(F94) Metodi statistici per l'apprendimento

Funzioni kernel

Docente: *Nicolò Cesa-Bianchi* versione 18 maggio 2018

I predittori lineari possono soffrire di un errore di bias molto elevato in quanto predittore il Bayesiano ottimo f^* è spesso molto lontano dall'essere lineare. Questo crea un potenziale rischio di underfitting tutte le volte che utilizziamo un algoritmo di apprendimento che genera predittori lineari.

Utilizzando una mappa non lineare per trasformare le istanze di un problema di apprendimento diventa possibile utilizzare predittori lineari per rappresentare predittori non lineari, alleviando il problema dell'underfitting. Consideriamo infatti una funzione non lineare $\boldsymbol{\phi} : \mathbb{R}^d \to \mathcal{V}$ che mappa le istanze $\boldsymbol{x} \in \mathbb{R}^d$ in un nuovo spazio $\mathcal{V}$ con un numero molto maggiore (anche infinito) di dimensioni. Costruendo la funzione $\boldsymbol{\phi}$ opportunamente, abbiamo che superfici non lineari in $\mathbb{R}^d$ vengono mappate in superfici lineari (iperpiani) in $\mathcal{V}$.

Per esempio, consideriamo la funzione $\boldsymbol{\phi} : \mathbb{R}^2 \to \mathbb{R}^6$, tale che $\boldsymbol{\phi}(x_1, x_2) = \big(1, x_1^2, x_2^2, x_1, x_2, x_1 x_2\big)$. Un iperpiano omogeneo in $\mathbb{R}^6$ con coefficienti $\boldsymbol{w} = (w_1, \dots, w_6)$ è descritto dall'insieme

$$\Big\{ \boldsymbol{z} \in \mathbb{R}^6 \,:\, \boldsymbol{w}^\top \boldsymbol{z} = 0 \Big\} \ .$$

Iperpiani di questa forma possono essere utilizzati per descrivere superfici nonlineari in $\mathbb{R}^2$. Infatti, per definizione di $\boldsymbol{\phi}$, $\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) = w_1 + w_2 x_1^2 + w_3 x_2^2 + w_4 x_1 + w_5 x_2 + w_6 x_1 x_2$. Ora notiamo che gli insiemi $\big\{ \boldsymbol{x} \in \mathbb{R}^2 \,:\, w_1 + w_2 x_1^2 + w_3 x_2^2 + w_4 x_1 + w_5 x_2 + w_6 x_1 x_2 = 0 \big\}$ descrivono curve coniche nello spazio Euclideo bidimensionale (ovvero la famiglia a cui appartengono ellissi, parabole e iperboli).

Data una generica mappa $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^N$, con $N \gg d$, consideriamo il classificatore $h : \mathbb{R}^d \to \{-1, +1\}$ definito da

$$h(\boldsymbol{x}) = \operatorname{sgn}\big(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})\big) \qquad \text{dove} \quad \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \, \phi(\boldsymbol{x})_i$$

Questo classificatore, che non è lineare in $\mathbb{R}^d$, equivale ad un classificatore lineare nello spazio $\boldsymbol{\phi}\big(\mathbb{R}^d\big) \subseteq \mathbb{R}^N$ definito come $\boldsymbol{\phi}\big(\mathbb{R}^d\big) \equiv \big\{ \boldsymbol{z} \in \mathbb{R}^N \,:\, (\exists \boldsymbol{x} \in \mathbb{R}^d)\, \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{z} \big\}$. Se $N \gg d$ il calcolo di $\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x})$, che richiede la somma di N prodotti, può però risultare computazionalmente eccessivo.

In realtà, questo problema computazionale può essere enormemente alleviato utilizzando certi tipi di mappe $\boldsymbol{\phi}$. Infatti, ricordando per esempio la regola di aggiornamento del Perceptrone $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_t \boldsymbol{x}_t$, possiamo scrivere un generico classificatore lineare h generato dal Perceptrone come

$$h(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{s \in S} y_s \boldsymbol{x}_s^\top \boldsymbol{x} \right)$$

dove S è l'insieme degli indici s di esempi $(\boldsymbol{x}_s, y_s)$ dove il Perceptrone ha eseguito un aggiornamento. Se quindi facciamo girare il Perceptrone in $\boldsymbol{\phi}\big(\mathbb{R}^d\big)$, il classificatore lineare h diventa

$$h_{\boldsymbol{\phi}}\big(\boldsymbol{x}\big) = \operatorname{sgn}\left(\sum_{s\in S} y_s \boldsymbol{\phi}(\boldsymbol{x}_s)^\top \boldsymbol{\phi}(\boldsymbol{x})\right) .$$

Per calcolare $h_{\boldsymbol{\phi}}(\boldsymbol{x})$ efficientemente basta poter calcolare ciascun termine $\boldsymbol{\phi}(\boldsymbol{x}_s)^\top \boldsymbol{\phi}(\boldsymbol{x})$ velocemente. La tecnica delle funzioni kernel ci dice come scegliere $\boldsymbol{\phi}$ in modo tale che esista una funzione $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ tale che

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}') \qquad \text{per ogni } \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d. \tag{1}$$

Per esempio, la funzione kernel corrispondente a $\boldsymbol{\phi}(x_1, x_2) = \big(1, x_1^2, x_2^2, \sqrt{2}\, x_1, \sqrt{2}\, x_2, \sqrt{2}\, x_1 x_2\big)$ è $K(\boldsymbol{x}, \boldsymbol{x}') = \big(1 + \boldsymbol{x}^\top \boldsymbol{x}'\big)^2$, come è immediato verificare.

Fissata una funzione kernel K, possiamo allora scrivere il classificatore lineare a soglia generato dal Perceptrone come

$$h_K(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{s\in S} y_s K(\boldsymbol{x}_s, \boldsymbol{x})\right) . \tag{2}$$

Si veda qui sotto per una descrizione del Perceptrone con kernel.

Algoritmo: **Perceptrone con kernel**
Inizializza S con l'insieme vuoto
Per ogni $t = 1, 2, \ldots$

1. Leggi il prossimo esempio $(\boldsymbol{x}_t, y_t)$
2. Calcola $\widehat{y}_t = \operatorname{sgn}\left(\sum_{s\in S} y_s\, K(\boldsymbol{x}_s, \boldsymbol{x}_t)\right)$
3. Se $\widehat{y}_t \neq y_t$ aggiungi t a S

La generalizzazione ovvia del kernel per apprendere coniche in $\mathbb{R}^2$ è il kernel polinomiale $K_n(\boldsymbol{x}, \boldsymbol{x}') = \big(1 + \boldsymbol{x}^\top \boldsymbol{x}'\big)^n$ con $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$. Questo kernel permette di apprendere classificatori di grado n in $\mathbb{R}^d$ rappresentandoli come iperpiani in uno spazio $\mathbb{R}^N$. Possiamo esplicitare la funzione $\boldsymbol{\phi}_n$ tale che $K_n(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}_n(\boldsymbol{x})^\top \boldsymbol{\phi}_n(\boldsymbol{x}')$ usando il teorema del binomio di Newton,

$$\big(1 + \boldsymbol{x}^\top \boldsymbol{x}'\big)^n = \sum_{k=0}^{n} \binom{n}{k} \big(\boldsymbol{x}^\top \boldsymbol{x}'\big)^k . \tag{3}$$

Osserviamo poi che vale

$$\big(\boldsymbol{x}^\top \boldsymbol{x}'\big)^k = \left(\sum_{i=1}^{d} x_i x_i'\right)^k = \sum_{\boldsymbol{v}\in\{1,\ldots,d\}^k} \left(\prod_{s=1}^{k} x_{v_s} x'_{v_s}\right) .$$

Quindi,

$$\boldsymbol{\phi}_n(\boldsymbol{x}) = \left(\sqrt{\binom{n}{k}} \prod_{s=1}^{k} x_{v_s}\right)_{k=0,\ldots,n,\ \boldsymbol{v}\in\{1,\ldots,d\}^k} . \tag{4}$$

In altre parole, il mapping $\boldsymbol{\phi}_n : \mathbb{R}^d \to \mathbb{R}^N$ associato al kernel polinomiale K_n è un vettore le cui componenti sono indicizzate da tutti i monomi $\prod_{s=1}^k x_{v_s}$ di grado al più n pesati dalle radici dei coefficienti binomiali.

Nel caso del kernel polinomiale di grado n, la dimensione N del codominio di $\boldsymbol{\phi}$ è

$$\sum_{k=0}^{n} \left|\{1,\ldots,d\}^k\right| = \sum_{k=0}^{n} d^k = \frac{d^{n+1}-1}{d-1}$$

ovvero $N = \Theta(d^n)$, esponenziale nel grado del polinomio.

Un ulteriore tipo di kernel è il kernel Gaussiano,

$$K_\gamma(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-\frac{1}{2\gamma}\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|^2\right) \qquad \gamma > 0\ .$$

Per trovare la mappa $\boldsymbol{\phi}_\gamma$ associata a K_γ procediamo come segue,

$$\begin{aligned}\exp\left(-\frac{1}{2\gamma}\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|^2\right) &= \exp\left(-\frac{1}{2\gamma}\left(\|\boldsymbol{x}\|^2 + \left\|\boldsymbol{x}'\right\|^2\right)\right)\exp\left(\frac{1}{\gamma}\left(\boldsymbol{x}^\top\boldsymbol{x}'\right)\right)\\ &= \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2\gamma}\right)\exp\left(-\frac{\|\boldsymbol{x}'\|^2}{2\gamma}\right)\sum_{n=0}^{\infty}\frac{1}{n!}\frac{\left(\boldsymbol{x}^\top\boldsymbol{x}'\right)^n}{\gamma^n}\end{aligned}$$

dove abbiamo usato l'espansione in serie di Taylor $e^x = 1 + x + \frac{x^2}{2!} + \cdots$. Osservando la formula (3) ci rendiamo conto che il kernel Gaussiano è una combinazione lineare di infiniti kernel polinomiali di grado crescente, ciascuno pesato dal reciproco del fattoriale del suo grado. Il parametro γ svolge il ruolo di fattore di scala per i prodotti $\boldsymbol{x}^\top\boldsymbol{x}'$. Infine i fattori $e^{-\|\boldsymbol{x}\|^2/(2\gamma)}e^{-\|\boldsymbol{x}'\|^2/(2\gamma)}$ normalizzano rispetto a $\boldsymbol{x}$ e $\boldsymbol{x}'$. Infatti, $K_\gamma(\boldsymbol{x},\boldsymbol{x}) = 1$ per ogni $\boldsymbol{x} \in \mathbb{R}^d$.

Notiamo che predittori della forma (2) che utilizzano kernel Gaussiani classificano un punto $\boldsymbol{x}$ calcolando una combinazione lineare, con coefficienti y_s, di funzioni Gaussiane $e^{-z^2/(2\gamma)}$ delle distanze $z = \|\boldsymbol{x}_s - \boldsymbol{x}\|$ per ogni $s \in S$. Questi classificatori sono quindi confrontabili con classificatori k-NN, dato che anche questi ultimi utilizzano la distanza dai punti di training per determinare la classificazione di un punto di test.

Il kernel Gaussiano è *universale*: per ogni $\gamma > 0$, per ogni funzione continua $f : \mathbb{R}^d \to \mathbb{R}$, e per ogni $\varepsilon > 0$, esiste $g \in \mathcal{H}_\gamma$, con

$$\mathcal{H}_\gamma = \left\{\sum_{i=1}^{N}\alpha_i\, K_\gamma(\boldsymbol{x}_i,\cdot)\ :\ \boldsymbol{x}_1,\ldots,\boldsymbol{x}_N \in \mathbb{R}^d,\ \alpha_1,\ldots,\alpha_N \in \mathbb{R},\ N \in \mathbb{N}\right\} \tag{5}$$

tale che h approssima f con errore al più ε. Questo significa che se (D,η) è un modello statistico tale che η sia continuo, allora possiamo approssimare in modo arbitrariamente accurato il classificatore Bayesiano ottimo $f^*(x) = \operatorname{sgn}\big(2\eta(x) - 1\big)$ usando funzioni della forma $\operatorname{sgn}(g)$ con $g \in \mathcal{H}_\gamma$.

Un algoritmo che genera classificatori della forma $h(\boldsymbol{x}) = \operatorname{sgn}\big(g(\boldsymbol{x})\big)$, con $g \in \mathcal{H}_\gamma$, è non parametrico. Infatti, pur essendo h un classificatore lineare, l'iperpiano $g \in \mathcal{H}_\gamma$ non è definito tramite un numero predeterminato di parametri. Mentre nel caso di kernel polinomiali di grado n, gli iperpiani sono

sempre esprimibili con un numero limitato (ancorché esponenziale in n) di coefficienti, nel caso di kernel Gaussiani gli iperpiani sono definiti tramite le combinazioni lineari

$$\sum_{i=1}^{N} \alpha_i \, K_\gamma(\boldsymbol{x}_i, \cdot)$$

dove N è sempre finito, ma arbitrariamente grande.

Dato un qualunque insieme $\mathcal{X}$ a cui appartengono le istanze $\boldsymbol{x}$ (non necessariamente $\mathbb{R}^d$), è possibile dimostrare che una qualunque funzione simmetrica $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ è un kernel se e solo se per ogni $m \in \mathbb{N}$ e per ogni $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathcal{X}$, la matrice $\boldsymbol{K}$ di dimensioni $m \times m$ e tale che $\boldsymbol{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ è positiva semidefinita. Ovvero, $\boldsymbol{z}^\top \boldsymbol{K} \boldsymbol{z} \geq 0$ per ogni $\boldsymbol{z} \in \mathbb{R}^m$. La rappresentazione della mappa $\phi_K : \mathcal{X} \to \mathcal{H}_K$ associata ad un kernel K e la rappresentazione dello spazio $\mathcal{H}_K$ non sono uniche. Però possiamo sempre rappresentare ϕ_K come $\phi_K(\boldsymbol{x}) = K(\boldsymbol{x}, \cdot)$ e $\mathcal{H}_K$ come lo spazio

$$\mathcal{H}_K \equiv \left\{ \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot) \,:\, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^d,\, \alpha_1, \ldots, \alpha_N \in \mathbb{R},\, N \in \mathbb{N} \right\}$$

equivalente a quello definito in (5) per i kernel Gaussiani.

Nel caso di kernel polinomiali K_n, sappiamo che $\mathcal{H}_K$ ha una rappresentazione equivalente come $\mathbb{R}^N$ (con $N = \Theta(d^n)$) e sappiamo scrivere $\boldsymbol{\phi}_n$ usando la formula (4). Quindi, nel caso dei kernel polinomiali, riusciamo a scrivere i classificatori lineari esplicitamente come $h(\boldsymbol{x}) = \operatorname{sgn}\big(\boldsymbol{w}^\top \boldsymbol{\phi}_n(\boldsymbol{x})\big)$, con $\boldsymbol{w} \in \mathbb{R}^N$. Per fare la stessa cosa con kernel arbitrari, per i quali non sappiamo scrivere $\boldsymbol{\phi}_K$ esplicitamente, usiamo il fatto che $K(\boldsymbol{x}, \boldsymbol{x}')$ implementa il prodotto interno fra $\phi_K(\boldsymbol{x})$ e $\phi_K(\boldsymbol{x}')$. Usando $\langle \cdot, \cdot \rangle_K$ per denotare il prodotto interno in $\mathcal{H}_K$, possiamo quindi definire

$$\big\langle K(\boldsymbol{x}, \cdot), K(\boldsymbol{x}', \cdot) \big\rangle_K = K(\boldsymbol{x}, \boldsymbol{x}')$$

Possiamo ora calcolare il prodotto interno fra $\phi_K(\boldsymbol{x}) = K(\boldsymbol{x}, \cdot)$ e un qualunque $f \in \mathcal{H}_K$,

$$\big\langle f, K(\boldsymbol{x}, \cdot) \big\rangle_K = \sum_{i=1}^{N} \alpha_i \big\langle K(\boldsymbol{x}_i, \cdot), K(\boldsymbol{x}, \cdot) \big\rangle_K = \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \boldsymbol{x}) = f(\boldsymbol{x})$$

dove abbiamo usato le proprietà di linearità che valgono per qualsiasi prodotto interno. Da ciò si vede che f calcola il prodotto interno fra se stessa e ϕ_K. Questa proprietà si chiama *reproducing* e per questa ragione $\mathcal{H}_K$ è anche chiamato un *reproducing kernel Hilbert space* (RKHS).

Di conseguenza, mentre in $\mathbb{R}^d$ il classificatore lineare era definito come $\operatorname{sgn}\big(\boldsymbol{w}^\top \boldsymbol{x}\big)$, in $\mathcal{H}_K$ il classificatore diventa $\operatorname{sgn}\big(f(\boldsymbol{x})\big)$ dove $f(\boldsymbol{x}) = \big\langle f, \boldsymbol{\phi}_K(\boldsymbol{x}) \big\rangle_K$.

Infine, vediamo come calcolare il prodotto interno $\big\langle f, g \big\rangle_K$ fra due elementi arbitrari di $\mathcal{H}_K$,

$$f = \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot) \qquad \text{e} \qquad g = \sum_{j=1}^{M} \alpha_j \, K(\boldsymbol{x}'_j, \cdot)$$

Abbiamo

$$\begin{aligned}\left\langle f, g \right\rangle_K &= \left\langle \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot), \sum_{j=1}^{M} \beta_j \, K(\boldsymbol{x}'_j, \cdot) \right\rangle_K \\ &= \sum_{i=1}^{N} \alpha_i \left\langle K(\boldsymbol{x}_i, \cdot), \sum_{j=1}^{M} \beta_j \, K(\boldsymbol{x}'_j, \cdot) \right\rangle_K \\ &= \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_i \beta_j \left\langle K(\boldsymbol{x}_i, \cdot), K(\boldsymbol{x}'_j, \cdot) \right\rangle_K \\ &= \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_i \beta_j K(\boldsymbol{x}_i, \boldsymbol{x}'_j)\end{aligned}$$

Dato che i predittori con kernel sono predittori lineari in $\mathcal{H}_K$, è possibile limitare il rischio (statistico o sequenziale) usando le tecniche utilizzate in precedenza per i predittori lineari. Il maggiorante sul rischio dipenderà dai parametri usuali, ovvero $\|\boldsymbol{w}\|$ e $\max_t \|\boldsymbol{x}_t\|$, dove però le norme sono calcolate in $\mathcal{H}_K$. In analogia con la definizione di norma di vettori $\|\boldsymbol{x}\|^2 = \boldsymbol{x}^\top \boldsymbol{x}$, la norma di $f \in \mathcal{H}_K$ è definita come $\|f\|_K^2 = \langle f, f \rangle_K$.

Questo ci dà subito il seguente risultato: $\|\phi_K(\boldsymbol{x})\|_K = \sqrt{\left\langle K(\boldsymbol{x}, \cdot), K(\boldsymbol{x}, \cdot) \right\rangle_K} = \sqrt{K(\boldsymbol{x}, \boldsymbol{x})}$.

Nel caso di un kernel K, per ogni $g \in \mathcal{H}_K$ abbiamo che

$$\|g\|_K = \left\| \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot) \right\|_K = \sqrt{\left\langle \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot), \sum_{j=1}^{N} \alpha_j \, K(\boldsymbol{x}_j, \cdot) \right\rangle_K} = \sqrt{\sum_{i,j=1}^{N} \alpha_i \alpha_j \, K(\boldsymbol{x}_i, \boldsymbol{x}_j)} \; .$$

Consideriamo un training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$ linearmente separabile in $\mathcal{H}_K$ per un qualche kernel K. Ovvero, esiste un $f \in \mathcal{H}_K$ tale che $\min_t y_t \, f(\boldsymbol{x}_t) \geq 1$. Il teorema di convergenza del Perceptrone (variante kernel) ci dà un maggiorante sul numero di aggiornamenti pari a

$$\|f\|_K^2 \max_t K(\boldsymbol{x}_t, \boldsymbol{x}_t) \; .$$

È anche possibile dimostrare una versione kernel del maggiorante sul rischio sequenziale del Perceptrone nel caso non linearmente separabile.

Un predittore lineare in uno spazio kernel può subire overfitting. Nel caso di kernel polinomiali K_n, aumentando il grado n del kernel riduciamo il training error (in quanto separiamo i dati in $\mathbb{R}^d$ con curve di grado n sempre più elevato). Se il grado n diventa troppo elevato rispetto alla dimensione del training set (e al livello di rumore nei dati) si crea quindi overfitting. Viceversa, quando $n = 1$ ricadiamo nel caso particolare di classificatori lineari e siamo quindi a rischio di underfitting.

Nel caso di kernel Gaussiani K_γ, il parametro γ corrisponde all'ampiezza delle Gaussiane centrate sui punti $\boldsymbol{x}_s$ —si veda (2). Se γ è molto piccolo rispetto alle distanze $\|\boldsymbol{x}_s - \boldsymbol{x}_t\|^2$ fra i punti di training e test, allora la classificazione di un punto $\boldsymbol{x}$ verrà essenzialmente determinata dal punto $\boldsymbol{x}_s$ ad esso più vicino. Questo permette di ottenere training error vicini a zero in quanto, analogamente

a quanto succede per 1-NN, tutti i punti di training che sono inseriti in S vengono classificati correttamente dal predittore di forma (2). Anche in questo caso, quindi, si può creare overfitting scegliendo γ troppo piccolo. Viceversa, quando γ è grande rispetto alle distanze $\|\boldsymbol{x}_s - \boldsymbol{x}_t\|^2$, allora le Gaussiane centrate sui punti $\boldsymbol{x}_s$ sono molto ampie. I classificatori diventano quindi simili a classificatori k-NN quando k è scelto vicino alla cardinalità del training set e siamo quindi a rischio di underfitting.

Abbiamo stabilito che una qualunque funzione simmetrica $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ è un kernel se e solo la matrice $\boldsymbol{K}$ è positiva semidefinita. Questo risultato è importante perché non pone condizioni sullo spazio $\mathcal{X}$ dei dati. Possiamo quindi definire kernel K su qualsiasi insieme $\mathcal{X}$: matrici, sequenze, alberi, grafi, eccetera. Il kernel può perciò essere visto come un modo di incapsulare lo spazio dei dati offrendo all'algoritmo di apprendimento un'interfaccia uniforme (ovvero, la funzione kernel). Per guidare l'intuizione nel disegno di una funzione kernel su un dato spazio $\mathcal{X}$, si può ricordare il fatto che $K(x, x')$ implementa il prodotto interno fra i due elementi $\phi_K(x)$ e $\phi_K(x')$ di uno spazio lineare $\mathcal{H}_K$, e può quindi essere visto come una misura di similarità fra x e x' in $\mathcal{X}$. Possiamo allora partire definendo una misura di similarità K su $\mathcal{X}$ e poi aggiustarne la definizione in modo che la mtrice risultante rispetti la condizione di semidefinitezza positiva.