Support vector machines

Course of Machine Learning
Master Degree in Computer Science
University of Rome "Tor Vergata"
a.a. 2021-2022

Giorgio Gambosi

Idea

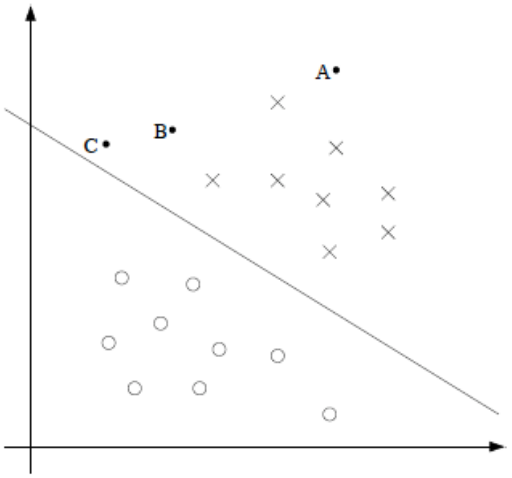
The binary classification problem is approached in a direct way, that is:

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
- We enrich and enlarge the feature space so that separation is (more) possible

Margins



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

Binary classifiers

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

Moreover, we consider linear classifier such as

$$h(\mathbf{x}) = g(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + w_0)$$

where $g(z) = 1$ if $z \geq 0$ and $g(z) = -1$ if $z < 0$. The prediction on the class of **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 **x** belongs to each class.

Margins

For any training set item $(\mathbf{x}_i, t_i)$, the *functional margin* of $(\mathbf{w}, w_0)$ wrt such item is defined as

$$\overline{\gamma}_i = t_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + w_0)$$

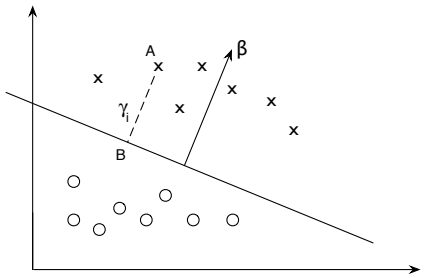
Observe that the resulting prediction is correct iff $\overline{\gamma}_i > 0$. Moreover, larger values of $\overline{\gamma}_i$ denote greater confidence on the prediction.

Given a training set $\mathcal{T} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)\}$ the functional margin of $(\mathbf{w}, w_0)$ wrt $\mathcal{T}$ is the minimum functional margin for all items in $\mathcal{T}$

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

Margins

The *geometric margin* γ_i of a training set item $\mathbf{x}_i, t_i$ is defined as the product of t_i and the distance from $\mathbf{x}_i$ to the boundary hyperplane, that is as the length of the line segment from $\mathbf{x}_i$ to its projection on the boundary hyperplane



Margins

Since, in general, the distance of a point $\overline{\mathbf{x}}$ from a hyperplane $\mathbf{w}^T\mathbf{x} = 0$ is $\dfrac{\mathbf{w}^T\overline{\mathbf{x}}}{||\mathbf{w}||}$, it results

$$\gamma_i = t_i \left(\frac{\mathbf{w}^T}{||\mathbf{w}||} \boldsymbol{\phi}(\mathbf{x}_i) + \frac{w_0}{||\mathbf{w}||} \right) = \frac{\overline{\gamma}_i}{||\mathbf{w}||}$$

So, differently from $\overline{\gamma}_i$, the geometric margin γ_i is invariant wrt parameter scaling. In fact, by substituting $c\mathbf{w}$ to $\mathbf{w}$ and cw_0 to w_0, we get

$$\overline{\gamma}_i = t_i(c\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + cw_0) = ct_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + w_0)$$

$$\gamma_i = t_i \left(\frac{c\mathbf{w}^T}{||c\mathbf{w}||} \boldsymbol{\phi}(\mathbf{x}_i) + \frac{cw_0}{||c\mathbf{w}||} \right) = t_i \left(\frac{\mathbf{w}^T}{||\mathbf{w}||} \boldsymbol{\phi}(\mathbf{x}_i) + \frac{w_0}{||\mathbf{w}||} \right)$$

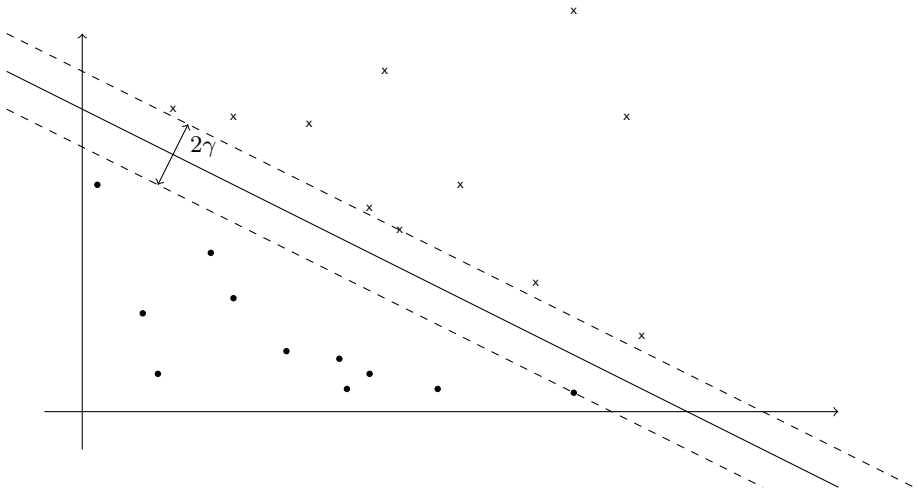
Margins

- 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 $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + w_0 = 0$, 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*.

Margins



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

$$\max_{\mathbf{w},w_0} \gamma$$
$$\text{where } \gamma_i = \frac{t_i}{||\mathbf{w}||}(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) \geq \gamma \qquad i = 1, \ldots, n$$

That is,

$$\max_{\mathbf{w},w_0} \gamma$$
$$\text{where } t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) \geq \gamma\,||\mathbf{w}|| \qquad i = 1, \ldots, n$$

Optimal margin classifiers

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

$$\gamma = \min_i t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) = 1$$

This can be obtained by assuming $||w|| = \frac{1}{\gamma}$, 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

$$\gamma_i = t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) \geq 1$$

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

$$t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) = 1$$

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

Optimal margin classifiers

For any element $\mathbf{x}, t$,

1. $t(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + w_0) > 1$ if $\boldsymbol{\phi}(\mathbf{x})$ is in the region corresponding to its class, outside the margin strip

2. $t(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})+w_0) = 1$ if $\boldsymbol{\phi}(\mathbf{x})$ is in the region corresponding to its class, on the maximum margin hyperplane
3. $0 < t(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + w_0) < 1$ if $\boldsymbol{\phi}(\mathbf{x})$ is in the region corresponding to its class, inside the margin strip
4. $t(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + w_0) = 0$ if $\boldsymbol{\phi}(\mathbf{x})$ is on the separating hyperplane
5. $-1 < t(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + w_0) < 0$ if $\boldsymbol{\phi}(\mathbf{x})$ is in the region corresponding to the other class, inside the margin strip
6. $t(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + w_0) = -1$ if $\boldsymbol{\phi}(\mathbf{x})$ is in the region corresponding to the other class, on the maximum margin hyperplane
7. $t(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + w_0) < -1$ if $\boldsymbol{\phi}(\mathbf{x})$ is in the region corresponding to the other class, outside the margin strip

Optimal margin classifiers

The optimization problem, is then transformed into

$$\max_{\mathbf{w},w_0} \gamma = ||\mathbf{w}||^{-1}$$
$$\text{where } t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) \geq 1 \qquad i = 1, \ldots, n$$

Maximizing $||\mathbf{w}||^{-1}$ is equivalent to minimizing $||\mathbf{w}||^2$ (we prefer minimizing $||\mathbf{w}||^2$ instead of $||\mathbf{w}||$ since it is smooth everywhere): hence we may formulate the problem as

$$\min_{\mathbf{w},w_0} \frac{1}{2}||\mathbf{w}||^2$$
$$\text{where } t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) \geq 1 \qquad i = 1, \ldots, n$$

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 the 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

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

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

Define the *Lagrangian*

$$L(\mathbf{x}, \lambda, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{k'} \mu_j h_j(\mathbf{x})$$

Lagrangian

Consider the maximum, which is a function of $\mathbf{x}$

$$\max_{\lambda,\boldsymbol{\mu}} L(\mathbf{x}, \lambda, \boldsymbol{\mu})$$
$$\lambda_i \geq 0 \qquad i = 1, \ldots, k$$

- if $\mathbf{x}$ is a feasible solution, then $h_j(\mathbf{x}) = 0$ for all j, and the value of μ_j does not affect the maximum, also, $g_i(\mathbf{x}) \leq 0$ and the maximum is obatined for $\lambda_i = 0$: as a consequence, the maximum is equal to $f(\mathbf{x})$ if
- if $\mathbf{x}$ is an unfeasible solution, then either $h_j(\mathbf{x}) \neq 0$ for some j, and the maximum is unbounded by setting the value of μ_j arbitrarily large with the same sign of $h_j(\mathbf{x})$, or $g_i(\mathbf{x}) > 0$ and the maximum is unbounded as λ_i grows indefinitely

As a consequence, the maximum is equal to $f(\mathbf{x})$ if $\mathbf{x}$ is feasible, while it is unbounded if $\mathbf{x}$ is not feasible.

Lagrangian

This results, in the case that an optimum $\mathbf{x}^*$ exists, into

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

In general, the weak duality property holds

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

where $\max_{\lambda\geq\mathbf{0},\boldsymbol{\mu}} \min_{\mathbf{x}} L(\mathbf{x}, \lambda, \boldsymbol{\mu})$ 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

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

Karush-Kuhn-Tucker conditions

The KKT conditions hold at the optimum, and can be used to simplify the dual problem definition.

For any value $\mathbf{x}$, such value is an optimum $\mathbf{x} = \mathbf{x}^*$ iff there exists $\lambda^*, \boldsymbol{\mu}^*$ such that

$$\begin{aligned}
\frac{\partial L(\mathbf{x}, \lambda, \boldsymbol{\mu})}{\partial x_i}\Big|_{\mathbf{x}^*,\lambda^*,\boldsymbol{\mu}^*} &= 0 && i = 1, \ldots, d \\
g_i(\mathbf{x}^*) &\geq 0 && i = 1, \ldots, k \\
h_j(\mathbf{x}^*) &= 0 && i = 1, \ldots, k' \\
\lambda_i^* &\geq 0 && i = 1, \ldots, k \\
\lambda_i^* g_i(\mathbf{x}^*) &= 0 && i = 1, \ldots, k
\end{aligned}$$

In order for the optimum to be a minimum, the second order condition must hold that the Hessian H_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 $g_i(\mathbf{x})$. 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 $t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) - 1 \geq 0$
- there is no $h_j(\mathbf{x})$
- Ω is the intersection of a set of hyperplanes, that is a polyhedron, hence convex.

The corresponding Lagrangian is

$$L(\mathbf{w}, w_0, \lambda) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \lambda_i \left(t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) - 1\right)$$
$$= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \lambda_i t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) + \sum_{i=1}^{n} \lambda_i t_i$$

and the dual problem (with same minimum) is

$$\max_{\lambda} \min_{\mathbf{w}, w_0} L(\mathbf{w}, w_0, \lambda)$$
$$\lambda_i \geq 0 \qquad i = 1, \ldots, k$$

Applying the KKT conditions

$\mathbf{w}^*, w_0^*$ is an optimum iff there exists λ^* such that:

$$\frac{\partial L(\mathbf{w}, w_0, \lambda)}{\partial \mathbf{w}_k}\Big|_{\mathbf{x}^*, \lambda^*} = w_k^* - \sum_{i=1}^{n} \lambda_i t_i \boldsymbol{\phi}_k(\mathbf{x}_i) = 0 \qquad k = 1, \ldots, m$$
$$\frac{\partial L(\mathbf{w}, w_0, \lambda)}{\partial w_0}\Big|_{\mathbf{x}^*, \lambda^*, \boldsymbol{\mu}^*} = \sum_{i=1}^{n} \lambda_i^* t_i = 0$$
$$t_i(\mathbf{w}^{*T}\boldsymbol{\phi}(\mathbf{x}_i) + w_0^*) - 1 \geq 0 \qquad i = 1, \ldots, n$$
$$\lambda_i^* \geq 0 \qquad i = 1, \ldots, n$$
$$\lambda_i^* \left(t_i(\mathbf{w}^{*T}\boldsymbol{\phi}(\mathbf{x}_i) + w_0^*) - 1\right) = 0 \qquad i = 1, \ldots, n$$

Observe that, since $H(\mathbf{w}) = \mathbf{I}$ is positive definite (symmetric with positive eigenvalues) the optimum is a minimum

Lagrange method: dual problem

We may modify the definition of the dual problem by applying the above relations to drop $\mathbf{w}$ and w_0 from $L(\mathbf{w}, w_0, \lambda)$ and from all constraints.

The new problem will have 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

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

Dual problem and kernel function

By defining the *kernel function*

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

the dual problem's formulation can be given as

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

Passing from primal to dual

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

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

The value of w_0^* can be obtained by observing that, for any support vector $\mathbf{x}_k$ (characterized by the condition $\lambda_k \geq 0$), it must be

$$1 = t_k \left(\boldsymbol{\phi}(\mathbf{x}_k)^T \mathbf{w}^* + w_0^* \right) = t_k \left(\sum_{j=1}^{n} \lambda_j^* t_j \boldsymbol{\phi}(\mathbf{x}_j)^T \boldsymbol{\phi}(\mathbf{x}_k) + w_0^* \right)$$
$$= t_k \left(\sum_{j=1}^{n} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_k) + w_0^* \right) = t_k \left(\sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_k) + w_0^* \right)$$

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

Deriving coefficients

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

$$t_k = \sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_k) + w_0^*$$

and

$$w_0^* = t_k - \sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_k)$$

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

$$w_0^* = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(t_i - \sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}_i) \right)$$

Classification through SVM

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

$$y(\mathbf{x}) = \sum_{i=1}^{m} w_i^* \phi_i(\mathbf{x}) + w_0^* = \sum_{j=1}^{n} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}) + w_0^*$$

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

$$y(\mathbf{x}) = \sum_{j \in \mathcal{S}} \lambda_j^* t_j \kappa(\mathbf{x}_j, \mathbf{x}) + w_0^*$$

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 linear separability hypothesis for the classes is quite restrictive
- In general, a suitable set of base functions ϕ, or a suitable kernel function $\kappa(\mathbf{x}_1, \mathbf{x}_2)$, may map all training set elements onto a larger-dimensional feature space where classes turn out to be (at least approximately) linearly separable.

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

$$t_i(\mathbf{w}^T \phi(\mathbf{x}_i) + w_0) \geq 1 \qquad i = 1, \ldots, n$$

- 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

Non separability in the training set

- This can be formalized as

$$\begin{aligned} \min_{\mathbf{w}, w_0, \boldsymbol{\xi}} \; & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \xi_i \\ & t_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + w_0) \geq 1 - \xi_i \qquad i = 1, \ldots, n \\ & \xi_i \geq 0 \qquad i = 1, \ldots, n \end{aligned}$$

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

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

$$\begin{aligned} & L(\mathbf{w}, w_0, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha}) \\ & = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \lambda_i (y_i(\mathbf{w}^T \boldsymbol{\phi}(x_i) + w_0) - 1 + \xi_i) - \sum_{i=1}^{n} \alpha_i \xi_i \\ & = \frac{1}{2}\sum_{i=1}^{n} w_i^2 + \sum_{i=1}^{n} (C - \alpha_i)\xi_i - \sum_{i=1}^{n} \lambda_i (t_i (\sum_{j=1}^{m} w_j \phi_j(\mathbf{x}_i)) + w_0) - 1 + \xi_i) \\ & = \frac{1}{2}\sum_{i=1}^{n} w_i^2 + \sum_{i=1}^{n} (C - \alpha_i - \lambda_i)\xi_i - \sum_{i=1}^{n}\sum_{j=1}^{m} \lambda_i t_i w_j \phi_j(\mathbf{x}_i) + w_0 \sum_{i=1}^{n} \lambda_i t_i + \sum_{i=1}^{n} \lambda_i \end{aligned}$$

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

KKT conditions

The Karush-Kuhn-Tucker conditions are now:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, w_0, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha}) &= \mathbf{0} && && \text{null gradient} \\
\frac{\partial}{\partial w_0} L(\mathbf{w}, w_0, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha}) &= 0 && && \text{null gradient} \\
\frac{\partial}{\partial \boldsymbol{\xi}} L(\mathbf{w}, w_0, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha}) &= \mathbf{0} && && \text{null gradient} \\
t_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + w_0) - 1 + \xi_i &\geq 0 && i = 1, \ldots, n && \text{constraints} \\
\xi_i &\geq 0 && i = 1, \ldots, n && \text{constraints} \\
\lambda_i &\geq 0 && i = 1, \ldots, n && \text{multipliers} \\
\alpha_i &\geq 0 && i = 1, \ldots, n && \text{multipliers} \\
\lambda_i \left(t_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + w_0) - 1 + \xi_i \right) &= 0 && i = 1, \ldots, n && \text{complementary slackness} \\
\alpha_i \xi_i &= 0 && i = 1, \ldots, n && \text{complementary slackness}
\end{aligned}$$

Deriving a dual formulation

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

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

Deriving a dual formulation

By plugging the above relations into $L(\mathbf{w}, w_0, \boldsymbol{\xi}, \lambda, \boldsymbol{\alpha})$, 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 $0 \leq \lambda_i \leq C$

Item characterization

Given a solution of the above problem, the elements of the training set can be partitioned into several subsets:

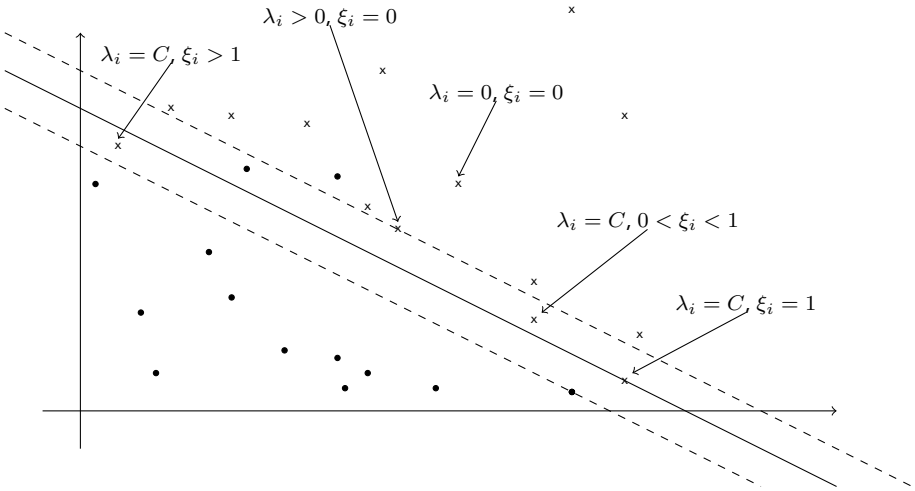
- elements correctly classified and not relevant, the ones such that $\lambda_i = 0$ and $\xi_i = 0$: such elements are in the correct halfspace, in terms of classification, and do not lie on the maximum margin hyperplanes (they are not support vectors)
- elements correctly classified and relevant, the ones such that $\lambda_i > 0$ and $0 \leq \xi_i < 1$: such elements are in the correct halfspace, in terms of classification, either on the maximum margin hyperplanes ($\xi_i = 0$) or within the margin region ($0 < \xi_i < 1$).
- elements incorrectly classified, the ones with $\lambda_i > 0$ and $\xi_i > 1$: such elements are in the wrong halfspace.

Item characterization

Let $\mathbf{x}_i$ be a training set element, then one of the following conditions holds:

1. $\xi_i = 0, \lambda_i = 0$ if $\phi(\mathbf{x}_i)$ is in the correct halfspace, outside the margin strip
2. $\xi_i = 0, 0 < \lambda_i < C$ if $\phi(\mathbf{x}_i)$ is in the correct halfspace, on the maximum margin hyperplane
3. $0 < \xi_i < 1, \lambda_i = C$ if $\phi(\mathbf{x}_i)$ is in the correct halfspace, within the margin strip
4. $\xi_i = 1, \lambda_i = C$ if $\phi(\mathbf{x}_i)$ is on the separating hyperplane
5. $\xi_i > 1, \lambda_i = C$ if $\phi(\mathbf{x}_i)$ is in the wrong halfspace

Item characterization



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

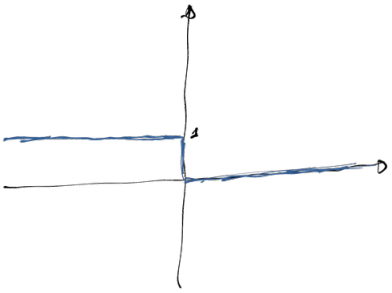
- Training time of the standard SVM is $O(n^3)$ (solving QP)
  - Can be prohibitive for large datasets
- Lots of research has gone into speeding up the SVMs

- Many approximate QP solvers are used to speed up SVMs
- Gradient descent? We need a loss function and its gradient

0/1 loss

The most "natural" loss function in (binary) classification

$$\underset{\mathbf{w},b}{\text{argmin}} \sum_{i=1}^{n} \mathbf{1}[t_i(\mathbf{w}^T\mathbf{x}_i + b) > 0]$$



0/1 loss

Consider the two cases:

- $\min_{\mathbf{w},b} \sum_{i=1}^{n} \mathbf{1}[t_i(\mathbf{w}^T\mathbf{x}_i + b) > 0] = 0$, that is, classes are linearly separable in the training set: in this case a simple Perceptron is guaranteed to find the optimal solution (even if referring to a different loss function)
- $\min_{\mathbf{w},b} \sum_{i=1}^{n} \mathbf{1}[t_i(\mathbf{w}^T\mathbf{x}_i + b) > 0] > 0$, that is, classes are not linearly separable in the training set: a Perceptron does not converge to any solution
  - indeed, in this case, it is not possible to minimize the function efficiently, unless $P = NP$: the problem is NP-hard even to approximate the solution, at least up to a small constant

0/1 loss e regularization

- Indeed, we do not really aim to optimize the loss function wrt the training set
- Need to generalize to test set, thus avoiding overfitting
- Regularization term

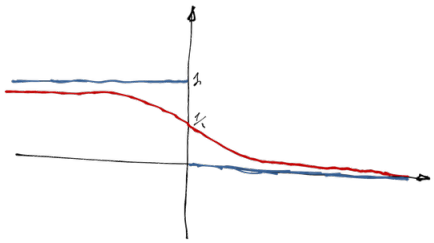
$$\underset{\mathbf{w},b}{\text{argmin}} \sum_{n} \mathbf{1}[t_n(\mathbf{w}^T\mathbf{x} + b) > 0] + \lambda R(\mathbf{w}, b)$$

Key issues

- How can we adjust the optimization problem so that there exist efficient algorithms for solving it?
- What are good regularizers $R(\mathbf{w}, b)$ for hyperplanes?
- Assuming we can adjust the optimization problem appropriately, what algorithms exist for efficiently solving this regularized optimization problem?

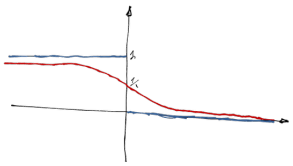
Surrogate smooth loss functions

- The problem of 0/1 loss is that
  - it is constant most of time: most variations of distance from the separating hyperplane do not correspond to loss change
  - sometimes, it varies abruptly: infinitesimal variations of distance result in relevant loss change
  - 0/1 loss is not *smooth*
- Use loss functions which approximate 0/1 loss and are smooth, such as



Surrogate smooth loss functions

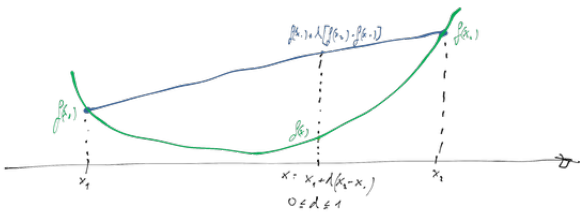
What can we expect when using this surrogate function?



- wrongly classified points do not weigh the same in terms of loss: they always weigh at least 0.5, but the difference in weights between points nearby the hyperplane and very distant points is about 0.5
- the same for well classified points: they always weigh at most 0.5, but the difference in weights between points nearby the hyperplane and very distant points is about 0.5
- loss varies smoothly wrt distance from the hyperplane: this makes it possible to use derivatives
- however we would prefer dealing with a *convex* loss function

Convex loss functions

Why do we like this type of functions?



- second derivative non negative everywhere: they have (at most) one local minimum
- as a consequence, finding the local minimum corresponds to a global minimization
- first derivative monotonically non-decreasing: at any point, the gradient points to the direction towards global minimum

Convex surrogate loss functions

- Approximate *from above* 0/1 loss: real 0/1 error always less than function loss
- Convex: unique local minimum = global minimum
- Smooth: may use derivatives to find minimum
- Main difference: relevance given to erroneous predictions

Convex surrogate loss functions

Let $y = \mathbf{w}^T\mathbf{x} + b$: some common loss functions

0/1 $L(t, y) = \mathbf{1}[yt \leq 0]$

Squared $L(t, y) = (t - y)^2$

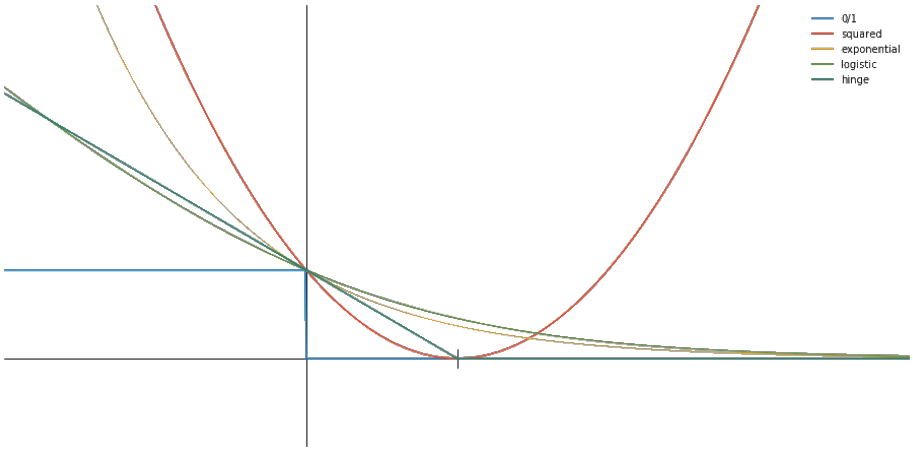
Logistic $L(t, y) = \frac{1}{\log 2} \log(1 + e^{-yt})$

Exponential $L(t, y) = e^{-yt}$

Hinge $L(t, y) = \max(0, 1 - yt)$

Convex surrogate loss functions

Plot of the defined loss functions: $y = \mathbf{w}^T\mathbf{x} + b$ on the horizontal axis, assuming $t = 1$



Hinge loss

Here, $L_H(y, t) = \max(0, 1 - ty)$. Not differentiable wrt to y at $ty = 1$

$$\frac{\partial}{\partial y} L_H = \begin{cases} -t & ty < 1 \\ 0 & ty > 1 \\ \text{undefined} & ty = 1 \end{cases}$$

Hinge loss

In order to apply gradient descent to minimize hinge loss, observe that, since $y = \mathbf{w}^T\mathbf{x} + b$,

$$\nabla_{\mathbf{w}} y = \mathbf{x}$$
$$\frac{\partial y}{\partial b} = 1$$

it results

$$\nabla_{\mathbf{w}} L_H = \begin{cases} -t\mathbf{x} & t(\mathbf{w}^T\mathbf{x} + b) < 1 \\ 0 & t(\mathbf{w}^T\mathbf{x} + b) > 1 \\ \text{undefined} & t(\mathbf{w}^T\mathbf{x} + b) = 1 \end{cases}$$

$$\frac{\partial}{\partial b} L_H = \begin{cases} -t & t(\mathbf{w}^T\mathbf{x} + b) \leq 1 \\ 0 & t(\mathbf{w}^T\mathbf{x} + b) > 1 \\ \text{undefined} & t(\mathbf{w}^T\mathbf{x} + b) = 1 \end{cases}$$

However, we may use it in gradient descent by referring to some *subgradient*

Subgradient

Given a convex function (such as hinge loss) f, at each differentiable point, the corresponding gradient $\nabla(x)$ provides a function which lower bounds f

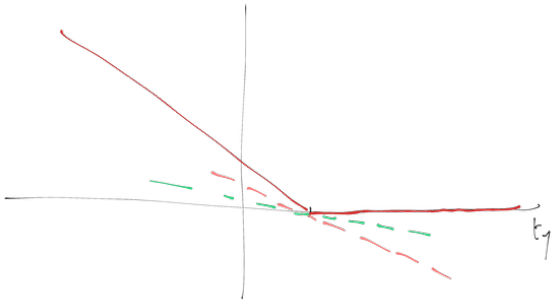
$$f(x') \geq f(x) + \nabla(x)(x - x')$$

If x is a singular point, where f is not differentiable and $\nabla(x)$ does not exist, a *subgradient* $\overline{\nabla}(x)$ is *any* function which lower bounds f

$$f(x') \geq f(x) + \overline{\nabla}(x)(x - x')$$

Subgradient and hinge loss

In the case of hinge loss, we may observe that any line whose slope in $[-t, 0]$ (if $t = 1$, in $[0, -t]$ if $t = -1$) is a subgradient



Subgradient and hinge loss

We may then choose the horizontal axis as the subgradient to use, which results into assuming that $\frac{\partial L_h}{\partial y} = 0$ at $ty = 1$ and, as a consequence

$$\overline{\nabla}_{\mathbf{w}} L_H = \begin{cases} -t\mathbf{x} & t(\mathbf{w}^T\mathbf{x} + b) < 1 \\ 0 & t(\mathbf{w}^T\mathbf{x} + b) \geq 1 \end{cases}$$

and, applying the same notation,

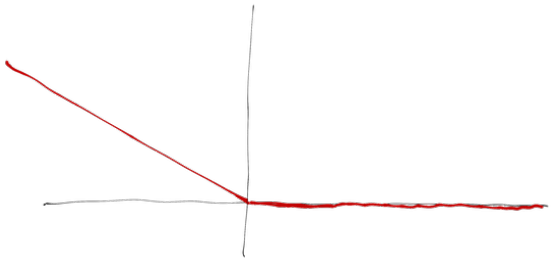
$$\frac{\overline{\partial}}{\overline{\partial b}} L_H = \begin{cases} -t & t(\mathbf{w}^T\mathbf{x} + b) \leq 1 \\ 0 & t(\mathbf{w}^T\mathbf{x} + b) \geq 1 \end{cases}$$

Subgradient, hinge loss and perceptron

Observe that, the hinge loss function translated by -1, that is

$$L_H(\mathbf{w}, b, \mathbf{x}, t) = \max(0, -y(\mathbf{w}^T\mathbf{x} + b))$$

is precisely the loss function used in the perceptron algorithm.



This loss function, even if convex and smooth almost everywhere (as discussed above), is not however a lower bound to 0/1 loss: for small errors (distance y from the hyperplane less than 1) it returns costs smaller than 1 (the value provided by 0/1 loss)

Logistic loss

Minimizing the logistic loss $L_l(t, y) = \frac{1}{\log 2}\log(1 + e^{-ty})$ is equivalent to minimizing cross entropy $t\log y + (1 - t)\log(1 - y)$ in logistic regression.

In fact, observe that, by considering the two cases $t = 1$, $t = -1$,

$$L_l(\mathbf{w}, b, t, \mathbf{x}) \propto \begin{cases} -\log(\sigma(\mathbf{w}^T\mathbf{x} + b)) = -\log(p(C_1|\mathbf{x})) & t = 1 \\ -\log(1 - \sigma(\mathbf{w}^T\mathbf{x} + b)) = -\log(p(C_0|\mathbf{x})) & t = -1 \end{cases}$$

The same results from cross entropy, where the two cases correspond to $t = 1$, $t = 0$,

$$\begin{cases} -\log(\sigma(\mathbf{w}^T\mathbf{x} + b)) = -\log(p(C_1|\mathbf{x})) & t = 1 \\ -\log(1 - \sigma(\mathbf{w}^T\mathbf{x} + b)) = -\log(p(C_0|\mathbf{x})) & t = 0 \end{cases}$$

Regularization term

Usually, in this framework, defined in terms of norms of $\mathbf{w}$

0 **norm** $R_0(\mathbf{w}) = \mathbf{1}[w_i \neq 0]$

1 **norm** $R_1(\mathbf{w}) = \sum_{i=1}^{d} |w_i|$

2 **norm** $R_2(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{d} w_i^2$

p **norm** $R_2(\mathbf{w}) = \left(\sum_{i=1}^{d} |w_i|^p\right)^{1/p}$

Known methods

Approaches considered before fit nicely into this framework

- Ridge regression corresponds to squared loss + R_2 regularization
- Lasso regression corresponds to squared loss + R_1 regularization
- Regularized logistic regression corresponds to logistic loss + R_2 regularization
- Perceptron corresponds to translated hinge loss, no regularization

- What about SVM?

SVM and gradient descent

Recall the formalization of the problem in the general case

$$\min_{\mathbf{w},w_0,\boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$
$$t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) \geq 1 - \xi_i \qquad i = 1,\ldots,n$$
$$\xi_i \geq 0 \qquad i = 1,\ldots,n$$

SVM and gradient descent

Given $\mathbf{w}, w_0$, the slack variable ξ_i is minimized as

$$\xi_i = \begin{cases} 0 & t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) \geq 1 \\ 1 - t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) & \text{otherwise} \end{cases}$$

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

$$L_H(\mathbf{w}, w_0, \mathbf{x}_i, t_i) = \max\left(0, 1 - t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0)\right)$$

We can then define the cost function to be minimized as

$$C(\mathbf{w}) = \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} L_H(\mathbf{w}, w_0, \mathbf{x}_i, t_i)$$
$$= \sum_{i=1}^{n} L_H(\mathbf{w}, w_0, \mathbf{x}_i, t_i) + R_2(\mathbf{w})$$

That is, SVM correspond to hinge loss with ridge regularization

SVM and gradient descent

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

$$\overline{\nabla}_w = \mathbf{w} - \sum_{\mathbf{x}_i \in L} t_i\boldsymbol{\phi}(\mathbf{x}_i)$$

where $x_i \in L$ iff $t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) < 1$.

The resulting iteration is

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \alpha\mathbf{w}^{(r)} + \alpha\sum_{\mathbf{x}_i \in L} t_i\boldsymbol{\phi}(\mathbf{x}_i)$$

SVM and SGD

In stochastic gradient descent, single items are considered at each iteration. This results in the following update rule

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \alpha\mathbf{w}^{(r)} + \alpha_i\boldsymbol{\phi}(\mathbf{x}_i) \qquad \text{if } \; t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0)) < 1$$
$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} \qquad \text{otherwise}$$

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 ϕ
- ϕ 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

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

- $\mathcal{F}$ 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 $\kappa : \mathbf{R}^n \times \mathbf{R}^n \mapsto \mathbf{R}^n$ 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

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

for all vectors˘.

Constructing kernel functions

Example 1. Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{R}^2$: $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2$ 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 $\phi(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, x_1x_2)^T$.

Constructing kernel functions

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

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

- 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 $\boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_2)$ requires $O(d^2)$ steps

Constructing kernel functions

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

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

for

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

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

Constructing kernel functions

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 $f, g : \mathbf{R}^n \mapsto \mathbf{R}$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = p(\kappa_1(\mathbf{x}_1, \mathbf{x}_2))$, for any polynomial $p : \mathbf{R}^q \mapsto \mathbf{R}$ 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 $\phi_i : \mathbf{R}^n \mapsto \mathbf{R}$ and for any kernel function $\kappa_3(\mathbf{x}_1, \mathbf{x}_2)$ in $\mathbf{R}^m$

Costructing kernel functions

$\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$ 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

Costructing kernel functions

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

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 $\mathbf{x}_1^T \mathbf{x}_2$.

SVM, kernels and SGD

Recall that by introducing kernels, a SVM may perform predictions by computing

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

instead of

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

However, we only know how to apply SGD to derive primal coefficients w_i^*. Can be apply it also to compute the dual coefficients λ_j^*?

It turns out that a suitable version of SGD (not discussed here) can be applied to learn the λ_i^* coefficients of a kernelized SVM