Metodi Kernel

La base dei Metodi Kernel consiste nell'impostare il problema dell'approssimazione della funzione desiderata come un problema di ottimizzazione in cui la funzione, appartenente a un certo insieme predefinito, minimizza una funzione che è data dalla somma di due termini. Il primo termine è dipendente dai dati e misura quanto ogni data funzione descriva bene i dati disponibili. Il secondo termine, che invece non dipende dai dati, esprime una misura della complessità della funzione prescelta.

Metodi Kernel per la Classificazione dei Problemi

Un problema di classificazione binaria è definito come segue:

- siano $\{x_i, t_i\}_{i=0}^{n}$ i dati disponibili
- si ha che $x_i \in R^m$ è l'i-esimo vettore di input
- e che $t_i \in \{-1, 1\}$ è il target di x_i, esso denota la classe di appartenenza dell'i-esimo esempio; 1 significa che appartiene alla classe 1 e -1 significa che appartiene alla classe 2

Metodi Kernel per la Classificazione dei Problemi (2)

I Metodi Kernel trasformano, quindi, x in uno Reproducing Kernel Hilbert Space, H tramite una funzione $\phi : R^m \rightarrow$ H e poi sviluppano un classificatore lineare in quello spazio. Un classificatore può essere come il seguente:

$$y(x) = w \cdot \phi(x) + b$$
$$y(x) > 0 \Rightarrow x \in Class1;\ y(x) < 0 \Rightarrow x \in Class2$$

L'operatore punto presente in H è chiamato Funzione Kernel, denotata come : $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ e la funzione ϕ è chiamata *feature map*. Tutte le computazioni sono effettuate utilizzando soltanto la funzione k.

Esempio di Funzioni Kernel

Facciamo un esempio:
sia $\phi(x)$ il vettore dei monomi su x di grado massimo d. Allora possiamo definire la Funzione Kernel come: $k(x_i, x_j) = (1 + x_i \cdot x_j)^d$. Questa viene chiamata Funzione Kernel Polinomiale e, per grandi valori del parametro d, la funzione diventa più flessibile e potente.
Un'altra Funzione Kernel molto usata è la Radial Basis Function Kernel (RBF Kernel) : $k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$. Anche in questo caso la flessibilità e la potenza della funzione dipendono dai valori del parametro γ.

Scelta del Classificatore

Per definire il classificatore da usare è necessario trovare il valore dei parametri (w,b), che si ottiene risolvendo il seguente problema di ottimizzazione:

$$\min_{w,b} \mathrm{E} = R + CL$$

In questo problema L è l'Errore Empirico definito come:

$$L = \sum_i l(y(x_i), t_i)$$

dove l è la funzione di perdita che descrive di quanto si allontana l'output $y(x_i)$ del classificatore dal target t_i.
Il parametro C viene sfruttato per stabilire un giusto equilibrio tra il valore di R e quello di L.

Regolarizzazione basata su margine

Il margine tra i piani definito da y(x) $= \pm 1$ è $2/\|w\|$. Rendere il margine grande equivale a far diminuire la funzione $R = \frac{1}{2}\|w\|^2$.
La funzione R viene chiamata *Regolarizzatore naturale*. Adesso possiamo riscrivere il nostro problema di ottimizzazione nella maniera seguente:

$$\min \frac{1}{2}\|w\|^2 + C \sum_i l(y(x_i), t_i)$$

Duale di Wolfe

La variabile w potrebbe risiedere in uno spazio dimensionale infinito, in questo caso è necessario maneggiare la soluzione tramite quantità dimensionalmente finite. Il Duale di Wolfe fa proprio questo. Siano w e $y(\cdot)$ definite come segue:

$$w = \sum_i \alpha_i t_i \phi_i(x_i),\ y(x) = \sum_i \alpha_i t_i k(x, x_i)$$

Soluzione diretta del Primale

Dato il Problema Primale:

$$\min \frac{1}{2}\|w\|^2 + C\sum_i l(y(x_i), t_i)$$

si sostituiscono le variabili seguenti:

$$\mathrm{w} = \sum_i \beta_i t_i \phi(x_i),\ y(x) = \sum_i \beta_i t_i k(x, x_i)$$

per ottenere questo problema di ottimizzazione:

$$\min \frac{1}{2}\sum_{i,j} t_i t_j \beta_i \beta_j k(x_i, x_j) + C\sum_i l(y(x_i), t_i)$$

In questo modo è possibile risolvere direttamente il problema per ottenere il vettore β.

Due approcci diversi

Gli approcci, chiamati Formulazioni Sparse, sono i seguenti:

- **Approccio 1:** Rimpiazzare il regolarizzatore con il seguente *regolarizzatore che induce la sparsità* $\sum_i \mid \beta_i \mid$ per ottenere il problema di ottimizzazione :

$$\min \sum_i \mid \beta_i \mid + C \sum_i l(y(x_i), t_i).$$

- **Approccio 2:** Aggiungere il regolarizzatore di sparsità $\sum_i \mid \beta_i \mid$ in maniera che sia pesato:

$$\min \lambda \sum_i \mid \beta_i \mid + \frac{1}{2} \sum_{i,j} t_i t_j \beta_i \beta_j k(x_i, x_j) + C \sum_i l(y(x_i), t_i).$$

Valori grandi di λ forzano le soluzioni sparse, mentre valori piccoli riportano il problema alle soluzioni kernel originali.

Algoritmi Per Problemi di Ottimizzazione

Sia

$$\min_{\theta \in Z} E(\theta)$$

Le tipologie dei problemi di ottimizzazione sono:

- E : Z$\rightarrow$R è *Linearmente Differenziabile*, Z $\subset R^n$
- Z $= R^n \Rightarrow$ *Non Vincolato*
- E = lineare, Z = poliedrica $\Rightarrow$ *Programmazione Lineare*
  E = quadratica, Z = poliedrica $\Rightarrow$ *Programmazione Quadratica*
  Altrimenti, *Programmazione Non Lineare*

Tra le varie tipologie di algoritmi risolutivi, parleremo del Metodo del Gradient Descent, dei Metodi di Newton e del Metodo del Gradiente Coniugato.

Metodo del Gradient Descent

Questo algoritmo sfrutta le informazioni del gradiente e, ad ogni iterazione sposta il vettore dei pesi nella direzione in cui la funzione di errore ha il massimo decremento. Una volta effettuato lo spostamento, il gradiente viene ricalcolato sfruttando il nuovo vettore dei pesi e il processo viene ripetuto.
Questo metodo ha una convergenza lineare, è molto semplice e buono localmente, ma risulta piuttosto lento e per questo poco usato in pratica.
Nel caso del problema di ottimizzazione in analisi si ha:

$$d = - \nabla E$$

Metodo di Newton

Con il metodo di Newton si costruisce la successione degli x_k per trovare la radice α di una funzione partendo da una stima iniziale x_0. La stima iniziale si suppone essere vicino alla radice. Si costruisce, quindi, la tangente della funzione in x_0 e si fa una prima approssimazione di α calcolando la radice della tangente. Ripetendo questo processo si ottiene la successione degli x_k.

Nell'usare questo metodo bisogna prima computare H $= \nabla^2 E(\theta)$, g $= \nabla E(\theta)$ e risolvere Hd $=$ -g. Per computare H si può usare la fattorizzazione di Cholesky, ovvero H $= LL'$. Il metodo di Newton potrebbe, in ogni caso, non convergere , o potrebbe perfino essere mal definito, se è stato aviato su un punto iniziale molto lontano dal minimo.

Metodo di Quasi-Newton

Questi metodi invece di calcolare la matrice Hessiana e invertirla, costruiscono un'approssimazione dell'inversa dell'Hessiano tramite una serie di passi sfruttando le informazioni del gradiente.

Metodo delle Direzioni Coniugate

Sia P una matrice simmetrica n x n, due vettori non nulli $d_1, d_2 \in R^n$ sono P-coniugati se $d_1' P d_2 = 0$.
Il metodo delle direzioni coniugate sfrutta, quindi, un insieme $\{d_0, d_1, ..d_{n-1}\}$ di direzioni P-coniugate tali che $d_i' P d_j = 0 \ \forall i \neq j$.
L'ottimo globale θ^* può essere espresso come:

$$\theta^* = \alpha_0 d_0 + ... + \alpha_{n-1} d_{n-1}$$

con adeguati coefficienti α_i. Riscritto in questo modo il problema di minimizzazione su R^n :

$$\mathrm{E}(\theta) = \tfrac{1}{2}(\textstyle\sum_{i=0}^{n-1} \alpha_i d_i)' P (\sum_{i=0}^{n-1} \alpha_i d_i) - q' (\sum_{i=0}^{n-1} \alpha_i d_i) = \sum_{i=0}^{n-1} [\tfrac{1}{2} \alpha_i^2 d_i' P d_i - \alpha_i q' d_i] = \sum_{i=0}^{n-1} E_i(\alpha_i)$$

viene trasformato in n problemi separati di minimizzazione su R, ovvero un problema per ogni $\alpha_i d_i$. Per trovare il minimo di E è necessario effettuare al massimo n ricerche.

Metodo del Gradiente Coniugato

Questo metodo sfrutta le direzioni coniugate e il gradiente per arrivare alla soluzione del problema. All'inizio si sceglie un punto iniziale θ_0 e si imposta $d_0 = -\nabla E(\theta_0)$. Ogni punto successivo viene calcolato nel modo seguente:

$$\theta_{k+1} = \theta_k + \eta_k d_k$$
$$\text{dove } \eta_k = arg\ min_\eta E(\theta_k + \eta d_k)$$

E ogni direzione successiva viene calcolata:

$$d_{k+1} = -\nabla E(\theta_{k+1}) + \beta_k d_k$$

Affinchè la nuova direzione sia coniugata con tutte le precedenti è sufficiente scegliere β_k tale che $d'_{k+1} A d_k = 0$.

Metodi di Decomposizione

Il Metodo di Decomposizione è una procedura iterativa. A ogni iterazione l'insieme delle variabili è separato in due sottoinsiemi B e N, dove B è l'insieme delle variabili di lavoro. L'insieme N, invece, viene definito in base a B come segue : $N = \{1,...,n\} \backslash B$.
L'insieme di lavoro B è molto importante. Se la cardinalità di B è grande, il numero di iterazioni necessarie sarà più piccolo ma ogni iterazione avrà un costo computazionale maggiore. D'altra parte, scegliendo una B con cardinalità più piccola, aumenta il numero di iterazioni necessarie.

Il Sottoproblema

Durante un'iterazione viene, quindi, risolto il sotto-problema seguente:

$$\begin{min}{\alpha_\beta} \frac{1}{2}[\alpha'_\beta (\alpha_N^k)'] \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix} - [e'_B (e_N^k)'] \begin{bmatrix} \alpha_B \\ \alpha_N^k \end{bmatrix} \\ \text{s.t. } 0 \leq \alpha_I \leq C,\ I \in B, \\ t'_B \alpha_B = -t'_N \alpha_N^k$$

e la funzione obiettivo viene riscritta come:

$$\frac{1}{2}\alpha'_B Q_{BB} \alpha_B + (-e_B + Q_{BN}\alpha_N^k)'\alpha_B + \text{costante} .$$

Grazie a questa costruzione è possibile evitare problemi di memoria in quanto, ad ogni iterazione, sono necessarie solo B colonne della matrice Q che vengono, quindi, calcolate solo quando necessario.

Algoritmo del Metodo di Decomposizione

Mostriamo ora i passi dell'algoritmo del Metodo di Decomposizione:

1. Trovare una possibile α^1
   Impostare k=1
2. Se α^k soddisfa le condizioni di ottimalità, stop.
   Altrimenti trovare il set di lavoro B.
   Definire N = {1,...,n} \B
3. Risolvere il sotto-Problema per α_B :
   $\min_{\alpha_B} \frac{1}{2}\alpha_B' Q_{BB}\alpha_B + (-e_B + Q_{BN}\alpha_N^k)'\alpha_B$
   s.t. $0 \leq \alpha_I \leq C$, $I \in B$, $t_B'\alpha_B = -t_N'\alpha_N^k$
4. $\alpha_N^{k+1} \equiv \alpha_N^k$.
   Impostare k = k+1;
   Tornare al passo 2.

Sequential Minimal Optimization

Per i Metodi di Decomposizione la parte cruciale consta nella scelta della dimensione di B.
Consideriamo $\mid B \mid = 2$ che è un caso estremo visto che $\mid B \mid \geq 2$ per soddisfare il vincolo di linearità. Il sotto-problema corrispondente può essere risolto analiticamente, senza il bisogno di utilizzare software di ottimizzazione specifici. Infatti si ha:

$$\begin{gathered} \min_{\alpha_i \alpha_j} \frac{1}{2}[\alpha_i \alpha_j] \begin{bmatrix} Q_{i,i} & Q_{i,j} \\ Q_{i,i} & Q_{j,j} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + (Q_{B,N}\alpha_N^k - e_B)' \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} \\ \text{s.t. } 0 \leq \alpha_i, \alpha_j \leq C, \\ t_i\alpha_i + t_j\alpha_j = -t_N'\alpha_N^k, \end{gathered}$$

In questo modo l'insieme B = {i,j} è composto solo dalle due colonne i, j, e, di conseguenza, non c'è bisogno di una condizione di fermata per l'agoritmo.

Condizione KKT di Ottimalità

Per le condizioni di ottimalità di KKT si ha che α è ottimale se e soltanto se:

$$\nabla f(\alpha) + bt = \lambda - \mu,$$
$$\lambda_i \alpha_i = 0,$$
$$\mu_i (C - \alpha_i) = 0 \ ,$$
$$\lambda_i \geq 0, \ \mu_i \geq 0, \ \text{i=1,...,n}$$
$$\text{dove } \nabla f(\alpha) \equiv Q\alpha - e$$

Questa condizione può essere riscritta come:

$$\nabla f(\alpha)_i + bt_i \geq 0 \qquad \text{se } \alpha_i < C$$
$$\nabla f(\alpha)_i + bt_i \leq 0 \qquad \text{se } \alpha_i > 0$$

Da notare che la variabile $t_i = \pm 1$.

Condizione KKT di Ottimalità (2)

A questo punto definiamo i seguenti insiemi:

$$I_{up}(\alpha) \equiv \{I \mid \alpha_I < C, t_I = 1 \text{ oppure } \alpha_I > 0, t_I = -1\},$$
$$I_{low}(\alpha) \equiv \{I \mid \alpha_I < C, t_I = -1 \text{ oppure } \alpha_I > 0, t_I = 1\}$$

Tutto questo per giungere alla conclusione che la variabile α è ottimale quando esiste e soddisfa i seguenti vincoli:

$$\max_{i \in I_{up}(\alpha)} -t_i \nabla f(\alpha)_i \leq \min_{j \in I_{low}(\alpha)} -t_j \nabla f(\alpha)_j$$

Coppia Violante

Una coppia che viola la condizione di ottimalità è tale quando:

$$i \in I_{up}(\alpha),\ j \in I_{low}(\alpha),\ \text{e si ha } -t_i \nabla f(\alpha)_i > -t_j \nabla f(\alpha)_j$$

La funzione $f(\alpha^k)$ è strettamente decrescente se e soltanto se B ha almeno una coppia violante. Scegliere una coppia violante, però, non è sufficiente per ottenere la convergenza. Per questo motivo viene scelta la coppia violante massimale definita come:

$$i \in arg \max_{I \in I_{up}(\alpha^I)} -t_I \nabla f(\alpha^k)_I,$$
$$j \in arg \min_{I \in I_{low}(\alpha^I)} -t_I \nabla f(\alpha^k)_I\ .$$

La coppia {i,j} può essere calcolata in O(n) operazioni.

Calcolo del Gradiente

Per risolvere i problemi legati al gradiente si eseguono i seguenti passi:

1. $\alpha^1 = 0$ implica che $\nabla f(\alpha) = Q \cdot 0 - e = -e$
2. aggiornare $\nabla f(\alpha)$ usando soltanto Q_{BB} e Q_{BN} :

$$\nabla f(\alpha^{k+1}) = \nabla f(\alpha^k) + Q(\alpha^{k+1} - \alpha^k) = \nabla f(\alpha^k) + Q_{:,B}(\alpha^{k+1} - \alpha^k)$$

In questo modo servono soltanto $\mid B \mid$ colonne di Q ad ogni iterazione per calcolare il gradiente.

Insieme Attivo

Siano le α_i le variabili del problema, esse vengono separate in tre gruppi:

- Gruppo O: $\alpha_i = 0$
- Gruppo C: $\alpha_i =$ Costante
- Gruppo A: solo le α_i appartenenti a questo insieme possono cambiare

$$\alpha = (\alpha_A, \alpha_C, \alpha_O),\ \alpha_C = Ce_C,\ \alpha_O = 0$$

Il problema di ottimizzazione che usa solo le variabili dell'insieme α_A è:

$$\begin{gathered} \min_{\alpha_B} \ \tfrac{1}{2}\alpha_A' Q_{AA}\alpha_A + (-e_A + CQ_{CA}e_C)'\alpha_A \\ \text{s.t. } 0 \leq \alpha_I \leq C,\ I \in A \\ t_A'\alpha_A = -Ct_C'e_C, \end{gathered}$$

Algoritmo dell'Insieme Attivo

Spieghiamo adesso come funziona questo metodo. Le iterazioni principali del metodo dell'Insieme Attivo sono:

- Risolvere il problema quadratico equamente vincolato che è stato specificato prima.
- Se la soluzione α_A viola un vincolo, spostare la prima variabile $i \in A$ che viola il vincolo, nell'insieme C o O.
- Se la soluzione α_A soddisfa tutti i vincoli, controllare se c'è una variabile in C o O che viola le condizioni (KKT) di ottimalità, se esiste spostarla in A.

Path Tracking

Consideriamo il seguente problema di ottimizzazione:

$$\min_{\beta} f(\beta) = \lambda J(\beta) + L(\beta)$$

dove $J(\beta) = \sum_j \mid \beta_j \mid$ e L è una funzione quadratica convessa e differenziabile a tratti.

Condizioni di Ottimalità PT

Sia g $= \nabla L$, e definiamo anche $\beta(\lambda)$ come minimizzatore di un qualsiasi λ, si viene a creare l'insieme A={ j: $\beta_j(\lambda) \neq 0$ } e A^c che è il suo complementare. Sotto queste ipotesi le condizioni di ottimalità diventano:

$$g_j + \lambda sgn(\beta_j) = 0 \ \forall j \in A \qquad (1)$$
$$| \ g_j \ | \leq \lambda \ \forall j \in A^c \qquad (2)$$

La condizione (1) definisce, entro una zona quadratica, un insieme di equazioni lineari β_j, $j \in A$. Chiamiamo γ la direzione nello β spazio così definita.
Per grandi valori di λ la soluzione è $\beta = 0$.
La funzione $L(\beta)$, invece, viene definita come segue:

$$L(\beta) = \frac{1}{2}\|\beta\|^2 + \sum_i l(y(x_i), t_i)$$

dove $y(x) = \beta' x$ e l è una funzione differenziabile e quadratica a tratti di perdita.

Algoritmo di Rosset-Zhu

I passi principali dell'algoritmo sono:

1. Inizializza $\beta = 0$, A = arg $max_j \mid g_j \mid$
2. ricava γ
3. while (max $\mid g_j \mid > 0$)
   - $d_1 = arg\ min_{d \geq 0}\ min_{j \in A^c} \mid g_j(\beta + d\gamma) \mid = \lambda + d$ (1).
   - $d_2 = arg\ min_{d \geq 0}\ min_{j \in A}\ (\beta + d\gamma)_j = 0$ (2) (una componente attiva è pari a 0).
   - Trovare d_3, ovvero il primo valore di d al quale il confine di una zona quadratica a tratti è violato.
   - Imposta d = min (d_1, d_2, d_3).
   - Se $d = d_1$ allora aggiungi la variabile che soddisfa (1) all'insieme A.
   - Se $d = d_2$ allora rimuovi la variabile che soddisfa (2) dall'insieme A.
   - $\beta \leftarrow \beta + d\gamma$
   - Aggiorna tutte le informazioni e calcola il nuovo vettore direzione γ

Metodo Finito di Newton

Diamo ora la definizione del problema di ottimizzazione riguardante il Metodo Finito di Newton:

$$\min_{\beta} f(\beta) = \frac{1}{2}\beta' R\beta + \sum_i l_i(\beta)$$

dove R è una matrice definita postiva e l_i è la funzione di perdita dell'i-esimo esempio. Possiamo assumere anche che l_i sia una funzione quadratica convessa e differenziabile a tratti (come menzionato nel Path Tracking).

Generica iterazione del metodo

Sia $\beta^k =$ il vettore iniziale all'iterazione k-esima, e sia $q_i =$ il valore della funzione quadratica l_i in β^k . Allora il problema risulta essere:

$$\bar{\beta} = \underset{\beta}{arg\,min}\ \tfrac{1}{2}\beta' R\beta + \textstyle\sum_i q_i(\beta)$$

Da notare che per ottenere $\bar{\beta}$ bisogna risolvere un sistema lineare.
Si definisce ora la direzione $d^k = \bar{\beta} - \beta^k$ e, successivamente, si calcola il prossimo punto come segue:

$$\beta^{k+1} = \beta^k + \delta^k d^k,$$
$$\text{dove } \delta^k = \underset{\delta}{arg\,min}\ f(\beta^k + \delta d^k)$$

Selezione Nei Kernel Lineari

Si inizia con nessuna variabile scelta e si aggiungono variabili in maniera greedy.
Sia $\beta =$ un vettore ottimo con un insieme di variabili corrente e sia β_j una variabile ancora non scelta. Il criterio di scelta è il seguente:

$$\bar{f}_j = \underset{\beta_j}{arg\,min}\ f(\beta, \beta_j), \text{ con } \beta \text{ fissata}$$

dove f è la funzione costo di training.
Si sceglie quindi la β_j con il valore di f_j più piccolo. Una volta scelta la migliore j, si risolve il $min(\beta, \beta_j)$ usando come valori iniziali $(\beta, 0)$.

Selezione nei Kernel Non Lineari Sparsi

L'idea alla base è simile a quella esposta in precedenza.
Si sfrutta la sostituzione primale $w = \beta_i t_i \phi(x_i)$ per ottenere il seguente problema di ottimizzazione:

$$\min\ L(\beta) = \frac{1}{2}\beta' Q\beta + \sum_i l(y(x_i), t_i)$$

dove $y(x) = \sum_i \beta_i t_i k(x, x_i)$.
Ad ogni passo del processo greedy è sufficiente restringere la valutazione su un piccolo sottoinsieme casuale delle β_j.

FINE