UNIVERSITY OF PISA

Risoluzione di un problema di Constrained Convex Quadratic Programming con Primal-Dual Interior Point Method

Computational Mathematics for Learning and Data Analysis project report

Cornacchia Giuliano, Salinas Mario Leonardo
Gruppo 21

INDICE

1 Introduzione

Nel seguente report viene descritto ed analizzato il problema numero 3 *noML* assegnato per il progetto finale di *Computational Mathematics for Learning and Data Analysis* e descritta la soluzione da noi proposta, insieme alle varie scelte implementative.

2 Descrizione del problema

Date le matrici $Q \in \mathbb{R}^{n\times n}$ ed $A \in \{0,1\}^{k\times n}$, con $Q \succeq 0$, e i vettori $q \in \mathbb{R}^n$ e $b = [1]^k$, il problema di ottimizzazione quadratica convessa primale P è definito come:

$$P := \begin{cases} min\ x^\top Qx + q^\top x \\ Ax = b \\ x \geq 0 \end{cases} \tag{2.1}$$

dove i vincoli nella matrice A formano k simplessi disgiunti della forma:

$$\sum_{i\in I^h} x_i = 1, h = 1,\dots,k \tag{2.2}$$

con $I^h, h = 1,\dots,k$ insiemi di indici che formano una partizione di $\{1,\dots,n\}$. Quindi il generico elemento della matrice A $a_{vi} = 1 \iff i \in I^v$.

Data la funzione da minimizzare in 2.1 $f(x) = x^\top Qx + q^\top x$ e la sua *funzione Lagrangiana* $L(x,\lambda_{eq},\lambda_s)$, a P si può associare il seguente *Problema Duale di Wolfe D*:

$$D := \begin{cases} \max\ L(x,\lambda_{eq},\lambda_s) \\ \nabla_x L(x,\lambda_{eq},\lambda_s) = 0 \\ \lambda_s \geq 0 \end{cases} = \begin{cases} \max\ x^\top Qx + q^\top x + \lambda_{eq}^\top (Ax-b) + \lambda_s^\top (-x) \\ 2Qx + q + A^\top \lambda_{eq} - \lambda_s = 0 \\ \lambda_s \geq 0 \end{cases} \tag{2.3}$$

Possiamo a questo punto scrivere il sistema KKT associato a P.

$$\begin{cases} \nabla_x L(x,\lambda_{eq},\lambda_s) = 0 \\ Ax - b = 0 \\ x_i \lambda_{s_i} = 0 \quad i = 1,\dots,n \\ (x,\lambda_s) \geq 0 \end{cases} \tag{2.4}$$

Scegliamo dunque di risolvere il sistema 2.4 applicando il metodo Primal-Dual Interior Point (PDIP): riformuliamo le condizioni di ottimalità 2.4 definendo una funzione $F : \mathbb{R}^{2n+k} \to \mathbb{R}^{2n+k}$ [1]:

$$F(x,\lambda_{eq},\lambda_s) = \begin{bmatrix} \nabla_x L(x,\lambda_{eq},\lambda_s) \\ Ax - b \\ XSe \end{bmatrix} = 0 \tag{2.5a}$$

$$(x,\lambda_s) \geq 0 \tag{2.5b}$$

dove

$$X = diag(x_1,\dots,x_n) \qquad S = diag(\lambda_{s_1},\dots,\lambda_{s_n}) \qquad e^\top = [1,\dots,1] \in \mathbb{R}^n \tag{2.6}$$

Il metodo PDIP, ad ogni iterazione k, genera triple $(x^k, \lambda_{eq}^k, \lambda_s^k)$ che soddisfano *strettamente* la 2.5b. La procedura con la quale si ricercano le direzioni $(\Delta x, \Delta\lambda_{eq}, \Delta\lambda_s)$ prende origine dal metodo di Newton per equazioni non lineari [1]: alla k-esima iterazione il metodo di Newton forma un modello lineare di F attorno al punto corrente $(x^k, \lambda_{eq}^k, \lambda_s^k)$ e ottiene le direzioni di ricerca risolvendo il seguente sistema lineare:

$$J(x, \lambda_{eq}, \lambda_s) \begin{bmatrix} \Delta x \\ \Delta\lambda_{eq} \\ \Delta\lambda_s \end{bmatrix} = -F(x, \lambda_{eq}, \lambda_s) \tag{2.7}$$

dove J è la Jacobiana di F. Nel nostro caso il sistema da risolvere diventa:

$$\begin{bmatrix} 2Q & A^\top & -I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda_{eq} \\ \Delta\lambda_s \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \\ XSe \end{bmatrix} \tag{2.8}$$

dove

$$r_d = 2Qx + q + A^\top\lambda_{eq} - \lambda_s \qquad r_p = Ax - b \tag{2.9}$$

Percorrere un passo intero lungo le direzioni trovate risolvendo 2.8 potrebbe violare il vincolo $(x, \lambda_s) \geq 0$, quindi aggiungiamo una parametro $\alpha \in (0, 1]$, detto *step-size*, che servirà a ridurre l'ampiezza del passo $(\Delta x, \Delta\lambda_{eq}, \Delta\lambda_s)$ per garantire il soddisfacimento del vincolo 2.5b.

$$(x^{k+1}, \lambda_{eq}^{k+1}, \lambda_s^{k+1}) = (x^k, \lambda_{eq}^k, \lambda_s^k) + \alpha(\Delta x, \Delta\lambda_{eq}, \Delta\lambda_s) \tag{2.10}$$

Data la corrente iterazione $(x^k, \lambda_{eq}^k, \lambda_s^k)$, che soddisfa 2.5b, introduciamo il *centering parameter* $\sigma \in [0, 1]$ e la *duality measure* $\mu = \frac{x^\top\lambda_s}{n}$; questi due parametri vengono utilizzati per direzionare il Newton step verso un punto per il quale valga $x_i\lambda_{s_i} = \sigma\mu$, piuttosto che a una soluzione diretta di 2.4. A seguito di questa considerazione, il nuovo step verrà calcolato risolvendo il *KKT perturbato* definito come:

$$\begin{bmatrix} 2Q & A^\top & -I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda_{eq} \\ \Delta\lambda_s \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \\ XSe - \sigma\mu e \end{bmatrix} \tag{2.11}$$

Il sistema 2.11 è non-simmetrico, lo trasformiamo in un sistema lineare equivalente simmetrico, eliminando la terza riga ed esprimendo $\Delta\lambda_s$ in funzione di Δx.

La terza riga di 2.11 può essere eliminata poichè durante le iterazioni x_i ed s_i rimangono strettamente positivi; per ricavare $\Delta\lambda_s$ in funzione di Δx dobbiamo prima isolare $\Delta\lambda_s$ sempre dalla terza riga del sistema:

$$\begin{aligned} S\Delta x + X\Delta\lambda_s &= -XSe + \sigma\mu e \\ X\Delta\lambda_s &= \sigma\mu e - XSe - S\Delta x \\ \Delta\lambda_s &= X^{-1}(\sigma\mu e - S\Delta x) - \lambda_s \end{aligned} \tag{2.12}$$

possiamo dunque riscrivere 2.11 sostituendo $\Delta\lambda_s$ come in 2.12:

$$\begin{aligned} 2Q\Delta x + A^\top\Delta\lambda_{eq} - \Delta\lambda_s &= -2Qx - q - A^\top\lambda_{eq} + \lambda_s \\ 2Q\Delta x + A^\top\Delta\lambda_{eq} - X^{-1}(\sigma\mu e - S\Delta x) + \cancel{\lambda_s} &= -2Qx - q - A^\top\lambda_{eq} + \cancel{\lambda_s} \\ (2Q + X^{-1}S)\Delta x + A^\top\Delta\lambda_{eq} &= -2Qx - q - A^\top\lambda_{eq} + X^{-1}\sigma\mu e \end{aligned} \tag{2.13}$$

ponendo $M = 2Q + X^{-1}S$ otteniamo il seguente sistema simmetrico detto anche *augmented KKT*:

$$\begin{bmatrix} 2Q + X^{-1}S & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda_{eq} \end{bmatrix} = - \begin{bmatrix} r_d - X^{-1}\sigma\mu e + \lambda_s \\ r_p \end{bmatrix} \tag{2.14a}$$

$$\Delta\lambda_s = X^{-1}(\sigma\mu e - S\Delta x) - \lambda_s \tag{2.14b}$$

la matrice a sinistra in 2.14a è simmetrica poichè:

- M è simmetrica perchè somma di una matrice simmetrica e una matrice diagonale
- il blocco inferiore sinistro e superiore destro sono l'uno il trasposto dell'altro

inoltre se A ha rango massimo essa è non-singolare, e 2.14a ammette soluzione. I vincoli in A, nel nostro caso di studio, formano k simplessi disgiunti, quindi $rank(A) = k$.

Il sistema 2.14a è dunque simmetrico, sparso e la matrice potrebbe essere mal condizionata a causa del prodotto $X^{-1}S$. Il metodo solitamente utilizzato per risolvere sistemi simmetrici sparsi è MINRES, ma non essendo stato affrontato durante il corso, utilizzeremo una sua generalizzazione, GMRES.

Data la simmetria del sistema lineare da risolvere, avremmo potuto anche optare per LDL-factorization, ma non è da escludere che nel nostro problema l'algoritmo possa incontrare degli zero-pivot. Sebbene esistano tecniche che consistono nel trattare un blocco 2x2 come pivot, ovviando cosi al problema, risolveremo il sistema 2.14a con il metodo iterativo GMRES.

3 Metodo Risolutivo

3.1 Primal-Dual Interior Point method

L'intuizione principale dei metodi Primal-Dual è quella di considerare sia il problema di minimizzazione P che il suo duale D per ottenere un limite superiore $v(P)$ ed inferiore $v(D)$ della soluzione. Da P e D si ricava quindi il sistema KKT 2.14a associato e una misura della distanza della soluzione attuale dall'ottimo, detta *complementary gap*; definita come la differenza fra il valore della funzione obiettivo in P e in D, eventualmente normalizzata.

Ad ogni iterazione, una volta trovata una soluzione per 2.14a, calcoliamo una nuova coppia di soluzioni primali/duali e riduciamo il complementary gap.

Con gli elementi presentati finora possiamo delineare la struttura dello pseudocodice del nostro metodo come segue:

Nell'algoritmo sopra descritto sono cruciali la scelta della tripla iniziale (riga 2) e la scelta di α (riga 10); nelle sezioni seguenti descriveremo le nostre scelte progettuali a riguardo. Le condizioni di stop scelte (riga 6) sono descritte in 3.4.

Mentre per la risoluzione della riga 8 utilizzeremo sia un approccio iterativo (GMRES) che diretto (versione modificata di LDL). Entrambi i metodi verranno presentati nella sezione 3.5.

Algorithm 1 pseudocodice Interior-Point Primal-Dual method

function PDIP(Q, q, A, b, eps, maxit)
  inizializzare $(x^0, \lambda_{eq}^0, \lambda_s^0) > 0$
  $\mu^0 \leftarrow (x^{0\top}\lambda_s^0)/n$
  $\sigma \in [0,1]$
  $k \leftarrow 0$
while *k<maxit & complementary_gap < eps*
  $\mu^{k+1} \leftarrow \sigma\mu^k$
  risolvere $\begin{bmatrix} 2Q + X^{-1}S & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{k+1} \\ \Delta\lambda_{eq}^{k+1} \end{bmatrix} = -\begin{bmatrix} r_d - X^{-1}\mu^{k+1}e + \lambda_s \\ r_p \end{bmatrix}$
  $\Delta\lambda_s^{k+1} \leftarrow X^{-1}(\mu^{k+1}e - S\Delta x^{k+1}) - \lambda_s^k$
  calcolare $(\alpha_x^{k+1}, \alpha_{\lambda_{eq}}^{k+1}, \alpha_{\lambda_s}^{k+1})$ *tale che* $(x^{k+1}, \lambda_s^{k+1}) > 0$
  $(x^{k+1}, \lambda_{eq}^{k+1}, \lambda_s^{k+1}) \leftarrow (x^k, \lambda_{eq}^k, \lambda_s^k) + (\alpha_x^{k+1}\Delta_x^{k+1}, \alpha_{\lambda_{eq}}^{k+1}\Delta_{\lambda_{eq}}^{k+1}, \alpha_{\lambda_s}^{k+1}\Delta_{\lambda_s}^{k+1})$
  $k \leftarrow k+1$

3.2 Punto inziale

Il nostro obiettivo è trovare una tripla iniziale $(x^0, \lambda_{eq}^0, \lambda_s^0)$ che sia ammissibile per P e D

Scelta x^0

Per la scelta di x^0 sfruttiamo la struttura della matrice dei vincoli A; ogni x_i appare in un solo vincolo ed il vincolo è definito come in 2.2, quindi:

$$x_i = \frac{1}{|I^h|}$$

dove I^h è l'insieme di indici, ovvero il simplesso, che contiene l'indice i.

Scelta λ_{eq}^0 e λ_s^0

Inizializziamo λ_{eq}^0 con numeri positivi $\in (0,1)$; mentre per garantire l'ammissibilità duale troviamo λ_s^0 risolvendo $\nabla_x L(x^0, \lambda_{eq}^0, \lambda_s^0) = 0$.

$$\overline{\lambda_s} = 2Qx^0 + q + A^\top\lambda_{eq}^0$$

Nel vettore λ_s^0 potrebbero essere presenti componenti negative, quindi definiamo Δ_s come:

$$\Delta_s = -(3/2)\ max(0, \min_i \overline{\lambda_{si}})$$

a questo punto scriviamo $\lambda_s^0 = \overline{\lambda_s} + e\Delta_s$ con

3.3 Scelta dello step-size α

Scegliamo di utilizzare uno step-size distinto per ogni vettore della tripla $(x, \lambda_{eq}, \lambda_s)$. Poichè vogliamo assicurare che, muovendoci lungo le direzioni trovate risolvendo il sistema 2.14a, i soli vincoli di non negatività in 2.5b siano soddisfatti, scegliamo il massimo α tale che i vincoli non vengano violati. Per il generico vettore $d \in \{x, \lambda_{eq}, \lambda_s\}$

$$\alpha_{max}^d = \min(1, \min_{i:\Delta d_i<0} -\frac{d_i}{\Delta d_i}) \tag{3.1}$$

E quindi scegliamo lo step α come segue:

$$\alpha^d = \min(1, \eta\alpha^d_{max}) \quad dove \;\; \eta \in [0.9, 1) \tag{3.2}$$

3.4 Criteri di stop

L'algoritmo può terminare sia perchè è stato raggiunto il numero massimo di iterazioni deciso dall'utente, oppure perchè il *complementary gap* è minore di ϵ.

3.5 Risoluzione del sistema lineare

Per la risoluzione del sistema in 2.14a abbiamo deciso di utilizzare e confrontare due diversi approcci; uno iterativo (GMRES) e l'altro diretto (LDL).

GMRES è un metodo iterativo per la soluzioni di sistemi lineari $Ax = b$ non simmetrici di grandi dimensioni. La soluzione esatta del sistema è $x_* = A^{-1}b$. L'idea del metodo GMRES è di approssimare la soluzione x_* al passo n con un vettore $x_n \in K_n$, dove K_n è il Krylov subspace $\langle b, Ab, \ldots, A^{n-1}b\rangle$, che minimizza la norma del residuo $r_n = b - Ax_n$. Questo si traduce in determinare x_n risolvendo un least squares problem.

Mentre per la risoluzione diretta del sistema, usiamo una versione modificata di LDL per matrici simmetriche, implementata da `MATLAB` attraverso l'operatore *mldivide*.

3.6 Convergenza del metodo

I risultati teorici sulla convergenza dei metodi DPIP [1] dimostrano che effettuando scelte appropriate sulla tripla iniziale $(x^0, \lambda^0_{eq}, \lambda^0_s)$, per $\epsilon > 0$ fissato, il limite superiore al numero delle iterazioni per garantire la convergenza dell'algoritmo è $log(n\, log(1/\epsilon))$ dove n è l'input size del problema.

Il nostro metodo è di tipo *infeasible*, ovvero ammette un punto iniziale che non soddisfi i vincoli in P e D, ma data la scelta di uno step size diverso per il problema primale e duale, e l'aggiunta del centering parameter - scelte che in pratica aumentano la velocità di convergenza - ci aspettiamo che il nostro metodo converga comunque in un numero di iterazioni di ≈ 100, per $n \to \infty$, come empiricamente osservato sui metodi di questa classe.

4 Codice prodotto e validazione

In questa sezione descriviamo il codice che implementa il metodo PDIP, la generazione delle istanze dei problemi e gli esperimenti.

4.1 Codice `MATLAB`

La nostra soluzione è composta da cinque files:

- `genProblem.m` script che genera un'instanza del problema (Q, q, A, b) usando funzioni ausiliare definite negli altri script.
- `genQF.m` script che dati dimensione n, $\delta \in (0, 1]$ e un vettore $v \geq 0$ genera un vettore $q \in \mathbb{R}^n$ ed una matrice $Q \in \mathbb{R}^{n\times n}$ con densità δ e autovalori v. Per generare Q con queste proprietà è stata utilizzata la funzione `MATLAB` `sprandsym`.

- `generateAdisjointed.m` script che genera la matrice dei vincoli A dato il numero di simplessi m e il numero di variabili n.
- `feasible_sp.m` script che dati Q, q ed A calcola il punto iniziale $(x^0, \lambda_{eq}^0, \lambda_s^0)$ come mostrato in 3.2.
- `PDIP.m` script che implementa il metodo primale duale del punto interno come in alg. 1.

4.2 Esperimenti

Allo scopo di testare l'implementazione proposta sono stati effettuati esperimenti considerando PDIP sia con risoluzione iterativa che diretta del sistema lineare. Inoltre tempi ed accuratezza del metodo da noi implementato sono stati confrontati con la funzione *built-in* di `MATLAB quadprog`. Gli esperimenti possono essere divisi in tre gruppi:

- Nel primo insieme di esperimenti abbiamo voluto testare la scalabilità del nostro metodo fissando m e densità al variare di n.
- Nel secondo gruppo di esperimenti abbiamo voluto investigare l'effetto del numero dei vincoli sulla convergenza. Abbiamo fatto variare m lasciando fisse n e densità.
- infine, nel terzo gruppo di esperimenti abbiamo investigato gli effetti della densità della matrice Q fissati m ed n.

Per ogni esperimento sono state effettuate 20 ripetizioni per poi calcolare media e deviazione standard delle metriche. Tali esperimenti sono stati eseguiti su un Laptop con processore *Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz (4 CPUs)* e 6 GB di RAM.

5 Risultati

Questa sezione è deidcata ai risultati sperimentali

SUBEXP 1

111

input size	**50**	**75**	**100**	**200**	**300**	**400**	**500**	**600**
PDIP-LDL	0.0064	0.0177	0.0211	0.0575	0.1418	0.3251	0.5271	0.8170
QUADPROG	0.0069	0.0115	0.0137	0.0334	0.0737	0.1377	0.2590	0.4541
$\frac{t_{PDIP-LDL}}{t_{QUADPROG}}$	0.9259	1.5432	1.5422	1.7229	1.9247	2.3617	2.0351	1.7989
input size	**700**	**800**	**900**	**1000**	**1250**	**1500**	**1750**	**2000**
PDIP-LDL	1.1532	1.4939	2.0655	2.5441	4.2911	6.8573	8.8513	11.7768
QUADPROG	0.6157	0.7584	1.0631	1.3172	2.2816	3.5041	5.3935	7.9170
$\frac{t_{PDIP-LDL}}{t_{QUADPROG}}$	1.8728	1.9697	1.9428	1.9315	1.8808	1.9569	1.6411	1.4875

Tabella 5.1: tabella

SUBEXP 2

222

333

6 CONCLUSIONI

sadasds

RIFERIMENTI BIBLIOGRAFICI

[1] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2 edition, 2006.