Machine learning

Support vector machines

Corso di Laurea Magistrale in Informatica

Università di Roma Tor Vergata

Prof. Giorgio Gambosi

a.a. 2021-2022

Idea

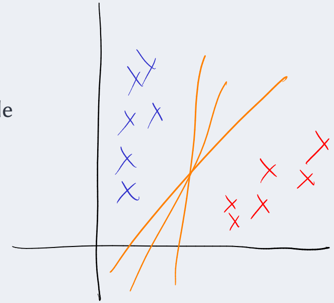
Metodo che cerca un iperpiano di separazione fra le classi, ma questo deve esistere. L'ipotesi iniziale (forte) e' che esista ⇒ le classi sono linearmente separabili.

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



Cerco un separatore lineare, ma non ce n'e' uno solo: posso far variare in modo infinitesimo i coefficienti dell'iperpiano ⇒ cerco il migliore in SVM.

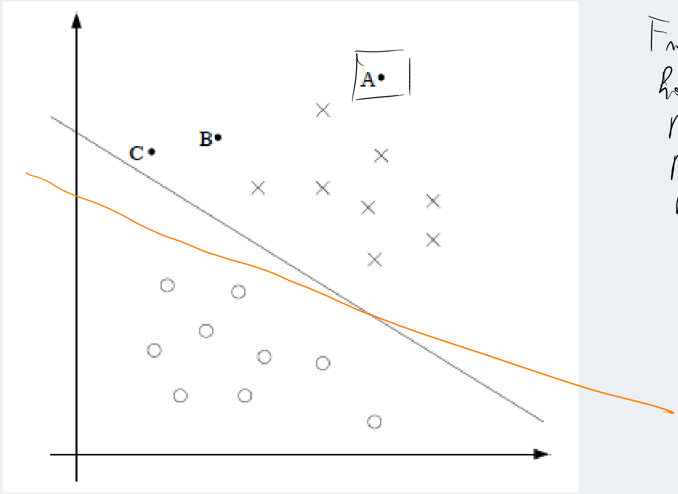
Se le classi non sono separabili, quello che posso fare è "ammorbidire" il concetto della separazione.

Qualunque iperpiano che prendo, ci sarà qualche elemento tagliato male, lo accetto ma gli assegno un costo, accetto quindi che ci siano elementi classificati male ma ne tengo conto quando scelgo l'iperpiano migliore.

Altrimenti porto i punti, mediante funzioni base, in uno spazio con più dimesioni e conto che le immagini così ottenute siano linearmente separabili.

Margins

A e' piu' lontano dalla regione di separazione $\Rightarrow$ mi piace di piu' di C e B. Dal punto di vista dell'iperpiano, mi piace di piu' dell'altro perche' i punti distano di piu'.



Fra gli infiniti che ho: mi piace di piu', fra due, quello per cui l'elemento piu' vicino e' piu' lontano.

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

Posso non considerare A, B, C ma prendere quello più vicino all'iperpiano: fra tutti, quello ha la classificazione più incerta.

Cambiando iperpiano, stavolta il punto linearmente più vicino sarà un altro, con la sua distanza dall'iper piano e così via …

Fra tutti, ci piace di più quello per cui: considerando, fra tutti gli elementi del training set classificati bene per ipotesi, quello più vicino all'iperpiano, cambiandolo ci sarà un elemento che sarà più vicino a quello nuovo con un'altra distanza.

L'iperpiano che più mi piace è quello per cui la distanza del punto più vicino è la massima possibile

Binary classifiers

Assumo di aver applicato f. kernel ϕ su x

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 $\mathscr{C}_0$ if $y = -1$ and to $\mathscr{C}_1$ 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 $\mathbf{x}$ 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 $p(\mathscr{C}_i|\mathbf{x})$ that $\mathbf{x}$ belongs to each class.

Margins

Supponiamo di aver definito l'iperpiano di separazione, abbiamo $\mathbf{w}$ e w_0
Lo pensiamo nello spazio delle feature proiettate.

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. $\Rightarrow$ più lontano dall'iperpiano di separazione.

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

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

Dato un determinato piano di separazione, ho un valore per gamma e posso rendere tutti i coefficienti più piccoli fra loro in modo che il prodotto risulti più piccolo (ma l'iperpiano è lo stesso), quindi anche i gamma segnati diminuiscono ma non ho cambiato nulla.

Questo è un problema, il fatto di poter diminuire i coefficienti: vorremmo far si che ad 1 iperpiano corrisponda solo una coppia di coefficienti.

Margins

Definiamo il margine geometrico: $\mathbf{w}$ e w_0 minimo.

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

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

normalizzato rispetto ai coeff
dell'iperpiano.

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

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

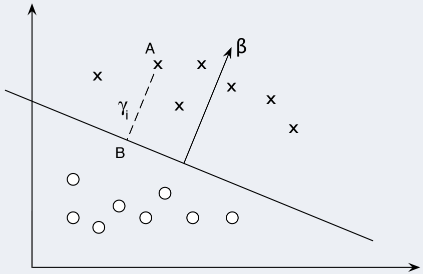
$$\gamma_i = t_i\left(\frac{c\mathbf{w}^T}{||c\mathbf{w}||}\phi(\mathbf{x}_i) + \frac{cw_0}{||c\mathbf{w}||}\right) = t_i\left(\frac{\mathbf{w}^T}{||\mathbf{w}||}\phi(\mathbf{x}_i) + \frac{w_0}{||\mathbf{w}||}\right)$$

Lo preferiamo, è invariante rispetto alla scalatura dei parametri: moltiplicando o dividendo i parametri per uno stesso valore, il risultato è lo stesso.

La nuova gamma non è altro che la distanza del punto rispetto all'iperpiano misurata in unità di misura pari a norma di w (quante norme di w ci sono fra l'iperpiano di separazione a w).

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

$(\mathbf{w}, w_0) \Rightarrow (\gamma_1, \gamma_2, \dots, \gamma_n)$ sono funzione dei parametri.

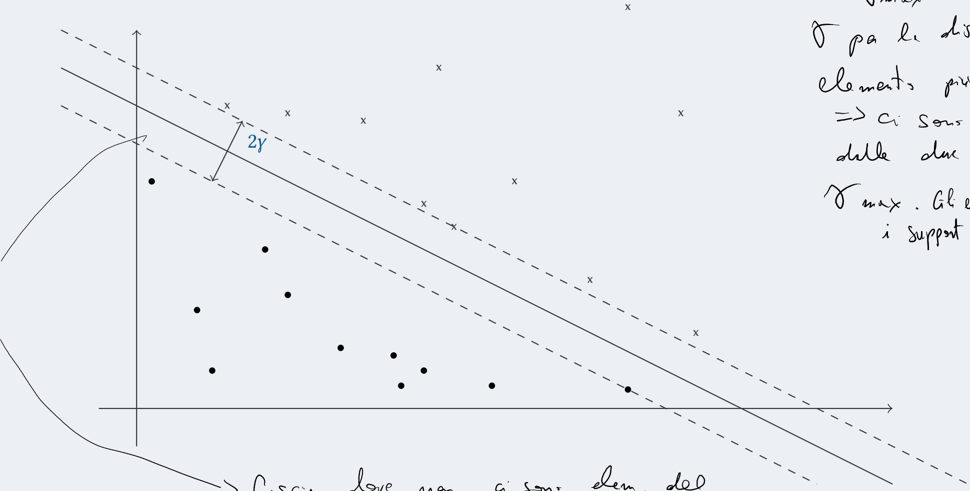
- The geometric margin wrt the training set $T = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_n, t_n)\}$ is then defined as the smallest geometric margin for all items $(\mathbf{x}_i, t_i)$

$$\gamma = \min_i \gamma_i$$

- a useful interpretation of γ is as half the width of the largest strip, centered on the hyperplane $\mathbf{w}^T\phi(\mathbf{x}) + w_0 = 0$, containing none of the points $\mathbf{x}_1, \dots, \mathbf{x}_n$
- 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.

Possiamo modificare l'iperpiano solo con rotazioni, traslazioni etc....

Margins



Se γ_{max} è il valore di γ per la distanza max dell' elemento più vicino.
$\Rightarrow$ ci sono almeno 2 elem delle due parti a distanza γ_{max}. Gli elementi a distanza γ_{max} sono i support vectors.

Vogliamo l'iperpiano di separazione in cui i margini sono alla massima distanza.

> fascia dove non ci sono elem. del train set. Possiamo crearne la più larga possibile.

Optimal margin classifiers

Given a training set T, we wish to find the hyperplane 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

$$\max_{\mathbf{w},w_0} \gamma$$

$$\text{where } \gamma_i = \underbrace{\frac{t_i}{\|\mathbf{w}\|}(\mathbf{w}^T\phi(\mathbf{x}_i) + w_0)}_{\text{distanza dell' i-esimo punto.}} \geq \gamma \qquad i = 1, \dots, n$$

⇒ altro modo di cercare max/min

That is,

$$\max_{\mathbf{w},w_0} \gamma$$

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

problema di ottimizzazione con vincoli.

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

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

$\longrightarrow$ scaliamo $\mathbf{w}, w_0$ per avere come risultato 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

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

$\longrightarrow$ se vale per $(\mathbf{x}_i, t_i)$

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

$\hookrightarrow$ almeno 2, uno per classe

non perdiamo in generalità se assumiamo $\gamma = 1$

Optimal margin classifiers

l'iperpiano e' il luogo dei punti per cui vale questo

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

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

Optimal margin classifiers

$pwm = \frac{1}{\gamma}$

The optimization problem, is then transformed into

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

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

$$\left.\begin{aligned}&\min_{\mathbf{w},w_0} \frac{1}{2}\|\mathbf{w}\|^2 \rightarrow \frac{1}{2}\, w_1^2 + w_2^2 + \dots + w_d^2\\ &\text{where } t_i(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) + w_0) \geq 1 \qquad i = 1, \dots, n\end{aligned}\right]$$

problema di minimizzazione migliore.

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 $\longrightarrow$ problema collegato all'originale.
- 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

$$\begin{aligned} g_i(\mathbf{x}) &\leq 0 \qquad & i = 1, \dots, k \\ h_j(\mathbf{x}) &= 0 \qquad & i = 1, \dots, k' \end{aligned}$$

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

Define the Lagrangian

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

funzione obiettivo a cui vengo integrati i vincoli, moltiplicati per i coefficienti Lagrangiani.

Lagrangian

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

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

- 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 $\mathbf{x}$ 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

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

Cerco l'ottimo di w

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

Siamo sul l'ottimo = (gradiente = 0)

$$\left.\frac{\partial L(\mathbf{x}, \Lambda, \boldsymbol{\mu})}{\partial x_i}\right|_{\mathbf{x}^*, \Lambda^*, \boldsymbol{\mu}^*} = 0 \qquad i = 1, \dots, d$$

è il duale

Se x^* è ottimo, è anche una soluzione ammissibile

$$\begin{cases} g_i(\mathbf{x}^*) \boxed{\geq} 0 & i = 1, \dots, k \\ h_j(\mathbf{x}^*) = 0 & i = 1, \dots, k' \end{cases}$$

$$\lambda_i^* \geq 0 \qquad i = 1, \dots, k$$

$$\boxed{\lambda_i^* g_i(\mathbf{x}^*) = 0} \qquad i = 1, \dots, k$$

importante

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

$$\max_{\Lambda}\min_{\mathbf{w},w_0} L(\mathbf{w},w_0,\Lambda)$$

$$\lambda_i\geq 0 \qquad\qquad i=1,\ldots,k$$

Applying the KKT conditions

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

$$\left.\frac{\partial L(\mathbf{w}, w_0, \Lambda)}{\partial \mathbf{w}_k}\right|_{\mathbf{x}^*,\Lambda^*} = w_k^* - \sum_{i=1}^{n} \lambda_i t_i \phi_k(\mathbf{x}_i) = 0 \qquad k = 1, \dots, m$$

$$\left.\frac{\partial L(\mathbf{w}, w_0, \Lambda)}{\partial w_0}\right|_{\mathbf{x}^*,\Lambda^*,\boldsymbol{\mu}^*} = \sum_{i=1}^{n} \lambda_i^* t_i = 0$$

$$t_i({\mathbf{w}^*}^T \boldsymbol{\phi}(\mathbf{x}_i) + w_0^*) - 1 \geq 0 \qquad i = 1, \dots, n$$

$$\lambda_i^* \geq 0 \qquad i = 1, \dots, n$$

$$\lambda_i^* \boxed{\left(t_i({\mathbf{w}^*}^T \boldsymbol{\phi}(\mathbf{x}_i) + w_0^*) - 1\right)} = 0 \qquad i = 1, \dots, n$$

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

$\rightarrow \neq 0$ per tutti i punti che non sono sull' iperpiano $\Rightarrow \lambda_i^* = 0$

Lagrange method: dual problem

Riformulo la definizione del duale applicando le relazioni precedenti.

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

$$\max_{\Lambda} \overline{L}(\Lambda) = \max_{\Lambda} \left(\sum_{i=1}^{n} \lambda_i - \frac{1}{2} \underbrace{\sum_{i=1}^{n} \sum_{j=1}^{n}}_{} \lambda_i \lambda_j t_i t_j \boldsymbol{\phi}(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_j) \right)$$

$$\lambda_i \geq 0 \qquad i = \ldots, n$$

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

trasformo in un insieme di valori e la soluzione ed ottengo l'ottimo

Qui la f. obiettivo e' nel quadrato dei termini: ho una somma per ogni coppia di punti dove moltiplico le coordinate, i target etc... la $\phi(\mathbf{x}_i)$ non compare mai da sola.

Dual problem and kernel function

Ho fissato le ϕ, entra in gioco il prodotto con un altro punto.

By defining the kernel function

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \Rightarrow$$ definisco una funzione binariata.

the dual problem's formulation can be given as

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

$$\lambda_i \geq 0 \qquad i = 1, \dots, n$$

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

$\rightarrow$ Lo posso calcolare a partire da $\phi(x_i)$, $\phi(x_j)$ e fare $\phi(x_i)^T \phi(x_j)$

Se posso calcolare $K(\phi(x_i), \phi(x_j))$ allora non devo passare per le ϕ, non devo calcolarle.

Potrei anche calcolarla senza passare da qui.

Ha senso non passare per le funzioni base perché se passo da uno spazio di dimesione d ad uno di dimensione m magari devo fare un prodotto scalare con più componenti.

Inoltre, in alcuni casi questo può risultare particolarmente utile: spesso viene usata la funzione kernel Guassiana, dove devo proiettare in uno spazio illimitato.

Più in generale, facendo così non mi interessa qual è l'immagine dell'elemento, mi interessa solo come riesco a misurare la relazione con un altro elemento.

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

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

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

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

where 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

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

and

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

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

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

Classification through SVM

o trovo w_0^*, w_1^* ...
oppure $\lambda_1^*, \lambda_2^*, \ldots \lambda_n^*, w_0^*$

(corrisponde ai due membri dell'equazione)

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

vedo le feature del punto

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

applico similitudine fra i punti + bias.

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

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

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.

→ Tutti i punti non critici (non sul margine) hanno $\lambda_i^* = 0$. Mi aspetto che valga per la grande maggioranza dei punti, la somma è fatta solo sui vettori di supporto.

$\exists \phi \quad \mathbb{R}^d \Rightarrow \mathbb{R}^m \quad \exists\, m > 0$ t.c.

$\forall x_1, x_2$ vale che $K(x_1, x_2) = \phi(x_1) \underset{\text{scalare}}{\cdot} \phi(x_2)$

allora esisteranno delle funzioni che rispettano questa cosa, le chiamo funzioni kernel.

possiamo usare un test che ci dice se una funzione è kernel.
Definiamo poi un insieme di regole che , fissate le funzioni che hanno questa caratteristica, componendole abbiamo ancora delle funzioni kernel.

Così, possiamo costruire una funzione sapendo che è kernel, ma non posso prenderne una a caso.
Invece di risolvere l'ottimo mediante metodi di ottimizzazione lineare, c'è un modo di vedere il problema come uno dove posso applicare una discesa del gradiente?
Se la applico, lo faccio su una funzione costo (cerco un minimo) in una situazione senza vincoli, serve quindi una tale funzione di costo senza vincoli.

Non separability in the training set

Caso generale, rilasciamo l'ipotesi di separabilità.

- The linear separability hypothesis for the classes is quite restrictive
- In general, a suitable set of base functions $\boldsymbol{\phi}$, 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

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

ci sarà almeno un elemento per cui vale <1

- 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

Trasformo in $t_i(w^T\phi(x_i) + w_0) \geq 1 - \xi_i$

Non separability in the training set

- This can be formalized as

voglio rispettare i vincoli abbassandoli il meno possibile

$$\min_{\mathbf{w},w_0,\boldsymbol{\xi}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^n \xi_i$$

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

$$\xi_i \geq 0 \qquad i = 1, \dots, n$$

ξ_i rappresenta di quanto il vincolo originale non e' rispettato

Servono per rendere i vincoli piu' semplici da superare, ma associo un costo all'abbassamento della soglia, portando le ξ_i nella funzione

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

- 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 $i = 1 \dots, n$.

KKT conditions

The Karush-Kuhn-Tucker conditions are now:

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

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

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

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

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

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:

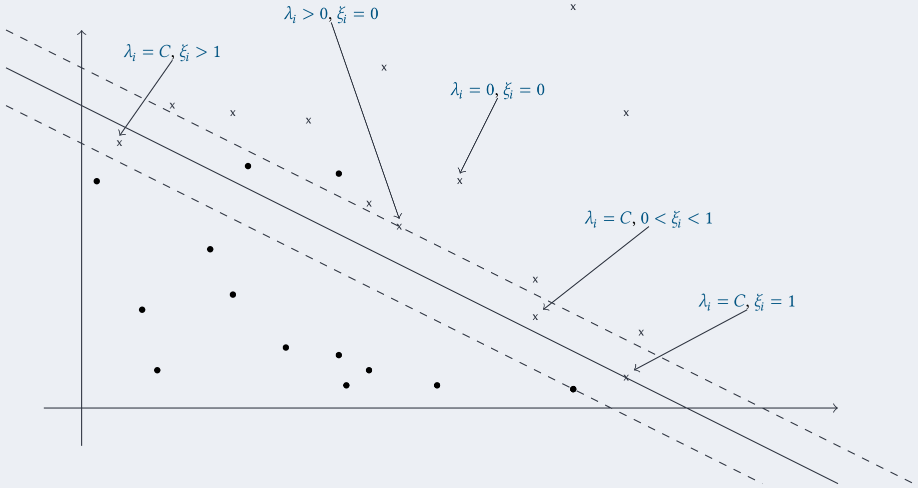
- elements correctly classified and not relevant, the ones such that $\lambda_i = 0$ and $\xi_i = 0$: such elements are in the correct halfspace, in terms of classification, and do not lie on the maximum margin hyperplanes (they are not support vectors)
- elements correctly classified and relevant, the 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 $\boldsymbol{\phi}(\mathbf{x}_i)$ is in the correct halfspace, outside the margin strip
2. $\xi_i = 0, 0 < \lambda_i < C$ if $\boldsymbol{\phi}(\mathbf{x}_i)$ is in the correct halfspace, on the maximum margin hyperplane
3. $0 < \xi_i < 1, \lambda_i = C$ if $\boldsymbol{\phi}(\mathbf{x}_i)$ is in the correct halfspace, within the margin strip
4. $\xi_i = 1, \lambda_i = C$ if $\boldsymbol{\phi}(\mathbf{x}_i)$ is on the separating hyperplane
5. $\xi_i > 1, \lambda_i = C$ if $\boldsymbol{\phi}(\mathbf{x}_i)$ is in the wrong halfspace

Item characterization


$\lambda_i > 0, \xi_i = 0$
$\lambda_i = C, \xi_i > 1$
$\lambda_i = 0, \xi_i = 0$
$\lambda_i = C, 0 < \xi_i < 1$
$\lambda_i = C, \xi_i = 1$

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

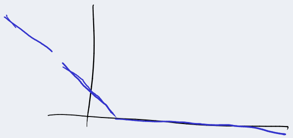
The approach can be extended to

- 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

Hinge loss



Let y, t the predicted value and the target, respectively. The hinge loss is defined as

$$L_H(y,t) = \max\left(0, 1 - ty\right)$$

If $y = \mathbf{w}^T\mathbf{x} + w_0$, we may define it as a function of $\mathbf{w}, w_0, \mathbf{x}$ (and t) as

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

In order to apply gradient descent to minimize hinge loss, the derivatives with respect to $\mathbf{w}$ and w_0 must be considered,

$$\begin{aligned} \frac{\partial L_H}{\partial w_i} &= \frac{\partial L_H}{\partial y}\frac{\partial y}{\partial w_i} = x_i \frac{\partial L_H}{\partial y} \qquad i = 1, \ldots, d \\ \frac{\partial L_H}{\partial w_0} &= \frac{\partial L_H}{\partial y}\frac{\partial y}{\partial w_i} = \frac{\partial L_H}{\partial y} \end{aligned}$$

y è il risultato dell'applicazione di $\mathbf{w}^T$, w_0 ed $\mathbf{x}$

Corrisponde alla funzione costo usata nel perceptrone

Hinge loss

Observe that L_H differentiable wrt to y at $ty = 1$

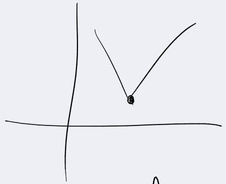
$$\frac{\partial L_H}{\partial y} = \begin{cases} -t & ty < 1 \\ 0 & ty > 1 \\ \text{undefined} & ty = 1 \end{cases}$$

which results into

$$\frac{\partial L_H}{\partial w_i} = \begin{cases} -tx_i & t(\mathbf{w}^T\mathbf{x} + w_0) < 1 \\ 0 & t(\mathbf{w}^T\mathbf{x} + w_0) > 1 \\ \text{undefined} & t(\mathbf{w}^T\mathbf{x} + w_0) = 1 \end{cases} \qquad i = 1, \ldots, d$$

$$\frac{\partial L_H}{\partial w_0} = \begin{cases} -t & t(\mathbf{w}^T\mathbf{x} + w_0) < 1 \\ 0 & t(\mathbf{w}^T\mathbf{x} + w_0) > 1 \\ \text{undefined} & t(\mathbf{w}^T\mathbf{x} + w_0) = 1 \end{cases}$$

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



qualunque funzione che fornisce valore nel punto < di quella originale e' un sub gradiente.

es: qualunque retta al di sotto della hinge loss

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. 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

$$\frac{\partial L_H}{\partial w_i} = \begin{cases} -tx_i & t(\mathbf{w}^T\mathbf{x} + w_0) < 1 \\ 0 & t(\mathbf{w}^T\mathbf{x} + w_0) \geq 1 \end{cases} \qquad i = 1, \dots, d$$

$$\frac{\partial L_H}{\partial w_0} = \begin{cases} -t & t(\mathbf{w}^T\mathbf{x} + w_0) < 1 \\ 0 & t(\mathbf{w}^T\mathbf{x} + w_0) \geq 1 \end{cases}$$

SVM and gradient descent

Recall the formalization of the problem in the general case

$$\min_{\mathbf{w}, w_0, \xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$

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

$$\xi_i \geq 0 \qquad i = 1, \ldots, n$$

Supponiamo di aver fissato $\mathbf{w}$ e w_0 dover minimizzare ξ_i rispettando i vincoli.

SVM and gradient descent

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

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

Se il vincolo e' rispettato, il punto e' dalla parte giusta

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 may then define the cost function to be minimized as

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

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

funzione di costo regolarizzata con una ridge.

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

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

where $x_k \in L$ iff $t_k(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_k) + w_0) < 1$.

The resulting iteration is

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

SVM and SGD

m $\cancel{\phi} \Rightarrow \mathbb{R}^d \to \mathbb{R}^m$ ma m è potenzialmente ∞.

Kernel RBF va bene, ma se cerco la soluzione sul primale mi serve la ϕ corrispondente

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

$$\begin{cases} w_i^{(r+1)} = (1-\alpha)w_i^{(r)} + \alpha\phi_i(\mathbf{x}_k) & \text{if } t_k(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_k)+w_0)) < 1 \\ w_i^{(r+1)} = w_i^{(r)} & \text{otherwise} \end{cases}$$

Qui non posso applicare le funzioni kernel. Esistono delle tecniche che partendo da funzioni base riescono a generare funzioni base abbastanza buone.

In SVM si ragiona in termini di funzioni kernel, il che implica che si possa tornare alla funzione base.

Abbiamo sempre visto tutto in termini del fatto che gli elementi siano vettori di punti, la funzione di base proietta su un altro spazio di vettori di punti e la funzione kernel stima la distanza fra questi elementi.

I kernel hanno però una valenza più ampia, perché un kernel è una funzione che dati due elementi restituisce una misura di quanto questi si assomigliano.

Facendo così, la relazione di "vicinanza" può essere fra oggetti più strutturati:

- grafi
- stringhe

quindi la k non richiede che l'elemento originale sia un vettore, si ragiona in termini di distanza fra gli elementi ma questa distanza la definisco io. La applichiamo in questo ambito con la restrizione che però abbiamo una funzione che prende usa sequenza di basi azotate e la fa diventare un vettore di numeri in uno spazio e la distanza è fra i vettori.

Basta che siano le $\phi_i(x_i)$ dei vettori, quindi i kernel ci permettono di classificare strutture più complesse.

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

$$\phi : \mathcal{X} \mapsto \mathcal{F}$$

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

- $\mathcal{F}$ 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

matematicamente deve esserlo, così che valga ad. es il prodotto scalare.

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 $(\mathbf{x}_1, \dots, \mathbf{x}_n)$, 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 vectorsˇ.

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

$$\begin{pmatrix} k(x_1, x_2) & & \\ & \ddots & \\ & & \end{pmatrix}$$

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

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

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

$$x = (x_1, x_2) \qquad \phi = \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1x_2 \\ x_1x_2 \end{bmatrix}$$

prodotto scalare $\Rightarrow$

è una funzione kernel

Constructing kernel functions

- 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

$$\phi(\mathbf{x}) = (x_1^2, \dots, x_d^2, x_1 x_2, \dots, x_1 x_d, x_2 x_1, \dots, x_d x_{d-1})^T$$

- 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

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,

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

for

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

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 $x_{i_1} x_{i_2} \dots x_{i_l}$ with $0 \le l \le t$.

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

E' complesso: per usare una funzione kernel lo devo dimostrare ma se non lo è non so dimostrarlo.

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 $\mathbb{R}^m$

$\underset{\text{prod. scalare.}}{x_1 \cdot x_2}$ e' f. kernel, con $\phi = I$. Posso partire a comporre da questa.

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 $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)$, with $\phi_i(\mathbf{x}) = \mathbf{x}$
2. c is a kernel function corresponding to the base functions $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)$, with $\phi_i(\mathbf{x}) = \dfrac{\sqrt{c}}{n}$
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

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

è kernel ma non so come è fatta ϕ.

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, $\dfrac{\mathbf{x}_1^T\mathbf{x}_2}{\sigma^2}$ is a kernel function, being the product of a kernel function with a constant $c = \dfrac{1}{\sigma^2}$
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 $\mathbf{x}_1^T \mathbf{x}_2$.

Le più diffuse quando si parte da spazio di vettori.

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