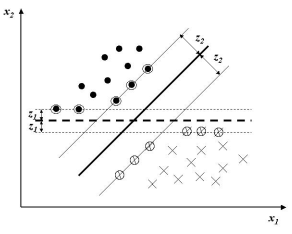

Fig. 4.2: Two directions with different margins: margin for direction 2 is larger than that for direction 1. The circled points are the support vectors. (Fig. from Theodoridis)

Note that the weight vector $\mathbf{w}$ here does not include w_0. The weight vector $\mathbf{w}$ and bias w_0 can now be scaled so that the value of $g(\mathbf{x})$ at the nearest points in ω_1, ω_2, is respectively equal to $+1$ and -1. These points are circled in Figure 4.2 and are known as the ***support vectors***. The support vectors are the samples closest to the separating hyperplane and are the most difficult samples to classify. The optimal hyperplane is completely defined by the support vectors, but of course we don't know which samples are support vectors without finding the optimal hyperplane. After this scaling, the total margin is

$$\frac{1}{\|\mathbf{w}\|} + \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{4.2}$$

Since we want to maximize the margin while ensuring correct classification, we can formulate the following optimization problem:

$$\min J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{4.3}$$

$$\text{subject to: } \delta_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq 1 \qquad i = 1, 2, ..., n \tag{4.4}$$

where δ_i is class indicator for $\mathbf{x}_i$ defined as: $\delta_i = +1$ if $\mathbf{x}_i \in \omega_1$, and $\delta_i = -1$ if $\mathbf{x}_i \in \omega_2$. n is the total number of available samples. Using the idea of Lagrange multipliers, the above problem can be recast as:

$$\min_{\mathbf{w}, w_0} L = \frac{1}{2}\mathbf{w}^\top \mathbf{w} - \sum_{i=1}^{n} \lambda_i \left(\delta_i(\mathbf{w}^\top \mathbf{x}_i + w_0) - 1 \right) \tag{4.5}$$

where λ_i is the Lagrange multiplier associated with the ith inequality. The two conditions that must then apply, for optimality, are

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \qquad \frac{\partial L}{\partial w_0} = 0 \tag{4.6}$$

This gives

$$\mathbf{w} = \sum_{i=1}^{n} \lambda_i \delta_i \mathbf{x}_i \qquad \sum_{i=1}^{n} \lambda_i \delta_i = 0 \tag{4.7}$$

This can be put into the Lagrange function in Eq. 4.5 from which we get a dual problem

Of the various Lagrange multipliers, only those corresponding to the active constraints $\delta_i(\mathbf{w}^\top \mathbf{x}_i + w_0) = 1$ will be non-zero, the remaining Lagrange multipliers will be zero. The data points for which the Lagrange multipliers are non-zero are the support vectors. From equation 4.12 it can be seen that the weights then depend only on the support vectors.

$$\max_{\lambda} \; W = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \delta_i \delta_j \lambda_i \lambda_j \mathbf{x}_i^\top \mathbf{x}_j \tag{4.8}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \lambda_i \delta_i = 0 \tag{4.9}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \tag{4.10}$$

$$\lambda_i(\delta_i(\mathbf{w}^\top \mathbf{x}_i + w_0) - 1) = 0 \tag{4.11}$$

where λ_i is the lagrangian multiplier for the constraint corresponding to the i^{th} data point. The above optimization problem can be solved for the optimal Lagrange multipliers λ_i using any optimization tool (such as those in Matlab). Once these Lagrange multipliers are computed, the weight vector $\mathbf{w}$ of the discriminating hyperplane can be calculated as:

$$\mathbf{w} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \lambda_{\mathbf{i}} \delta_{\mathbf{i}} \mathbf{x}_{\mathbf{i}} \tag{4.12}$$

while the bias w_0 can be estimated as

$$w_0 = \delta_K - \mathbf{x}_K \mathbf{w} \qquad \text{where } K = \arg\max_i \lambda_i \tag{4.13}$$

Example 4.1. It is required to separate class 1 from 2 where class 1 has 3 points: $(1, 6)$, $(1, 10)$ and $(4, 11)$, and class 2 has $(5, 2)$, $(7, 6)$ and $(10, 4)$.

Soln:

First note that W inEq.4.8 may be written as

$$W = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \begin{bmatrix} \lambda_1 \\ \dots \\ \lambda_n \end{bmatrix}^\top \mathbf{H} \begin{bmatrix} \lambda_1 \\ \dots \\ \lambda_n \end{bmatrix} \tag{4.14}$$

where the (i, j) element in $\mathbf{H}$ is $\delta_i \delta_j \mathbf{x}_i^\top \mathbf{x}_j$. The data is rearranged into $\mathbf{x}$ and $\boldsymbol{\delta}$:

$$\mathbf{x} = \begin{bmatrix} 1 & 6 \\ 1 & 10 \\ 4 & 11 \\ 5 & 2 \\ 7 & 6 \\ 10 & 4 \end{bmatrix} \qquad \boldsymbol{\delta} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

Then $\mathbf{H}$ will turn out to be

In matlab $\mathbf{H}$ can be computed as $(\mathbf{x} * \mathbf{x}^\top) .* (\boldsymbol{\delta} * \boldsymbol{\delta}^\top)$.

$$\mathbf{H} = \begin{bmatrix} 37 & 61 & 70 & -17 & -43 & -34 \\ 61 & 101 & 114 & -25 & -67 & -50 \\ 70 & 114 & 137 & -42 & -94 & -84 \\ -17 & -25 & -42 & 29 & 47 & 58 \\ -43 & -67 & -94 & 47 & 85 & 94 \\ -34 & -50 & -84 & 58 & 94 & 116 \end{bmatrix}$$

The quadratic programming problem posed above needs to be solved in Matlab using the `quadprog` command. Unfortunately the problem as stated above is not in canonical form as long as Matlab is concerned, and hence needs to be rearranged to handle minimization and $\leq$ inequalities. We use

$$\mathbf{f} = \begin{bmatrix} -1 \\ \dots \\ -1 \end{bmatrix} = -\texttt{ones}(6,1) \qquad \Rightarrow \qquad \min\ \mathbf{f}^{\mathsf{T}}\boldsymbol{\lambda} + \frac{1}{2}\boldsymbol{\lambda}\mathbf{H}\boldsymbol{\lambda}$$

$$\mathbf{a} = \begin{bmatrix} 0 \\ \dots \\ 0 \end{bmatrix} = \texttt{zeros}(6,1) \qquad \mathbf{A} = \begin{bmatrix} -1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -1 \end{bmatrix} = -\texttt{eye}(6) \qquad \Rightarrow \qquad \mathbf{A}\boldsymbol{\lambda} \leq \mathbf{a}$$

Eq. 4.9 can be rewritten as follows

$$\mathbf{B} = \begin{bmatrix} \delta_1 & \cdots & \delta_6 \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} = [[\boldsymbol{\delta}]; [\texttt{zeros}(5,6)]] \qquad \mathbf{b} = \begin{bmatrix} 0 \\ \dots \\ 0 \end{bmatrix} = \texttt{zeros}(6,1) \qquad \Rightarrow \qquad \mathbf{B}\boldsymbol{\lambda} = \mathbf{b}$$

Then

$$\boldsymbol{\lambda} = \texttt{quadprog}(\mathbf{H} + \texttt{eye}(6) * 0.001, \mathbf{f}, \mathbf{A}, \mathbf{a}, \mathbf{B}, \mathbf{b})$$

The $\texttt{eye}(6) * 0.001$ term is added to the objective function for stability reasons: this addition ensures that $\mathbf{H}$ is positive definite.

The solution is

$$\boldsymbol{\lambda} = [0.036, 0, 0.039, 0, 0.076, 0]^{\mathsf{T}}$$

which implies the the 1st, 3rd and 5th points in $\mathbf{x}$ are the support vectors. The discriminant is obtained from

$$\mathbf{w} = \sum_{i=1}^{n} \lambda_i \delta_i \mathbf{x}_i = (\boldsymbol{\lambda}. * \boldsymbol{\delta})^{\mathsf{T}}\mathbf{x} = \begin{bmatrix} -0.33 \\ 0.20 \end{bmatrix}$$

Since $\lambda_1 > 0$, the offset is easily found as

$$w_0 = \frac{1}{\delta_1} - \mathbf{w}^{\mathsf{T}}\mathbf{x}_1 = 0.13$$

Non-separable Classes

For situations where the samples are linearly non-separable, the above optimization procedure is not valid. This situation is illustrated in Figure 4.3. It is not possible to draw a hyperplane that will have a class separation band around it, with no data points inside this band. The training vectors now belong to one of the following three categories:

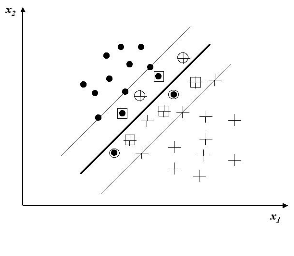

Fig. 4.3: Nonseparable classes case: some points fall inside the class separation band. (Fig. from Theodoridis)

- Vectors that fall outside the band and are correctly classified. These vectors satisfy

$$\delta_i(\mathbf{w}^{\mathsf{T}}\mathbf{x}_i + w_0) \geq 1$$

- Vectors that fall inside the band but are correctly classified. These vectors satisfy

$$0 \leq \delta_i(\mathbf{w}^\top \mathbf{x}_i + w_0) < 1$$

- Vectors that are misclassified. These satisfy

$$\delta_i(\mathbf{w}^\top \mathbf{x}_i + w_0) < 0$$

All these three cases can be represented by a single constraint by introducing a new variable ξ_i as:

$$\delta_i(\mathbf{w}^\top \mathbf{x}_i + w_0) \geq 1 - \xi_i; \qquad i = 1, 2, ..., n \tag{4.15}$$

The first category of data corresponds to $\xi_i = 0$, the second to $0 < \xi_i \leq 1$, and the third to $\xi_i > 1$. The variables ξ_i are known as slack variables. With this modification, we can now solve the following optimization problem:

$$\min J(\mathbf{w}, w_0, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \tag{4.16}$$

$$\begin{aligned} \text{subject to} \quad & \delta_i[\mathbf{w}^\top \mathbf{x}_i + w_0] \geq 1 - \xi_i, \quad i = 1, 2, ..., n \\ & \xi_i > 0, \quad i = 1, 2, ..., n \end{aligned}$$

where the parameter C is a positive constant that controls the relative influence of the two competing terms. As before, based on the use of Lagrange multipliers, the above problem can be recast as:

$$\max_{\lambda} \; W = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \delta_i \delta_j \lambda_i \lambda_j \mathbf{x}_i^\top \mathbf{x}_j \tag{4.17}$$

$$\begin{aligned} \text{subject to} \quad & \sum_{i=1}^{n} \lambda_i \delta_i = 0 \\ & 0 \leq \lambda_i \leq C, \qquad i = 1, 2, ..., n \end{aligned}$$

Note that the only difference between this formulation and the one presented earlier for the linearly separable case is that the Lagrange multipliers λ_i are bounded above by C. The linearly separable case corresponds to $C \to \infty$. Further, also note that the slack variables ξ_i and their corresponding Lagrange multipliers do not appear in the formulation. Their presence is indirectly reflected through C.

Allowing noisy, misclassified, data suggests that we now have a 'soft margin'.

Kernel Trick

There is another technique available to solve linearly inseparable problems. We define a kernel function, using an inner product between the data. This facilitates a nonlinear transformation, from the input space, to a higher dimension space, where the problem may be linearly separable. Let $\mathbf{\Phi} : \mathbf{X} \to \mathbf{H}$ indicate a nonlinear transformation from the input space $\mathbf{X}$ to the feature space $\mathbf{H}$.

$$\mathbf{w}^{\mathbf{\Phi}\top} \mathbf{\Phi}(\mathbf{x}) + w_0 = 0 \tag{4.18}$$

Without losing generality, w_0 can be set to zero:

$$\mathbf{w}^{\mathbf{\Phi}\top} \mathbf{\Phi}(\mathbf{x}) = 0 \tag{4.19}$$

Example 4.2. Consider the mapping $\mathbf{\Phi} : \mathbb{R}^2 \to \mathbb{R}^3$ with $(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. If you collect the monomial features of degree 2, we can have $([x]_1, [x]_2) \mapsto ([x]_1^2, [x]_2^2, [x]_1[x]_2)$.

The optimal weight vector $\mathbf{w}^{\mathbf{\Phi}*}$ in the new feature space is

This is similar to the derivation based on the Lagrange multiplier method, above.

$$\mathbf{w}^{\mathbf{\Phi}*} = \sum_{i=1}^{n} \lambda_i^* \delta_i \mathbf{\Phi}(\mathbf{x}_i) \tag{4.20}$$

The optimal hyperplane in the feature space is

The trick here is that we do not need to explicitly solve this problem in the (typically higher dimension) feature space. Therefore trying to derive a linear classifier in a higher dimension space is equivalent to solving a linearly inseparable problem in the original space, efficiently.

$$\sum_{i=1}^{n} \lambda_i^* \delta_i \mathbf{\Phi}^\top(\mathbf{x}_i)\mathbf{\Phi}(\mathbf{x}) = 0 \tag{4.21}$$

The term $\mathbf{\Phi}^\top(\mathbf{x}_i)\mathbf{\Phi}(\mathbf{x})$ is the inner product kernel:

$$\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{\Phi}^\top(\mathbf{x}_i)\mathbf{\Phi}(\mathbf{x}_j) \tag{4.22}$$

When the $\mathbf{\Phi}$ refer to second order polynomials, the kernel is simply the square of the dot product in input space:

$$x_{i,1}^2 x_{j,1}^2 + x_{i,2}^2 x_{j,2}^2 + 2x_{1,i}x_{1,j}x_{2,i}x_{2,j} = (\mathbf{x}_i \cdot \mathbf{x}_j)^2 \tag{4.23}$$

This can be seen to extend to d dimensions:

$$\mathbf{\Phi}_i^\top \mathbf{\Phi}_j = (\mathbf{x}_i \cdot \mathbf{x}_j)^d \tag{4.24}$$

The net result is that we can compute the value of the dot product without explicitly carrying out a mapping involving monomials. This dot product trick becomes important, because building a d degree polynomial given m input features would involve collecting $N(m; d)$ number of terms (i.e. monomials of degree d), and this can blow up really fast: for $m = 100$ and $d = 6$ we are looking at 1.6×10^9 terms.

Note that effectively we replace dot products in $\mathbf{x}$ with dot products in $\mathbf{\Phi}$.

$$N(d; m) = {}^{d+m-1}\mathrm{C}_d = \frac{(d+m-1)!}{d!(m-1)!} \tag{4.25}$$

The optimal hyperplane then is

$$\sum_{i=1}^{n} \lambda_i^* \delta_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) = 0 \tag{4.26}$$

Soft margins and kernels

Equation 4.17 can be modified to include the kernel function:

$$\max_{\lambda} \; W = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \delta_i \delta_j \lambda_i \lambda_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \tag{4.27}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \lambda_i \delta_i = 0$$
$$0 \le \lambda_i \le C, \qquad i = 1, 2, ..., n$$

The optimal classifier will be

$$\sum_{i=1}^{n} \lambda_i^* \delta_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) + w_o^* = 0 \tag{4.28}$$

where $w_0^* = 1 - \sum_{i=1}^{n} \lambda_i \delta_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}_s)$, for a positive support vector $\delta_s = +1$.

Kernel choices

A typical way to set up a nonlinear problem is to create a kernel from a weighted combination of classical kernels.

- Given a proposed feature space representation $\mathbf{\Phi}(x) = (\mathbf{\Phi}_1(x), \mathbf{\Phi}_2(y))$, then

$$\mathbf{K}(x,y) = \mathbf{\Phi}(x)\cdot\mathbf{\Phi}(y) = \mathbf{\Phi}_1(x)\cdot\mathbf{\Phi}_1(y) + \mathbf{\Phi}_2(x)\cdot\mathbf{\Phi}_2(y) = \mathbf{K}_1(x,y) + \mathbf{K}_2(x,y) \qquad (4.29)$$

- If the feature representation is of the form $\mathbf{\Phi}(x) = \sqrt{\alpha}\mathbf{\Phi}_1(x)$, then $\mathbf{K}(x,y) = \alpha\mathbf{K}_1(x,y)$ for $\alpha > 0$.
- If $\mathbf{\Phi}_{ij} = \mathbf{\Phi}_1(x)_i\mathbf{\Phi}_2(x)_j$ (the dot product of the ith component of $\mathbf{\Phi}_1(x)$ and the jth component of $\mathbf{\Phi}_2(x)$), then

$$\begin{aligned}\mathbf{K}(x,y) = \mathbf{\Phi}(x)\cdot\mathbf{\Phi}(y) &= \sum_{i,j}\mathbf{\Phi}(x)_{i,j}\mathbf{\Phi}(y)_{i,j} = \sum_{i,j}\mathbf{\Phi}_1(x)_i\mathbf{\Phi}_2(x)_j\mathbf{\Phi}_1(y)_i\mathbf{\Phi}_2(y)_j \\ &= \left(\sum_i \mathbf{\Phi}_1(x)_i\mathbf{\Phi}_1(y)_i\right)\left(\sum_j \mathbf{\Phi}_2(x)_i\mathbf{\Phi}_2(y)_j\right) = (\mathbf{\Phi}_1(x)\cdot\mathbf{\Phi}_1(y))(\mathbf{\Phi}_2(x)\cdot\mathbf{\Phi}_2(y)) \\ &= \mathbf{K}_1(x,y)\mathbf{K}_2(x,y) \qquad (4.30)\end{aligned}$$

- Note that if instead of polynomials of degree d, we look for polynomials *up to* degree d, then we are combining monomials of the form (for $d = 2$)

$$\mathbf{\Phi}(x)\mathbf{\Phi}(y) = \sum_{i=0}^{d}(x.y)^d = (xy)^0 + (xy)^1 + (xy)^2 + (yx)^1 = (xy+1)^2 \qquad (4.31)$$

  In general, then we get $\mathbf{K}(x,y) = (\mathbf{x}\cdot\mathbf{y}+1)^d$.
- A favorite kernel is the radial basis kernel:

$$\mathbf{K}(\mathbf{x}_i,\mathbf{x}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{\sigma^2}\right) \qquad (4.32)$$

Typically σ is chosen $= 1$.

- The sigmoid kernel is of the form

$$\mathbf{K}(\mathbf{x}_i,\mathbf{x}) = \tanh(\eta\mathbf{x}_i\cdot\mathbf{x}+\nu) \qquad (4.33)$$

Example 4.3. The XOR problem. We have class 1 $= \mathbf{x}_1 = [1,-1]^\top$, $\mathbf{x}_2 = [-1,1]^\top$, class 2 $= \mathbf{x}_3 = [1,1]^\top$, $\mathbf{x}_4 = [-1,-1]$. We use a polynomial kernel of degree 2. Then $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i\cdot\mathbf{x}_j+1)^2$. The kernel corresponds to a mapping

$$\mathbf{\Phi}(\mathbf{x}) = [1,\ \sqrt{2}\mathbf{x}^{(1)},\ \sqrt{2}\mathbf{x}^{(2)},\ \sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(2)},\ (\mathbf{x}^{(1)})^2,\ (\mathbf{x}^{(2)})^2]$$

We need to maximize W subject to $\lambda_i \geq 0$ and $\lambda_1+\lambda_2-\lambda_3-\lambda_4 = 0$.

$$\max_\lambda W = \sum_{i=1}^4\lambda_i - \frac{1}{2}\sum_{i=1}^4\sum_{j=1}^4\delta_i\delta_j\lambda_i\lambda_j\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)$$

W can be rewritten as

$$w(\boldsymbol{\lambda}) = \sum_{i=1}^4\lambda_i - \frac{1}{2}\boldsymbol{\lambda}^\top\mathbf{H}\boldsymbol{\lambda} \qquad \text{where } \mathbf{H} = \begin{bmatrix} 9 & 1 & -1 & -1 \\ 1 & 9 & -1 & -1 \\ -1 & -1 & 9 & 1 \\ -1 & -1 & 1 & 9\end{bmatrix}$$

Differentiating W w.r.t. $\boldsymbol{\lambda}$ and setting equal to zero gives

$$\frac{d}{d\boldsymbol{\lambda}}W = \begin{bmatrix}1\\1\\1\\1\end{bmatrix} - \begin{bmatrix}9 & 1 & -1 & -1\\ 1 & 9 & -1 & -1\\ -1 & -1 & 9 & 1\\ -1 & -1 & 1 & 9\end{bmatrix}\boldsymbol{\lambda} = \mathbf{0}$$

which is solved to get $\boldsymbol{\lambda} = [0.25, 0.25, 0.25, 0.25]^{\mathsf{T}}$. All constraints are satisfied, but since $\boldsymbol{\lambda} > \mathbf{0}$, all samples are support vectors. The weight vector $\mathbf{w}$ is

$$\mathbf{w} = \sum_{i=1}^{4} \lambda_i \delta_i \boldsymbol{\Phi}(\mathbf{x}_i) = 0.25(\boldsymbol{\Phi}(\mathbf{x}_1) + \boldsymbol{\Phi}(\mathbf{x}_2) + \boldsymbol{\Phi}(\mathbf{x}_3) + \boldsymbol{\Phi}(\mathbf{x}_4)) = [0, 0, 0, -\sqrt{2}, 0, 0]$$

The nonlinear discriminant function therefore is

$$g(\mathbf{x}) = \mathbf{w}\boldsymbol{\Phi}(\mathbf{x}) = \sum_{i=1}^{6} \mathbf{w}_i \boldsymbol{\Phi}_i(\mathbf{x}) = -\sqrt{2}(\sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(2)}) = -2\mathbf{x}^{(1)}\mathbf{x}^{(2)}$$