UNIVERSITÀ DEGLI STUDI
DI PERUGIA

Dipartimento di Ingegneria

INGEGNERIA INFORMATICA E ROBOTICA

Sviluppo e confronto tra modelli di Support Vector Machine (SVM): approccio centralizzato e approccio splitted by examples tramite algoritmo ADMM

Gianluca Coletta, 340208

Anno Accademico 2022/2023

Indice

1 Introduzione

Lo scopo di questo progetto è l'implemntazione di una **Support Vector Machine** tramite **ADMM** (Alternating Direction Method of Multipliers) in due differenti versioni:

- Versione ADMM centralizzata
- Versione ADMM distribuita splitted by examples

L'importanza dell'ottimizzazione **distribuita** è ormai conosciuta, visto il diffondersi massivo di cluster per il calcolo, che prevedono architetture di tipo distribuito, le quali presentano un certo numero di nodi connessi da una rete, ognuno con una sua capacità di calcolo e di storage.

Le **Support Vector Machines** (SVM) sono potenti algoritmi di apprendimento automatico ampiamente utilizzati in problemi di classificazione e regressione. Ciò che le rende particolarmente interessanti è la loro capacità di gestire efficacemente sia dati lineari che non lineari in uno spazio multidimensionale.

Il principio chiave di una SVM è la ricerca dell'iperpiano ottimale in uno spazio ad alta dimensione che possa separare distintamente le diverse classi di dati. Questo iperpiano è definito in modo da massimizzare il margine tra le classi, ovvero la distanza tra l'iperpiano e i punti dati più vicini di ciascuna classe, noti come *support vector*. Questo approccio è chiamato "max-margin".

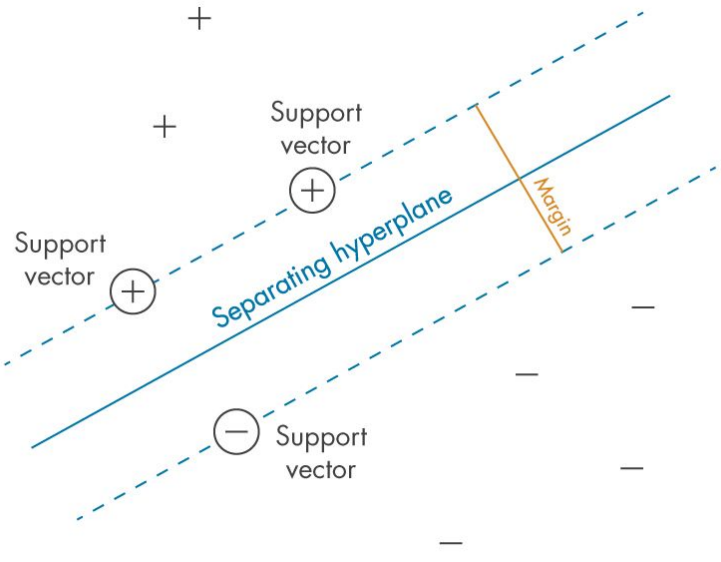


Figura 1: Support Vector Machine

La formulazione di una **SVM** è espressa dalla seguente formula:

$$\sum_{i=1}^{n} min_{\beta,\beta_0} \left\{1 - y_i(\underline{\beta}^T \underline{p_i} + \beta_0)\right\} + \lambda \left\|\underline{\beta}\right\|_2^2$$

Gli algoritmi di ottimizzazione possono essere suddivisi in due categorie: tecniche di ottimizzazione primale e tecniche di ottimizzazione primale-duale. Nella seconda categoria rientra il metodo **ADMM**. Questo è un metodo iterativo utilizzato per risolvere problemi di apprendimento generale, come:

$$min_{x,z} f(x) + g(z)$$

$$s.t.\ A\underline{x} + B\underline{z} = \underline{c}$$

L'**ADMM** è un algoritmo di ottimizzazione iterativo che risolve problemi complessi suddividendoli in sotto-problemi più gestibili. Funziona attraverso tre passaggi principali:

1. **Aggiornamento delle variabili primarie**: Si aggiornano le variabili del problema principale.
2. **Aggiornamento delle variabili duali**: Si aggiornano le variabili duali legate ai vincoli del problema.
3. **Aggiornamento dei moltiplicatori**: Si aggiornano i moltiplicatori lagrangiani.

La formulazione generica è la seguente:

$$\underline{x}^{k+1} = argmin_{\underline{x}} L_\rho(\underline{x}, \underline{z}^{(k)}, \underline{\nu}^{(k)})$$

$$\underline{z}^{k+1} = argmin_{\underline{z}} L_\rho(\underline{x}^{(k+1)}, \underline{z}, \underline{\nu}^{(k)})$$

$$\underline{\nu}^{k+1} = \underline{\nu}^k + \rho(A\underline{x}^{(k+1)} + B\underline{z}^{(k+1)} - \underline{c})$$

2 Tecnologie utilizzate

In questa sezione verranno presentate e descritte brevemente le tecnologie utilizzate nel progetto.

2.1 MATLAB

MATLAB, acronimo di "Matrix Laboratory," è un potente **ambiente di programmazione e calcolo numerico** utilizzato in una vasta gamma di discipline scientifiche ed ingegneristiche. Si basa principalmente sulla manipolazione efficiente di matrici e offre una vasta libreria di funzioni per l'analisi dei dati, la simulazione, la visualizzazione e la risoluzione di problemi matematici complessi. L'interfaccia utente interattiva di MATLAB consente agli utenti di eseguire comandi e scrivere script per risolvere una varietà di problemi in modo efficiente, rendendolo uno strumento essenziale per la ricerca e lo sviluppo tecnologico.

2.2 CVX

CVX è un **ambiente di modellazione e ottimizzazione** progettato per semplificare la risoluzione di problemi di ottimizzazione convessa. Conosciuto per la sua semplicità e facilità d'uso, CVX consente agli utenti di formulare problemi di ottimizzazione convessa in modo chiaro e intuitivo, senza doversi preoccupare di dettagli complessi di implementazione.

Gli utenti possono esprimere i propri problemi di ottimizzazione convessa utilizzando una sintassi simile alla notazione matematica standard, rendendo CVX uno strumento ideale per ricercatori, ingegneri e scienziati che vogliono risolvere problemi complessi di ottimizzazione in modo efficiente.

2.3 Python

Python è un **linguaggio di programmazione** ampiamente utilizzato nell'ambito dell'Exploratory Data Analysis (**EDA**). Con Python, puoi eseguire un'analisi esaustiva dei dati in modo efficiente. Python offre librerie come pandas, numpy, matplotlib e seaborn che semplificano l'importazione, la manipolazione, la visualizzazione e la comprensione dei dati. L'EDA in Python consente di esplorare dati, identificare tendenze, rilevare outlier e preparare i dati per l'analisi statistica o la modellazione.

3 Il dataset utilizzato

Il dataset che stiamo esaminando contiene una vasta gamma di informazioni relative alle proprietà chimiche e fisiche di diversi tipi di vini. Il dataset è composto da 1599 campioni e contiene le seguenti **features**:

- **Fixed acidity**
- **Volatile acidity**
- **Citric acid**
- **Residual sugar**
- **Chlorides**
- **Free sulfur dioxide**
- **Total sulfur dioxide**
- **Density**
- **pH**
- **Sulphates**
- **Alcohol**
- **Quality**

L'ultima caratteristica, denominata **Quality**, rappresenta l'etichetta di interesse in questo dataset. Può assumere uno dei due valori: "buona" (*good*) o "cattiva" (*bad*). Per facilitare la creazione di un classificatore binario utilizzando **SVM**, questi due valori sono stati convertiti in numeri interi. In particolare, il valore "buona" è stato codificato come 1, mentre il valore "cattiva" è stato codificato come -1. In alcuni casi, come nella creazione di una *confusion matrix*. la codifica può richiedere anche il valore 0, ma l'uso principale è quello di associare chiaramente una classe positiva (1) e una classe negativa (-1) per scopi di classificazione binaria.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	-1
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	-1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	-1
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	1
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	-1

Figura 2: Prime 5 righe del dataset

La metrica scelta per valutare l'efficacia del modello SVM è l'***accuracy***. Questa scelta è motivata dal fatto che si può osservare un bilanciamento quasi uniforme tra le due classi, come evidenziato nella figura seguente:

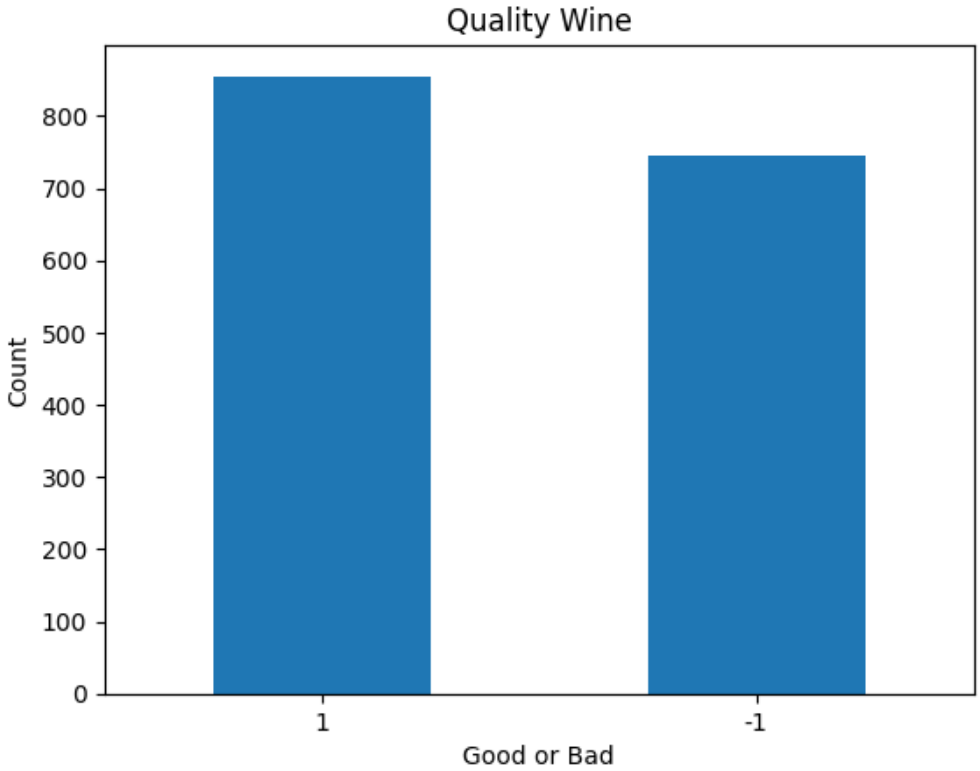


Figura 3: Distrubizione campioni per classi

Infine il dataset è stato suddiviso randomicamente secondo le seguenti proporzioni:

- L'80 % dei dati è stato allocato per la fase di addestramento (**training set**).
- Il restante 20 % dei dati è stato riservato per la fase di test (**test set**).

Il dataset è disponibile al seguente link.

4 Approccio SVM centralizzato

Ricordiamo che la formulazione di una **SVM** è espressa dalla seguente formula:

$$\sum_{i=1}^{n} min_{\beta,\beta_0} \left\{1 - y_i(\underline{\beta}^T \underline{p_i} + \beta_0)\right\} + \lambda \left\|\underline{\beta}\right\|_2^2$$

Per ricondurci ad un problema del tipo

$$l(Ax - b) + r(x)$$

dove **A** è la matrice delle features e **x** è il vettore contenente tutti i parametri da stimare, andiamo a rappresentare il problema di ottimizzazione nella seguente forma equivalente. Con:

$$\underline{x}^T = \left[\underline{\beta}^T, \beta_0\right]$$

$$\underline{a_i} = y_i \begin{pmatrix} p_i \\ 1 \end{pmatrix}$$

$$\underline{\widetilde{x}} = \underline{\beta} = C\underline{x} = \begin{bmatrix} I & \underline{0} \\ \underline{0}^T & 0 \end{bmatrix} \left[\underline{\beta}\ \beta_0\ \right]$$

Il problema diventa:

$$min_{\underline{x}} \sum_{i=1}^{n} max\ \left\{0, 1 - \underline{a_i}^T \underline{x}\right\} + \lambda \left\|\widetilde{x}\right\|_2^2$$

Che può essere riscritto come:

$$min_{\underline{x}}\, \underline{1}^T max\left\{0, \underline{1}^T - A\underline{x}\right\} + \lambda \left\|Cx\right\|_2^2$$

La loss function è **additiva** e la regolarizzazione è separabile quindi la formulazione si presta ad approcci di calcolo distribuito sia per sample che per features. Per la versione centralizzata abbiamo che:

$$min_{\underline{x}}\, \underline{1}^T max\left\{0, \underline{1}^T - A\underline{x}\right\} + \lambda \left\|\underline{x}\right\|_2^2$$

In MATLAB, tramite la libreria CVX, questa espressione può essere espressa nel seguente modo:

```
cvx_begin quiet
variables x_v(n+1)
    minimize (sum( max(0, 1 - A*x_v)) + lambda*sum_square(x_v))
cvx_end
```

Figura 4: SVM centralizzata tramite CVX

5 Approccio SVM splitted by examples

La versione distribuita (o parallela) dell'algoritmo di ADMM si basa sulla seguente architettura

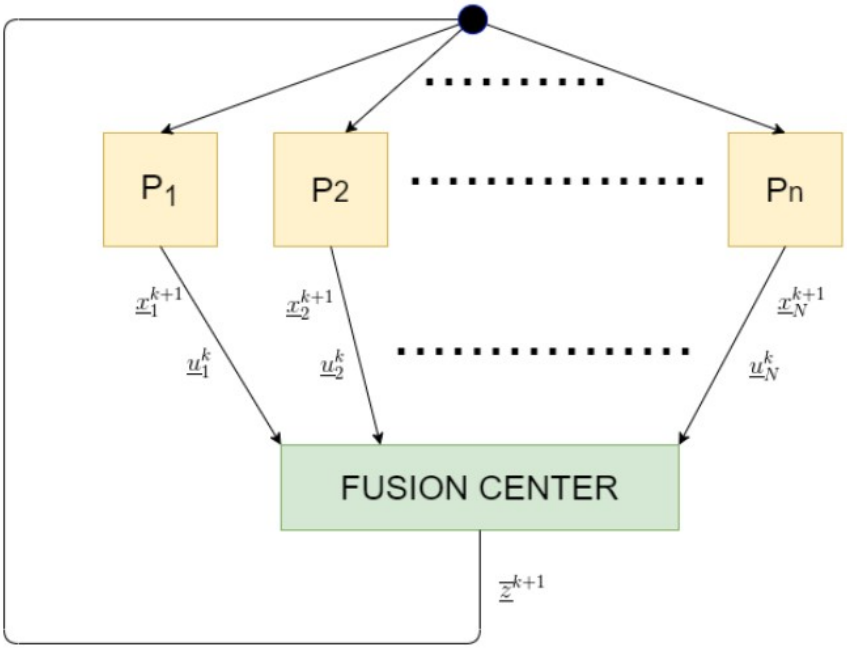


Figura 5: ADMM distribuito

in cui

$$P_i,\ i = 1, \dots N$$

sono i processori che lavorano in parallelo, x e u sono le variabili locali, mentre z è la variabile globale.

Per scrivere il problema di classificazione tramite ADMM, bisogna innanzitutto dividere la matrice A e il vettore b per samples, ossia per righe, in modo da distribuire un numero simile di sample in ogni processore rimanendo vincolati a raggiungere la stessa soluzione dal vettore z.

Quindi il problema è:

$$min_{\underline{x}} \sum_{j=1}^{N} \underline{1}_{n_j}^T max \left\{0, \underline{1}_{n_j} - A_j \underline{x}_j\right\} + \lambda \left\|z\right\|_2^2$$

Quindi separiamo le varie formule iterative tra le variabili:

$$\underline{x}^{k+1} = argmin_{\underline{x}} L_\rho(\underline{x}, \underline{z}^{(k)}, \underline{\nu}^{(k)})$$

Che diventa:

$$\underline{x}^{k+1} = argmin_{\underline{x}_i} \sum_{j=1}^{n_j} max\left\{0, 1 - \underline{a}_{ij}^T \underline{x}_i\right\} + \frac{\rho}{2} \left\| \underline{x}_i - \underline{z}^{(k)} + \underline{u}^{(k)} \right\|_2^2$$

In MATLAB l'espressione è scritta nel seguente modo:

```
for i = 1:N
    cvx_begin quiet
        variable x_v(n+1)
        minimize (sum(max(0, A_split{i}*x_v+1)) + (rho/2)*sum_square(x_v - z + u(:,i)))
    cvx_end
    X(:,i) = x_v;
end
```

Per quanto riguarda Z:

$$\underline{z}^{k+1} = argmin_{\underline{z}} L_\rho(\underline{x}^{(k+1)}, \underline{z}, \underline{\nu}^{(k)})$$

Che è uguale a:

$$\underline{z}^{k+1} = argmin_{\underline{z}} \lambda \left\| Cz \right\|_2^2 + \frac{N\rho}{2} \left\| \overline{\underline{x}}^{(k)} - \underline{z} + \overline{\underline{u}}^{(k)} \right\|$$

Svolengdo i calcoli diventa:

$$\underline{z}^{k+1} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\overline{\underline{x}}^{(k)} + \overline{\underline{u}}^{(k)} \right)$$

Con

$$\alpha = \frac{N\rho}{2\lambda + N\rho}$$

In MATLAB:

```
z = (N*rho/(2*lambda + N*rho))*mean(X + u, 2);
```

Infine la variabile U corrisponde a:

$$\underline{u}_i^{(k+1)} = \underline{u}_i^{(k)} - \underline{z}^{(k)} + \underline{x}_i^{(k)}$$

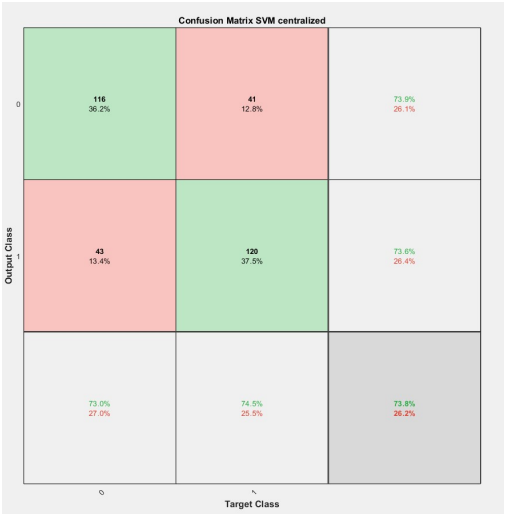
In MATLAB:

```
for i = 1:N
    u(:,i)=u(:,i) + X(:,i) - z;
end
```

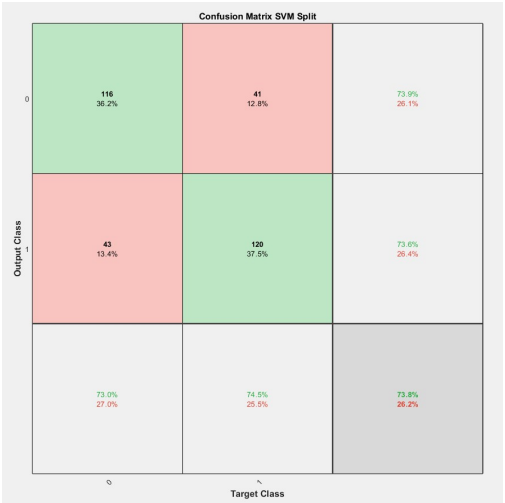
6 Risultati e conclusioni

Implementando le soluzioni in versione centralizzata e distribuita, quello che si ottiene è che i parametri trovati nella soluzione distribuita convergono con consenso quasi agli stessi valori corrispondenti a quelli ottenuti risolvendo la versione centralizzata.

Di seguito vengono mostrati i grafici rigurdanti la *confusion matrix* e la *ROC* dei due diversi approcci utilizzati: La *confusione matrix* è una tabella che aiuta a valutare le prestazioni di un modello di classificazione, rivelando quanti casi sono stati previsti correttamente e quanti in modo errato. La curva *ROC*, d'altra parte, è un grafico che rappresenta quanto bene il modello distingue tra le classi positive e negative al variare della soglia di decisione. Entrambi sono strumenti cruciali per valutare la capacità di un modello di classificazione.

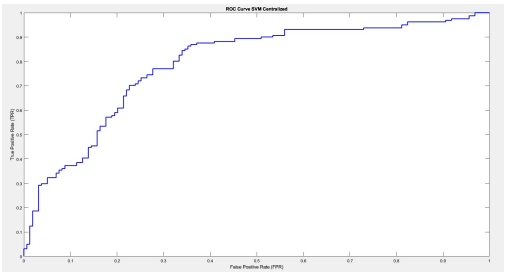


(a) Approccio centralizzato

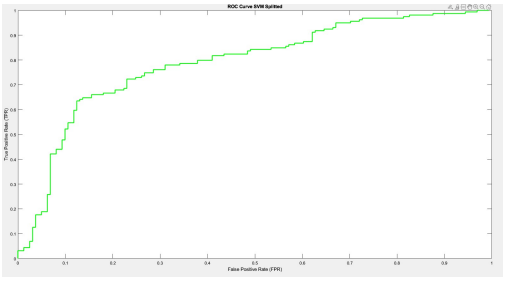


(b) Approccio splitted

Figura 6: Confronto confusion matrix



(a) Approccio centralizzato

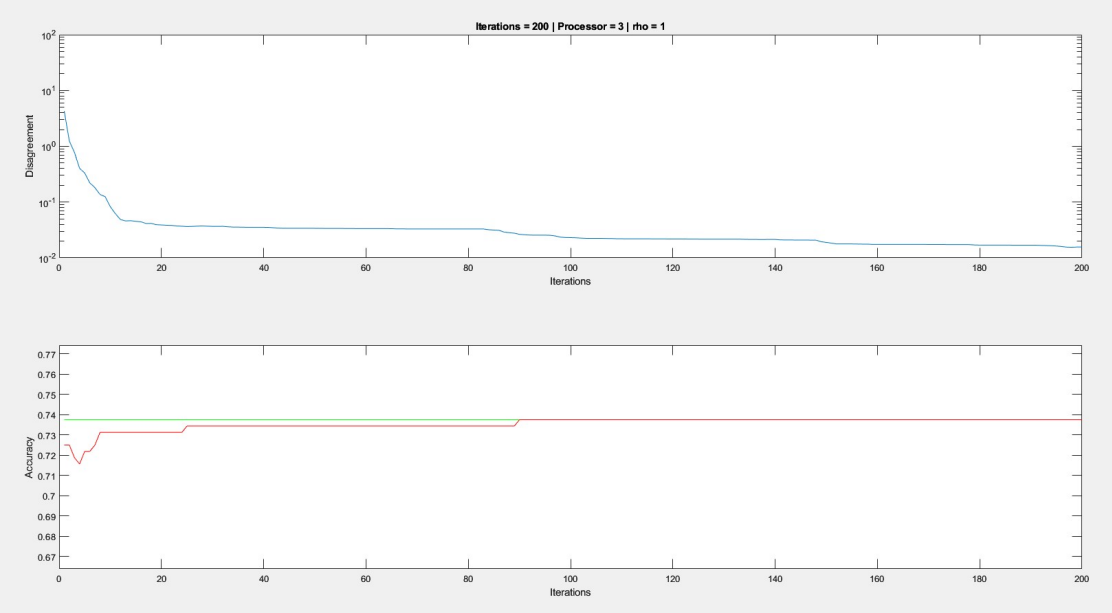


(b) Approccio splitted

Figura 7: Confronto ROC

Infine, di seguito sono mostrati i seguenti grafici:

- Il primo grafico mostra la somma dei valori del vettore differenza di $\underline{x}_i$ con la media di tutti gli $\underline{x}_i$ in ogni processore, al variare delle iterazioni.
- Il secondo mostra l'accuracy della versione centralizzata (costante e in verde) insieme a quella della versione distribuita (in rosso) al variare delle iterazioni.



Dal secondo grafico si può osservare che dopo alcune iterazioni, l'accuratezza della versione distribuita tende a convergere verso il valore ottenuto dalla versione centralizzata.