Support vector machines

Course of Machine Learning
Master Degree in Computer Science
University of Rome "Tor Vergata"
a.a. 2024-2025

Giorgio Gambosi

Idea

The binary classification problem is approached in a direct way, that is we try and find a plane that separates the classes in feature space (indeed, a "best" plane, according to a reasonable characteristic). If this is not possible, we get creative in two ways:

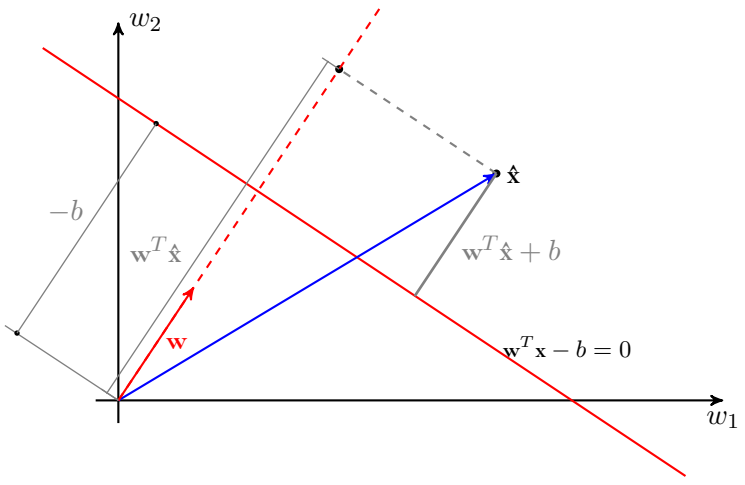
- We soften what we mean by "separates", and/or
- We enrich and enlarge the feature space so that separation is (more) possible

First of all, we remark that most classification algorithms output not just a class label but a score, and the predicted label is assigned on the basis of such score, which is compared to a threshold. In linear classification, the score is computed as a linear combination $\mathbf{w}^T\mathbf{x} + b$ of the features.

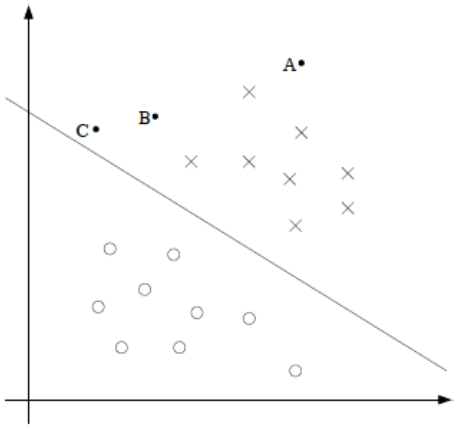
For example, logistic regression returns the class probability

$$p(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b) \in [0, 1]$$

and me may follow the strategy of predicting class 1 if the class probability is > 0.5. Essentially, the score is an estimate of confidence: the farthest it is from the threshold, the most we are confident about the prediction. In geometric terms, the score also represents how far the point corresponding to the given item lies from the decision boundary, which is the hyperplane $\mathbf{w}^T\mathbf{x} - b = 0$



For example, in the following case,



A can be assigned to $\mathcal{C}_1$ with greater confidence than B and even greater confidence than C.

These considerations, when applied to the training phase, result into preferring decision boundaries such that all items in the training set are correctly classified (they lie in the correct decision region) and referring to that boundary makes us confident of such classification (they lie far enough from the boundary).

Intuitively, we want to select boundaries which correctly classify the items in the training set are characterized by a space **margin** around them that does not contain any item.

More in detail, consider a binary classifier which, for any element $\mathbf{x}$, returns a value $y \in \{-1, 1\}$, where we assume that $\mathbf{x}$ is assigned to $\mathcal{C}_0$ if $y = -1$ and to $\mathcal{C}_1$ if $y = 1$.

Moreover, we consider linear classifiers such as

$$h(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + b) = g(\overline{\mathbf{w}}^T\overline{\mathbf{x}})$$

where $g(z) = 1$ if $z \geq 0$ and $g(z) = -1$ if $z < 0$. The prediction on the class of $\mathbf{x}$ is then provided by deriving a value in $\{-1, 1\}$ just as in the case of a perceptron, that is with no estimation of the probabilities $p(\mathcal{C}_i|\mathbf{x})$ that $\mathbf{x}$ belongs to each class.

For any training set item $(\mathbf{x}_i, t_i)$, the **functional margin** of $(\mathbf{w}, b) = \overline{\mathbf{w}}$ with respect to such item is defined as

$$\overline{\gamma}_i = t_i(\mathbf{w}^T\mathbf{x}_i + b) = t_i\overline{\mathbf{w}}^T\overline{\mathbf{x}}_i$$

Observe that the resulting prediction is correct iff $\overline{\gamma}_i > 0$. Moreover, we assume that larger values of $\overline{\gamma}_i$ denote greater confidence on the prediction.

Given a training set $\mathcal{T} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)\}$ the functional margin of $\overline{\mathbf{w}}$ wrt $\mathcal{T}$ is the minimum functional margin for all items in $\mathcal{T}$

$$\overline{\gamma} = \min_i \overline{\gamma}_i$$

However, we have a problem with this definition:

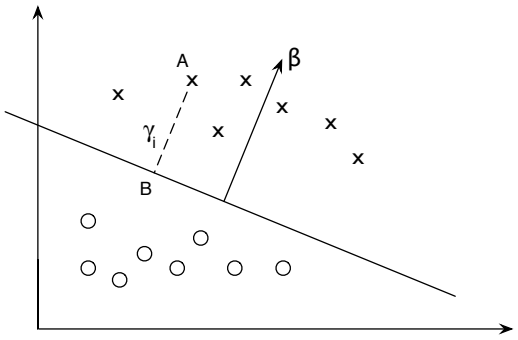
- If we rescale the parameters $\overline{\mathbf{w}}$, by a scalar $\alpha > 0$, we get new parameters $\alpha\overline{\mathbf{w}}$, that is $\alpha\mathbf{w}, \alpha b$.
- Using $\alpha\overline{\mathbf{w}}$ doesn't change the classification of points.
- However, the margin $\alpha\overline{\mathbf{w}}^T\overline{\mathbf{x}}_i = \alpha\mathbf{w}^T\mathbf{x}_i + \alpha b$ is now scaled by α.

It doesn't make sense that the same classification boundary may have different margins when we rescale it.

In order to overcome this inconvenience, we define the **geometric margin** γ_i with respect to a training example $(\mathbf{x}_i, t_i)$ as

$$\gamma_i = t_i\left(\frac{\mathbf{w}^T}{||\mathbf{w}||}\mathbf{x}_i + \frac{b}{||\mathbf{w}||}\right) = \frac{\overline{\gamma}_i}{||\mathbf{w}||}$$

where, as usual, $||\mathbf{w}|| = \sqrt{\sum_{i=1}^{d} w_i^2}$. The geometric margin γ_i equals the (signed) distance between $\mathbf{x}_i$ and the hyperplane defined by $\mathbf{w}, b$, that is as the length of the line segment from $\mathbf{x}_i$ to its projection on the boundary hyperplane, expressed using $||\mathbf{w}||$ as distance unit.



Differently from $\overline{\gamma}_i$, the geometric margin γ_i is invariant with respect to parameter scaling. In fact, by substituting $\alpha\overline{\mathbf{w}}$ to $\overline{\mathbf{w}}$, we have

$$t_i \overline{\mathbf{w}}^T \overline{\mathbf{x}}_i = t_i(\mathbf{w}^T \mathbf{x}_i + b) = \overline{\gamma}_i$$
$$t_i (\alpha\overline{\mathbf{w}})^T \overline{\mathbf{x}}_i = t_i(\alpha \mathbf{w}^T \mathbf{x}_i + \alpha b) = \alpha t_i(\mathbf{w}^T \mathbf{x}_i + b) = \alpha(t_i \overline{\mathbf{w}}^T \overline{\mathbf{x}}_i) = \alpha\overline{\gamma}_i$$

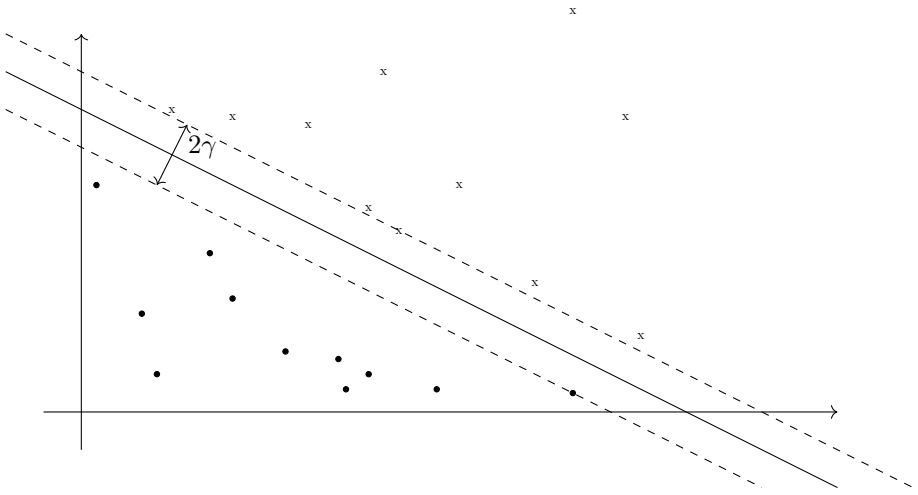
while

$$t_i \frac{\overline{\mathbf{w}}^T}{||\mathbf{w}||} \overline{\mathbf{x}}_i = t_i \left(\frac{\mathbf{w}^T}{||\mathbf{w}||} \mathbf{x}_i + \frac{b}{||\mathbf{w}||} \right) = \gamma_i$$
$$t_i \frac{(\alpha\overline{\mathbf{w}})^T}{||(\alpha\mathbf{w})||} \overline{\mathbf{x}}_i = t_i \left(\frac{\alpha\mathbf{w}^T}{\alpha\,||\mathbf{w}||} \mathbf{x}_i + \frac{\alpha b}{\alpha\,||\mathbf{w}||} \right) = t_i \left(\frac{\mathbf{w}^T}{||\mathbf{w}||} \mathbf{x}_i + \frac{b}{||\mathbf{w}||} \right) = \gamma_i$$

- The geometric margin wrt the training set $\mathcal{T} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)\}$ is then defined as the smallest geometric margin for all items $(\mathbf{x}_i, t_i)$

$$\gamma = \min_i \gamma_i$$

- a useful interpretation of γ is as half the width of the largest strip, centered on the hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$, containing none of the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
- the hyperplanes on the boundary of such strip, each at distance γ from the hyperplane and passing (at least one of them) through some point $\mathbf{x}_i$ are said **maximum margin hyperplanes**.



Optimal margin classifiers

Given a training set $\mathcal{T}$, we wish to find the hyperplanes which separates the two classes (if one does exist) and has maximum γ: by making the distance between the hyperplanes and the set of points corresponding to elements as large as possible, the confidence on the provided classification increases.

Assume classes are linearly separable in the training set: hence, there exists a hyperplane (an infinity of them, indeed) separating elements in C_1 from elements in C_2. In order to find the one among those hyperplanes which maximizes γ, we have to solve the following optimization problem

$$\begin{aligned} &\max_{\mathbf{w},b} \gamma \\ &\text{where } \gamma_i = \frac{t_i}{||\mathbf{w}||}\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq \gamma \qquad i = 1, \ldots, n \end{aligned}$$

which maximizes the smallest margin over the training set $(\mathbf{x}_i, t_i)$. It guarantees each point has margin at least γ.

This corresponds to,

$$\begin{aligned} &\max_{\mathbf{w},b} \gamma \\ &\text{where } t_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq \gamma\,||\mathbf{w}|| \qquad i = 1, \ldots, n \end{aligned}$$

As observed, if all parameters are scaled by any constant α, all geometric margins γ_i between elements and hyperplane are unchanged: we may then exploit this freedom to introduce the constraint

$$\gamma = \min_i \mathbf{w}^T\mathbf{x}_i t_i = 1$$

This can be obtained by assuming $||\mathbf{w}|| = \frac{1}{\gamma}$, which corresponds to considering a scale where the maximum margin has width 2. This results, for each element $\mathbf{x}_i, t_i$, into a constraint

$$\gamma_i = t_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1$$

An element (point) is said **active** if the equality holds, that is if

$$(\mathbf{w}^T\mathbf{x}_i + b)t_i = 1$$

and **inactive** if this does not hold. Observe that, by definition, there must exist at least one active point.

For any item $(\mathbf{x}, t)$, different cases correspond to the value of the signed linear combination $t(\mathbf{w}^T\mathbf{x} + b)$:

1. $t(\mathbf{w}^T\mathbf{x} + b) > 1 \Rightarrow \mathbf{x}$ is in the region corresponding to its class, outside the margin strip
2. $t(\mathbf{w}^T\mathbf{x} + b) = 1 \Rightarrow \mathbf{x}$ is in the region corresponding to its class, on the maximum margin hyperplane
3. $0 < t(\mathbf{w}^T\mathbf{x} + b) < 1 \Rightarrow \mathbf{x}$ is in the region corresponding to its class, inside the margin strip
4. $\mathbf{w}^T\mathbf{x} + b = 0 \Rightarrow \mathbf{x}$ is on the separating hyperplane
5. $-1 < t(\mathbf{w}^T\mathbf{x} + b) < 0 \Rightarrow \mathbf{x}$ is in the region corresponding to the other class, inside the margin strip
6. $t(\mathbf{w}^T\mathbf{x} + b) = -1 \Rightarrow \mathbf{x}$ is in the region corresponding to the other class, on the maximum margin hyperplane
7. $t(\mathbf{w}^T\mathbf{x} + b) < -1 \Rightarrow \mathbf{x}$ is in the region corresponding to the other class, outside the margin strip

The optimization problem, is then transformed into

$$\max_{\mathbf{w},b} \gamma = ||\mathbf{w}||^{-1}$$
$$\text{where } t_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 \qquad i = 1, \ldots, n$$

Maximizing $||\overline{\mathbf{w}}||^{-1}$ is equivalent to minimizing $||\overline{\mathbf{w}}||^2$: hence we may formulate the problem as

$$\min_{\mathbf{w},b} \frac{1}{2} ||\mathbf{w}||^2$$
$$\text{where } t_i \left(\mathbf{w}^T \mathbf{x}_i + b\right) \geq 1 \qquad i = 1, \ldots, n$$

This is a **convex quadratic** optimization problem. The function to be minimized is in fact convex and the set of points satisfying the constraint is a convex polyhedron (intersection of half-spaces).

Duality

From optimization theory it derives that, given the problem structure (linear constraints + convexity):

- there exists a **dual formulation** of the problem
- the optimum of the dual problem is the same of the original (**primal**) problem

Lagrangian

Consider the optimization problem

$$\min_{\mathbf{x} \in \Omega} f(\mathbf{x})$$

where Ω is the feasible region, defined by the constraints

$$g_i(\mathbf{x}) \leq 0 \qquad i = 1, \ldots, k$$

where $f(\mathbf{x})$, $g_i(\mathbf{x})$are convex functions and Ω is a convex set.

The **Lagrangian** is defined as

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x})$$

Consider the maximum wrt to non negative λ of the lagrangian

$$\max_{\lambda} L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \max_{\lambda} \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x})$$
$$\lambda_i \geq 0 \qquad i = 1, \ldots, k$$

which is a function of $\mathbf{x}$.

- if $\mathbf{x}$ is a feasible solution, $g_i(\mathbf{x}) \leq 0$ for all i and the maximum is obtained for $\lambda_i = 0$: as a consequence,

$$\max_{\lambda : \lambda_i \geq 0} L(\mathbf{x}, \lambda) = f(\mathbf{x})$$

- if $\mathbf{x}$ is an unfeasible solution, then $g_i(\mathbf{x}) > 0$ for some i and the maximum is unbounded, as λ_i can be arbitrarily great

As a consequence, the maximum of the lagrangian is equal to $f(\mathbf{x})$ if $\mathbf{x}$ is feasible, while it is unbounded if $\mathbf{x}$ is not feasible. This results, in the case that a minimum $\mathbf{x}^*$ exists, into

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) = \min_{\mathbf{x}\in\Omega} \max_{\lambda:\lambda_i\geq 0} L(\mathbf{x},\lambda)$$

In general, the **weak duality property** holds

$$\max_{\lambda:\lambda_i\geq 0} \min_{\mathbf{x}\in\Omega} L(\mathbf{x},\lambda) \leq \min_{\mathbf{x}\in\Omega} \max_{\lambda:\lambda_i\geq 0} L(\mathbf{x},\lambda) = \min_{\mathbf{x}\in\Omega} f(\mathbf{x})$$

where $\max_{\lambda:\lambda_i\geq 0} \min_{\mathbf{x}\in\Omega} L(\mathbf{x},\lambda)$ is the **dual** problem of $\min_{\mathbf{x}\in\Omega} f(\mathbf{x})$

Moreover, in the case of convex optimization (our case here) the strong duality property holds

$$\max_{\lambda:\lambda_i\geq 0} \min_{\mathbf{x}\in\Omega} L(\mathbf{x},\lambda) = \min_{\mathbf{x}\in\Omega} \max_{\lambda:\lambda_i\geq 0} L(\mathbf{x},\lambda) = \min_{\mathbf{x}\in\Omega} f(\mathbf{x})$$

Karush-Kuhn-Tucker conditions

The following necessary and sufficient conditions hold at the optimum $(\mathbf{x}^*, \lambda^*)$, and can be used to simplify the dual problem definition.

$$\begin{aligned}
\nabla_{\mathbf{x}} L(\mathbf{x},\lambda)\Big|_{\mathbf{x}^*,\lambda^*} &= \mathbf{0} \\
\frac{\partial L(\mathbf{x},\lambda)}{\partial \lambda_i}\Big|_{\mathbf{x}^*,\lambda^*} &= g_i(\mathbf{x}^*) \geq 0 && i = 1,\dots,k \\
\lambda_i^* &\geq 0 && i = 1,\dots,k \\
\lambda_i^* g_i(\mathbf{x}^*) &= 0 && i = 1,\dots,k
\end{aligned}$$

In order for the optimum to be a minimum, the second order condition must hold that the Hessian $H_{\mathbf{x}}$ evaluated at $\mathbf{x}^*$ must be positive definite.

Note: the last condition (**complementary slackness**) states that a Lagrangian multiplier λ_i^* can be non-zero only if $g_i(\mathbf{x}^*) = 0$, that is of $\mathbf{x}^*$ is"at the limit" for the constraint $g_i(\mathbf{x}) \leq 0$. In this case, the constraint is said **active**.

Application to SVM

In our case,

- $f(\mathbf{x})$ corresponds to $\frac{1}{2}||\mathbf{w}||^2$
- $g_i(x)$ corresponds to $t_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 \geq 0$
- Ω is the intersection of a set of hyperplanes, that is a polyhedron, hence convex.

The corresponding Lagrangian is

$$\begin{aligned}
L(\overline{\mathbf{w}},\lambda) &= \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \lambda_i \left((\mathbf{w}^T\mathbf{x}_i + b)t_i - 1\right) \\
&= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \lambda_i \mathbf{w}^T\mathbf{x}_i t_i - b\sum_{i=1}^{n} \lambda_i t_i + \sum_{i=1}^{n} \lambda_i
\end{aligned}$$

and the dual problem (with same optimum) is

$$\max_{\lambda} \min_{\overline{\mathbf{w}}} L(\overline{\mathbf{w}}, \lambda)$$
$$\lambda_i \geq 0 \qquad i = 1, \ldots, k$$

Applying the KKT conditions

In order to state the dual problem as a function of λ, let us derive the values of the coefficients $\overline{\mathbf{w}}$ at the optimum of $L(\overline{\mathbf{w}}, \lambda)$, by applying the KKT conditions

First of all, we compute:

$$\frac{\partial L(\overline{\mathbf{w}}, \lambda)}{\partial w_k}\Big|_{\mathbf{w}^*, b^*} = w_k^* - \sum_{i=1}^{n} \lambda_i t_i x_{ik} = 0$$
$$\frac{\partial L(\overline{\mathbf{w}}, \lambda)}{\partial b}\Big|_{\mathbf{w}^*, b^*} = \sum_{i=1}^{n} \lambda_i t_i = 0$$

that is,

$$\nabla_{\mathbf{w}} L(\overline{\mathbf{w}}, \lambda)\Big|_{\mathbf{w}^*, b^*} = \mathbf{w}^* - \sum_{i=1}^{n} \lambda_i t_i \mathbf{x}_i = 0$$
$$\frac{\partial L(\overline{\mathbf{w}}, \lambda)}{\partial b}\Big|_{\mathbf{w}^*, b^*} = \sum_{i=1}^{n} \lambda_i t_i = 0$$

The resulting KKT conditions are

$$\mathbf{w}^* = \sum_{i=1}^{n} \lambda_i t_i \mathbf{x}_i$$
$$\sum_{i=1}^{n} \lambda_i t_i = 0$$
$$t_i(\mathbf{w}^{*T}\mathbf{x}_i + b^*) - 1 \geq 0 \qquad i = 1, \ldots, n$$
$$\lambda_i \geq 0 \qquad i = 1, \ldots, n$$
$$\lambda_i \left(t_i(\mathbf{w}^{*T}\mathbf{x}_i + b^*) - 1 \right) = 0 \qquad i = 1, \ldots, n$$

By substituting the values of the coefficients $\mathbf{w}^*$ according to the above equations and observing that b^* is multiplied by zero, , we may state that at the optimum $\overline{\mathbf{w}}^*$ the dual problem asks to maximize

$$
\begin{aligned}
L(\lambda) &= \frac{1}{2}\mathbf{w}^{*T}\mathbf{w}^* - \sum_{i=1}^{n}\lambda_i t_i \mathbf{w}^{*T}\mathbf{x}_i - b^*\sum_{i=1}^{n}\lambda_i t_i + \sum_{i=1}^{n}\lambda_i \\
&= \frac{1}{2}\left(\sum_{i=1}^{n}\lambda_i t_i \mathbf{x}_i\right)^T \sum_{j=1}^{n}\lambda_j t_j \mathbf{x}_j - \sum_{i=1}^{n}\lambda_i t_i \mathbf{x}_i^T \sum_{j=1}^{n}\lambda_j t_j \mathbf{x}_j + \sum_{i=1}^{n}\lambda_i \\
&= \frac{1}{2}\sum_{i=1}^{n}\lambda_i t_i \mathbf{x}_i^T \sum_{j=1}^{n}\lambda_j t_j \mathbf{x}_j - \sum_{i=1}^{n}\lambda_i t_i \mathbf{x}_i^T \sum_{j=1}^{n}\lambda_j t_j \mathbf{x}_j + \sum_{i=1}^{n}\lambda_i \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j t_i \mathbf{x}_i^T t_j \mathbf{x}_j + \sum_{i=1}^{n}\lambda_i \\
&= \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j
\end{aligned}
$$

with the constraints on λ

$$
\begin{aligned}
\sum_{i=1}^{n}\lambda_i t_i &= 0 \\
\lambda_i &\geq 0 \qquad\qquad i = 1, \ldots, n
\end{aligned}
$$

The remaining two KKT conditions can be proved to be always verified.

Dual SVM problem

We have modified the definition of the dual problem by applying the KKT conditions to drop the occurrences of coefficients $\mathbf{w}, b$ from $L(\mathbf{w}, b, \lambda)$.

The new problem has the same optimum of the original primal, where the KKT conditions will indeed hold, connecting the values of the optimal solutions of the two problems

$$
\begin{aligned}
\max_{\lambda} L(\lambda) &= \max_{\lambda}\left(\sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j t_i t_j \mathbf{x}_i^T \mathbf{x}_j\right) \\
\lambda_i &\geq 0 \qquad i = \ldots, n \\
\sum_{i=1}^{n}\lambda_i t_i &= 0
\end{aligned}
$$

All the considerations above clearly hold if we assume a set of base functions $\boldsymbol{\phi}$ is applied, thus resulting into the dual problem

$$
\begin{aligned}
\max_{\lambda} L(\lambda) &= \max_{\lambda}\left(\sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j t_i t_j \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j)\right) \\
\lambda_i &\geq 0 \qquad i = \ldots, n \\
\sum_{i=1}^{n}\lambda_i t_i &= 0
\end{aligned}
$$

By defining the kernel function

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j)$$

the dual problem's formulation can be given as

$$\begin{aligned} \max_{\lambda} L(\lambda) &= \max_{\lambda} \left(\sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j t_i t_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right) \\ \lambda_i &\geq 0 \qquad i = 1, \ldots, n \\ \sum_{i=1}^{n} \lambda_i t_i &= 0 \end{aligned}$$

Passing from primal to dual

Disadvantage The number variables increases from m to n (in particular, if $\mathbf{x} = \mathbf{x}$, from d to n).

Advantage The number of variables to be considered, which are relevant for classification, turns out to be quite smaller than n.

Deriving coefficients

By solving the dual problem, the optimal values of Langrangian multipliers λ^* are obtained.

The optimal values of parameters $\mathbf{w}^*$ are then derived through the relations

$$w_i^* = \sum_{j=1}^{n} \lambda_j^* t_j \phi_i(\mathbf{x}_j) \qquad\qquad i = 1, \ldots, m$$

The value of b^* can be obtained by observing that, for any support vector $\mathbf{x}_k$ (characterized by the condition $\lambda_k \geq 0$), it must be

$$\begin{aligned} 1 = t_k \left(\mathbf{w}^{*T} \boldsymbol{\phi}(\mathbf{x}_k) + b^* \right) &= t_k \left(\sum_{j=1}^{n} \lambda_j^* t_j \boldsymbol{\phi}(\mathbf{x}_j)^T \boldsymbol{\phi}(\mathbf{x}_k) + b^* \right) \\ = t_k \left(\sum_{j=1}^{n} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_k) + b^* \right) &= t_k \left(\sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_k) + b^* \right) \end{aligned}$$

where $\mathcal{S}$ is the set of indices of support vectors.

As a consequence, since $t_k = \pm 1$, in order to have a unitary product it must be

$$t_k = \sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_k) + b^*$$

and

$$b^* = t_k - \sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_k)$$

A more precise solution can be obtained as the mean value obtained considering all support vectors

$$b^* = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(t_i - \sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_i) \right)$$

Classification through SVM

A new element $\mathbf{x}$ can be classified, given a set of base functions $\boldsymbol{\phi}$ or a kernel function κ, by checking the sign of

$$y(\mathbf{x}) = \sum_{i=1}^{m} w_i^* \phi_i(\mathbf{x}) + b^* = \sum_{j=1}^{n} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}) + b^*$$

As observed, if $\mathbf{x}_i$ is not a support vector, then it must be $\lambda_i^* = 0$. Thus, the above sum can be written as

$$y(\mathbf{x}) = \sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}) + b^*$$

The classification performed through the dual formulation, using the kernel function, does not take into account all training set items, but only support vectors, usually a quite small subset of the training set.

Non separability in the training set

- The approach described before, when applied to non linearly separable sets, does not provide acceptable solutions: it is in fact impossibile to satisfy all constraints

$$t_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 \qquad i = 1, \ldots, n$$

- These constraints must then be relaxed in order to allow them to not hold, at the cost of some increase in the objective function to be minimized
- A **slack variable** ξ_i is introduced for each constraint, to provide a measure of how much the constraint is not verified

This can be formalized as

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \; & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \xi_i \\ & t_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1 - \xi_i \qquad i = 1, \ldots, n \\ & \xi_i \geq 0 \qquad i = 1, \ldots, n \end{aligned}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$

- By introducing suitable multipliers, the following Lagrangian can be obtained

$$\begin{aligned} & L(\mathbf{w}, b, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha}) = \\ & = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \lambda_i (y_i(\mathbf{w}^T \boldsymbol{\phi}(x_i) + b) - 1 + \xi_i) - \sum_{i=1}^{n} \alpha_i \xi_i \\ & = \frac{1}{2}\sum_{i=1}^{n} w_i^2 + \sum_{i=1}^{n} (C - \alpha_i)\xi_i - \sum_{i=1}^{n} \lambda_i (t_i (\sum_{j=1}^{m} w_j \phi_j(\mathbf{x}_i)) + b) - 1 + \xi_i) \\ & = \frac{1}{2}\sum_{i=1}^{n} w_i^2 + \sum_{i=1}^{n} (C - \alpha_i - \lambda_i)\xi_i - \sum_{i=1}^{n}\sum_{j=1}^{m} \lambda_i t_i w_j \phi_j(\mathbf{x}_i) + b\sum_{i=1}^{n} \lambda_i t_i + \sum_{i=1}^{n} \lambda_i \end{aligned}$$

where $\alpha_i \geq 0$ and $\lambda_i \geq 0$, for $i = 1 \ldots, n$.

The Karush-Kuhn-Tucker conditions are now:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha}) &= \mathbf{0} && && \text{null gradient} \\
\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha}) &= 0 && && \text{null gradient} \\
\frac{\partial}{\partial \boldsymbol{\xi}} L(\mathbf{w}, b, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha}) &= \mathbf{0} && && \text{null gradient} \\
t_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b) - 1 + \xi_i &\geq 0 && i = 1, \ldots, n && \text{constraints} \\
\xi_i &\geq 0 && i = 1, \ldots, n && \text{constraints} \\
\lambda_i &\geq 0 && i = 1, \ldots, n && \text{multipliers} \\
\alpha_i &\geq 0 && i = 1, \ldots, n && \text{multipliers} \\
\lambda_i \left(t_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b) - 1 + \xi_i \right) &= 0 && i = 1, \ldots, n && \text{complementary slackness} \\
\alpha_i \xi_i &= 0 && i = 1, \ldots, n && \text{complementary slackness}
\end{aligned}$$

Deriving a dual formulation

From the null gradient conditions wrt w_i, b, ξ_j it derives

$$\begin{aligned}
w_i &= \sum_{j=1}^{n} \lambda_j t_j \phi_i(\mathbf{x}_j) \qquad i = 1, \ldots, m \\
0 &= \sum_{i=1}^{n} \lambda_i t_i \\
\lambda_i &= C - \alpha_i \leq C \qquad i = 1, \ldots, n
\end{aligned}$$

By plugging the above relations into $L(\overline{\mathbf{w}}, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha})$, the dual problem results

$$\begin{aligned}
\max_{\lambda} \tilde{L}(\lambda) = \max_{\lambda} & \left(\sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j t_i t_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right) \\
0 \leq \lambda_i &\leq C \qquad i = 1, \ldots, n \\
\sum_{i=1}^{n} \lambda_i y_i &= 0
\end{aligned}$$

Observe that the only difference wrt the linearly separable case is given by constraints $0 \leq \lambda_i$ transformed into in $0 \leq \lambda_i \leq C$

Classification

From the optimal solution λ^* of the dual problem, the coefficients $\mathbf{w}^*$ and b^* can be derived just as done in the linearly separable case.

A new element $\mathbf{x}$ can then be classified, again, through the sign of

$$y(\mathbf{x}) = \sum_{i=1}^{m} w_i^* \phi_i(\mathbf{x}) + b^*$$

or, equivalently, of

$$y(\mathbf{x}) = \sum_{i \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + b^*$$

Extensions

The approach can be extended to

- More than 2 classes (multiclass classification): solve one vs all binary classification problem for all classes
- Real-valued outputs (support vector regression)

Computational issues

- Training time of the standard SVM is $O(n^3)$ (solving QP)
  - Can be prohibitive for large datasets
- Lots of research has gone into speeding up the SVMs
  - Many approximate QP solvers are used to speed up SVMs
  - Gradient descent faster and with possibility of limiting the computation time

SVM and gradient descent

Recall the formalization of the problem in the general case

$$\begin{aligned} \min_{\mathbf{w}, \boldsymbol{\xi}} \; & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \xi_i \\ & t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1 - \xi_i \qquad i = 1, \ldots, n \\ & \xi_i \geq 0 \qquad i = 1, \ldots, n \end{aligned}$$

Given $\mathbf{w}, b$, the slack variable ξ_i is minimized as

$$\xi_i = \begin{cases} 0 & t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1 \\ 1 - t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + b) & \text{otherwise} \end{cases}$$

The optimal value of ξ_i corresponds to the **hinge loss** of the corresponding item

$$L_H(\mathbf{w}, b, \mathbf{x}_i, t_i) = \max\left(0, 1 - t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + b)\right)$$

We may then define the cost function to be minimized as

$$\begin{aligned} C(\mathbf{w}) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} L_H(\mathbf{w}, b, \mathbf{x}_i, t_i) \\ &\propto \sum_{i=1}^{n} L_H(\mathbf{w}, b, \mathbf{x}_i, t_i) + \frac{1}{2C}||\mathbf{w}||^2 \end{aligned}$$

That is, SVM correspond to hinge loss with ridge regularization

Since hinge loss is not differentiable a $x = 1$, as discussed above, subgradient descent can be applied to iteratively find the optimal solution, with

$$\frac{\partial L_H}{\partial w_i} = w_i - \sum_{\mathbf{x}_k \in L} t_k \phi_i(\mathbf{x}_k)$$

where $x_k \in L$ iff $t_k(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_k) + b) < 1$.

The resulting iteration is

$$w_i^{(r+1)} = w_i^{(r)} - \eta w_i^{(r)} + \eta \sum_{\mathbf{x}_k \in L} t_k \phi_i(\mathbf{x}_k) = (1-\eta) w_i^{(r)} + \alpha \sum_{\mathbf{x}_k \in L} t_k \phi_i(\mathbf{x}_k)$$

SVM and SGD

In stochastic gradient descent, single items are considered at each iteration. This results in the following update rule, assuming $\mathbf{x}_k$ is the element considered at the current step

$$\begin{aligned} w_i^{(r+1)} &= (1-\eta) w_i^{(r)} + \alpha \phi_i(\mathbf{x}_k) && \text{if } \; t_k(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_k) + b)) < 1 \\ w_i^{(r+1)} &= w_i^{(r)} && \text{otherwise} \end{aligned}$$

Kernel methods motivation

- Often we want to **capture nonlinear patterns** in the data
  - Nonlinear Regression: Input-output relationship may not be linear
  - Nonlinear Classification: Classes may not be separable by a linear boundary
- Linear models (e.g., linear regression, linear SVM) are not just rich enough
- Kernels: Make linear models work in nonlinear settings
  - By mapping data to higher dimensions where it exhibits linear patterns
  - Apply the linear model in the new input space
  - Mapping changing the feature representation
- Note: Such mappings can be expensive to compute in general
  - Kernels give such mappings for (almost) free
    - In most cases, the mappings need not be even computed
    - .. using the Kernel Trick!

Kernels: Formally Defined

- Recall: Each kernel k has an associated basis function $\boldsymbol{\phi}$
- $\boldsymbol{\phi}$ takes input $x \in \mathcal{X}$ (input space) and maps it to $\mathcal{F}$ (feature space)

- Kernel $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ takes two inputs and gives their similarity in $\mathcal{F}$ space

$$\begin{aligned} &\phi : \mathcal{X} \mapsto \mathcal{F} \\ &\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R} \qquad\qquad \kappa(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) \end{aligned}$$

- $\mathcal{F}$ needs to be a vector space with a dot product defined on it (Hilbert space)
- Can just any function be used as a kernel function?
  - No. It must satisfy a suitable condition

Verifying a given function is a kernel

A necessary and sufficient condition for a function $\kappa : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$ to be a kernel is that, for all sets $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, the Gram matrix $\mathbf{K}$ such that $k_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ is semidefinite positive, that is

$$\mathbf{v}^T \mathbf{K} \mathbf{v} \geq 0$$

for all vectors[v].

This is equivalent to the condition that all eigenvalues of $\mathbf{K}$ are non negative.

Constructing kernel functions

Example:
Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^2$: $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2$ is a valid kernel function?
This can be verified by observing that

$$\begin{aligned} \kappa(\mathbf{x}_1, \mathbf{x}_2) &= (x_{11}x_{21} + x_{12}x_{22})^2 \\ &= x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11}x_{12}x_{21}x_{22} \\ &= (x_{11}^2, x_{12}^2, x_{11}x_{12}, x_{11}x_{12}) \cdot (x_{21}^2, x_{22}^2, x_{21}x_{22}, x_{21}x_{22}) \\ &= \phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2) \end{aligned}$$

and by defining the base functions as $\mathbf{x} = (x_1^2, x_2^2, x_1x_2, x_1x_2)^T$.

- In general, if $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ then $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2 = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2)$, where

$$\mathbf{x} = (x_1^2, \ldots, x_d^2, x_1x_2, \ldots, x_1x_d, x_2x_1, \ldots, x_dx_{d-1})^T$$

- the d-dimensional input space is mapped onto a space with dimension $m = d^2$
- observe that computing $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ requires time $O(d)$, while deriving it from $\phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2)$ requires $O(d^2)$ steps

The function $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^2$ is a kernel function. In fact,

$$\begin{aligned} \kappa(\mathbf{x}_1, \mathbf{x}_2) &= (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^2 \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} x_{1i}x_{1j}x_{2i}x_{2j} + \sum_{i=1}^{n} (\sqrt{2}cx_{1i})(\sqrt{2}cx_{2i}) + c^2 \\ &= \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) \end{aligned}$$

for

$$\mathbf{x} = (x_1^2, \ldots, x_d^2, x_1x_2, \ldots, x_1x_d, x_2x_1, \ldots, x_dx_{d-1}, \sqrt{2c}x_1, \ldots, \sqrt{2c}x_d, c)^T$$

This implies a mapping from a d-dimensional to a $(d+1)^2$-dimensional space.

Function $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^t$ is a kernel function corresponding to a mapping from a d-dimensional space to a space of dimension

$$m = \sum_{i=0}^{t} d^i = \frac{d^{t+1} - 1}{d - 1}$$

corresponding to all products $x_{i_1}x_{i_2} \ldots x_{i_l}$ with $0 \leq l \leq t$.

Observe that, even if the space has dimension $O(d^t)$, evaluating the kernel function requires just time $O(d)$.

Techniques for constructing kernel functions

Given kernel functions $\kappa_1(\mathbf{x}_1, \mathbf{x}_2)$, $\kappa_2(\mathbf{x}_1, \mathbf{x}_2)$, the function $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ is a kernel in all the following cases

- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = e^{\kappa_1(\mathbf{x}_1, \mathbf{x}_2)}$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa_1(\mathbf{x}_1, \mathbf{x}_2) + \kappa_2(\mathbf{x}_1, \mathbf{x}_2)$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa_1(\mathbf{x}_1, \mathbf{x}_2)\kappa_2(\mathbf{x}_1, \mathbf{x}_2)$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = c\kappa_1(\mathbf{x}_1, \mathbf{x}_2)$, for any $c > 0$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{A} \mathbf{x}_2$, with $\mathbf{A}$ positive definite
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = f(\mathbf{x}_1)\kappa_1(\mathbf{x}_1, \mathbf{x}_2)g(\mathbf{x}_2)$, for any $f, g : \mathbb{R}^n \mapsto \mathbb{R}$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = p(\kappa_1(\mathbf{x}_1, \mathbf{x}_2))$, for any polynomial $p : \mathbb{R}^q \mapsto \mathbb{R}$ with non-negative coefficients
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa_3(\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2))$, for any vector $\boldsymbol{\phi}$ of m functions $\phi_i : \mathbb{R}^n \mapsto \mathbb{R}$ and for any kernel function $\kappa_3(\mathbf{x}_1, \mathbf{x}_2)$ in $\mathbb{R}^m$

$\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$ is a kernel function. In fact,

1. $\mathbf{x}_1 \cdot \mathbf{x}_2 = \mathbf{x}_1^T \mathbf{x}_2$ is a kernel function corresponding to the base functions $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)$, with $\phi_i(\mathbf{x}) = \mathbf{x}$
2. c is a kernel function corresponding to the base functions $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)$, with $\phi_i(\mathbf{x}) = \frac{\sqrt{c}}{n}$
3. $\mathbf{x}_1 \cdot \mathbf{x}_2 + c$ is a kernel function since it is the sum of two kernel functions
4. $(\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$ is a kernel function since it is a polynomial with non negative coefficients (in particular $p(z) = z^d$) of a kernel function

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = e^{-\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{2\sigma^2}}$$

is a kernel function. In fact,

1. since $||\mathbf{x}_1 - \mathbf{x}_2||^2 = \mathbf{x}_1^T\mathbf{x}_1 + \mathbf{x}_2^T\mathbf{x}_2 - 2\mathbf{x}_1^T\mathbf{x}_2$, it results

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = e^{-\frac{\mathbf{x}_1^T\mathbf{x}_1}{2\sigma^2}} e^{-\frac{\mathbf{x}_2^T\mathbf{x}_2}{2\sigma^2}} e^{\frac{\mathbf{x}_1^T\mathbf{x}_2}{\sigma^2}}$$

2. $\mathbf{x}_1^T \mathbf{x}_2$ is a kernel function (see above)
3. then, $\frac{\mathbf{x}_1^T \mathbf{x}_2}{\sigma^2}$ is a kernel function, being the product of a kernel function with a constant $c = \frac{1}{\sigma^2}$
4. $e^{\frac{\mathbf{x}_1^T \mathbf{x}_2}{\sigma^2}}$ is the exponential of a kernel function, and as a consequence a kernel function itself
5. $e^{-\frac{\mathbf{x}_1^T \mathbf{x}_1}{\sigma^2}} e^{-\frac{\mathbf{x}_1^T \mathbf{x}_1}{2\sigma^2}} e^{\frac{\mathbf{x}_1^T \mathbf{x}_2}{\sigma^2}}$ is a kernel function, being the product of a kernel function with two functions $f(\mathbf{x}_1) = e^{-\frac{\mathbf{x}_1^T \mathbf{x}_1}{2\sigma^2}}$ and $g(\mathbf{x}_2) = e^{-\frac{\mathbf{x}_2^T \mathbf{x}_2}{2\sigma^2}}$

Relevant kernel functions

1. Polynomial kernel
$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + 1)^d$$
2. Sigmoidal kernel
$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \tanh\left(c_1 \mathbf{x}_1 \cdot \mathbf{x}_2 + c_2\right)$$
3. Gaussian kernel
$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{2\sigma^2}\right)$$
where $\sigma \in \mathbb{R}$

Observe that a gaussian kernel can be derived also starting from a non linear kernel function $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ instead of $\mathbf{x}_1^T \mathbf{x}_2$.