Trace estimation by random sampling

Alessandra G. C. Cattafi

Sommario

Questa relazione tratta il problema della stima della traccia di una matrice, risolvendolo usando gli strumenti della probabilità e statistica, giungendo allo sviluppo di algoritmi di tipo Montecarlo.

I metodi probabilistici in seguito descritti nascono da un'idea naturale: costruire uno stimatore non distorto per la traccia, e produrne k copie indipendenti per ridurre la varianza della stima.

In seguito, vengono usati i metodi standard di probabilità e statistica per sviluppare garanzie a priori per i nostri stimatori di traccia. Particolarmente, vengono trattati gli stimatori probabilistici di Girard.

1 Notazioni e definizioni preliminari

1.1 Algebra lineare

Lavoreremo nel campo reale ($\mathbb{R}$) o nel campo complesso ($\mathbb{C}$). Il simbolo $\mathbb{F}$ si riferisce equivalentemente al campo reale o complesso, nei casi in cui la scelta tra i due sia irrilevante.

Il simbolo * è riservato alla trasposta (coniugata) di una matrice di vettori. Una matrice che soddisfa $\mathbf{A} = \mathbf{A}^*$ viene detta *auto-aggiunta*. È utile distinguere lo spazio $\mathbb{H}_n$ delle matrici auto-aggiunte n x n sul campo scalare.

La semplice norma $\|\cdot\|$ si riferisce alla *norma spettrale* di una matrice. Restituisce il massimo valore singolare del suo argomento.

1.2 Probabilità

Lavoriamo in uno spazio di probabilità ambiente abbastanza ricco da supportare tutte le variabili random che definiamo. Il termine *variabili random* racchiude scalari, vettori e matrici.

La mappa $\mathbb{P}(E)$ restituisce la probabilità di un evento E. L'operatore $\mathbb{E}$ restituisce il *valore atteso* di una variabile aleatoria. Per vettori e matrici può essere calcolato componente per componente. Il valore atteso è lineare. In riferimento a variabili aleatorie indipendenti, spesso includiamo la specifica 'statisticamente indipendenti', per fare una distinzione con 'linearmente indipendenti'. Abbreviamo il termine (statisticamente) 'indipendenti e identicamente distribuite' con i.i.d.

Diciamo che U è uno *stimatore non distorto* di V se $\mathbb{E}[U] = V$.

1.2.1 Distribuzioni

In questo contesto $\sim$ significa 'ha la stessa distribuzione di'.

Scriviamo $UNIF$ per indicare la *distribuzione uniforme* su un insieme finito (di misura). In particolare, una variabile aleatoria scalare *Rademacher* ha la distribuzione $UNIF\{\pm 1\}$. Un vettore randomico di Rademacher ha coordinate $i.i.d$, ognuna distribuita come una variabile scalare di Rademacher.

Scriviamo $NORMAL(\mu, C)$ per indicare la *distribuzione normale* su $\mathbb{F}^n$, con valore atteso $\mu \in \mathbb{F}^n$ e matrice di covarianza semidefinita positiva $C \in \mathbb{H}_n$. Un vettore (o scalare) appartenente alla distribuzione normale standard ha valore atteso 0 e matrice di covarianza l'identità. Spesso usiamo il termine *Gaussiana* per riferirci alla distribuzione normale.

2 Introduzione

Il nostro obiettivo è stimare la traccia di una matrice generica $\mathbf{M}$,

$$\operatorname{trace}(\mathbf{M}) = \sum_{i=1}^{n} \mathbf{M}_{ii}.$$

Questo è semplice se si ha accesso diretto alle entrate della matrice $\mathbf{M}$, perché basta leggerne gli n elementi diagonali; non lo è altrettanto se l'operazione primitiva a nostra disposizione è, ad esempio, il prodotto matrice-vettore $u \mapsto \mathbf{M}u$: in questo caso vorremmo evitare di applicare n volte la primitiva. Cioè, siano δi i vettori della base canonica:

$$\delta_i \mapsto \mathbf{M}\delta_i$$

Inizialmente, ci ridurremo al problema di stimare la traccia di una matrice non nulla, semi-definita positiva $\mathbf{A} \in \mathbb{H}_n$. Il nostro scopo è di produrre un'approssimazione della traccia di $\mathbf{A}$, e una misura della sua qualità.

Tutti i metodi esposti in seguito usano informazioni lineari sulla matrice in input.

Ovvero, vengono estratti i dati rilevanti dalla matrice $\mathbf{A}$ calcolando il prodotto $\mathbf{Y} = \mathbf{AP}$, dove $\mathbf{P} \in \mathbb{F}^{n \times k}$ è una matrice casuale di prova.

Tutte le operazioni implementate coinvolgono solo la matrice campione $\mathbf{Y}$ e la matrice di prova $\mathbf{P}$.

Storicamente, la prima applicazione della stima probabilistica della traccia era stimare a priori e a posteriori l'errore per problemi ai minimi quadrati di grosse dimensioni. Nello specifico, è usato per accellerare la procedura di cross-validation per la stima dei parametri ottimali di regolarizzazione nelle smoothing spline [3, 4].

Sono presenti anche interessanti applicazioni contemporanee nel machine learning e nella quantificazione dell'incertezza [2].

3 Idee dietro gli algoritmi proposti

L'idea di base della stima probabilistica della traccia è che è facile costruire una variabile casuale il cui valore atteso eguagli la traccia della matrice input.

Si consideri un vettore random $w \in \mathbb{F}^n$ che sia isotropico ($\mathbb{E}[ww^*] = I$). Per la ciclicità della traccia e per linearità:

$$X = w^*(\mathbf{A}w) \text{ soddisfa } \mathbb{E}[X] = \text{trace}(\mathbf{A}) \tag{1}$$

In altre parole, la variabile casuale X è uno stimatore non distorto della traccia.

Si noti che la distribuzione di X dipende dalla matrice incognita $\mathbf{A}$.

Un solo campione di X è raramente adeguato, poiché avrà varianza elevata.

Il più comune meccanismo per ridurre la varianza è di creare k copie indipendenti di X.

Per $k \in \mathbb{N}$, definiamo

$$\bar{X_k} = \frac{1}{k}\sum_{i=1}^{k} X_i \text{ dove } X_i \sim X \text{ sono } i.i.d. \tag{2}$$

Per linearità $\bar{X_k}$ è uno stimatore non distorto della traccia. I campioni sono statisticamente indipendenti, quindi la varianza decresce.

Infatti:

$$\mathbb{E}[\bar{X_k}] = \text{trace}(\mathbf{A}) \text{ e } Var[\mathbf{A}] = \frac{1}{k}Var[X]$$

Lo stimatore (2) può essere considerato il metodo più elementare nell'algebra lineare randomizzata.

Per calcolare $\bar{X_k}$, dobbiamo simulare k copie indipendenti del vettore random $w \in \mathbb{F}^n$ e performare k prodotti matrice-vettore con $\mathbf{A}$, con $O(kn)$ calcoli aggiuntivi.

3.0.1 Matrici generiche

Assumere che $\mathbf{A}$ sia semidefinita positiva, ci permette di concludere che la deviazione standard dello stimatore probabilistico è più piccola della traccia della matrice. Lo stesso metodo ci permette di stimare la traccia di una generica matrice quadrata, ma la varianza dello stimatore potrebbe non essere più confrontabile con la traccia.

3.1 Garanzie a priori

Possiamo ottenere garanzie a priori sulla performance dello stimatore di traccia usando l'analisi teorica. Questi risultati illuminano quali caratteristiche della matrice di input influenzano la qualità della stima della traccia, e ci dicono quanti campioni k sono sufficienti ad ottenere una data tolleranza dell'errore. Si noti, a ogni modo, che questi *bound* dipendono dalle proprietà della matrice input che spesso sono sconosciute a chi usa gli stimatori. Indipendentemente

dalla distribuzione del vettore test isotropico w, la disuguaglianza di Chebyshev ci dona un semplice *bound* probabilistico per lo stimatore di traccia:

$$\mathbb{P}\{|\bar{X}_k - \text{trace}(\mathbf{A})| \geq t\} \leq \frac{Var[X]}{kt^2} \text{ per } t > 0 \tag{3}$$

Possiamo specializzare questo risultato a specifici stimatori di traccia inserendo la varianza.

3.2 Stime dell'errore a posteriori

In pratica, raramente abbiamo a disposizione le informazioni necessarie per stimare l'errore a priori. È più saggio saggiare la qualità delle stime dalle informazioni che raccogliamo. Dato che abbiamo piena conoscenza del processo stocastico che genera gli stimatori della traccia, possiamo usare alcuni approcci della statistica classica. Al livello base, la varianza del campione è uno stimatore non distorto per la varianza dei singoli campioni:

$$X_k = \frac{1}{k-1}\sum_{i=1}^{k}(X_i - \bar{X}_k)^2 \text{ soddisfa } \mathbb{E}[S_k] = Var(X) \tag{4}$$

La varianza di S_k dipende dal momento quarto della variabile random X. Una stima standard è:

$$Var[S_k] \leq \mathbb{E}[(X - \mathbb{E}[X])^4] \tag{5}$$

Bound e stime empiriche della varianza di S_k possono essere ottenute usando la disuguaglianza di Effron-Stein [1].

Per $\alpha \in (0, \frac{1}{2})$ possiamo costruire un intervallo di confidenza (Bilatero, t di Student) di livello $1 - 2\alpha$:

$$trace(\mathbf{A}) \in \bar{X}_k \pm t_{\alpha,k-1}\sqrt{S_K} \tag{6}$$

dove $t_{\alpha,t-1}$ è l'α-quantile della distribuzione t di Student con $k-1$ gradi di libertà. Questo risultato ci dice che trace($\mathbf{A}$) giace nell'intervallo specificato con probabilità circa $1 - 2\alpha$. La pratica suggerisce siano maggiormente efficaci se $k \geq 30$, mentre $\alpha \geq 0.25$.

4 Stimatore di Girard

Consideriamo un vettore aleatorio normale standard $w \sim NORMAL(\mathbf{0}, \mathbf{1})$. La varianza dello stimatore di traccia risultante (1)-(2) soddisfa:

$$Var[\bar{X}_k] = \frac{2}{k}\sum_{i,j=1}^{n}|(\mathbf{A})_{ij}|^2 = \frac{2}{k}\|\mathbf{A}\|_{\mathbb{F}}^2 \leq \frac{2}{k}\|\mathbf{A}\|\,\text{trace}(\mathbf{A}) \tag{7}$$

L'invarianza per rotazioni della distribuzione normale standard ci permette di caratterizzare il comportamento di questo stimatore in pieno dettaglio.

Girard [3] ha anche studiato gli stimatori ottenuti pescando w in maniera uniformemente casuale dalla sfera $\sqrt{n}\mathbb{S}^{n-1}(\mathbb{F})$ per $\mathbb{F} = \mathbb{R}$.

Quando $\mathbb{F} = \mathbb{C}$, questo approccio ha la varianza minima tra tutti gli stimatori di traccia della forma (1)-(2).

4.1 Garanzie a Priori

Definiamo la dimensione intrinseca di A come: $\text{intdim}(\mathbf{A}) = \frac{\text{trace}(\mathbf{A})}{\|\mathbf{A}\|}$.

Se il vettore test w è un vettore aleatorio normale standard, lo stimatore di traccia $\bar{X_k}$ soddisfa:

$$\mathbb{P}\{|\bar{X_k} - \text{trace}(\mathbf{A})| \geq t \cdot \text{trace}(\mathbf{A})\} \leq \frac{2}{k \cdot intdim(\mathbf{A})t^2} \tag{8}$$

Il bound segue da (4), (3). Intuitivamente: il bound dell'errore relativo dello stimatore di traccia è più preciso quando la dimensione intrinseca di A è grande.

4.2 Implementazione

Questo algoritmo stima la traccia della matrice in ingresso, creandone k stimatori non distorti indipendenti e usando la loro media come stima finale. La performance dello stimatore X dipende dalla distribuzione dei vettori test, che in questo caso è Gaussiana.

```
function [X, S]=GirardTE(A,k)
%Prende in ingresso una matrice semidefinita positiva
   A, e il numero k di
%campioni richiesti. Restituisce X, stima della
   traccia e S, varianza dei
%campioni
[m,~]=size(A);
sum=0;
%definisco un array 'dinamico' di valori
a(1)=0;
for i=1:k
 w=normrnd(0,1,m,1); %vettore test isotropico estratto
     da una popolazione Gaussiana
 a(i)=w'*(A*w);
 sum=sum+a(i);
end
X=sum/k;%diverra' la stima della traccia
sum1=0;
for i=1:k
    sum1= (a(i)+X)^2;
end
S=sum1/(k-1); %diverra' la stima della varianza
```

```
end
```

4.3 Errore a posteriori: bootstrap confidence interval

Qui viene usato l'approccio di Miles Lopez per algoritmi di algebra lineare numerica [5]. Sia $\chi = (X_1, ..., X_k)$ il campione empirico da (2). La procedura di bootstrap consiste nell'estrarre con reinserimento ulteriori campioni da χ, per ottenere maggiori informazioni sulla distribuzione campione dello stimatore di traccia $\bar{X}_k$, in questo caso ricaviamo intervalli di fiducia.

```
function [m, n]=BootstrapTraceEstimateG(A, k, B, a)
%Prende in ingresso una matrice semidefinita positiva
   A, il numero k di
%campioni richiesti, il numero B di duplicati
   bootstrap e il parametro a
%per il livello di fiducia.
% Restituisce gli estremi della regione di fiducia di
   livello 1-2a
[m,~]=size(A);
sum=0;
x(1)=0;
for i=1:k
 w=normrnd(0,1,m,1); %vettore test isotropico estratto
     da una popolazione Gaussiana
 x(i)=w'*(A*w);
 sum=sum+x(i);
end
X=sum/k;%diverra' la stima della traccia
e(1)=0;
sum1=0;
for i=1:B
    %estraggo un campione da x con reinserimento
    y=datasample(x, k);
    for j=1:k
    sum1=sum+y(j);
    end
    Y=sum1/k;
    e(i)= Y-X;
end
%calcolo i quantili del vettore di errore (a, 1-a)
q1=quantile(e, a);
q2=quantile(e,1-a);
m=X-q1;
n=X+q2;
end
```

5 Stimatore di Hutchinson

Si consideri un vettore random di Rademacher $w \sim UNIF\{\pm 1\}^n$. La varianza dello stimatore di traccia risultante soddisfa:

$$Var[\bar{X}_k] = \frac{4}{k} \sum_{1 \le i < j \le n} |(\mathbf{A})_{ij}|^2 < \frac{2}{k} \|\mathbf{A}\|_{\mathbb{F}}^2 \le \frac{2}{k} \|\mathbf{A}\| \operatorname{trace}(\mathbf{A}) \tag{9}$$

Questo è lo stimatore di traccia con varianza minima tra quelli generati da un vettore random w isotropico con coordinate statisticamente indipendenti. Inoltre evita la simulazione di variabili normali.

5.1 Implementazione

Questo algoritmo stima la traccia della matrice in ingresso, creandone k stimatori non distorti indipendenti e usando la loro media come stima finale. La performance dello stimatore X dipende dalla distribuzione dei vettori test, che in questo caso è Rademacher.

```
function [X, S]=HutchinsonTE(A,k)
%Prende in ingresso una matrice semidefinita positiva
   A, e il numero k di
%campioni richiesti. Restituisce X, stima della
   traccia e S, varianza dei
%campioni
[m,~]=size(A);
sum=0;
%definisco un array 'dinamico' di valori
a(1)=0;
for i=1:k
 %creo un vettore test isotropico estratto da  una
    popolazione Rademacher
 mp = [-1 1];
 w= (mp((rand(1,m)<.5)+1))';
 a(i)=w'*(A*w);
 sum=sum+a(i);
end
X=sum/k;%diverra' la stima della traccia
sum1=0;
for i=1:k
    sum1= (a(i)+X)^2;
end
S=sum1/(k-1); %diverra' la stima della varianza
end
```

5.2 Errore a posteriori: bootstrap confidence interval

Di seguito, la mia implementazione dell'algoritmo per determinare gli intervalli di confidenza adatti per gli stimatori di Hutchinson.

```
function [m, n]=BootstrapTraceEstimateH(A, k, B, a)
%Prende in ingresso una matrice semidefinita positiva
   A, il numero k di
%campioni richiesti, il numero B di duplicati
   bootstrap e il parametro a
%per il livello di fiducia.
% Restituisce gli estremi della regione di fiducia di
   livello 1-2a
[m,~]=size(A);
sum=0;
x(1)=0;
for i=1:k
 mp = [-1 1];
 w= (mp((rand(1,m)<.5)+1))'; %vettore test estratto da
     popolazione Rademacher
 x(i)=w'*(A*w);
 sum=sum+x(i);
end
X=sum/k;%diverra' la stima della traccia
e(1)=0;
sum1=0;
for i=1:B
    %estraggo un campione da x con reinserimento
    y=datasample(x, k);
    for j=1:k
    sum1=sum+y(j);
    end
    Y=sum1/k;
    e(i)= Y-X;
end
%calcolo i quantili del vettore di errore (a, 1-a)
q1=quantile(e, a);
q2=quantile(e,1-a);
m=X-q1;
n=X+q2;
end
```

6 Sperimentazione

Le sperimentazioni seguenti sono state effettuate utilizzando matlab online. La seguente tabella presenta le matrici test ottenute sul sito (https://sparse.tamu.edu/)

e le loro proprietà.

Nome	SDP	Taglia	Tipo
finance256.mat	Sì	37376x37376	'optimization problem'
inline_1.mat	Sì	503712x503712	'structural problem'
minsurfo.mat	Sì	40806x40806	'optimization problem'
psd10.mat	Sì	16558x16558	'optimization problem'
s3dkq4m2.mat	Sì	90449x90449	'structural problem'

Tabella 1: Questa tabella racchiude le matrici test e le loro proprietà

La seguente tabella presenta il confronto sule matrici test dei due algoritmi da me implementati con il calcolo effettivo della traccia operato da Matlab con la funzione trace.

Matrici	Matlab	Girard	Hutchinson
finance256.mat	3.7376e+04	3.7230e+04	3.7345e+04
inline_1.mat	3.6467e+11	3.6449e+11	3.6472e+11
minsurfo.mat	8.2559e+04	8.2694e+04	8.2572e+04
psd10.mat	1.6558e+04	1.6561e+04	1.6436e+04
s3dkq4m2.mat	6.3413e+07	6.3332e+07	6.3401e+07

Tabella 2: Questa tabella racchiude e confronta le tracce calcolate dai diversi algoritmi, con $k = 100$

La seguente tabella confronta i tempi (in secondi) di esecuzione dei due algoritmi.

Matrici	Girard	Hutchinson
finance256.mat	0.648488	0.085415
minsurfo.mat	0.087097	0.690184
psd10.mat	0.492181	0.263839
s3dkq4m2.mat	1.978267	2.184183

Tabella 3: Questa tabella racchiude e confronta i tempi di esecuzione dei due algoritmi con $k = 100$

6.1 Grafici

Di seguito è possibile osservare, tramite i grafici seguenti, come gli stimatori di traccia approssimino sempre meglio il valore della traccia al crescere di k.

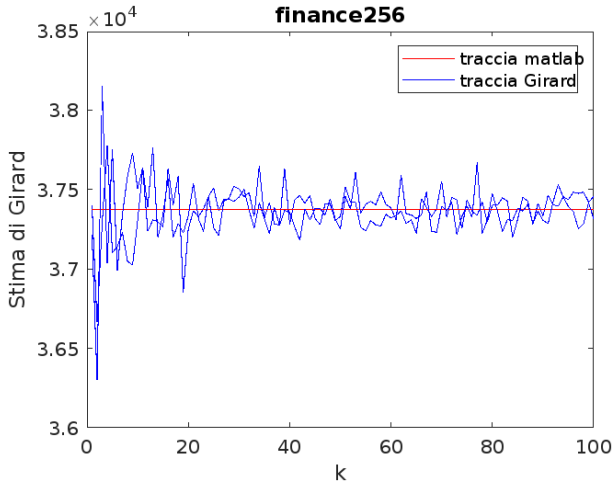

Figura 1: Stima della traccia di Girard al variare di k

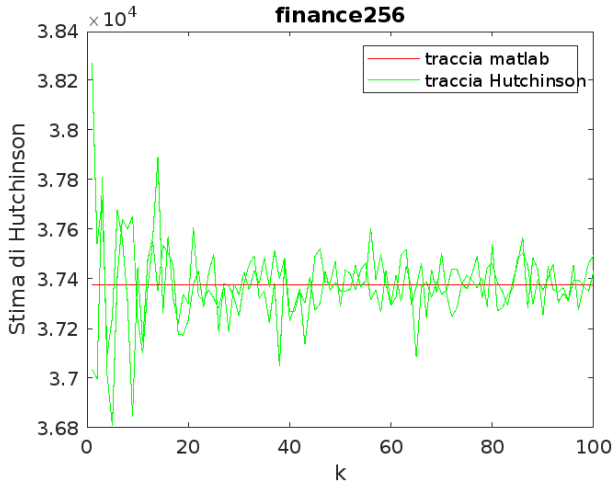

Figura 2: Stima della traccia di Hutchinson al variare di k

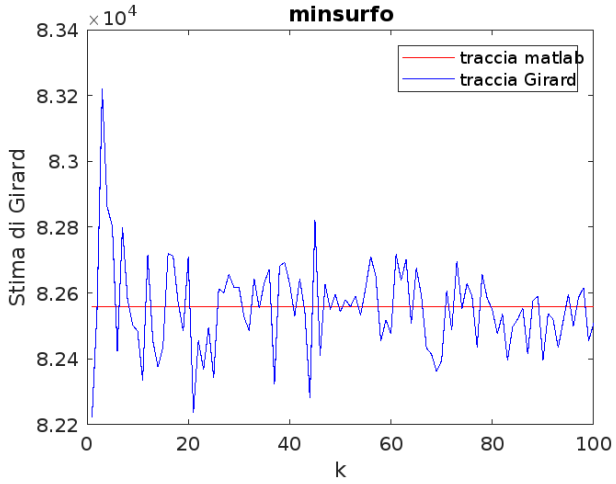

Figura 3: Stima della traccia di Girard al variare di k

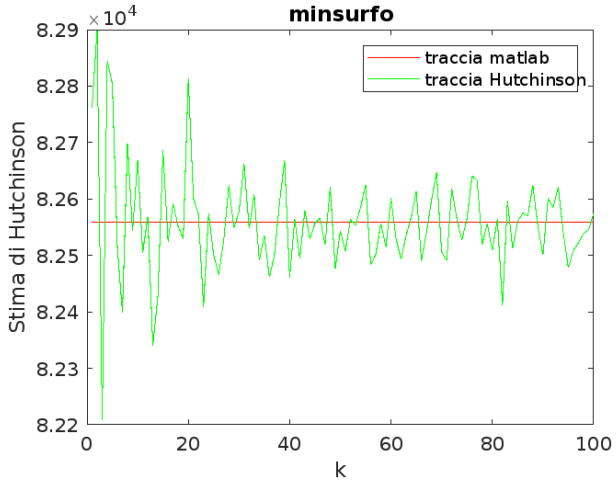

Figura 4: Stima della traccia di Hutchinson al variare di k

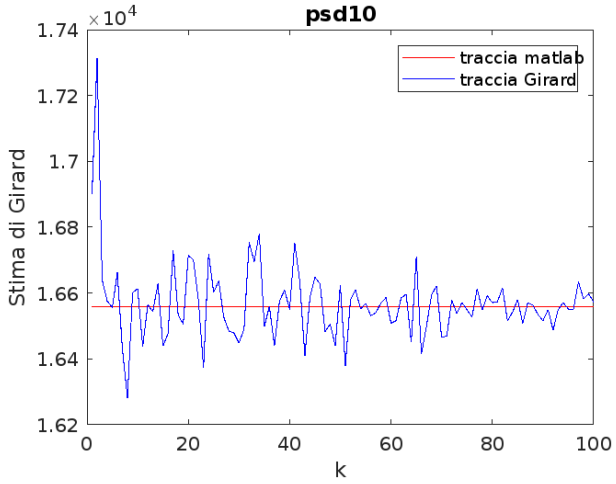

Figura 5: Stima della traccia di Girard al variare di k

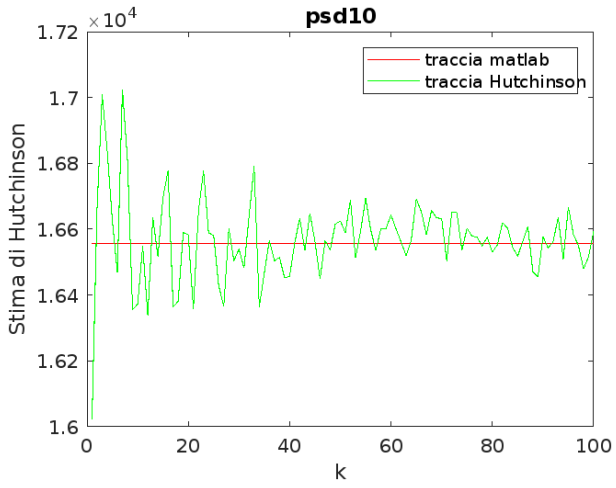

Figura 6: Stima della traccia di Hutchinson al variare di k

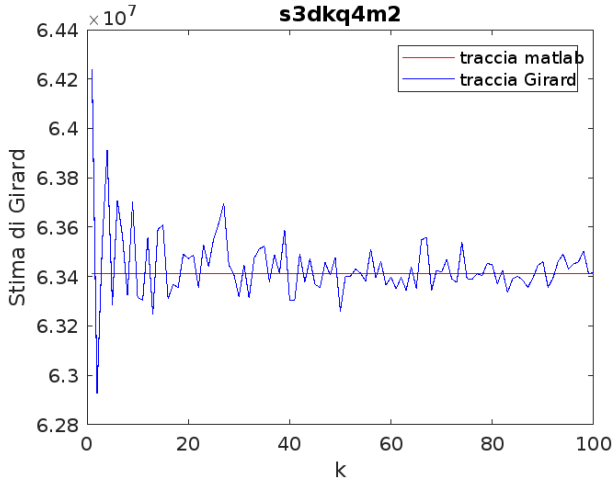

Figura 7: Stima della traccia di Girard al variare di k

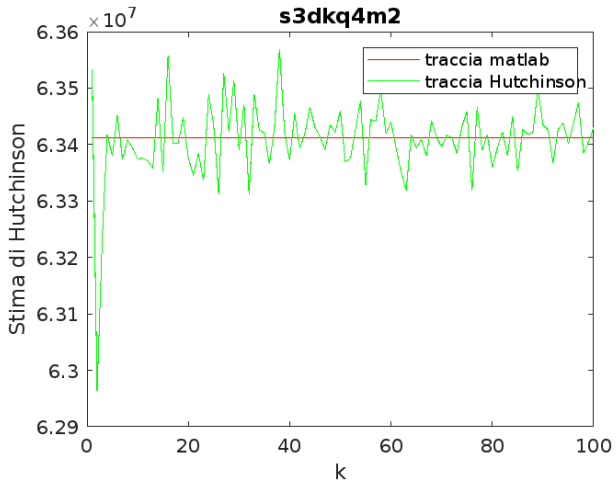

Figura 8: Stima della traccia di Hutchinson al variare di k

6.2 Errori a priori

Ai fini dello studio dell'errore a priori mi servo di matrici di test da me generate, poiché le matrici sopra indicate hanno una taglia troppo elevata perché si possa calcolare la loro norma su matlab in full.

Nome	SDP	Taglia	$\|\| \cdot \|\|$
A	Sì	12000x12000	1.8000e+04
B	Sì	6600x6600	9.8995e+03
C	Sì	23400x23400	3.5100e+04
D	Sì	5000x5000	7.4997e+03

Tabella 4: Questa tabella racchiude le principali informazioni sulle matrici test

Matrici	Girard	Hutchinson
A	5.1841e+10	5.1841e+10
B	8.6251e+09	8.6251e+09
C	3.8440e+11	3.8440e+11
D	3.7503e+09	3.7503e+09

Tabella 5: Questa tabella racchiude e confronta gli errori a priori dei due algoritmi, $k = 100$, $t = 0.1$

Come si poteva già notare dai risultati teorici, l'errore a priori dei due algoritmi ha il medesimo *upper bound*.

6.3 Errori a posteriori

La tabella seguente confronta gli errori a posteriori dei due algoritmi considerati, calcolati utilizzando l'algoritmo di Bootstrap confidence interval da me implementato.

Matrici	Girard	Hutchinson
finance256.mat	[3.7099e+04 , 3.7848e+04]	[3.7047e+04 , 3.7796e+04]
inline_1.mat	[3.6097e+11 , 3.6826e+11]	[3.6106e+11 , 3.6835e+11]
minsurfo.mat	[8.1829e+04 , 8.3481e+04]	[8.1727e+04 , 8.3379e+04]
psd10.mat	[1.6392e+04 , 1.6723e+04]	[1.6413e+04 , 1.6746e+04]
s3dkq4m2.mat	[6.2672e+07 , 6.3939e+07]	[6.2722e+07 , 6.3990e+07]

Tabella 6: Questa tabella racchiude e confronta gli intervalli di confidenza dei due algoritmi, utilizzando i dati raccolti con $k = 100$, a=0.05, B=200

Alla luce delle nostre sperimentazioni, l'algoritmo di Hutchinson risulta più efficace.

Riferimenti bibliografici

[1] Stéphane Boucheron, Gábor Lugosi e Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[2] Jack K Fitzsimons et al. "Improved stochastic trace estimation using mutually unbiased bases". In: *arXiv preprint arXiv:1608.00117* (2016).

[3] Antoine Girard. "A fast 'Monte-Carlo cross-validation' procedure for large least squares problems with noisy data". In: *Numerische Mathematik* 56 (1989), pp. 1–23.

[4] Michael F Hutchinson. "A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines". In: *Communications in Statistics-Simulation and Computation* 18.3 (1989), pp. 1059–1076.

[5] Miles E Lopes. "Estimating the algorithmic variance of randomized ensembles via the bootstrap". In: *The Annals of Statistics* 47.2 (2019), pp. 1088–1112.