UNIVERSITÀ DI PISA

Corso di Laurea Magistrale in Data Science and Business Informatics

Optimization For Data Science

Camilla Andreazzoli
Davide Ricci

ANNO ACCADEMICO 2022-2023

Indice

Capitolo 1

Introduzione

Dato il training set $\{(x_i, y_i)\}_{i=1}^N$ dove $x_i \in \mathbb{R}^h$, $y_i \in \mathbb{R}$ per $i = 1..N$, considerato un problema di classificazione multiclasse non linearmente separabile tramite SVR soggetto ad iperparametri C ed ϵ, ci proponiamo di risolvere il seguente problema primale quadratico:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{N} \xi_i$$

con vincoli dati da

$$\begin{cases} \xi_i \geq 0 \\ wx_i - b - y_i - \epsilon \leq \xi_i \\ -wx_i + b + y_i - \epsilon \leq \xi_i \end{cases}$$

Equivalentemente possiamo considerare il relativo problema duale:

$$\min_{\lambda,\lambda^*} \psi(\lambda, \lambda^*) = \min_{\lambda,\lambda^*} \frac{1}{2}\sum_{i,j=1}^{N} (\lambda_i - \lambda_i^*)\langle x_i, x_j\rangle (\lambda_j - \lambda_j^*) + \epsilon\sum_{i=1}^{N}(\lambda_i + \lambda_i^*) - \sum_{i=1}^{N} y_i(\lambda_i - \lambda_i^*)$$

dipendente da 2N parametri e soggetto ai seguenti vincoli

$$\begin{cases} \lambda, \lambda^* \in \mathbb{R}^N \\ \sum_{i=1}^{N}(\lambda_i - \lambda_i^*) = 0 \\ 0 \leq \lambda_i \leq C \quad \forall i = 1..N \\ 0 \leq \lambda_i^* \leq C \quad \forall i = 1..N \end{cases} \tag{1.1}$$

Definiamo dunque la funzione

$$\phi\colon \mathbb{R}^h \to \mathbb{R}^{h_1}$$

$$x_i \mapsto \phi(x_i)$$

con $h_1 > h$, che mappi gli x_i in uno spazio di dimensione maggiore in modo non lineare, così da rendere i dati del training set linearmente separabili in tale spazio. Dunque, scelta una funzione kernel $K\colon \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}$ tale che $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, è possibile risolvere il seguente problema:

$$\min_{\lambda,\lambda^*} \frac{1}{2} \sum_{i,j=1}^{N} (\lambda_i - \lambda_i^*) K(x_i, x_j)(\lambda_j - \lambda_j^*) + \epsilon \sum_{i=1}^{N} (\lambda_i + \lambda_i^*) - \sum_{i=1}^{N} y_i(\lambda_i - \lambda_i^*) \qquad (1.2)$$

soggetto nuovamente ai vincoli di (1.1).
Il problema (1.2) può essere riscritto in forma matriciale come:

$$\psi(\lambda, \lambda^*) = \frac{1}{2}(\lambda, \lambda^*)^T Q(\lambda, \lambda^*) + q(\lambda, \lambda^*) \qquad (1.3)$$

dove $q = (\epsilon - y_1, .., \epsilon - y_n, \epsilon + y_1, .., \epsilon + y_n)$ e Q è la matrice seguente:

$$\begin{pmatrix} K & -K \\ -K & K \end{pmatrix}$$

dove K è la gram matrix relativa alla funzione kernel utilizzata, ovvero:

$$\begin{pmatrix} K(x_1, x_1) & \cdots & \cdots & K(x_1, x_n) \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ K(x_n, x_1) & \cdots & \cdots & K(x_n, x_n) \end{pmatrix}$$

Detto X il dominio della funzione ψ, preso un punto $\lambda_i = (\lambda, \lambda^*) \in X$, una direzione $d \in \mathbb{R}^{2N}$ è detta direzione ammissibile di X nel punto λ_i se $\exists \quad \overline{\epsilon} > 0 : \lambda_i + \overline{\epsilon} d \in X$. Definiamo $F_X(\lambda_i)$ l' insieme delle direzioni ammissibili di X nel punto λ_i. Definiamo, inoltre, il cono delle direzioni ammissibili nel punto $\lambda_i \in X$ del primo ordine l'insieme:
$D_X(\lambda_i) = \{d \in \mathbb{R}^{2N} : \langle \nabla g_i(\lambda_i), d \rangle \leq 0 \quad i \in \mathcal{A}(\lambda_i), \langle \nabla h_j(\lambda_i), d \rangle = 0 \quad j \in J\}$ dove $\mathcal{A}(\lambda_i)$ è l'active set del punto λ_i, J l'insieme degli indici dei vincoli descritti da uguaglianze, g_i, h_j le funzioni che descrivono rispettivamente vincoli di disuguaglianza e uguaglianza.
Osserviamo che ciascuna delle funzioni che descrivono i vincoli in (1.1) è affine, perciò vale l'uguaglianza tra gli insiemi $F_X(\lambda_i)$ e $D_X(\lambda_i)$, relazione che sfrutteremo nell'algoritmo per calcolare algebricamente l'insieme delle direzioni ammissibili del punto (λ, λ^*).
Ci proponiamo di risolvere il problema di ottimizzazione sopra descritto attraverso l'implementazione del metodo della proiezione del gradiente, in particolare sfruttando la definizione "standard" dell'algoritmo, immediatamente descritta qui sotto.

Per notazione indichiamo con $\lambda_i = (\lambda, \lambda^*)$ il punto corrente dell'iterata i-esima dell'algoritmo.
Considerata dunque l'iterata i-esima, si calcola $-\nabla\psi(\lambda_i)$:

- $-\nabla\psi(\lambda_i) \in F_X(\lambda_i)$
  In questo caso si implementa una line search lungo tale direzione per individuare lo step length α ottimale e muoversi nel nuovo punto $\lambda_{i+1} = \lambda_i - \alpha\nabla\psi(\lambda_i)$;

- $-\nabla\psi(\lambda_i) \notin F_X(\lambda_i)$
  In questo caso bisogna proiettare $-\nabla\psi(\lambda_i)$ in $F_X(\lambda_i)$, ovvero bisogna risolvere il seguente problema di ottimizzazione:

  $$d_* = \min_d \{\frac{1}{2}\|d + \nabla\psi(\lambda_i)\|_2^2 : d \in F_X(\lambda_i)\}$$

  per poi implementare una line search lungo la proiezione d^* trovata ed infine muoversi nel nuovo punto $\lambda_{i+1} = \lambda_i - \alpha d^*$.

A questo punto è possibile osservare che, in base alla posizione del punto λ_i all'interno del dominio della funzione $\psi(\lambda, \lambda^*)$, il cono $F_X(\lambda_i)$ varia:

1. il punto λ_i è interno al dominio della funzione, ovvero

   $$\forall j = 1, .., N \quad 0 < \lambda_j < C, \quad 0 < \lambda_j^* < C, \quad \sum_{j=1}^{N}(\lambda_j - \lambda_j^*) = 0$$

   In questo caso, sfruttando il fatto che $F_X(\lambda_i) = D_X(\lambda_i)$, che l'active set di λ_i è vuoto e che $\nabla\Big(\sum_{i=1}^{N}(\lambda_i - \lambda_i^*)\Big)(\lambda_i)$ è un vettore di lunghezza $2N$ con prime N entrate pari ad 1 e ultime N a -1, l'insieme delle feasible directions nel punto λ_i in cui ci troviamo risulta essere

   $$\begin{aligned} F_X(\lambda_i) &= \{d = (d, d^*) \in \mathbb{R}^{2N} : \langle \nabla h(\lambda_i), d\rangle = 0\} \\ &= \{d \in \mathbb{R}^{2N} : \sum_{i=1}^{N}(d_i - d_i^*) = 0\} \end{aligned}$$

   ovvero l'insieme delle feasible directions nel caso in cui un punto sia interno al dominio della funzione è dato dall'insieme di tutte le direzioni che giacciono sull' iperpiano descritto dall'equazione in (1.1). Dunque, in questo caso particolare, la proiezione d^* risulta:

   $$\begin{aligned} d_* &= \min_d \{\frac{1}{2}\|d + \nabla\psi(\lambda_i)\|_2^2 : \sum_{i=1}^{N}(d_i - d_i^*) = 0\} \\ &= \min_d \{\frac{1}{2}d^t d + \nabla\psi(\lambda_i)^t d : \sum_{i=1}^{N}(d_i - d_i^*) = 0\} \end{aligned}$$

   Questo è un problema di ottimizzazione con vincolo di uguaglianza del tipo $Ad = 0$, dove la matrice A è in realtà un vettore orizzontale unitario di lunghezza 2N con prime N componenti uguali ad 1, le restanti a -1, dunque con

rango massimo pari ad 1, e dal momento che la matrice che descrive la funzione quadratica relativa a questo sottoproblema è l'identità I, quindi strettamente definita positiva, possiamo risolverlo in modo diretto risolvendo il Poorman's KKT-system seguente:

$$\begin{pmatrix} I & A' \\ A & 0 \end{pmatrix} \begin{pmatrix} d^* \\ \mu \end{pmatrix} = \begin{pmatrix} -\nabla\psi(\lambda_i) \\ 0 \end{pmatrix} \tag{1.4}$$

Dunque, risolvendo il sistema 1.4 sopra, indicando con $\nabla_j\psi(\lambda_i)$ la componente j-esima di $\nabla\psi(\lambda_i)$, si ottiene:

$$\begin{cases} \mu = \frac{-\sum_{j=1}^{N}\nabla_j\psi(\lambda_i)+\sum_{j=N+1}^{2N}\nabla_j\psi(\lambda_i)}{2N} \\ d_* = -(A*\mu+\nabla\psi(\lambda_i)) \end{cases}$$

A questo punto se

$$\|d_*\|_2 < \epsilon$$

λ_i è il minimo cercato e l'algoritmo termina, altrimenti si implementa una line search lungo la direzione d_* e si passa al nuovo punto

$$\lambda_{i+1} = \lambda_i + \alpha d^*$$

2. il punto λ_i si trova nel bordo del dominio della funzione, ovvero esiste almeno un indice $i \in \{1, 2, ..., N\}$ per cui almeno una delle disuguaglianze in (1.1) è in realtà una uguaglianza. Dunque, per $j = 1..N$, possono essere soddisfatte le seguenti uguaglianze:

   - $\lambda_j = 0 \vee \lambda_j = C$
   - $\lambda_j^* = 0 \vee \lambda_j^* = C$

   Chiameremo con $g_{1,j}(\lambda, \lambda^*) = \lambda_j - C$ nel caso in cui si abbia $\lambda_j = C$, $g_{2,j}(\lambda, \lambda^*) = -\lambda_j$ nel caso in cui $\lambda_j = 0$, e con stessa notazione $g_{1,j}^* = \lambda_j^* - C$ se $\lambda_j^* = C$ ed infine $g_{2,j}^* = -\lambda_j^*$ se $\lambda_j^* = 0$.
   In questo caso, indicando con $\mathcal{A}(\lambda_i)$ l'active set di λ_i, si ha che $F_X(\lambda_i)$ è descritto da i vari prodotti scalari dei gradienti delle funzioni $g_{i,j}, g_{i,j}^*$ che eventualmente risultano attive con la direzione $d \in \mathbb{R}^{2N}$ posti minori o uguali a zero, ed inoltre anche dall'equazione $\sum_{i=1}^{N}(d_i - d_i^*) = 0$, già presente nella definizione del cono delle direzioni ammissibili del punto precedente. Indicando con $A_{\mathcal{A}(\lambda_i)}$ la matrice che descrive tutti i vincoli presenti nel cono $F_X(\lambda_i)$, possiamo ridurci a risolvere il seguente problema di ottimizzazione

$$d_* = \min_d\{\frac{1}{2}d'd + \nabla\psi(\lambda)'d : A_{\mathcal{A}(\lambda_i)}d \leq 0, \sum_{i=1}^{N}(d_i - d_i^*) = 0\} \tag{1.5}$$

   che risulta essere un problema di ottimizzazione con vincolo dato da una disuguaglianza. Questa tipologia di problema fa parte dei CQKnP, dunque possiamo pensare di risolvere questo sotto problema sfruttando un ulteriore metodo ad hoc, che descriveremo nella sezione 1.1 seguente. Se viene soddisfatta la condizione di stop da d_* l'algoritmo termina, altrimenti viene implementata una line search lungo la direzione trovata e si aggiorna il nuovo punto in $\lambda_{i+1} = \lambda_i + \alpha d^*$.

1.1 Analisi Sottoproblema (1.5)

Il problema che ci proponiamo di risolvere fa parte della famiglia dei "Convex Continous Quadratic Knapsack Problem", ovvero un problema di ottimizzazione in cui la funzione obiettivo è quadratica, continua, convessa e soggetta a determinati vincoli di uguaglianza e disuguaglianza. In particolare, il problema a cui siamo interessati si presenta nella seguente forma generale:

$$\min_x \{\sum_{i=1}^{N} d_i x_i^2 + c_i x_i : \sum_{i=1}^{N} a_i x_i = v, l_i \leq x_i \leq u_i\}$$

dove $d_i \geq 0$, $l_i \in \{\mathbb{R}, -\infty\}, u_i \in \{\mathbb{R}, +\infty\}, l_i \leq u_i$.
Nello specifico del problema (1.5) abbiamo che $d_i = \frac{1}{2}\ \forall i = 1, .., N$, $c_i = \nabla_i(\psi)$, $a_i \in \{-1, +1\}$, $v = 0$, $l_i \in \{-\infty, 0\}$, $u_i \in \{0, +\infty\}$ dipendentemente dalle equazioni di vincolo attivo che abbiamo nel punto corrente λ, in particolare nel caso in cui una delle componenti del punto λ corrente che si trova nel bordo del dominio della funzione ψ siano nulle, si ha $l_i = 0, u_i = +\infty$, il caso opposto $l_i = -\infty$, $u_i = 0$ si presenta invece nel caso in cui una delle componenti del punto corrente λ sia uguale a C, mentre $l_i = -\infty, u_i = +\infty$ nel caso in cui non avvenga nessuna delle possibilità precedenti, ovvero nel caso in cui $0 < \lambda_i, \lambda_i^* < C$. Infine,la variabile x da cui dipende la funzione è la direzione d. Perciò il problema si presenta sotto la seguente forma:

$$\min_d \{\sum_{i=1}^{2N} \frac{1}{2} d_i^2 + \nabla_i(\psi) d_i : \sum_{i=1}^{2N} a_i d_i = 0, l_i \leq d_i \leq u_i\}$$

L'approccio che ci proponiamo di seguire è quello lagrangiano in cui andremo a risolvere l'equivalente problema duale.

La lagrangiana del nostro problema , considerando solo il vincolo di uguaglianza, è data da:

$$L(d, \mu) = \sum_{i=1}^{2N} \frac{1}{2} d_i^2 + c_i d_i + \mu a_i d_i$$

dove $\forall i, d_i$ può essere soggetta a determinati vincoli, ovvero potrebbe essere $d_i \in [-\infty, 0]$ oppure $d_i \in [0, +\infty)$, $\mu \in \mathbb{R}$, inoltre per semplicità di notazione indichiamo con $c_i = \nabla_i(\psi)$. Calcolando le derivate parziali rispetto a ciascuna componente d_i e dovendo però tenere di conto anche dei vari vincoli possibili, è necessario distinguere 3 casi possibili (no vincolo, vincolo dato da $(-\infty, 0]$, vincolo dato da $[0, +\infty)$) e quindi è possibile osservare che l'espressione corrispondente al minimo varia nel seguente modo:

- no vincoli: $d_i = -(c_i + \mu a_i)$
- vincolo $(-\infty, 0]$: allora in questo caso si ha

$$d_i = \begin{cases} -(c_i + \mu a_i) & \text{se } -(c_i + \mu a_i) \leq 0 \\ 0 & \text{altrimenti} \end{cases}$$

  ed osserviamo che la condizione $-(c_i + \mu a_i) \leq 0$ è equivalente a $\mu \geq -c_i$ per $i = 1, ...N$, $\mu \leq c_i$ per $i = N + 1, .., 2N$

- vincolo $[0, +\infty)$: in questo caso si ha:

$$d_i = \begin{cases} -(c_i + \mu a_i) & \text{se } -(c_i + \mu a_i) \geq 0 \\ 0 & \text{altrimenti} \end{cases}$$

  ed in quest'altro caso la condizione $-(c_i + \mu a_i) \geq 0$ è equivalente a $\mu \leq -c_i$ per $i = 1, .., N$, $\mu \geq c_i$ per $i = N + 1, .., 2N$

Derivando invece la lagrangiana rispetto al moltiplicatore μ si ottiene nuovamente la condizione di vincolo $\sum_{i=1}^{2N} a_i d_i = 0$.
Si sostituisce dunque il relativo valore di $d_i(\mu)$ dipendentemente dal vincolo che deve rispettare. Dunque l'equazione che bisogna risolvere è la seguente:

$$\psi'(\mu) = \sum_{i=1}^{2N} a_i d_i(\mu) = 0 \tag{1.6}$$

con l'espressione di $d_i(\mu)$ definita a tratti come visto sopra, dipendentemente dal valore assunto da μ. A questo punto, si ordinano i vari breakpoints, ovvero i punti rispetto ai quali l' espressione varia. Questi risultano essere rispettivamente $-c_i$ per eventuali vincoli relativi a valori del punto corrente λ nella prima metà ($i = 1, .., N$) e c_i per eventuali vincoli relativi alla seconda metà. Una volta disposti in ordine crescente si provano uno ad uno nell'espressione (1.6) per verificarla. Se viene soddisfatta, abbiamo trovato il massimo, ci fermiamo ed otteniamo l'espressione di $d^*(\mu^*)$ ottimale, altrimenti cerchiamo i primi due breakpoints per i quali si verifica che $\psi' > 0$ e $\psi' < 0$ e si implementa un'interpolazione per cercare il punto che va ad annullare la derivata.
Nel caso in cui il primo breakpoint restituisce derivata negativa, visto che la funzione $\psi'(\mu)$ è decrescente, è stato deciso di decrementare il punto fino a quando non si raggiunge un punto che presenta derivata nulla o positiva. Se è nulla allora l'ottimo è stato raggiunto, quindi si può calcolare il valore d^*, altrimenti si implementa un'interpolazione per trovare il punto che va a soddisfare l'equazione (1.6). Allo stesso modo, nel caso in cui l'ultimo breakpoint restituisce derivata positiva, si incrementa il valore fino a quando non si trova il punto che annulla la derivata $\psi'(\mu)$, oppure chen presenta derivata negativa e quindi si procede nuovamente con un'interpolazione.

Capitolo 2

Convergenza

Le seguenti proposizioni che andremo ad enunciare descrivono la convergenza del metodo della proiezione del gradiente. Ciascuna di queste assume sempre l'ipotesi di convessità del dominio della funzione ψ da ottimizzare, e differenziabilità, dunque tali ipotesi verranno sottointese e non saranno ripetute all'interno degli enunciati.

Proposizione 1. *Sia $\{\lambda_k\}$ una sequenza generata dal metodo della proiezione del gradiente con α_k tramite line search esatta o la regola di Armijo lungo la direzione ammissibile d^*. Allora ogni punto limite di $\{\lambda_k\}$ è stazionario. [2]*

Proposizione 2. *Sia $\{\lambda_k\}$ una sequenza generata dal metodo della proiezione del gradiente con $\alpha_k = \alpha \quad \forall k$ costante. Si supponga che ψ sia L-smooth, ovvero che $\exists L > 0$ tale che*

$$\|\nabla\psi(\lambda_1) - \nabla\psi(\lambda_2)\| \leq L\|\lambda_1 - \lambda_2\| \quad \forall\lambda_1, \lambda_2 \in Dom(\psi)$$

Allora, se $0 < \alpha < \frac{2}{L}$, ogni punto limite di $\{\lambda_k\}$ è stazionario.[2]

Osserviamo che, in generale senza l'ipotesi di convessità della funzione da minimizzare, non è assicurato che il punto limite sia un punto di minimo locale o globale. Se invece aggiungiamo l'ipotesi di convessità della funzione, il punto limite risulta essere un minimo globale ed in particolare, se la funzione è strettamente convessa, esso risulta essere unico.
I teoremi seguenti analizzano la convergenza sublineare del metodo della proiezione del gradiente, quando la funzione da ottimizzare è convessa ed L-smooth.

Teorema 1. *Sia ψ convessa ed L-smooth, con $\alpha = \frac{1}{L}$. Allora il metodo della proiezione del gradiente soddisfa*

$$\psi(\lambda_k) - \psi(\lambda^*) \leq \frac{3L\|\lambda_0 - \lambda^*\|^2 + \psi(\lambda_0) - \psi(\lambda^*)}{k}$$

[3]

Teorema 2. *Sia ψ convessa e strettamente L-smooth, con $\alpha = \frac{1}{L}$ costante. Allora il metodo della proiezione del gradiente soddisfa*

$$\psi(\lambda_k) - \psi(\lambda^*) \leq \frac{L}{2k}\|\lambda_0 - \lambda^*\|^2 \tag{2.1}$$

[5]

E' possibile inoltre mostrare che, scegliendo uno stepsize α tramite line search, la convergenza risulta sublineare, come nel caso di uno stepsize $\alpha = \frac{1}{L}$.

Nel caso particolare in cui

- la funzione ψ da ottimizzare sia quadratica e strettamente convessa.
- lo stepsize α che viene fatto ad ogni iterazione è costante.
- indicando con $f(x) = [x]^+$ la proiezione di un punto $x \in \mathbb{R}^n$ nel dominio e con m,M l'autovalore minore e maggiore di Q

si ottiene la stima seguente:

$$\begin{aligned}\|\lambda_{k+1} - \lambda^*\| &= \|(\lambda_k - \alpha[\nabla\psi(\lambda_k)]^+) - (\lambda^* - \alpha[\nabla\psi(\lambda^*)]^+)\| \\ &\leq \|(\lambda_k - \alpha\nabla\psi(\lambda_k)) - (\lambda^* - \alpha\nabla\psi(\lambda^*))\| \\ &= \|(I - \alpha Q)(\lambda_k - \lambda^*)\| \\ &\leq \max\{|1 - \alpha m|, |1 - \alpha M|\}\|\lambda_k - \lambda^*\|\end{aligned}$$

e scegliendo $\alpha = \frac{2}{M+m}$ si ottiene il seguente tasso di convergenza:

$$\frac{\|\lambda_{k+1} - \lambda^*\|}{\|\lambda_k - \lambda^*\|} \leq \frac{M - m}{M + m}$$

Dunque, in questo caso particolare si raggiunge una convergenza lineare. Osserviamo che il numero di iterazioni k per raggiungere una precisione ϵ è pari a

$$k \geq \frac{\ln \frac{e_1}{\epsilon}}{\ln \frac{M+m}{M-m}}$$

dove indichiamo con $e_1 = \|\lambda_0 - \lambda^*\|$.

Da questi risultati è possibile osservare come la complessità dell'algoritmo dipenda dalla bontà della scelta del punto iniziale λ_0 che non può essere controllato a priori, dall'accuratezza che si vuole raggiungere (una precisione maggiore richiederà un maggiore numero di iterazioni) e dagli autovalori dell'hessiana (più m si avvicina allo zero, più il numero di iterazioni andrà ad aumentare).

Nel nostro caso di studio le ipotesi di convessità sia del dominio che della funzione e di differenziabilità sono verificate, dunque il punto a cui il metodo converge è un minimo globale. Inoltre, poichè la funzione obiettivo è quadratica, è possibile eseguire una line search esatta per la ricerca dello stepsize α ottimale, che soddisfa le condizioni di Armijo-Wolfe e quindi rientra nei casi precedentemente studiati. Infine, ci aspettiamo che la convergenza del metodo sia nel caso peggiore sublineare e nel caso ottimale lineare, poichè non conosciamo gli autovalori dell'hessiana che dipendono anche dalla funzione kernel scelta.

Capitolo 3

Implementazione Algoritmica

L'algoritmo principale descritto sotto prevede come dati di input la matrice Q ed il vettore q che descrivono in forma compatta il problema (1.2), gli iperparametri C ed ϵ relativi all'SVR, un punto di partenza λ_0 che soddisfa i vincoli descritti in (1.1), il numero N di dati nel training set, il numero massimo di iterazioni che può eseguire l'algoritmo ed un valore soglia utilizzato per le stime numeriche all'interno dell'algoritmo.

Algorithm 1 Projected Gradient Method Algorithm

procedure PGM(Q,q,ϵ,C,λ_0,N, max_iter,tol)
anti_grad $\leftarrow$ -(Q*$\lambda_0 + q$)
isin_border $\leftarrow$ any($\lambda_0 \leq$ tol | $\lambda_0 \geq$ C-tol)
curr_iter $\leftarrow$ 1
$[\lambda_n, f_{min}, min, d^*, norma, \alpha]$ $\leftarrow$ projection_routine(Q,q,λ_0,anti_grad,N,ϵ,C, isin_border)
while $\|d^*\| > \epsilon$ **and** $curr\_iter \leq max\_iter$ **do**
anti_grad $\leftarrow$ -(Q*$\lambda_n + q$)
isin_border $\leftarrow$ any($\lambda_n \leq$ tol | $\lambda_n \geq$ C-tol)
$[\lambda_n, f_{min}, min, d^*, norma, \alpha] \leftarrow$ projection_routine(Q,q,λ_n,anti_grad,N,ϵ,C, isin_border)
curr_iter $\leftarrow$ curr_iter+1
end while
end procedure

I valori restituiti in output sono rispettivamente il punto dell'iterata corrente, il valore assunto dalla funzione ψ che descrive il problema di ottimizzazione (1.2), il minimo, due array contenenti i valori assunti dalla funzione in ciascuna iterata ed il minimo relativo a ciascuna iterata ed infine il valore dell'iterata alla quale si è fermato l'algoritmo (utili per costruire grafici). All'interno dell'algoritmo viene richiamata la funzione projection_routine che permette di restituire il punto dell'iterata corrente, il valore assunto dalla funzione $\psi(\lambda, \lambda^*)$ in tale punto, la proiezione d^* dell'antigradiente del punto corrente nel cono delle direzioni ammissibili, la norma della proiezione ed infine lo step α che viene calcolato tramite un line search esatta per muoversi lungo la direzione d^*. La routine segue le logiche descritte precedentemente nell'Introduzione, ne riportiamo lo pseudocodice sotto. Questo algoritmo va

a richiamare la funzione AP per la proiezione dell'antigradiente nel caso di punto sul bordo. La descrizione di tale funzione con relativo pseudocodice viene fatta nelle pagine seguenti.

Algorithm 2 Direction Computation Algorithm

procedure PROJECTION_ROUTINE(Q,q,λ_n,anti_grad,N,ϵ,C,tol, isin_border)
  if isin_border **then**
    $A \leftarrow$ matrix describing active constraints computation
    isin_feasible $\leftarrow$ **not** $any(A * anti\_grad) > tol$
    if isin_feasible **and** $|\sum_{i=1}^{N} anti\_grad(i) - \sum_{i=N+1}^{2N} anti\_grad(i)| \leq \epsilon$ **then**
      $d^* \leftarrow anti\_grad$
    else
      $grad_\lambda \leftarrow -anti\_grad$
      $d^* \leftarrow$ Calling the function $AP(grad_\lambda, \lambda_n, N, C, tol)$ for projecting antigradient into the cone
    end if
  else
    if $|\sum_{i=1}^{N} anti\_grad(i) - \sum_{i=N+1}^{2N} anti\_grad(i)| \leq \epsilon$ **then**
      $d^* \leftarrow anti\_grad$
    else
      solve the KKT system

$$\begin{pmatrix} I & A' \\ A & 0 \end{pmatrix} \begin{pmatrix} d^* \\ \mu \end{pmatrix} = \begin{pmatrix} -\nabla\psi(\lambda_i) \\ 0 \end{pmatrix}$$

    end if
  end if
  norma $\leftarrow \|d^*\|$
  if norma $\leq \epsilon$ **then**
    $f_{min} \leftarrow \frac{1}{2}\lambda_n^T Q \lambda_n + q\lambda_n$
    min$\leftarrow \lambda_n$
    $\alpha \leftarrow 0$
  else
    $f_{min} \leftarrow \frac{1}{2}\lambda_n^T Q \lambda_n + q\lambda_n$
    min $\leftarrow \lambda_n$
    $\alpha \leftarrow$ Do an exact line search for finding the optimal step
    $\lambda_n \leftarrow \lambda_n + \alpha d^*$
  end if
end procedure

Dal momento che il cono delle direzioni ammissibili varia in base ai vincoli relativi al punto corrente, per prima cosa viene verificato se il punto è nel bordo o interno. In base a questo fatto, viene controllato se è necessario proiettare l'antigradiente nel cono. Nuovamente, la proiezione varia in base alla natura del punto: nel caso di punto sul bordo viene richiamata la funzione AP per la proiezione del gradiente, altrimenti, nel caso in cui il punto sia interno, la proiezione viene eseguita diretta-

mente risolvendo il KKT system (1.4). Dunque, dopo aver calcolato la proiezione d^* dell'antigradiente, si verifica se la norma della direzione ottenuta risulta minore di ϵ, in caso affermativo ci troviamo nel punto ottimale, altrimenti viene aggiornato il punto corrente spostandosi di uno step α ottenuto tramite una line search esatta.

Infine riportiamo lo pseudocodice dell'algoritmo che si propone di risolvere il problema (1.5) implementando le logiche descritte in sezione 1.1. I parametri input sono il gradiente del punto corrente, il punto corrente, N,C e tol. Restituisce in output la proiezione dell'antigradiente nel cono delle direzioni ammissibili del punto corrente.

Algorithm 3 Antigradient Projection Algorithm

procedure AP($grad_\lambda$,λ_n,N, C, tol)
brk_sort ← Breakpoints array creation without duplicates, sorted in ascending order
$last\_\psi' \leftarrow 0$
for $i = 1 : n$ **do** ▷ Iterate over the breakpoints
m← brk_sort(i)
d← Direction computation depending on the constraints valid at the current point and on μ
$\psi' \leftarrow$ Derivative computation defined in (1.6) using d just calculated
if $|\psi'| \leq tol$ **then**
$d^* = d$ ▷ d is the optimum
break
end if
if $i == 1$ **and** $\psi' < -tol$ **then**
Decreasing brk_sort(1) till we find a point with positive derivative
Do an interpolation for finding the point μ^* with null derivative
$d \leftarrow$ Direction computation depending on the constraints valid at the current point and on μ^*
$d^* \leftarrow d$
break
end if
if $i == n$ **and** $\psi' > tol$ **then**
Increasing brk_sort(n) till we find a point with negative derivative
Do an interpolation for finding the point μ^* with null derivative
$d \leftarrow$ Direction computation depending on the constraints valid at the current point and on μ^*
$d^* \leftarrow d$
break
end if
if $last\_\psi' > tol$ **and** $\psi' < -tol$ **then**
Do an interpolation for finding the point with null derivative
$d^* \leftarrow d$
break
end if
$last\_\psi' \leftarrow \psi'$
end for
end procedure

Capitolo 4

Esperimenti

Gli esperimenti sono stati eseguiti sui dataset Energy Community (288 record ed un'unica feature) ed Housing (506 record e 13 features). La matrice Q che va a descrivere il problema di ottimizzazione (1.3) risulta avere una dimensione pari a 576x576 nel primo caso e 1012x1012 nel secondo. Per ciascun dataset è stato testato l'algoritmo fissando gli iperparametri relativi all'SVR, la matrice kernel (definita con funzione radiale o polinomiale) ed i suoi parametri ed il punto iniziale da cui far partire le iterazioni. Ciascun risultato è stato confrontato con il risultato ottenuto dalla routine quadprog di MATLAB. Per evitare ambiguità indicheremo con ϵ_{SVR} l'iperparametro relativo all'SVR e con ϵ l'accuratezza che si vuole raggiungere relativa alla condizione di terminazione dell'algoritmo.
Per osservare gli effetti dei vari parametri sull'algoritmo, i parametri $\epsilon = 10^{-2}$, $\epsilon_{SVR} = 10^{-2}$, max_iter= 1000 sono stati mantenuti costanti, mentre sono stati fatti variare i valori di C nell'intervallo $logspace(-2, 3, 6)$, dei parametri delle funzioni kernel testate ed i punti iniziali da cui far partire l'algoritmo, che sono stati settati rispettivamente a $(0, .., 0), (\frac{C}{2}, .., \frac{C}{2}), (C, .., C)$. Per quanto riguarda la funzione kernel radiale il parametro σ è stato variato nell'intervallo $logspace(-3, 3, 7)$, mentre per la polinomiale è stato mantenuto il grado 2 costante.
È stato deciso poi di aumentare il numero di iterazioni per i casi in cui l'algoritmo non riuscisse a convergere entro le 1000 iterazioni, per osservare se fosse possibile ottenere risultati più soddisfacenti. Infine durante l'analisi dei vari risultati è stato sempre verificato che la curva relativa ai punti di minimo restituiti nelle varie iterate sia sempre decrescente, proprietà base del projected gradient method.

4.1 Dataset Energy Community

Ciascuna combinazione di parametri sopra citata va a definire un problema di ottimizzazione differente, in quanto la matrice Q che definisce il problema di ottimizzazione dipende dalla funzione kernel scelta (e quindi dai suoi parametri) ed il parametro C va a delimitare il range in cui ricercare la soluzione ottimale. Dunque l'algoritmo è stato testato su diversi problemi e di uno stesso problema sono stati provati vari punti di partenza.
È risultato in modo evidente come la scelta del punto iniziale vada ad influenzare la convergenza dell'algoritmo.

4.1.1 Kernel Radiale

Partendo dagli esperimenti conseguiti usando una funzione kernel radiale, sono stati comparati i risultati ottenuti con uno step esatto (exact line search) e step costante $\alpha = \frac{1}{L}$ (dove L risulta essere l'autovalore maggiore della matrice Q). Per tutti i casi testati è risultato evidente come una buona scelta del punto iniziale λ_0 vada ad influire sul numero di iterazioni necessarie per ottenere un'accuratezza pari ad ϵ. Inoltre, la scelta di un passo adattivo è risultata sempre vincente rispetto ad una con passo costante. I grafici sottostanti (4.1) riportano la convergenza ottenuta tramite line search esatta insieme a quella relativa ad uno step costante. In particolare viene riportato l'andamento dell'errore relativo definito come $\frac{f_k - f^*}{max(abs(f^*),1)}$ in funzione del numero di iterazioni k: è chiaro che, quando il punto iniziale risulta vicino all'ottimo (caso $\lambda_0 = (0, .., 0), (\frac{C}{2}, .., \frac{C}{2})$), si ottiene la convergenza entro il numero massimo di iterazioni. Inoltre, è possibile osservare come la scelta di un passo adattivo sia vincente rispetto ad una con passo costante.

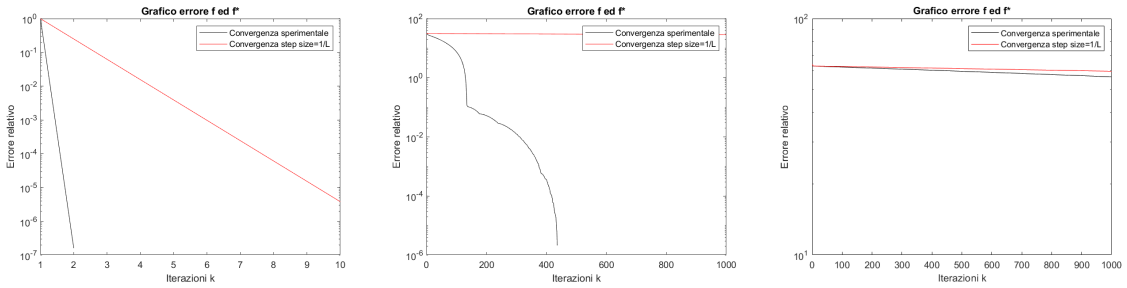

Figura 4.1: I grafici mostrano l'errore in funzione del numero di iterazioni eseguite dall'algoritmo, con funzione kernel radiale con parametro $\sigma = 10^{-3}$, $C = 100$ e punti iniziali rispettivamente $\lambda_0 = (0, .., 0), (\frac{C}{2}, .., \frac{C}{2}), (C, .., C)$

Riportiamo per ciascuno di questi tre casi la distanza del punto iniziale dall'ottimo ed il tempo impiegato dall'algoritmo prima di stopparsi, rapportato al tempo relativo alla funzione Quadprog (Q time) nella tabella seguente.

λ_0	$\|\lambda_0 - \lambda^*\|$	Elapsed time	Q time
$(0, .., 0)$	4.26	0.043 s	0.101 s
$(C, .., C)$	$2.39 * 10^3$	0.323 s	0.066 s
$(\frac{C}{2}, .., \frac{C}{2})$	$1.19 * 10^3$	0.792 s	0.074 s

A livello sperimentale è stata raggiunta una convergenza di tipo sublineare ($O(\frac{1}{k})$), in accordo con quanto discusso nel Capitolo 2, in quanto la matrice Q presentava sempre il minimo degli autovalori pressochè nullo. A tal proposito i grafici in figura (4.2) riportano le due quantità che definiscono la disequazione (2.1) del Teorema 2:

$$\psi(\lambda_k) - \psi(\lambda^*) \leq \frac{L}{2k} \|\lambda_0 - \lambda^*\|^2$$

in cui si vede come, a livello asintotico, la disuguaglianza venga rispettata. Lo step utilizzato, secondo le ipotesi del teorema, è quello costante.

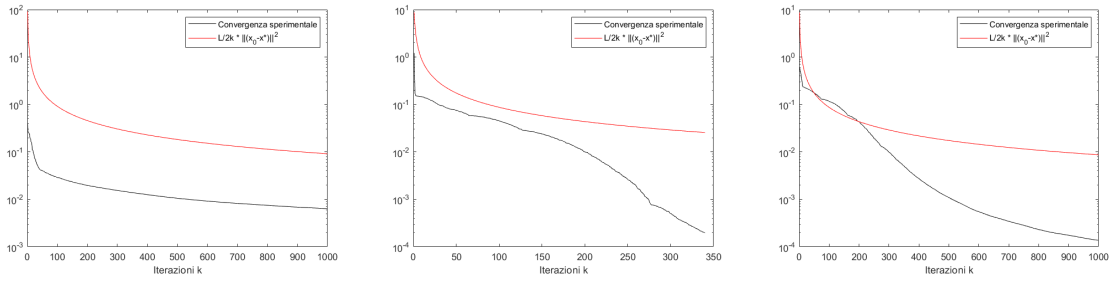

Figura 4.2: I grafici mostrano l'errore in funzione del numero di iterazioni eseguite dall'algoritmo, con funzione kernel radiale con parametri $\sigma = 10^{-1}, 10^{-2}, 100$ e $C = 1, 10^{-1}, 10^{-2}$ e punti iniziali $\lambda_0 = (0, .., 0), (\frac{C}{2}, .., \frac{C}{2}), (C, .., C)$ rispettivamente.

Infine sono stati testati nuovamente tutti i casi che non hanno raggiunto la convergenza entro le 1'000 iterazioni, aumentando il limite a 10'000 iterazioni. Riportiamo nella tabella sottostante i relativi risultati.

λ_0	Iterations	σ	C	Elapsed time	Q time
$(0, .., 0)$	1592	10	10^3	7.093 s	0.725 s
$(\frac{C}{2}, .., \frac{C}{2})$	1203	10	10^3	5.027 s	0.642 s
$(C, .., C)$	1575	10	10^3	7.674 s	0.657 s
$(0, .., 0)$	3962	1	10	18.454 s	0.405 s
$(\frac{C}{2}, .., \frac{C}{2})$	3377	1	10	15.1 s	0.371 s
$(C, .., C)$	1248	10^{-3}	10	0.973 s	0.052 s

In conclusione si può affermare che, con un passo adattivo, il punto iniziale $\lambda_0 = (0, .., 0)$ permette di raggiungere la convergenza entro il numero massimo di iterazioni nel 90% circa dei casi testati, seguito da $\lambda_0 = (\frac{C}{2}, .., \frac{C}{2})$ con il 70% dei casi ed infine $(C, .., C)$ con il 66%.

4.1.2 Kernel Polinomiale

Per quanto riguarda gli esperimenti sul kernel polinomiale, la corrispondente funzione è stata testata mantenendo costante il grado 2 e variando il valore di C dell' SVR. Anche in questo caso sono stati confrontati i risultati ottenuti sia con step esatto che con step costante, che è risultato sempre meno performante del primo. Per quanto riguarda la convergenza è emerso come, per piccoli valori del parametro C, l'algoritmo riesca a migliorare convergendo entro circa le 1000 iterazioni a partire da ciascun punto iniziale testato; invece, per i valori di $C = 10, 100, 1000$ non si riesce ad ottenere la convergenza da nessun punto iniziale testato, anche aumentando il numero di iterazioni massime a 10^4. Nella tabella seguente sono riportate le performance dei casi convergenti:

λ_0	Iterations	C	Elapsed time	Q time
$(0, .., 0)$	353	10^{-2}	1.711 s	0.051 s
$(\frac{C}{2}, .., \frac{C}{2})$	562	10^{-2}	2.075 s	0.043 s
$(C, .., C)$	372	10^{-2}	1.811 s	0.052 s
$(0, .., 0)$	468	10^{-1}	1.929 s	0.049 s
$(\frac{C}{2}, .., \frac{C}{2})$	562	10^{-1}	2.069 s	0.061 s
$(C, .., C)$	508	10^{-1}	2.068 s	0.062 s
$(0, .., 0)$	1013	1	4.132 s	0.357 s
$(\frac{C}{2}, .., \frac{C}{2})$	735	1	2.642 s	0.336 s
$(C, .., C)$	913	1	4.097 s	0.331 s

Inoltre si riportano i grafici relativi alla convergenza per ciascun punto iniziale testato nella figura (4.3) seguente, con passo adattivo.

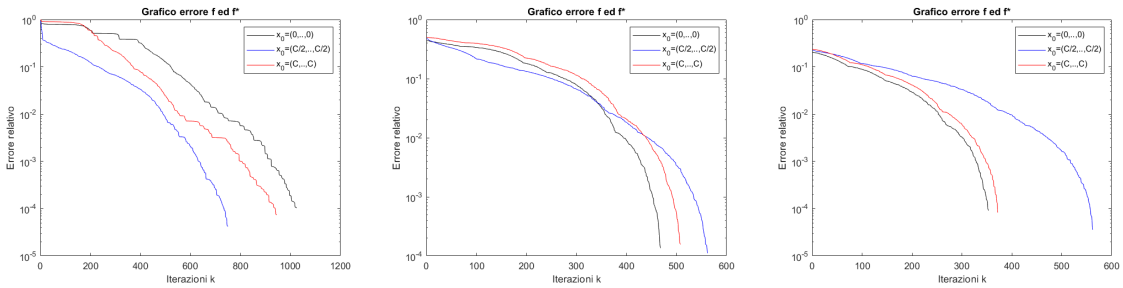

Figura 4.3: Convergenza con $C = 1, 10^{-1}, 10^{-2}$ kernel polinomiale

La convergenza ottenuta risulta ancora una volta sublineare, se ne riporta una dimostrazione nei grafici (4.4) sottostanti, eseguiti con passo adattivo.

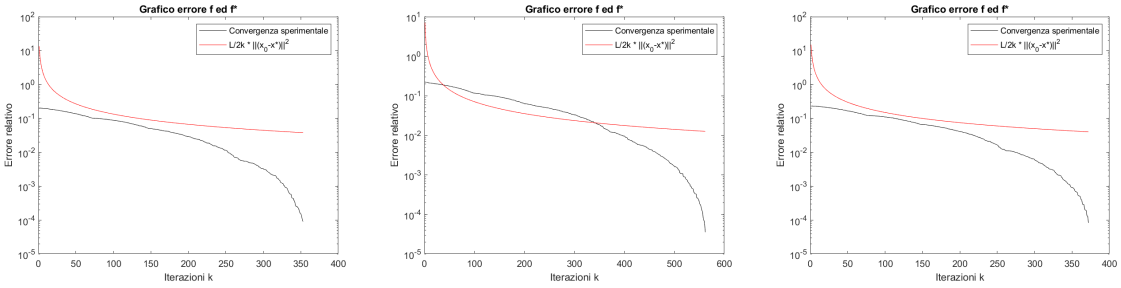

Figura 4.4: Convergenza con $C = 10^{-2}$ costante e $\lambda_0 = (0, .., 0), (\frac{C}{2}, .., \frac{C}{2}), (C, .., C)$

Ancora, i casi che non hanno ottenuto la convergenza entro le 1'000 iterazioni sono stati nuovamente testati settando il numero di iterazioni massime a 10'000, ma diversamente dal caso precedente con kernel radiale non sono stati ottenuti dei miglioramenti significativi, se non per $C = 10, \lambda_0 = (\frac{C}{2}, .., \frac{C}{2})$ con un tempo di esecuzione pari a 48.024 s (figura (4.5)).

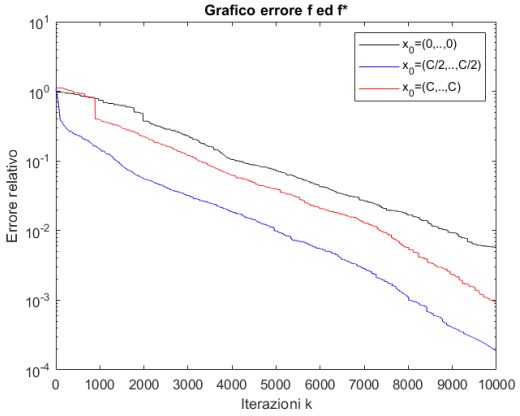

Figura 4.5: Convergenza con $C = 10$ kernel polinomiale

4.2 Dataset Housing

Gli stessi test che sono stati eseguiti con Energy Community sono stati eseguiti anche con questo dataset, ed i risultati sono simili a quelli ottenuti precedentemente: la scelta di uno step esatto domina su quello costante, come mostrato nei grafici in figura (4.6); la convergenza ottenuta è risultata nuovamente sublineare, in quanto l'hessiana Q presenta autovalore minore nullo; infine, con il kernel radiale, nella maggior parte dei casi si ottiene nuovamente la convergenza entro il numero massimo di iterazioni. Differentemente al caso di studio esaminato in sezione 4.1, nel caso polinomiale non si riescono mai a raggiungere dei buoni risultati, anche aumentando il numero massimo di iterazioni e provando diversi punti iniziali oltre a quelli tipicamente testati fino ad ora. È possibile pensare che la non convergenza sia fortemente influenzata dagli autovalori della matrice hessiana, infatti nei casi testati con il kernel polinomiale risulta che l'autovalore L maggiore è dell'ordine di 10^8, e sappiamo dunque che il numero di iterazioni k da eseguire per ottenere un'accuratezza dell'ordine di ϵ è proporzionale ad L, dunque impossibile da raggiungere in pratica. Diverso il caso polinomiale relativo al dataset Energy Community (sezione 4.1), in cui l'hessiana presenta L dell'ordine 10^2, dunque in tal caso la convergenza può essere raggiunta molto più facilmente.

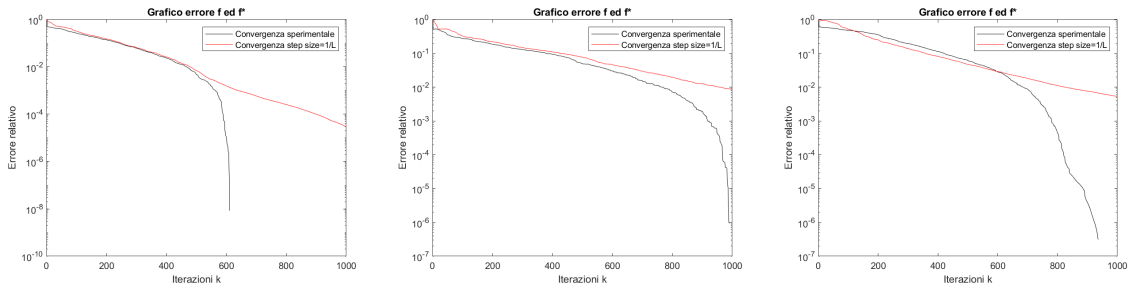

Figura 4.6: I grafici mostrano l'errore in funzione del numero di iterazioni eseguite dall'algoritmo, con funzione kernel radiale con parametri $\sigma = 10^{-2}, 10^3, 1$ rispettivamente, $C = 1$ costante e punti iniziali $\lambda_0 = (0, .., 0), (\frac{C}{2}, .., \frac{C}{2}), (C, .., C)$

Per il caso radiale è stata ottenuta ancora convergenza sublineare. I grafici in figura (4.7) mostrano quanto ottenuto con passo adattivo.

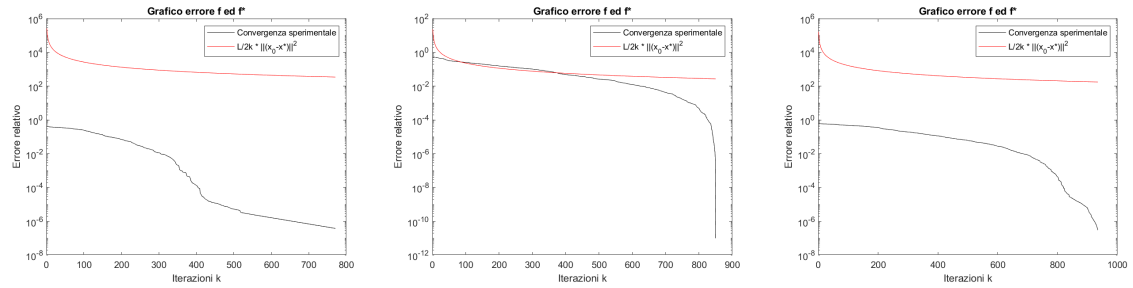

Figura 4.7: I grafici mostrano l'errore in funzione del numero di iterazioni eseguite dall'algoritmo, con funzione kernel radiale con parametri $\sigma = 10^{-3}$ per le prime 2 immagini e $\sigma = 1$ per la restante; invece $C = 10, 10^{-1}, 1$ e punti iniziali $\lambda_0 = (0, .., 0), (\frac{C}{2}, .., \frac{C}{2}), (C, .., C)$ rispettivamente.

Come per il dataset precedente, è stato aumentato il parametro MaxIter per gli esperimenti in cui non è stata ottenuta la convergenza entro le 1000 iterazioni. A tal proposito è stato ottenuto un risultato interessante per $\sigma = 10^{-3}, C = 100$ e step esatto, riportato in figura (4.8), in cui è possibile osservare nuovamente come la scelta del punto iniziale vada ad influire sulla velocità di convergenza. In particolare si ottiene la convergenza da $\lambda_0 = (\frac{C}{2}, .., \frac{C}{2})$ (grafico blu) e $\lambda_0 = (0, .., 0)$ (grafico nero). Inoltre, è possibile osservare come la velocità di convergenza rallenti dopo circa 2500 iterazioni a partire da $(0, .., 0)$, mentre a partire dagli altri due punti iniziali si ha un netto miglioramento fino circa alle prime 1000 iterazioni, dopo di che si assiste ad una fase di stallo che si prolunga fino a MaxIter per il punto $(C, .., C)$, mentre per $(\frac{C}{2}, .., \frac{C}{2})$ si assiste ad un improvviso miglioramento che permette di raggiungere il risultato più preciso.

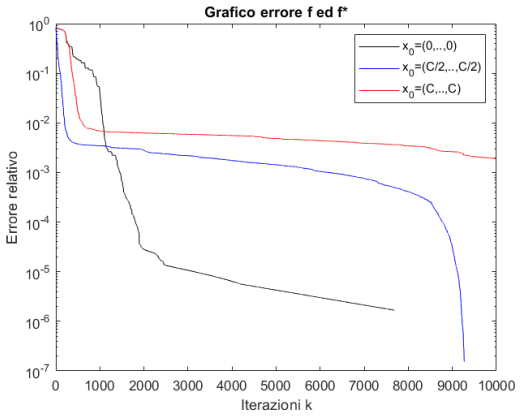

Figura 4.8: Convergenza con $C = 100$, kernel radiale con $\sigma = 10^{-3}$

In linea generale per questo dataset è emerso che, differentemente dal dataset Energy Community, non c'è un punto di partenza che permetta di ottenere un numero più elevato di casi convergenti, in quanto per tutti i λ_0 testati si ottiene la convergenza dell'algoritmo nel 70% dei casi circa. In conclusione è emerso che, a prescindere dal punto iniziale testato, le uniche combinazioni in cui non si ottiene la convergenza sono quelle relative a $C \geq 100$ e $\sigma \leq 10$. Si riportano dunque nella tabella seguente i risultati degli esperimenti relativi ai casi $\sigma = 10^2, 10^3$ e $C = 10^2, 10^3$ con exact line search.

λ_0	Iterations	σ	C	Elapsed time	Q time
$(0, .., 0)$	1388	10^2	10^2	16.628 s	2.299 s
$(\frac{C}{2}, .., \frac{C}{2})$	1352	10^2	10^2	13.682 s	2.372 s
$(C, .., C)$	1888	10^2	10^2	26.014 s	2.244 s
$(0, .., 0)$	610	10^3	10^2	6.467 s	0.194 s
$(\frac{C}{2}, .., \frac{C}{2})$	1010	10^3	10^2	8.654 s	0.177 s
$(C, .., C)$	616	10^3	10^2	7.049 s	0.168 s
$(0, .., 0)$	613	10^3	10^3	6.264 s	2.711 s
$(\frac{C}{2}, .., \frac{C}{2})$	1008	10^3	10^3	8.306 s	2.652 s
$(C, .., C)$	657	10^3	10^3	7.171 s	2.516 s

Bibliografia

[1] Enrico Gorgone Antonio Frangioni. A library for continuous convex separable quadratic knapsack problems. *European Journ al of Operational Research*, 2013.

[2] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, Massachusetts, 1999.

[3] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *In Foundations and Trends in Machine Learning*, 2015.

[4] Michael Patriksson. A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research*, 2006.

[5] Y.Sun. Notes on first-order methods for minimizing smooth functions, 2015.