

Approssimazione universale e kernel

Vincenzo Bonifaci

IN550 – Machine Learning

Feature e basi di funzioni

Parlando di regressione polinomiale abbiamo visto l'utilità di introdurre *feature* non lineari rispetto ai dati di input

Ad esempio, una regressione cubica

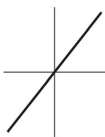
$$h(x) = w_3x^3 + w_2x^2 + w_1x + w_0 \quad (x \in \mathbb{R}^1)$$

può essere ricondotta ad una regressione lineare introducendo il *vettore delle feature* $\phi(x) = (1, x, x^2, x^3)$:

$$h(x) = w_3\phi_3 + w_2\phi_2 + w_1\phi_1 + w_0\phi_0 = w^\top \phi$$

$$\phi(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix} \in \mathbb{R}^4$$

$\{1, x, x^2, x^3\}$ può essere vista come una *base di funzioni*



$$f(x) = x$$



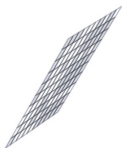
$$f(x) = x^2$$



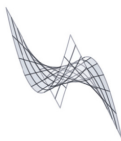
$$f(x) = x^3$$



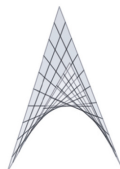
$$f(x) = x^4$$



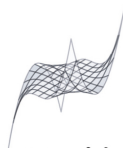
$$f(\mathbf{x}) = x_2$$



$$f(\mathbf{x}) = x_1 x_2^2$$



$$f(\mathbf{x}) = x_1 x_2$$



$$f(\mathbf{x}) = x_1^2 x_2^3$$

Combinando “unità” polinomiali $f_1(x) = x$, $f_2(x) = x^2$, $f_3(x) = x^3, \dots$ possiamo cercare di *approssimare* funzioni arbitrarie

$$y(x) \approx w_0 + w_1 f_1(x) + w_2 f_2(x) + \dots + w_B f_B(x)$$

Approssimazione universale

La base di funzioni $\{1, x, x^2, \dots, x^k, \dots\}$ ha interessanti caratteristiche:

- Ciascuna “unità” (monomio) ha una *forma fissa*: non ha nessun parametro “interno”
- Esistono infinite unità possibili:

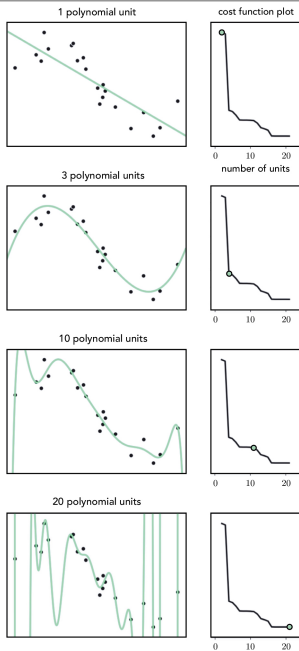
$$f_k(x) = x^k, \quad x \in \mathbb{R}^1$$

Nel caso di input d -dimensionali,

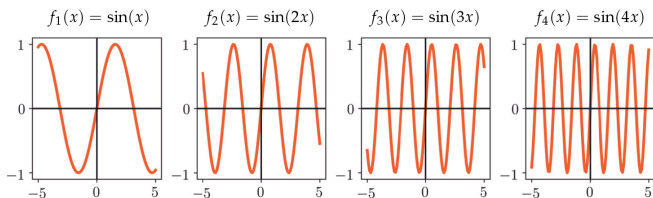
$$f_k(x) = x_1^{k_1} x_2^{k_2} \dots x_d^{k_d}, \quad x \in \mathbb{R}^d$$

- Sufficienti unità possono approssimare in maniera **arbitrariamente precisa** qualunque funzione continua (vedi *teorema di approssimazione di Weierstrass*)

⇒ I polinomi sono un esempio di *approssimatori universali* (a forma fissa)



Approssimatori di Fourier



Un altro esempio di approssimatori universali a forma fissa è dato dalla *base di Fourier*:

$$f_{2k-1}(x) = \sin(2\pi kx) \quad x \in \mathbb{R}^1$$

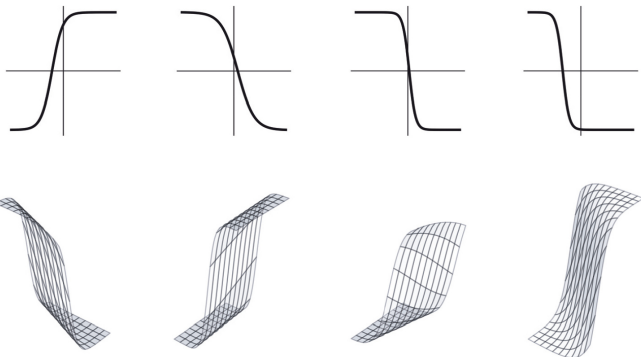
$$f_{2k}(x) = \cos(2\pi kx)$$

In d dimensioni,

$$f_k(x) = \exp(2\pi i k_1 x_1) \exp(2\pi i k_2 x_2) \dots \exp(2\pi i k_d x_d) \quad x \in \mathbb{R}^d$$

Approssimatori neurali

Anche i neuroni artificiali possono essere visti come degli approssimatori universali, ma a *forma variabile* (ogni unità ha dei parametri interni)



$$f_k(x) = \tanh(w_{k,0} + w_{k,1}x) \quad x \in \mathbb{R}^1$$

$$f_k(x) = \tanh(w_{k,0} + w_{k,1}x_1 + \dots + w_{k,d}x_d) \quad x \in \mathbb{R}^d$$

Elevato grado di approssimazione: difficoltà

Per ottenere un elevato grado di approssimazione, è necessario usare una famiglia di approssimatori altamente rappresentativa

Es.: tutti i monomi di grado $\leq D$, con D grande

Questo porta a due difficoltà:

- La varianza del modello aumenta
- La complessità computazionale dell'apprendimento aumenta

Esempi

Il numero di parametri di un modello polinomiale di grado totale D per un'input d -dimensionale è

$$n = \binom{d+D}{D} - 1 = \frac{(d+D)!}{d!D!} - 1$$

Il numero di parametri di un modello di Fourier di grado D è

$$n = (2D + 1)^d - 1$$

Elevato grado di approssimazione: difficoltà

La varianza può essere controllata:

- aumentando il numero di esempi nel training set
- utilizzando tecniche di regolarizzazione

Per combattere l'esplosione combinatoria si introduce invece il concetto di *kernelizzazione*

- Permette di lavorare nello spazio delle feature in maniera **implicita**
- Fornisce un guadagno computazionale quando n (numero di feature) è più grande di m (numero di esempi)

Feature map e spazio delle feature

Spazio di Hilbert

Uno *spazio di Hilbert* è uno spazio vettoriale (completo) dotato di un **prodotto interno** $\langle a, b \rangle$.

Feature map e spazio delle feature

Sia \mathcal{X} lo spazio degli input. Una *feature map* è una funzione

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

\times $\phi(\times)$

dove \mathcal{F} è uno spazio di Hilbert. \mathcal{F} è detto *spazio delle feature*.

Spesso (ma non sempre), $\mathcal{F} \equiv \mathbb{R}^n$ per qualche n
 Teoricamente, \mathcal{F} può essere infinito dimensionale

Kernel

→ L'osservazione cruciale (vedi oltre) è che molti metodi di apprendimento aggiornano i parametri solo in base al valore di prodotti interni della forma

$$\langle \phi(x^{(j)}), \phi(x^{(i)}) \rangle \rightarrow K(x^{(j)}, x^{(i)})$$

Kernel di ϕ

Sia $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ la funzione definita da

$$K(x, x') \stackrel{\text{def}}{=} \langle \phi(x), \phi(x') \rangle$$

per tutti gli $x, x' \in \mathcal{X}$. K è chiamato il *kernel* di ϕ .

Se sapessimo calcolare $K(x, x')$ **senza** calcolare $\phi(x)$ e $\phi(x')$, potremmo sperare in un guadagno computazionale

Esempio: Least Mean Squares

L'algoritmo di regressione Least Mean Squares, quando esamina l'esempio i -esimo, applica la regola di aggiornamento

$$w \leftarrow w - 2\eta(w^\top \phi^{(i)} - y^{(i)})\phi^{(i)}$$

Per induzione, il vettore w apparterrà allo spazio vettoriale generato da

$$\{\phi^{(1)}, \phi^{(2)}, \dots, \phi^{(m)}\}$$

quindi può essere scritto come $w = \sum_{j=1}^m \alpha_j \phi^{(j)}$ e una regola equivalente è

$$\alpha_i \leftarrow \alpha_i - 2\eta \left(\sum_{j=1}^m \alpha_j \underbrace{\langle \phi^{(j)}, \phi^{(i)} \rangle}_{K(x^{(j)}, x^{(i)})} - y^{(i)} \right)$$

Se sapessimo calcolare il kernel, potremmo lavorare in \mathbb{R}^m anziché \mathbb{R}^n

Teorema del rappresentante

Teorema del rappresentante [Representer theorem]

Si consideri un problema di ottimizzazione della forma

$$\rightarrow \underset{w}{\text{minimize}} g(\langle w, \phi^{(1)} \rangle, \dots, \langle w, \phi^{(m)} \rangle) + R(\|w\|) \quad (*)$$

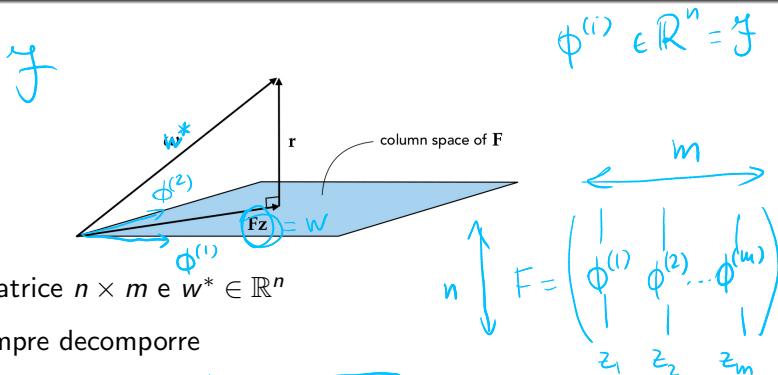
con $g : \mathbb{R}^m \rightarrow \mathbb{R}$ arbitraria e $R : \mathbb{R}_+ \rightarrow \mathbb{R}$ nondecrescente.

Allora esiste $\alpha \in \mathbb{R}^m$ tale che $w \stackrel{\text{def}}{=} \sum_{i=1}^m \alpha_i \phi^{(i)}$ è una soluzione ottima del problema.

$$(\phi^{(i)} = \phi(x^{(i)}))$$

$$\alpha = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{pmatrix} \in \mathbb{R}^m$$

Una semplice decomposizione vettoriale



Sia F una matrice $n \times m$ e $w^* \in \mathbb{R}^n$

Possiamo sempre decomporre

$$w^* = Fz + r$$

dove r è ortogonale allo spazio generato dalle colonne di F

(ed r è nullo se e solo se w^* è nello spazio generato dalle colonne di F)

Dimostrazione del teorema del rappresentante

Sia w^* una soluzione ottima del problema di ottimizzazione (*)
 Definiamo

$$F = (\phi^{(1)} \quad \phi^{(2)} \quad \dots \quad \phi^{(m)}) \in \mathbb{R}^{n \times m}$$

Applichiamo la decomposizione $w^* = F\alpha + r = \sum_{i=1}^m \alpha_i \phi^{(i)} + r$

Considerando $w = w^* - r$, osserviamo:

- $\langle r, \phi^{(i)} \rangle = 0 \quad (\forall i)$
- $\|w^*\|^2 = \|w\|^2 + \|r\|^2 \geq \|w\|^2$
- $R(\|w^*\|) \geq R(\|w\|)$
- $\langle w, \phi^{(i)} \rangle = \langle w^* - r, \phi^{(i)} \rangle = \langle w^*, \phi^{(i)} \rangle$

$$\begin{aligned} \langle w^*, w^* \rangle &= \langle w+r, w+r \rangle \\ &= \langle w, w \rangle + 2\langle w, r \rangle \\ &\quad + \langle r, r \rangle \\ &= \|w\|^2 + \|r\|^2 + 2\langle w, r \rangle \end{aligned}$$

~~$2\langle w, r \rangle$~~
0

Quindi anche w è una soluzione ottima del problema, e soddisfa
 $w = \sum_{i=1}^m \alpha_i \phi^{(i)}$.

Esempio di kernel: kernel polinomiale

$$\begin{array}{l}
 \textcircled{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} \\
 u, v \in \mathbb{R}^3
 \end{array}
 \quad
 \phi(x) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}
 \quad
 \begin{array}{l}
 w_0 \\
 w_1/\sqrt{2} \\
 w_2/\sqrt{2} \\
 \vdots \\
 w_5
 \end{array}
 \quad
 \begin{array}{l}
 \phi(u) \\
 1 \\
 \sqrt{2}u_1 \\
 \sqrt{2}u_2 \\
 u_1^2 \\
 \sqrt{2}u_1u_2 \\
 u_2^2
 \end{array}$$

$$\begin{aligned}
 \boxed{K(u, v)} &= \phi(u)^\top \phi(v) \\
 &= 1 + \underbrace{2u_1v_1}_{\leftarrow} + \underbrace{2u_2v_2}_{\leftarrow} + u_1^2v_1^2 + \underbrace{2u_1u_2v_1v_2}_{\leftarrow} + u_2^2v_2^2 \\
 &= (1 + u_1v_1 + u_2v_2)^2 \\
 &= \underbrace{(u^\top v)^2}.
 \end{aligned}$$

Per una feature map polinomiale di grado D : $\boxed{K(u, v)} = (u^\top v)^D$

Esempio di kernel: kernel di Fourier

$$\phi(x) = \begin{pmatrix} 1 \\ \sqrt{2} \cos(2\pi x) \\ \sqrt{2} \sin(2\pi x) \\ \dots \\ \sqrt{2} \cos(2\pi D x) \\ \sqrt{2} \sin(2\pi D x) \end{pmatrix} \quad \text{cos}(0 \cdot 2\pi x)$$

Si può dimostrare (vedi libro di testo) che

$$K(u, v) = \frac{\sin((2D + 1)\pi(u - v))}{\sin(\pi(u - v))}$$

Esiste anche la variante d -dimensionale della formula

Kernel come misura di similarità/correlazione

$$\phi \rightsquigarrow K(\cdot, \cdot)$$

Un kernel è (per certi versi) analogo ad una misura di similarità o di correlazione tra vettori di input:

$$K(x, x')$$

- Se $\phi(x)$ e $\phi(x')$ sono vicini, ci aspettiamo che

$$K(x, x') = \langle \phi(x), \phi(x') \rangle \text{ sia grande}$$

- Se $\phi(x)$ e $\phi(x')$ sono quasi ortogonali, ci aspettiamo che

$$K(x, x') = \langle \phi(x), \phi(x') \rangle \text{ sia piccolo}$$

Possiamo costruire un kernel valido **partendo** da una misura di similarità?

Per esempio, esiste una feature map ϕ il cui kernel coincida con

$$\text{RBF}(x, x') \stackrel{\text{def}}{=} \exp\left(-\beta \|x - x'\|^2\right) \quad ?$$

(*Radial Basis Function*; $\beta > 0$ è un iperparametro)

Caratterizzazione dei kernel validi

La risposta è data dal seguente teorema

$$K(x, x') = K(x', x)$$

Teorema di Mercer

Sia $K: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ una funzione data. Allora K è un kernel valido se e solo se, per ogni m e ogni $x^{(1)}, \dots, x^{(m)} \in \mathcal{X}$, la matrice $m \times m$

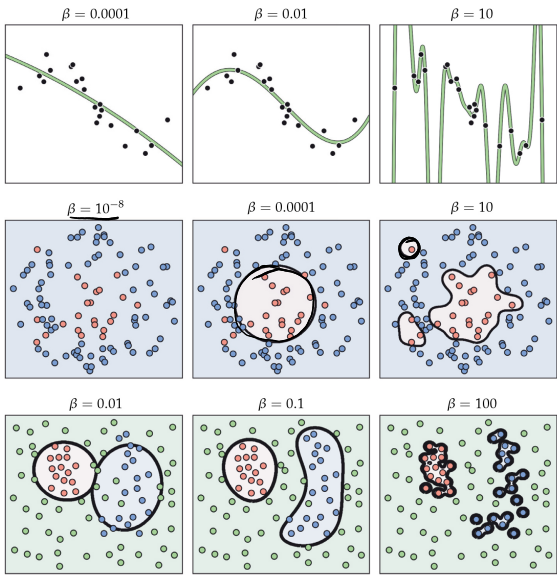
$$K \stackrel{\text{def}}{=} \begin{pmatrix} K(x^{(1)}, x^{(1)}) & K(x^{(1)}, x^{(2)}) & \dots & K(x^{(1)}, x^{(m)}) \\ K(x^{(2)}, x^{(1)}) & K(x^{(2)}, x^{(2)}) & \dots & K(x^{(2)}, x^{(m)}) \\ \dots & \dots & \dots & \dots \\ K(x^{(m)}, x^{(1)}) & K(x^{(m)}, x^{(2)}) & \dots & K(x^{(m)}, x^{(m)}) \end{pmatrix}$$

è **simmetrica** e **semidefinita positiva** (\equiv ha tutti gli autovalori ≥ 0).

Ad esempio, per la Radial Basis Function, si dimostra che la condizione è soddisfatta; quindi la RBF è un kernel valido (con uno spazio delle feature \mathcal{F} infinito-dimensionale, ma la cosa non è rilevante computazionalmente)

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

~~$$\phi(x)$$~~



Regressione

Classific. binaria

Classific. multiclasse

Esempi di regressione, classificazione binaria e classificazione multiclasse basate su Radial Basis Function con vari valori di β

Effettuare predizioni con un metodo kernelizzato

All'arrivo di un **nuovo** input x , per poter calcolare la predizione, qualunque sia l'algoritmo utilizzato occorrerà calcolare

$$K(x^{(i)}, x)$$

per ogni $x^{(i)}$ nel training set

In altri termini, per ogni singola predizione occorre confrontare x con tutti gli esempi nel training set (come succedeva in K -nearest neighbor)

Esempio: in una regressione lineare kernelizzata, la predizione sarà

$$\underbrace{w^\top \phi(x)} = \sum_{i=1}^m \alpha_i \phi^{(i)\top} \phi(x) = \sum_{i=1}^m \alpha_i K(x^{(i)}, x)$$

Conclusione sui metodi kernelizzati

Vantaggi:

- Permettono di utilizzare un alto numero n di feature **implicitamente**
- La rappresentazione del modello è in \mathbb{R}^m anziché \mathbb{R}^n
(conveniente se $\underline{m < n}$)
- La funzione kernel è definibile dall'utente (ma in tal caso deve soddisfare il criterio di Mercer)

Svantaggi:

- Ogni predizione richiede di scandire **tutto** il training set
- La rappresentazione del modello è in \mathbb{R}^m anziché \mathbb{R}^n
(sconveniente se $\underline{m > n}$)
- Non sempre il kernel $K(x, x')$ è calcolabile in modo efficiente; dipende dalla feature map ϕ