Comparazione tra diverse implementazioni del metodo del gradiente

Tommaso Botarelli
tommaso.botarelli@edu.unifi.it
7136210

Abstract

Le performances del metodo del gradiente possono variare in base al tipo di line search che si applica. In questo report si riassume la sperimentazione effettuata sulle performances ottenute dall'applicazione di 3 diverse tecniche di line search e 2 metodi per la scelta del passo iniziale. In particolare sono state considerate tecniche di line search che permettono piccolo incrementi temporanei della funzione da ottimizzare che dovrebbe garantire una migliore efficienza nel raggiungimento di un punto di minimo globale ed una migliore flessibilità nell'affrontare punti di minimo locale. La sperimentazione ha evidenziato i punti di forza e di debolezza di ciascun metodo, sia in termini di velocità di convergenza che di qualità della soluzione, mettendo in luce il ruolo cruciale della scelta del metodo relativo al passo nella performance complessiva del metodo del gradiente.

1. Introduzione

Si considera un problema di ottimizzazione non vincolata:

$$\min_{x \in \mathbb{R}^n} f(x)$$

dove $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ è continuamente differenziabile. I metodi iterativi producono una sequenza $x_0, x_1, ...$ dove x_{k+1} è generato a partire da x_k, la direzione d_k e uno step size α_k usando:

$$x_{k+1} = x_k + \alpha_k d_k$$

In particolare in questo lavoro è stato usato il *metodo del gradiente* che pone come direzione quella dell'antigradiente $d_k = \nabla f(x_k)$. Il metodo del gradiente quindi lascia aperto il problema della scelta del passo α_k. I metodi che permettono la scelta del passo si chiamano *metodi di line search*. Fra questi metodi ci sono metodi monotoni e non monotoni.

I metodi di line search monotoni (come Armijo) scelgono α_k in modo tale da avere $f(x_{k+1}) < f(x_k)$.

I metodi di line search non monotoni invece permettono anche la scelta di α_k che producono piccoli incrementi della funzione obiettivo, cioé anche quando la condizione $f(x_{k+1}) < f(x_k)$ non è soddisfatta.

Questi metodi possono incrementare la probabilità di trovare una soluzione ottima ed incrementare allo stesso tempo il tempo di convergenza nei casi in cui uno schema monotono è costretto a "navigare" lungo un punto di minimo locale identificato come un avvallamento stretto della funzione obiettivo.

2. Metodi di Line Search

In questa sezione vengono descritti brevemente i metodi di line search implementati ed utilizzati nella sperimentazione.

2.1. Armijo

La line search di Armijo è una strategia classica per determinare un passo che garantisca una sufficiente diminuzione della funzione obiettivo lungo una direzione di discesa. Dato un punto x_k ed una direzione di discesa d_k il metodo cerca il più piccolo α tale che sia verificata la condizione di Armijo:

$$f(x_k + \alpha d_k) \leq f(x_k) + \gamma \alpha \nabla f(x_k)^T d_k$$

Algorithm 1 Metodo di Armijo

1: α = INITIALSTEP
2: **while** $f(x_k + \alpha d_k) > f(x_k) + \gamma \alpha \nabla f(x_k)^T d_k$ **do**
3: $\quad \alpha = \alpha \delta$
4: **end while**
5: **return** α

2.2. Grippo-Lampariello-Lucidi

Il metodo GLL [1] è una variante *non monotona* della classica line search di Armijo, progettata per migliorare l'efficienza del metodo del gradiente evitando le limitazioni imposte dalla richiesta di riduzione a ogni iterazione.

A differenza della condizione di Armijo, che confronta il valore della funzione $f(x_k + \alpha_k d_k)$ con $f(x_k)$, il metodo

GLL permette temporanei incrementi della funzione confrontando con un valore massimo recente:

$$f(x_k + \alpha d_k) \leq \max_{0 \leq j \leq M_k} f(x_{k-j}) + \gamma \alpha \nabla f(x_k)^T d_k,$$

dove: M_k è un parametro che controlla la *finestra di memoria*, cioè il numero di iterazioni passate da considerare.

Questa condizione consente di avere una funzione che *non debba necessariamente diminuire a ogni passo*, purché sia inferiore al massimo recente penalizzato. In questo modo, il metodo può *accettare passi più grandi*, favorendo un miglioramento della velocità di convergenza in situazioni in cui la discesa monotona sarebbe troppo restrittiva.

Algorithm 2 Metodo GLL

1: α = INITIALSTEP
2: **while** $f(x_k + \alpha d_k) > \max_{0 \leq j \leq M_k} f(x_{k-j}) + \gamma \alpha \nabla f(x_k)^T d_k$ **do**
3: $\quad \alpha = \alpha \delta$
4: **end while**
5: **return** α

2.3. Non Monotone Line Search Algorithm

Questo algoritmo di line search non monotono ha la forma generale di GLL ma la massimizzazione è rimpiazzata da una media del valore della funzione obiettivo nei passi precedenti [3].

Algorithm 3 Metodo NLSA

1: α = INITIALSTEP
2: **while** $f(x_k + \alpha d_k) > C_k + \gamma \alpha \nabla f(x_k)^T d_k$ **do**
3: $\quad \alpha = \alpha \delta$
4: **end while**
5: **return** α

Infatti questo algoritmo definisce due termini, C_k e Q_k che vengono inizializzati rispettivamente a $f(x_0)$ e 1. Dopodichè ad ogni iterazione del metodo del gradiente si effettua l'aggiornamento:

- $Q_{k+1} = \eta_k Q_k + 1$
- $C_{k+1} = \frac{\eta_k Q_k C_k + f(x_{k+1})}{Q_{k+1}}$

Si può osservare che C_{k+1} è una combinazione convessa di C_k e $f(x_{k+1})$. Questo fa si che η_k controlli il grado di non monotonicità. Se $\eta_k = 0$ per ogni k allora il metodo collassa nel classico metodo di Armijo. Se invece $\eta_k = 1$ per ogni k allora $C_k = \sum_{i=0}^{k} f(x_i)$ cioè la media dei valori della funzione obiettivo nei precedenti passi.

Nome	Grandezza soluzione
LOGHAIRY	2
MARATOSB	2
TOINTGOR	50
SENSORS	100
FLETCHCR	1000
DIXMAANL	3000
DIXMAANB	3000
DIXMAANI1	3000
ARWHEAD	5000
POWELLSG	5000
SCHMVETT	5000
INDEF	5000
SCURLY20	10000
BOX	10000
SPARSQUR	10000

Table 1: Problemi utilizzati nella sperimentazione e grandezza corrispondente

3. Scelta del passo iniziale

Anche la scelta del passo iniziale può garantire migliorie in termini di convergenza. Una scelta iniziale troppo piccola rallenta la convergenza, mentre una scelta troppo grande può causare instabilità o il fallimento della line search. Per semplicità, si assume spesso un passo iniziale costante, come per esempio $\alpha_0 = 1$ ma questa scelta non si adatta bene a tutti i problemi.

Metodo Barzilai-Borwein. Per migliorare la qualità della discesa, sono state proposte tecniche più sofisticate per stimare un buon valore iniziale del passo. Tra queste, il metodo Barzilai-Borwein (BB) [2] si è dimostrato particolarmente efficace. Esso propone un passo iniziale basato sull'approssimazione a due punti dell'equazione secante alla base dei metodi quasi-Newton.

In questo metodo il passo iniziale può essere scelto da due stime:

$$\alpha_1 = \frac{\Delta x \Delta g}{\Delta g \Delta g}$$

$$\alpha_2 = \frac{\Delta x \Delta x}{\Delta x \Delta g}$$

Dove $\Delta x = x_k - x_{k-1}$ e $\Delta g = \nabla f(x_k) - \nabla f(x_{k-1})$

4. Sperimentazione

La sperimentazione [1] è stata effettuata implementando in Python i metodi appena descritti: Armijo (*A*), Grippo con $M = 5$ (*G(5)*), Grippo con $M = 10$ (*G(10)*), Non

[1] Repository GitHub: Link

Monotone Line Search Algorithm (*NLSA*). Ogni tecnica è stata sperimentata con 3 diverse scelte del passo iniziale: costante e pari ad 1 (*CS*), Barzilai-Borwein α_1 (*BB1*) e Barzilai-Borwein α_2 (*BB2*). È stato poi aggiunto alle metodologie di confronto il solver *L-BFGS* della libreria *Scipy* [5].

I problemi sono stati scelti dalla lista di problemi della libreria *PyCUTEst* [4]. In particolare sono stati scelti 15 problemi di varia grandezza (Tabella 1).

4.1. Risultati

Nella sperimentazione sono stati collezionati i tempi di esecuzione di ogni metodo ottenuti come media da 10 iterazioni (Tabella 2). Per quei metodi che impiegano visibilmente più tempo sono state diminuite le iterazioni a 2 per garantire tempi di esecuzione più rapidi.

Oltre al tempo di esecuzione è stato collezionato anche il valore raggiunto dal metodo in questione per osservare quale metodo conduce al miglior risultato (Tabella 3).

4.2. Discussione

Dai valori dei tempi di esecuzione in tabella 2 si può vedere come in generale i tempi di esecuzione siano più alti nel caso di metodo di Armijo classico. Soprattutto al crescere della grandezza del problema il metodo di Armijo risulta essere di gran lunga più lento. Questo comportamento è particolarmente evidente quando si considera una scelta del passo inziale costante. Questo era prevedibile dal momento che questa combinazione risulta essere quella meno "intelligente" fra quelle proposte.

Il trend generale dei risultati mostra come l'inizializzazione costante (CS) porta a tempi di esecuzione più grandi e spesso anche di ordini di grandezza (vedi caso SCURLY20, BOX, SPARSQUR, problemi a grandezza maggiore).

Questo dimostra come un inizializzazione del passo più intelligente come quella adottata dai metodi BB permetta effettivamente di convergere ad una soluzione in modo più veloce.

Fra i diversi metodi implementati si nota come G(10) converga spesso in modo più veloce rispetto agli altri. Anche questo è in linea con la teoria di questo metodo. Il metodo permette di superare minimi locali ed il raggiungimento della soluzione è più rapido.

È interessante confrontare questo risultato con quello ottenuto da G(5). Infatti si vede come quasi sempre la soluzione più rapida è ottenuta da G(10). Permettere al metodo di guardare ad un maggior numero di valori della funzione passati (consentendo quindi un incremento temporaneo potenzialmente maggiore) permette al tempo stesso di superare velocemente minimi locali ed una velocità di convergenza maggiore.

Nonostante il metodo GLL permetta spesso di battere LBFGS nel caso del problema 'INDEF' si ha che LBFGS raggiunge la soluzione con un ordine di grandezza minore rispetto a tutti gli altri metodi. Questo consente di concludere che la scelta del metodo utilizzato deve essere in ogni caso valutata caso per caso.

Dalla tabella 3 si può osservare invece come i risultati siano più dispersi e meno concentrati. Avendo i vari metodi una condizione di uscita analoga ($||\nabla f(x)||^2 < 10^{-3}$), i valori ottenuti sono simili. Tuttavia si può notare come una scelta del passo iniziale con BB2 permetta non solo di ottenere un tempo di esecuzione minore ma spesso anche un risultato finale migliore.

5. Conclusione

La sperimentazione effettuata conferma l'importanza della scelta del metodo di line search per il metodo del gradiente. In particolare conferma come metodi più intelligenti come GLL permettano di convergere più velocemente. La sperimentazione ha evidenziato come anche una implementazione *from scratch* di questi metodi permetta di raggiungere soluzioni in valore comparabile ma con velocità spesso maggiori di metodi ampiamente utilizzati come LBFGS della libreria *Scipy*.

In particolare è emerso quanto anche la scelta del passo iniziale abbia un impatto importante sulle prestazioni del metodo del gradiente. Anche in questo caso una scelta più accurata del passo iniziale permette di migliorare le prestazioni del metodo del gradiente con line search basata su passo costante.

References

[1] S. Lucidi L. Grippo F. Lampariello. "A Nonmonotone Line Search Technique for Newton's Method". In: *SIAM Journal on Optimization* (1986).

[2] J. Barzilai and J. M. Borwein. "Two-point step size gradient methods". In: *MA Journal of Numerical Analysis* (1988).

[3] H. Zhang W. Hager. "A Nonmonotone Line Search Technique and Its Application to Unconstrained Optimization". In: *SIAM Journal on Optimization* (2004).

[4] *PyCUTEst*. URL: `https://jfowkes.github.io/pycutest/_`

[5] *Scipy*. URL: `https://docs.scipy.org/doc/scipy/tutori`

Nome	LBFGS	SA			G(10)			G(5)			NLSA		
		CS	BB1	BB2	CS	BB1	BB2	CS	BB1	BB2	CS	BB1	BB2
LOGHAIRY	1.5e-03	7.7e-02	8.9e-04	1.7e-03	8.8e-02	9.4e-04	1.8e-03	8.2e-02	8.9e-04	1.6e-03	9.0e-02	1.0e-03	1.9e-03
MARATOSB	6.5e-02	4.7e-01	1.8e-01	1.0e-01	8.4e-01	8.3e-02	9.2e-02	7.4e-01	1.6e-01	1.0e-01	6.4e-01	1.1e-01	9.4e-02
TOINTGOR	2.2e-02	5.7e-02	1.6e-02	1.6e-02	1.1e-01	1.7e-02	1.6e-02	7.6e-02	1.7e-02	1.6e-02	6.9e-02	1.6e-02	1.5e-02
SENSORS	5.6e-02	4.3e-01	1.5e-01	1.4e-01	1.7e1	1.3e-01	9.0e-02	3.6e0	1.5e-01	8.6e-02	2.9e0	1.4e-01	8.7e-02
FLETCHCR	1.2e0	2.5e0	2.4e-01	1.3e-01	2.4e0	1.4e-01	8.8e-02	2.3e0	2.0e-01	9.2e-02	2.3e0	1.9e-01	9.4e-02
DIXMAANL	5.1e-02	2.9e-01	1.2e-01	2.2e-01	1.3e1	7.2e-02	1.4e-01	8.2e0	7.9e-02	1.4e-01	1.3e1	1.0e-01	1.4e-01
DIXMAANB	1.8e-02	1.1e-02	1.4e-02	1.4e-02	2.6e-02	1.1e-02	8.3e-03	1.7e-02	1.1e-02	1.0e-02	6.0e-02	9.6e-03	1.1e-02
DIXMAANI1	7.7e-02	8.4e-01	1.5e-01	1.2e-01	5.2e-01	1.0e-01	6.0e-02	5.2e-01	9.4e-02	6.0e-02	5.3e-01	8.4e-02	6.5e-02
ARWHEAD	2.8e-02	8.3e0	2.3e-02	2.4e-02	1.1e1	1.8e-02	2.0e-02	1.0e1	1.8e-02	1.9e-02	8.5e0	2.1e-02	2.0e-02
POWELLSG	4.5e-02	4.7e0	5.5e-01	8.9e-02	4.0e0	7.1e-02	6.2e-02	4.1e0	1.3e-01	6.1e-02	3.9e0	6.4e-02	6.6e-02
SCHMVETT	5.0e-02	3.2e-01	1.2e-01	1.0e-01	1.1e0	7.0e-02	7.3e-02	4.4e-01	7.2e-02	7.3e-02	5.9e-01	6.1e-02	7.5e-02
INDEF	1.1e-01	6.3e0	1.7e1	6.3e0	3.8e0	9.9e0	4.1e0	3.8e0	9.9e0	3.9e0	3.8e0	1.0e1	4.0e0
SCURLY20	8.7e0	1.3e2	1.5e1	1.2e1	1.4e2	8.9e0	7.6e0	1.3e2	9.3e0	7.6e0	1.4e2	9.1e0	7.7e0
BOX	3.5e-02	4.2e1	1.1e2	8.1e-02	3.7e1	8.5e-02	5.3e-02	3.7e1	8.7e-02	5.3e-02	3.6e1	8.1e-02	5.6e-02
SPARSQUR	9.3e-02	2.0e-01	7.6e-02	8.0e-02	1.0e0	5.2e-02	5.1e-02	4.5e-01	4.9e-02	5.0e-02	5.5e-01	4.9e-02	5.0e-02

Table 2: Tempi di esecuzione in secondi. Evidenziati i tempi migliori.

Nome	LBFGS	SA			G(10)			G(5)			NLSA		
		CS	BB1	BB2	CS	BB1	BB2	CS	BB1	BB2	CS	BB1	BB2
LOGHAIRY	6.5e0	6.5e0	6.6e0	6.4e0	6.5e0	6.6e0	6.4e0	6.5e0	6.6e0	6.4e0	6.5e0	6.6e0	6.4e0
MARATOSB	-1.0e0	1.0e0	-8.9e-01	-1.0e0	1.0e0	-1.0e0	-1.0e0	1.0e0	-1.0e0	-1.0e0	1.0e0	-1.0e0	-1.0e0
TOINTGOR	1.4e3	1.4e3	1.4e3	1.4e3	1.4e3	1.4e3	1.4e3	1.4e3	1.4e3	1.4e3	1.4e3	1.4e3	1.4e3
SENSORS	-2.1e3	-2.1e3	-2.1e3	-2.1e3	-2.1e3	-2.1e3	-2.1e3	-2.1e3	-2.1e3	-2.1e3	-2.1e3	-2.1e3	-2.1e3
FLETCHCR	8.5e-09	2.5e-04	8.5e-07	6.7e-07	2.7e-04	8.7e-07	5.8e-08	2.7e-04	9.5e-07	5.8e-08	2.6e-04	9.8e-07	5.8e-08
DIXMAANL	1.0e0	2.6e0	2.6e0	2.5e0	2.3e3	2.6e0	2.6e0	1.7e1	2.6e0	2.6e0	2.3e3	2.5e0	2.6e0
DIXMAANB	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0
DIXMAANI1	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0	1.0e0
ARWHEAD	6.5e-09	3.4e-08	1.6e-11	4.4e-12	4.9e-09	1.6e-11	4.4e-12	5.2e-09	1.6e-11	4.4e-12	1.5e-08	1.6e-11	4.4e-12
POWELLSG	5.7e-04	7.4e-03	1.7e-04	1.1e-04	7.4e-03	1.3e-04	7.9e-05	7.4e-03	1.4e-04	6.6e-05	7.4e-03	1.3e-04	7.0e-05
SCHMVETT	-1.5e4	-1.5e4	-1.5e4	-1.5e4	-1.5e4	-1.5e4	-1.5e4	-1.5e4	-1.5e4	-1.5e4	-1.5e4	-1.5e4	-1.5e4
INDEF	-2.8e40	-2.5e7	-1.5e156	-2.8e7	-2.5e7	-1.5e156	-1.6e6	-2.5e7	-1.5e156	-4.2e10	-2.5e7	-1.5e156	-3.2e10
SCURLY20	-6.8e5	1.7e11	2.7e9	5.9e6	3.3e12	5.5e10	2.2e7	1.3e12	3.2e10	4.0e7	5.8e12	2.4e10	6.6e6
BOX	-1.9e3	-1.7e3	-1.9e3	-1.9e3	-1.6e3	-1.9e3	-1.9e3	-1.6e3	-1.9e3	-1.9e3	-1.6e3	-1.9e3	-1.9e3
SPARSQUR	1.7e-06	1.4e-05	8.8e-06	1.1e-05	1.5e-05	8.8e-06	1.1e-05	1.4e-05	8.8e-06	1.1e-05	1.5e-05	8.8e-06	1.1e-05

Table 3: Valori della funzione a convergenza. Evidenziati i valori minori.