Ottimizzazione dei Metodi Kernel

Jacek Filipczuk

Luglio 2013

Indice

1 Introduzione

In questo lavoro si parlerà di metodi adattivi per la risoluzione di problemi di approssimazione di funzioni a partire da un numero finito di esempi.
Questi metodi sono noti come Metodi Kernel e vengono sfruttati nella risoluzione di una vasta tipologia di problemi.
Nei seguenti capitoli saranno affrontati problemi di ottimizzazione che nascono dalle soluzioni del noto problema della Classificazione dei Problemi.
Inizialmente verrà fornita una definizione dei Metodi Kernel, seguita da un analisi dei metodi più usati. I capitoli successivi, invece, spiegheranno gli argomenti riguardanti i Metodi Duali e i Metodi Primali.

2 Metodi Kernel e i Problemi di Ottimizzazione

In questo capitolo verrranno spiegate le definizioni basilari necessarie per la comprensione degli argomenti trattati successivamente.

2.1 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. Risulta chiaro che la minimizzazione del solo primo termine all'interno di un insieme predefinito molto grande porta a un'ottima capacità descrittiva, grazie anche all'impiego di una funzione molto complesssa. La minimizzazione del solo secondo termine, invece, consiste nella costruzione di una funzione molto semplice il cui difetto è quello di non descrivere molto bene i dati disponibili. In entrambi i casi, il risultato finale è una funzione dotata di scarso potere di generalizzazione. La scelta migliore, per avere un apprendimento efficace, è avere un corretto bilanciamento tra i due parametri durante la minimizzazione, e questo si può ottenere grazie ad un parametro aggiuntivo, detto di Regolarizzazione, che svolge proprio il compito di bilanciamento.
Un'altra importante caratteristica dei Metodi Kernel è data dal fatto che la misura di complessità può essere costruita ad hoc tenendo conto della specificità dei dati e del problema. L'uso della misura più appropriata corrisponde alla scelta della migliore rappresentazione dei dati, ovvero della migliore funzione kernel.
La scelta di una funzione kernel corrisponde alla ricerca di una rappresentazione dei dati in uno spazio dotato di un grande numero di dimensioni. In questo spazio è possibile controllare la complessità delle soluzioni perchè è possibile restringere l'insieme delle funzioni predefinite a funzioni lineari. Nella pratica le soluzioni possono essere arbitrariamente complesse.
Infine citiamo due Metodi Kernel molto comuni:

- Support Vector Machines (SVM): basati sull'uso di un termine dipendente dai dati e derivato da considerazioni di statistica.
- Kernel Logistic Regression (KLR) : sono una forma non-lineare di regressione logistica.

2.2 Metodi Kernel per la Classificazione dei Problemi

I Metodi Kernel sono sfruttati nella risoluzione di un vasto insieme di problemi come la classificazione (binaria/multiclasse), la regressione, la regressione ordinale, il ranking e l'apprendimento non supervisionato. In questa sezione ci concentreremo solo sui problemi di classificazione binaria.
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

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 : $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. Tutte le computazioni sono effettuate utilizzando soltanto la funzione k.
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.
Per definire il classificatore da usare è necessario trovare il valore dei parametri (w,b), che si ottiene risolvendo il seguente problema di ottimizzazione:

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

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.
A questo punto è facile pensare che la cosa più importante sia minimizzare L, questo però porta a un overfitting dei dati. La funzione di regolarizzazione R aiuta a prevenire l'overfitting e a rendere il modello più semplice.
Infine il parametro C viene sfruttato per stabilire un giusto equilibrio tra il valore di R e quello di L. C viene chiamato *hyperparametro*.

2.2.1 Funzioni di perdita

Alcune delle funzioni di perdita più comuni sono:

- *SVM (Hinge) loss* : l(y,t) = 1 - ty se $ty < 1$; 0 altrimenti .
- *KLR (Logistic) loss* : l(y,t) = -log(1 + $e^{(-ty)}$) .
- *L_2-SVM loss* : l(y,t) = $(1 - ty)^2/2$ se $ty < 1$; 0 altrimenti .
- *Modified Huber loss* : l(y,t) vale 0 se $\xi \geq 0$; $\xi^2/2$ se $0 < \xi < 2$; infine $2(\xi - 1)$ se $\xi \geq 2$, dove ξ = 1 - ty

2.2.2 Regolarizzazione basata su margine

Il margine tra i piani definito da y(x) = ± 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:

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

2.2.3 Duale di Wolf

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 Wolf fa proprio questo. Siano w e $y(\cdot)$ definite come segue:

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

allora le funzioni kernel precedentemente esposte vengono riscritte nella maniera seguente:

- **SVM dual:**

$$\begin{gathered}\min E(\alpha) = \frac{1}{2}\sum_{i,j} t_i t_j \alpha_i \alpha_j k(x_i, x_j) - \sum_i \alpha_i \\ \text{s.t. } 0 \leq \alpha_i \leq C,\ \sum_i t_i \alpha_i = 0\end{gathered}$$

- **KLR dual:**

$$\begin{gathered}\min\ E(\alpha) = \tfrac{1}{2}\textstyle\sum_{i,j} t_i t_j \alpha_i \alpha_j k(x_i, x_j) + C\sum_i g(\tfrac{\alpha_i}{C}) \\ \text{s.t. } \textstyle\sum_i t_i \alpha_i = 0 \\ \text{where } g(\delta) = \delta \log \delta + (1-\delta)\log(1-\delta)\end{gathered}$$

- **L_2-SVM dual:**

$$\begin{gathered}\min\ E(\alpha) = \tfrac{1}{2}\textstyle\sum_{i,j} t_i t_j \alpha_i \alpha_j \tilde{k}(x_i, x_j) - \sum_i \alpha_i \\ \text{s.t. } \alpha_i \geq 0,\ \textstyle\sum_i t_i \alpha_i = 0 \\ \text{where } \tilde{k}(x_i, x_j) = k(x_i, x_j) + \tfrac{1}{C}\delta_{ij}\end{gathered}$$

2.2.4 Soluzione diretta del Primale

Nel caso in cui la funzione di perdità è differenziabile, come nei metodi KLR e L_2-SVM, un approccio migliore potrebbe essere quello di risolvere direttamente il primale del problema.
Dato il Problema Primale:

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

si sostituiscono le variabili seguenti:

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

per ottenere questo problema di ottimizzazione:

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

In questo modo è possibile risolvere direttamente il problema per ottenere il vettore β. Inoltre ci sono due approcci per minimizzare il numero dei parametri α diversi da zero.
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 :

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

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

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

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

3 Algoritmi per Problemi di Ottimizzazione

In questo capitolo saranno presentati alcuni algoritmi per la risoluzione dei problemi di ottimizzazione. Verranno elencate le varie tipologie di problemi di ottimizzazione con alcuni algoritmi risolutivi più usati.
Sia

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

Le tipologie dei problemi di ottimizzazione sono:

- E : Z→R è *Linearmente Differenziabile*, Z $\subset R^n$
- Z = R^n ⇒ *Non Vincolato*
- E = lineare, Z = poliedrica ⇒ *Programmazione Lineare*
  E = quadratica, Z = poliedrica ⇒ *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.

3.1 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:

$$\mathrm{d} = - \nabla \mathrm{E}$$

3.2 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.
Nel caso del problema di ottimizzazione in analisi si ha:

$$\mathrm{d} = -[\nabla^2 E]^{-1} \nabla E$$

Questo metodo può essere interpretato in vari modi:

- $\theta + d$ minimizza l'approssimazione di secondo ordine

$$\hat{E}(\theta + d) = E(\theta) + \nabla E(\theta)'d + \tfrac{1}{2}d'\nabla^2 E(\theta)d$$

- $\theta + d$ risolve la condizione di ottimalità linearizzata

$$\nabla E(\theta + d) \approx \nabla \hat{E}(\theta + d) = \nabla E(\theta) + \nabla^2 E(\theta)d = 0$$

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.

3.2.1 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.

3.3 Metodo delle Direzioni Coniugate

Per problemi di grandi dimensioni. si desidera rapidità di convergenza maggiore del metodo del gradiente e complessità computazionale minore del metodo di Newton, ovvero senza dover invertire matrici. Poichè nell'intorno di un ottimo locale/globale una qualsiasi funzione è data da un'approssimazione quadratica, si considerano funzioni quadratiche:

$$\min \mathrm{E}(\theta) = \tfrac{1}{2}\theta' P\theta - q'\theta$$

con P matrice simmetrica definita positiva (di cui l'ottimo globale θ^* è l'unica soluzione di $P\theta = q$) e poi si estende l'approccio a funzioni generiche. 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.

3.4 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 $d'_{k+1} A d_k = 0$.

3.5 Paragone tra i metodi

Tra i vari metodi esposti , quelli di Quasi-Newton sono robusti, ma utilizzano O(n^2) spazio di memoria per salvare l'approssimazione dell'inversa della matrice Hessiana, e di conseguenza sono inadatti per problemi di grandi dimensioni. D'altra parte i metodi del Gradiente Coniugato sono ben strutturati e lavorano bene con problemi di grandi dimensioni ma necessitano di essere implementati con molta attenzione. In alcune situazioni, invece, è preferibile usare il metodo del Block Coordinate Descent.Questo metodo consiste, semplicemente, nell'ottimizzare un sottoinsieme delle variabili alla volta.

4 Metodi di Decomposizione

4.1 Problema Duale SVM

Riprendiamo in esame il Problema Duale SVM che è definito come:

$$\begin{gathered}\min \tfrac{1}{2}\sum_{i,j} t_i t_j \alpha_i \alpha_j k(x_i, x_j) - \sum_i \alpha_i \\ \text{s.t. } 0 \leq \alpha_i \leq C,\ \sum_i t_i \alpha_i = 0\end{gathered}$$

e riscriviamolo in notazione matriciale come segue:

$$\begin{gathered}\min_{\alpha} \tfrac{1}{2}\alpha' Q\alpha - e'\alpha \\ \text{s.t. } 0 \leq \alpha_i \leq C,\ \text{i=1,...,n} \\ t'\alpha = 0, \\ \text{dove } Q_{i,j} = t_i t_j k(x_i, x_j) \text{ e il vettore } e = [1, ..., 1]' \ .\end{gathered}$$

In questa formulazione la matrice $Q_{i,j} \neq 0$ è piena. Per questo motivo lo spazio occupato da Q è grande, e metodi come quelli di Newton o Quasi-Newton non possono essere direttamente usati perchè richiedono uno spazio di memoria pari a $O(n^2)$. Anche il metodo del Gradiente Coniugato, che richiede uno spazio di memoria pari a *O(n)*, non può essere utilizzato perchè $\nabla f = Q\alpha - e$ richiede l'uso dell'intero kernel della matrice.
E proprio in queste situazioni che vengono sfruttati i Metodi di Decomposizione.

4.2 Metodo 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}\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.
Durante un'iterazione viene, quindi, risolto il sotto-problema seguente:

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

e la funzione obiettivo viene riscritta come:

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

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.

4.3 Algoritmo del Metodo di Decomposizione

Questo algoritmo risulta avere una velocità di convergenza inferiore rispetto al Metodo di Newton o al Metodo di Quasi-Newton ma, in compenso, la variabile α non deve essere molto accurata, molte α_i restano a zero tutto il tempo e questo rende la computazione più efficiente. 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 :
   $\min\limits_{\alpha_B} \frac{1}{2}\alpha'_B Q_{BB}\alpha_B + (-e_B + Q_{BN}\alpha^k_N)'\alpha_B$
   s.t. $0 \leq \alpha_l \leq C$, $l \in B$, $t'_B\alpha_B = -t'_N\alpha^k_N$

4. $\alpha^{k+1}_N \equiv \alpha^k_N$.
   Impostare k = k+1;
   Tornare al passo 2.

4.4 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:

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

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.

4.4.1 Violazione dei vincoli KKT

Un modo per scegliere le colonne i,j dell'insieme B è quello di sfruttare le condizioni di Karush-Kuhn-Tucker. Le variabili che violano tali condizioni vengono scelte per far parte dell'insieme B.
Ricordiamo che il nostro problema di ottimizzazione è:

$$\begin{aligned} &\min_{\alpha} \text{ f}(\alpha) = \tfrac{1}{2}\alpha' Q\alpha - e'\alpha, \\ &\text{s.t. } t'\alpha = 0,\ 0 \le \alpha_i \le C\ . \end{aligned}$$

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

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

Questa condizione può essere riscritta come:

$$\begin{aligned} \nabla f(\alpha)_i + bt_i \ge 0 \qquad & \text{se } \alpha_i < C \\ \nabla f(\alpha)_i + bt_i \le 0 \qquad & \text{se } \alpha_i > 0 \end{aligned}$$

Da notare che la variabile $t_i = \pm 1$.
A questo punto definiamo i seguenti insiemi:

$$\begin{aligned} I_{up}(\alpha) &\equiv \{l \mid \alpha_l < C, t_l = 1 \text{ oppure } \alpha_l > 0, t_l = -1\}, \\ I_{low}(\alpha) &\equiv \{l \mid \alpha_l < C, t_l = -1 \text{ oppure } \alpha_l > 0, t_l = 1\} \end{aligned}$$

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

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

4.4.2 Coppia Violante

Una volta che è stata definita la condizione di ottimalità, bisogna trovare la coppia di variabili che viola tale condizione.
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:

$$\begin{aligned} i &\in arg \max_{l \in I_{up}(\alpha^l)} -t_l \nabla f(\alpha^k)_l, \\ j &\in arg \min_{l \in I_{low}(\alpha^l)} -t_l \nabla f(\alpha^k)_l\ . \end{aligned}$$

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

4.4.3 Calcolo del Gradiente

Per calcolare la coppia violante è necessario calcolare prima il gradiente. Il calcolo del gradiente, però, comporta dei problemi di memoria in quanto $\nabla f(\alpha) = Q\alpha - e$ implica l'uso di Q. 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.

5 Programmazione Quadratica

Passiamo ora a parlare della Programmazione Quadratica e di un metodo in particolare, il metodo dell'Insieme Attivo.

5.1 Metodo dell'Insieme Attivo

Per prima cosa diamo una definizione di un Insieme Attivo per poi spiegarlo nel dettaglio.
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:

$$\begin{aligned} \min_{\alpha_B}\ & \tfrac{1}{2}\alpha_A' Q_{AA}\alpha_A + (-e_A + CQ_{CA}e_C)'\alpha_A \\ & \text{s.t. } 0 \le \alpha_l \le C,\ l \in A \\ & t_A'\alpha_A = -Ct_C'e_C, \end{aligned}$$

Questo problema è uguale all'originale tranne per il fatto che si basa su un insieme di variabili più piccolo. Supponiamo che tutte le variabili soddisfano il seguente vincolo: $0 < \alpha_i < C$, $i \in A$, e risolviamo il problema che prende il nome di problema quadratico equamente vincolato:

$$\begin{aligned} \min_{\alpha_B}\ & \tfrac{1}{2}\alpha_A' Q_{AA}\alpha_A + (-e_A + CQ_{CA}e_C)'\alpha_A \\ & \text{s.t. } 0 \le \alpha_l \le C,\ l \in A \\ & t_A'\alpha_A = -Ct_C'e_C, \end{aligned}$$

La soluzione di questo problema può essere ottenuta risolvendo un sistema lineare $H\gamma = g$ dove γ include sia le α_A sia le b. Una fattorizzazione di H viene mantenuta e aggiornata durante le iterazioni dell'algoritmo.
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.

Questo algoritmo può essere inizializzato scegliendo un insieme di partenza piccolo, ad esempio solo due variabili. Con appropriati criteri di scelta per le variabili da aggiungere, l'algoritmo dimostra di avere una convergenza finita. Inoltre, come è facile intuire, il punto cruciale dell'algoritmo è il metodo usato per scegliere dagli insiemi C o O la variabile da inserire nell'Insieme Attivo. L'algoritmo originale sceglie la variabile che viola maggiormente le condizioni di ottimalità (KKT), ma questo può risultare costoso. Una possibile variante è quella che visita in modo casuale gli indici e sceglie la prima variabile che viola le condizioni di ottimalità.

5.1.1 Considerazioni sul Metodo dell'Insieme Attivo

Spieghiamo brevemente i vantaggi e gli svantaggi di questo metodo:

- **Vantaggi:** convergenza finita, e quindi indipendente dalla tolleranza introdotta dalle condizioni di fermata, buoni risultati quando l'insieme A non è più grande di poche migliaia e, infine, si adatta molto bene quando vengono usati metodi basati sul gradiente per la selezione del modello.
- **Svantaggi:** la memorizzazzione delle variabili e la fattorizzazione di H possono risultare troppo costose quando la dimensione di A supera alcune migliaia, inoltre la fattorizzazione richiede di essere eseguita più volte.

6 Metodi di Path Tracking

In questo capitolo ci occuperemo dei Metodi di Path Tracking.

6.1 Formulazione Generale del Problema

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. Dire che L è differenziabile a tratti significa che il β-spazio è suddiviso in un finito numero di zone, in ognuna di esse L è una funzione quadratica convessa e, ai bordi delle zone, i pezzi di L combaciano perfettamente per mantenere la differenziabilità.
L'obiettivo, in questo caso, è di trovare un minimizzatore per f in funzione di λ.
Sia g = ∇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:

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

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 , e più precisamente quando si ha $\lambda > max_j \mid g_j(0) \mid$, la soluzione è $\beta = 0$.
La funzione $L(\beta)$, invece, viene definita come segue:

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

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

6.2 Algoritmo di Rosset-Zhu

Il seguente è un algoritmo di Path Tracking.
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

6.3 Classificatori di Kernel Non-Lineare

Consideriamo ora un problema primale di kernel non-lineare:

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

dove l è una funzione quadratica differenziabile a tratti di perdita.
Effettuiamo la semplice sostituzione $w = \beta_i t_i \phi(x_i)$ per ottenere :

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

dove $y(x) = \sum_i \beta_i t_i k(x, x_i)$.
Con queste modifiche quando il minimo di $f = \lambda J + L$ è calcolato rispetto a λ, si ha che $\beta = 0$ per λ di grandi valori, e si ottiene il minimizzatore di L quando $\lambda \to 0$. Valori intermedi per λ generano approssimazioni dove è presente sparsità.

7 Metodo Finito di Newton

In alcune situazioni il metodo Finito di Newton risulta essere più efficiente e meglio strutturato rispetto al metodo della decomposizione duale.

7.1 FNM: Formulazione Generale

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

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

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).
Ci soffermiamo ora su una generica iterazione del metodo per rendere chiaro il suo funzionamento.
Sia β^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:

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

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:

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

Si dimostra che anche questo metodo gode della convergenza finita.
Per quanto riguarda la selezione delle variabili, esistono due metodi principali : Selezione nei Classificatori di Kernel Lineari e quella nei Kernel Non Lineari Sparsi.

7.1.1 Selezione nei Kernel Lineari

Si inizia con nessuna variabile scelta e si aggiungono variabili in maniera greedy.
Sia β = 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:

$$\bar{f_j} = \underset{\beta_j}{argmin} \ f(\beta, \beta_j), \text{ con } \beta \text{ fissata}$$

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 $(\beta, 0)$.

7.1.2 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 $y(x) = \sum_i \beta_i t_i k(x, x_i)$.
Da notare che il problema è essenzialmente nella forma di un classificatore lineare, tranne che per il regolarizzatore ($\beta' Q\beta \backslash 2$) che è nella forma di una generica funzione quadratica.
Le nuove variabili β_j diverse da zero posssono essere scelte in maniera greedy, come nell'approccio precedente. Ad ogni passo del processo greedy è sufficiente restringere la valutazione su un piccolo sottoinsieme casuale delle β_j. Il risultato finale è un algoritmo efficace con complessità pari a $O(d^2 n)$ nella selezione di d funzioni kernel di base.
Su numerosi dataset, un piccolo valore di d garantisce quasi la stessa accuratezza di un classificatore kernel pieno che sfrutta tutte le funzioni di base.