EE 649 Pattern Recognition

Support Vector Machines

Ulisses Braga-Neto

ECE Department

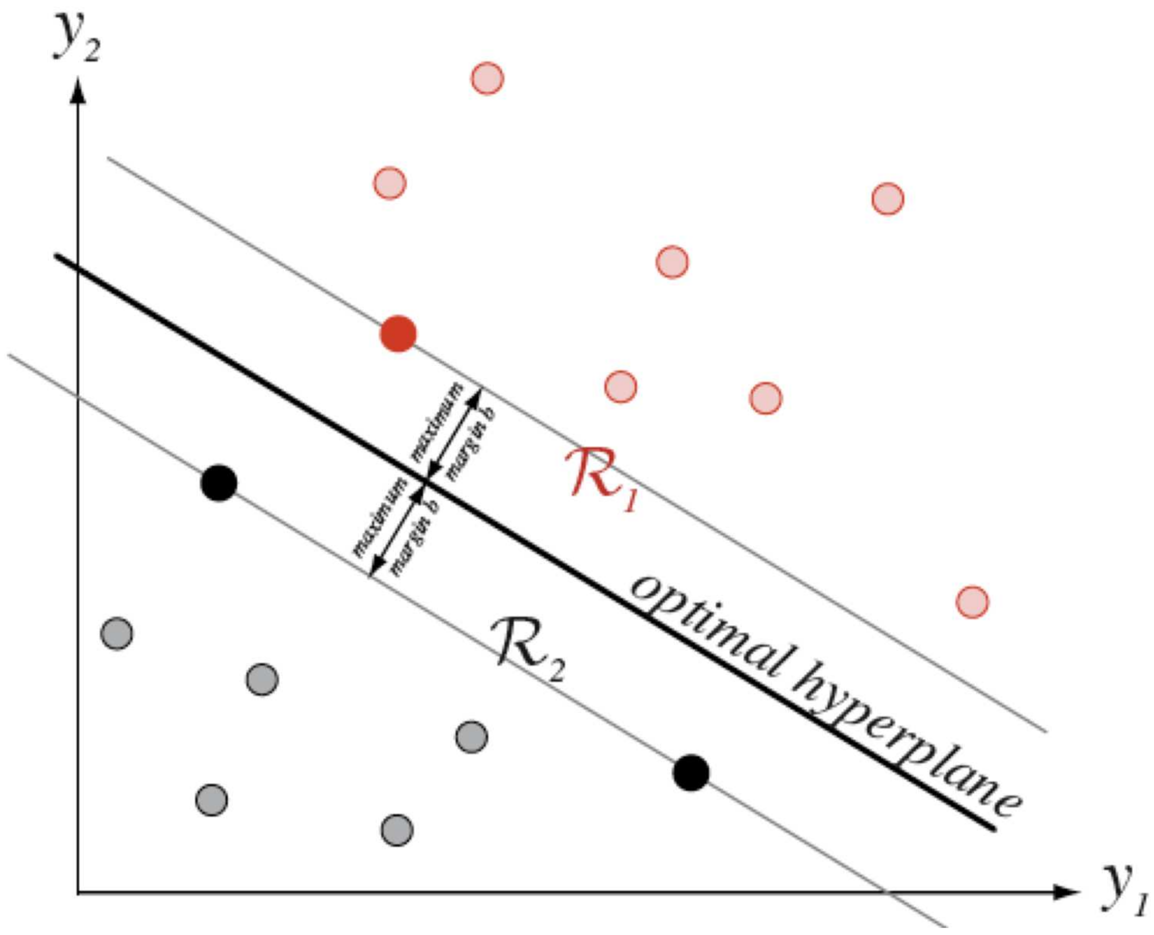
Texas A&M University

Main Ideas

- Adjust linear discriminant with margins such that the margin is *maximal* — this is called the *maximum margin algorithm*.
- The points closest to the hyperplane are called the *support vectors* and determine a lot of the properties of the classifier.
- A solution for the case where the data is not linearly separable can be readily found by using slack variables in the optimization problem.
- Alternatively, project the data to a high-dimensional space, where it is linearly separable. The solution is such that the classifier can be written in terms of *kernels* in the original space.

Maximum Margin Hyperplane (MMH)

Graphical Example.



Linear Discriminants

Linear Discriminant Function:

$$g(x) = a^T x + a_0$$

We have that

$$\psi_n(x) = \begin{cases} 1, & a^T x + a_0 > 0 \\ 0, & \text{otw} \end{cases}$$

Therefore, all training points are correctly classified if:

$$y_i(a^T x_i + a_0) > 0, \quad i = 1, \ldots, n$$

where $y_i = \pm 1$ (instead of the usual 0 and 1).

Discrimination with a Margin

Recall that if we want to have a margin $b > 0$, the constraint becomes:

$$y_i(a^T x_i + a_0) \geq b, \quad i = 1, \ldots, n$$

The perceptron solution puts all points at a distance at least $b/||a||$ from the hyperplane. Since a, a_0 and b can be freely scaled, we can set $b = 1$ without loss of generality:

$$y_i(a^T x_i + a_0) \geq 1, \quad i = 1, \ldots, n$$

The margin is $1/||a||$ and the points that are at this exact distance from the hyperplane are called *support vectors*.

Maximal Margin Hyperplane

The idea is to maximize the margin $1/||a||$.

For this, it suffices to minimize $\frac{1}{2}\,||a||^2 = \frac{1}{2}\,a^T a$ subject to the constraints:

$$y_i(a^T x_i + a_0) \geq 1, \quad i = 1, \ldots, n$$

The solution vector a^* determines the Maximal Margin Hyperplane (MMH).

Note: the corresponding optimal value a_0^* will be determined later from a^* and the constraints.

Primal Problem

The method of *Lagrange Multipliers* allows us to turn this into an unconstrained problem (unconstrained in a and a_0).

Consider the *primal Lagrangian functional*:

$$L_P(a, a_0, \lambda) = \frac{1}{2} a^T a - \sum_{i=1}^{n} \lambda_i \left(y_i(a^T x_i + a_0) - 1 \right)$$

where $\lambda_i \geq 0$ for $i = 1, \ldots, n$ are the langrange multipliers.

It can be shown that the solution to the previous constrained problem can be found by simultaneously minimizing L_p with respect to a and a_0 and maximizing it with respect to λ (hence, we search for a *saddle point* in L_p).

Dual Problem

Since L_p is unconstrained for a and a_0, a necessary condition for optimality is that the derivatives of L_p with respect to a and a_0 be zero (this is part of the "Karush-Kuhn-Tucker" - KKT conditions for the original problem), which yields the equations:

$$a = \sum_{i=1}^{n} \lambda_i y_i x_i \quad \text{and} \quad \sum_{i=1}^{n} \lambda_i y_i = 0$$

Substituting these back into L_p eliminates a and a_0, yielding the *dual Lagrangian functional*:

$$L_D(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

Dual Problem - II

Note that $L_D(\lambda)$ is a quadratic function of λ and is a function of the inner products $x_i^T x_j$ (this will be important for kernels later).

The functional $L_D(\lambda)$ must be maximized with respect to λ_i subject to the constraints:

$$\lambda_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \lambda_i y_i = 0$$

This problem can be solved using quadratic programming techniques.

Support Vectors

According to the KKT theorem, the solution λ^* corresponds to the sensitivity of L_P to the constraints.

We have that $\lambda_i^* = 0$ whenever

$$y_i(a^T x_i + a_0) > 1 \text{ (inactive or slack constraint)}$$

The support vectors are the points $\{(x_i, y_i); i \in \mathcal{S}\}$ such that $\lambda_i > 0$. For these points, we have:

$$y_i(a^T x_i + a_0) = 1 \text{ (active or tight constraint)}$$

The larger λ_i^* is, the "tighter" the constraint is and the higher the influence of the support vector is (the degenerate case where the constraint is active for a point but $\lambda_i^* = 0$ can happen; this would not be considered a support vector).

The Solution

Once λ^* is found, the MMH vector a^* is determined by

$$a^* = \sum_{i=1}^{n} \lambda_i^* y_i x_i = \sum_{i \in \mathcal{S}} \lambda_i^* y_i x_i$$

while a_0^* can be determined from any of the active constraints $a^T x_i + a_0 = y_i$ (support vectors), or from their sum:

$$n_s a_0 + (a^*)^T \sum_{i \in \mathcal{S}} x_i = \sum_{i \in \mathcal{S}} y_i$$

where $n_s = \text{Card}(\mathcal{S})$ is the number of support vectors.

The Solution - II

The previous equations yields:

$$a_0^* = -\frac{1}{n_s} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} \lambda_i^* y_i x_i^T x_j + \frac{1}{n_s} \sum_{i \in \mathcal{S}} y_i$$

The MMH classifier is therefore given by $\psi_n(x) = 1$ if

$$\sum_{i \in \mathcal{S}} \lambda_i^* y_i x_i^T x - \frac{1}{n_s} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} \lambda_i^* y_i x_i^T x_j + \frac{1}{n_s} \sum_{i \in \mathcal{S}} y_i > 0$$

with $\psi_n(x) = 0$ otherwise.

The MMH classifier is a function *only* of the support vectors and of inner products of the form $x^T x$.

Non-Separable Case

If the data is not linearly separable, it is still possible to formulate the problem and find a solution by introducing *slack variables* ξ_i, $i = 1, \ldots, n$, for each of the constraints, resulting in a new set of $2n$ constraints:

$$y_i(a^T x_i + a_0) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, \quad i = 1, \ldots, n$$

Therefore, if $\xi_i > 0$, the corresponding training point is an "outlier," i.e., it can lie closer to the hyperplane than the margin, or even be misclassified. We introduce a penalty term $C \sum_{i=1}^{n} \xi_i$ in the functional, which then becomes:

$$\frac{1}{2} a^T a + C \sum_{i=1}^{n} \xi_i$$

Slack Penalty Constant

The constant C modulates how large the penalty for the presence of outliers are. If C is small, the penalty is small and a solution is more likely to incorporate outliers. If C is large, the penalty is large and therefore a solution is unlikely to incorporate many outliers.

This means that small C favors a "softer" margin and therefore less overfitting, whereas large C means that the solution will attempt to adjust to the training data so as to minimize the number of outliers, leading to more overfitting. In summary, the amount of overfitting is directly proportional to the magnitude of C. Too small a C on the other hand may lead to *underfitting*, that is, too much slackness is allowed and the classifier does not fit the data at all.

Non-Separable Case Primal Problem

We need to encode the constraints in the functional, as in the separable case. This time there are $2n$ constraints, so there will be $2n$ Lagrange multipliers: λ_i and ρ_i, for $i = 1, \ldots, n$.

The primal functional can then be written as:

$$L_P(a, a_0, \xi, \lambda, \rho)$$

$$= \frac{1}{2} a^T a + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \lambda_i \left(y_i(a^T x_i + a_0) - 1 + \xi_i \right) - \sum_{i=1}^{n} \rho_i \xi_i$$

Non-Separable Case Dual Problem

Setting the derivatives of L_p with respect to a and a_0 to zero yields the same equations as in the separable case:

$$a = \sum_{i=1}^{n} \lambda_i y_i x_i \quad \text{and} \quad \sum_{i=1}^{n} \lambda_i y_i = 0 \tag{1}$$

Setting the derivatives of L_p with respect to ξ_i to zero yields

$$C - \lambda_i - \rho_i = 0, \quad i = 1, \ldots, n \tag{2}$$

Non-Separable Case Dual Problem - II

Substituting these equations back into L_P leads to the same expression for the dual Lagrangian functional as in the separable case:

$$L_D(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

which must be maximized with respect to λ_i. The first set of constraints come from equation (1)

$$\sum_{i=1}^{n} \lambda_i y_i = 0$$

Non-Separable Case Dual Problem - III

One also has the following equations, which are derived from equation (2) and the non-negativity of the lagrange multipliers $\lambda_i, \rho_i \geq 0$:

$$0 \leq \lambda_i \leq C, \quad i = 1, \ldots, n$$

The outliers are points for which $\xi_i > 0 \Rightarrow \rho_i = 0 \Rightarrow \lambda_i = C$. The points for which $0 < \lambda_i < C$ are called *margin vectors*.

The separable case corresponds to $C = \infty$. In that case, no outlier is possible and all support vectors ($\lambda_i > 0$) are margin vectors (the case $\lambda_i = C$ is not possible). For finite C, outliers are possible. If C is small, more of the constraints $\lambda_i \leq C$ may be active ($\lambda_i = C$), i.e., there can be more outliers, and vice-versa if C is large, which supports the conclusion that C controls overfitting.

Solution in Non-Separable Case

Once the optimal λ^* is found, the solution vector a^* is determined by (1)

$$a^* = \sum_{i=1}^{n} \lambda_i^* y_i x_i = \sum_{i \in \mathcal{S}} \lambda_i^* y_i x_i$$

where $\mathcal{S} = \{i \,|\lambda_i > 0\}$ is the support vector index set. The value of a_0^* can be determined from any of the active constraints $a^T x_i + a_0 = y_i$ with $\xi_i = 0$, that is, the constraints for which $0 < \lambda_i < C$ (the margin vectors), or from their sum:

$$n_m a_0 + (a^*)^T \sum_{i \in \mathcal{S}_m} x_i = \sum_{i \in \mathcal{S}_m} y_i$$

Solution in Non-Separable Case - II

In the previous equation, $\mathcal{S}_m = \{i \,|0 < \lambda_i < C\}$ is the margin vector index set, and $n_m = \text{Card}(\mathcal{S}_m)$ is the number of margin vectors.

From this it follows that:

$$a_0^* = -\frac{1}{n_m} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}_m} \lambda_i^* y_i x_i^T x_j + \frac{1}{n_m} \sum_{i \in \mathcal{S}_m} y_i$$

The optimal discriminant is thus given by

$$(a^*)^T x + a_0^* = \sum_{i \in \mathcal{S}} \lambda_i^* y_i x_i^T x + a_0^* \geq 0$$

Nonlinear Support Vector Machines

Idea: Find transformation $\phi(x)$ that takes the original feature space into a higher-dimensional space, and apply the MMH algorithm there. The corresponding classifier back in the original space will generally be non-linear.

The entire previous derivation goes through with $\phi(x)$ in place of x.

Thus, the nonlinear SVM classifier is given by $\psi_n(x) = 1$ if

$$\sum_{i \in \mathcal{S}} \lambda_i^* y_i \phi^T(x_i)\phi(x) - \frac{1}{n_s} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} \lambda_i^* y_i \phi^T(x_i)\phi(x_j) + \frac{1}{n_s} \sum_{i \in \mathcal{S}} y_i > 0$$

with $\psi_n(x) = 0$ otherwise.

Kernel “Trick”

Let us introduce a *Kernel function*

$$K(x, y) = \phi^T(x)\phi(y)$$

We can avoid computing $\phi(x)$ completely by using $k(x, y)$.

The dual functional can be written as:

$$L_D(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j k(x_i, x_j)$$

which must be maximized with respect to λ, subject to

$$\lambda_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \lambda_i y_i = 0$$

SVM Kernel Classifier

Once the solution is found, the SVM kernel classifier is:

$$\psi_n(x) = \begin{cases} 1, & \sum_{i \in \mathcal{S}} \lambda_i^* y_i k(x_i, x) + a_0^* > 0 \\ 0, & \text{otw} \end{cases}$$

where

$$a_0^* = -\frac{1}{n_s} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} \lambda_i^* y_i k(x_i, x_j) + \frac{1}{n_s} \sum_{i \in \mathcal{S}} y_i$$

We are back to kernel classification, but a peculiar one:

- Only a subset of the training set, the support vectors, have any influence.
- The kernel influence is weighted by the corresponding support vector sensitivity λ_i^*.

Mercer's Condition

To be admissible, a kernel must be expressible as an inner product in a feature space (which can be an infinite-dimensional space). It can be shown that this is true if and only if

- K is symmetric: $K(x, y) = K(y, x)$
- K is positive semi-definite:

$$\int K(x, y) g(x) g(y) \, dx \, dy \geq 0$$

for any function g satisfying:

$$\int g^2(x) \, dx < \infty$$

Examples of Kernel

Some examples of kernels used in applications:

- Polynomial: $k(x, y) = (1 + x^T y)^p$
- Gaussian: $k(x, y) = \exp(-|x - y|^2/\sigma^2)$
- Sigmoid: $k(x, y) = \tanh(kx^T y - \delta)$

It can be shown that in each case above $k(x, y)$ satisfies Mercer's condition (for the sigmoid kernel, this will be true for certain values of k and δ), so that $k(x, y) = \phi^T(x)\phi(y)$ for a suitable mapping ϕ.

Example: XOR Problem

This is the simplest non-linearly separable problem (with minimal number of points).

The data set consists of:

- Class 0: $\{(-1, 1), (1, -1)\}$
- Class 1: $\{(-1, -1), (1, 1)\}$

We consider the polynomial kernel of order 2:

$$\begin{aligned} k(x, y) &= (1 + x^T y)^2 = (1 + x_1 y_1 + x_2 y_2)^2 \\ &= 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2 \\ &= \phi^T(x)\phi(y) \end{aligned}$$

where $\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2)$.

Example: XOR Problem - II

Note that the dual functional (for any nonlinear SVM problem) can be written compactly as:

$$L_D(\lambda) = \lambda^T \mathbf{1} - \frac{1}{2}\lambda^T H \, \lambda$$

where $H_{ij} = y_i y_j K(x_i, x_j)$.

Setting

$$\frac{\partial L_D}{\partial \lambda_i} = 0$$

gives a system of n equations in the n variables λ_i (again this is true for any nonlinear SVM problem):

$$H\lambda = 1$$

Example: XOR Problem - III

In the case of the XOR problem, it is easy to see that

$$H = \begin{bmatrix} 9 & -1 & -1 & 1 \\ -1 & 9 & 1 & -1 \\ -1 & 1 & 9 & -1 \\ 1 & -1 & -1 & 9 \end{bmatrix}$$

Leading to the following system of equations

$$\begin{aligned} 9\lambda_1 - \lambda_2 - \lambda_3 + \lambda_4 &= 1 \\ -\lambda_1 + 9\lambda_2 + \lambda_3 - \lambda_4 &= 1 \\ -\lambda_1 + \lambda_2 + 9\lambda_3 - \lambda_4 &= 1 \\ \lambda_1 - \lambda_2 - \lambda_3 + 9\lambda_4 &= 1 \end{aligned}$$

Example: XOR Problem - IV

The solution to these equations is $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{8}$. Because matrix H is positive definite, the functional $L_D(\lambda)$ is strictly concave and the condition $\frac{\partial L_D}{\partial \lambda_i} = 0$ is necessary and sufficient for an unconstrained global maximum. But since the vector $\lambda = (\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8})^T$ is a point in the feasible region,

$$\lambda_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \lambda_i y_i = 0\,,$$

this is in fact the solution λ^* to the dual problem (note that all 4 training points are support vectors).

Example: XOR Problem - V

The optimal solution is given by:

$$a^* = \sum_{i=1}^{4} \lambda_i^* y_i \phi(x_i) = (0, 0, 0, \frac{1}{\sqrt{2}}, 0, 0)^T$$

and

$$a_0^* = -\frac{1}{4} \sum_{i=1}^{4} \sum_{j=1}^{4} \lambda_i^* y_i k(x_i, x_j) + \frac{1}{4} \sum_{i=1}^{4} y_i = 0$$

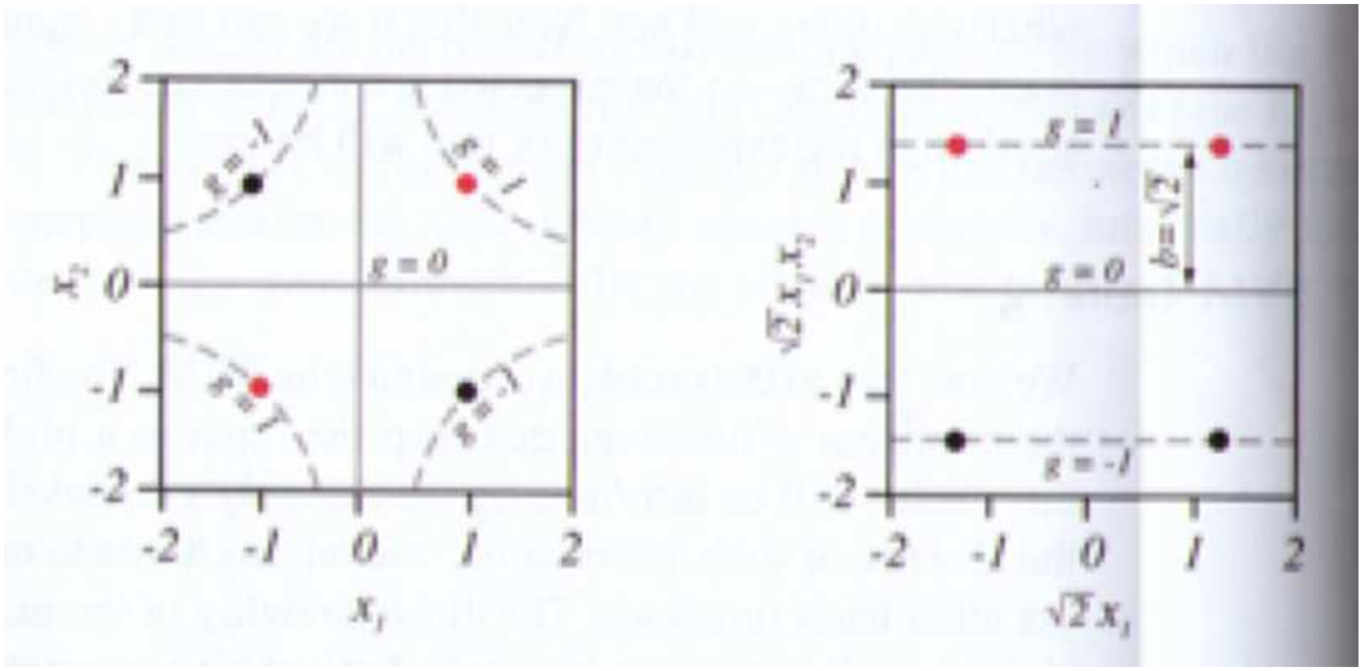
so that the MMH in transformed space is determined by:

$$(a^*)^T \phi(x) + a_0^* = x_1 x_2 = 0$$

with optimal margin $\frac{1}{||a||} = \sqrt{2}$.

Example: XOR Problem - VI

The classifier in the original and transformed feature spaces is depicted below.



General Case

In the general case, one both uses a mapping to a higher-dimensional spaces and allows slackness.

The dual functional is written as:

$$L_D(\lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j k(x_i, x_j)$$

which must be maximized with respect to λ, subject to

$$0 \leq \lambda_i \leq C \quad \text{and} \quad \sum_{i=1}^{n} \lambda_i y_i = 0$$

where C is the regularization parameter.

General SVM Kernel Classifier

Once the solution is found, the SVM kernel classifier is:

$$\psi_n(x) = \begin{cases} 1, & \sum_{i \in \mathcal{S}} \lambda_i^* y_i k(x_i, x) + a_0^* > 0 \\ 0, & \text{otw} \end{cases}$$

where

$$a_0^* = -\frac{1}{n_m} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}_m} \lambda_i^* y_i k(x_i, x_j) + \frac{1}{n_m} \sum_{i \in \mathcal{S}_m} y_i$$

where $\mathcal{S}_m = \{i \,|0 < \lambda_i < C\}$ is the margin vector index set, and $n_m = \text{Card}(\mathcal{S}_m)$ is the number of margin vectors (i.e., those vectors for which $0 < \lambda_i < C$).