Università degli Studi di Perugia

Dipartimento di Ingegneria

ingegneria informatica e robotica

SVM: un confronto in termini di performance tra sharing optimization

Dervishaj Ergys

Luca Fagiolo

Indice

1. Introduzione

L'elaborato ha come obiettivo quello di documentare il confronto tra varie formulazioni di SVM analizzando anche un eventuale miglioramento del problema risolto mediante l'integrazione di PCA concentrandosi nelle differenze in termini di performance sia qualitative che prestazionali. SVM è un algoritmo di Machine Learning di tipo Supervised utilizzato per risolvere il task di classificazione. In questo caso, il problema di classificazione affrontato mediante l'algoritmo SVM è quello di Loan Approval, utilizzando il dataset "Loan Approval Classification". Questo dataset è una versione sintetica ispirata al dataset originale Credit Risk presente su Kaggle ed è stato arricchito con variabili aggiuntive basate su dati di Financial Risk per l'approvazione dei prestiti.

2. Obiettivi del progetto

Il confronto tra le varie formulazioni di SVM è realizzato con due obiettivi principali: principali:

- dimostrare che dopo un numero sufficiente di iterazioni le soluzioni ottenute in modo distribuito coincidono con la soluzione centralizzata
- confrontare le tre formulazioni in termini di performance

Le principali fasi operative del progetto sono:

- implementazione e risoluzione del problema di SVM in versione centralizzata in codice MATLAB
- formulazione teorica del problema in forma distribuita secondo l'approccio ADMM in configurazione parallela considerando uno schema di suddivisione rispetto ai dati (Splitting Across Examples) e uno schema di suddivisione rispetto alle feature (Splitting Across Features)
- implementazione e risoluzione delle due versioni distribuite in codice MATLAB
- utilizzo di PCA per ridurre la dimensionalità del problema con l'obiettivo di migliorare le performance cercando di preservare l'accuracy
- analisi critica dei risultati ottenuti, con particolare attenzione sull'efficacia delle diverse formulazioni in termini di accuratezza, efficienza e convergenza

INSERIRE IMMAGINE SCHEMA ADMM PARALLELO

2.1 Formulazione teorica

Support Vector Machine (SVM) è utilizzato principalmente per task di classificazione. Il principio fondamentale consiste nell'individuare un iperpiano ottimale di separazione tra le diverse categorie presenti nei dati. Tale iperpiano è definito in modo da massimizzare il margine, ossia la distanza minima che intercorre tra esso e i punti di ciascuna classe. Massimizzare questo margine equivale a garantire la maggiore capacità di generalizzazione possibile del modello, rendendolo meno sensibile alle variazioni dei dati di addestramento. I Support Vectors rappresentano le istanze critiche del dataset: sono i punti appartenenti alle classi che risultano più prossimi al confine di separazione. Essi determinano univocamente la posizione e l'orientamento dell'iperpiano ottimale, poiché racchiudono l'informazione essenziale per la definizione del margine. In altre parole, l'intera soluzione della SVM dipende esclu-

sivamente da questi punti, che rappresentano i campioni più difficili da classificare in maniera corretta. Sia dato un insieme di dati di training

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}, \quad \mathbf{x}_i \in \mathbb{R}^n,\ y_i \in \{-1, +1\}.$$

L'obiettivo di Support Vector Machine (SVM) è individuare un iperpiano che separi le due classi con il margine massimo. Un iperpiano può essere espresso come:

$$\mathbf{w}^\top \mathbf{x} + b = 0,$$

dove $\mathbf{w} \in \mathbb{R}^n$ rappresenta il vettore dei pesi e $b \in \mathbb{R}$ è il termine di bias. Affinché i dati siano separati correttamente, si impone:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i,$$

e il margine geometrico tra le due classi è dato da:

$$\frac{2}{\|\mathbf{w}\|}.$$

Massimizzare il margine equivale dunque a minimizzare la norma di $\mathbf{w}$.

2.1.1 Problema di ottimizzazione

La formulazione primale del problema è:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{soggetto a} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i. \end{aligned}$$

Si tratta di un problema convesso, la cui soluzione dipende esclusivamente dai punti di frontiera detti **Support Vectors**.

2.1.2 Introduzione delle slack variables

Per gestire dati non perfettamente separabili, si introducono le variabili di scarto $\xi_i \geq 0$, ottenendo:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \\ \text{soggetto a} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0,\ \forall i, \end{aligned}$$

dove $C > 0$ controlla il compromesso tra ampiezza del margine e penalizzazione degli errori.

2.1.3 Forma duale

Applicando i moltiplicatori di Lagrange, si ottiene la forma duale:

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

$$\text{soggetto a} \quad 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{m} \alpha_i y_i = 0.$$

La soluzione ottimale risulta:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i.$$

2.1.4 Hinge Loss

Il problema SVM può essere riscritto come minimizzazione mediante la **Hinge Loss**:

$$L_{\text{hinge}}(y_i, f(\mathbf{x}_i)) = \max(0, 1 - y_i f(\mathbf{x}_i)),$$

dove $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$. La formulazione equivalente è:

$$\min_{\mathbf{w}, b} \ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)).$$

2.2 Alternating Direction Method of Multipliers (ADMM)

Alternating Direction Method of Multipliers (ADMM) è un algoritmo iterativo per la risoluzione di problemi di ottimizzazione convessa, particolarmente efficace quando la funzione obiettivo è separabile in più termini. La logica fondamentale di ADMM consiste nell'alternare l'ottimizzazione di due sotto-problemi accoppiati da un vincolo lineare di uguaglianza, aggiornando a ogni iterazione una variabile duale che funge da coordinatore tra i due sotto-problemi.

2.2.1 Forma generale del problema

Si consideri un problema di ottimizzazione convessa nella forma:

$$\min_{x \in \mathbb{R}^n,\, z \in \mathbb{R}^m} \ f(x) + g(z) \quad \text{soggetto a} \quad Ax + Bz = c,$$

dove:

- f e g sono funzioni convesse e non necessariamente differenziabili;
- $A \in \mathbb{R}^{p \times n}$ e $B \in \mathbb{R}^{p \times m}$ sono matrici che definiscono il vincolo di accoppiamento lineare tra le variabili;
- $c \in \mathbb{R}^p$ è un vettore che specifica il termine noto del vincolo.

2.2.2 Lagrangiano aumentato

Il Lagrangiano aumentato associato al problema è definito come:

$$\mathcal{L}_\rho(x, z, \nu) = f(x) + g(z) + \nu^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2,$$

dove $\nu \in \mathbb{R}^p$ rappresenta il vettore dei moltiplicatori duali associati al vincolo lineare e $\rho > 0$ è il parametro di penalità aumentata. Il termine quadratico aggiunto, proporzionale al quadrato della norma euclidea della violazione del vincolo, garantisce la stabilità numerica dell'algoritmo e favorisce la convergenza anche in presenza di funzioni non differenziabili o mal condizionate.

È utile introdurre la variabile duale scalata $u = \frac{1}{\rho}\nu$, che permette di riscrivere il Lagrangiano in una forma equivalente più compatta:

$$\mathcal{L}_\rho(x, z, u) = f(x) + g(z) + \frac{\rho}{2}\|Ax + Bz - c + u\|_2^2 - \frac{\rho}{2}\|u\|_2^2,$$

dove il termine $-\frac{\rho}{2}\|u\|_2^2$ è una costante rispetto alle variabili primali e può essere omesso durante l'ottimizzazione. Questa formulazione rende gli aggiornamenti duali più intuitivi e facilita l'analisi di convergenza.

2.2.3 Schema iterativo generale

L'algoritmo ADMM procede alternando tre fasi di aggiornamento ad ogni iterazione k:

Passo 1 - Minimizzazione rispetto a x:

$$x^{(k+1)} := \arg\min_{x \in \mathbb{R}^n} \left(f(x) + \frac{\rho}{2}\|Ax + Bz^{(k)} - c + u^{(k)}\|_2^2 \right).$$

In questo passo si minimizza il Lagrangiano aumentato rispetto alla variabile x, mantenendo fisse le altre variabili ai loro valori correnti. Il termine quadratico aggiunto trasforma il problema in uno fortemente convesso.

Passo 2 - Minimizzazione rispetto a z:

$$z^{(k+1)} := \arg\min_{z \in \mathbb{R}^m} \left(g(z) + \frac{\rho}{2}\|Ax^{(k+1)} + Bz - c + u^{(k)}\|_2^2 \right).$$

Questo passo minimizza il Lagrangiano rispetto a z, utilizzando il valore appena calcolato di $x^{(k+1)}$. **Passo 3 - Aggiornamento duale:**

$$u^{(k+1)} := u^{(k)} + (Ax^{(k+1)} + Bz^{(k+1)} - c).$$

Questo aggiornamento corrisponde a un passo di ascesa del gradiente sul problema duale. Ogni iterazione riduce progressivamente la discrepanza tra le variabili primali e spinge il sistema verso il soddisfacimento del vincolo, garantendo la convergenza al punto di ottimo sotto condizioni appropriate.

2.2.4 Interpretazione distribuita e ADMM con consenso

Una delle caratteristiche più rilevanti di ADMM è la sua naturale capacità di distribuzione computazionale, che lo rende ideale per l'ottimizzazione su larga scala e per sistemi multi-agente. Se la funzione obiettivo $f(x)$ è separabile rispetto a blocchi di variabili, ovvero:

$$f(x) = \sum_{i=1}^{N} f_i(x_i), \quad \text{con } x = (x_1, \ldots, x_N) \text{ e } x_i \in \mathbb{R}^{n_i},$$

si può introdurre una variabile globale di consenso $z \in \mathbb{R}^n$ e imporre vincoli di accoppiamento del tipo:

$$A_i x_i = z, \quad i = 1, \ldots, N,$$

dove ogni matrice $A_i \in \mathbb{R}^{n \times n_i}$ proietta la variabile locale x_i nello spazio comune. Il problema complessivo diventa:

$$\min_{x_1,\ldots,x_N,z} \sum_{i=1}^{N} f_i(x_i) + g(z) \quad \text{soggetto a} \quad A_i x_i = z,\ i = 1, \ldots, N.$$

In questo schema, ogni agente i risolve localmente il proprio sotto-problema di ottimizzazione:

$$x_i^{(k+1)} := \arg\min_{x_i \in \mathbb{R}^{n_i}} \left(f_i(x_i) + \frac{\rho}{2} \| A_i x_i - z^{(k)} + u_i^{(k)} \|_2^2 \right),$$

che dipende solo dalla propria funzione obiettivo locale f_i e dalla propria variabile duale u_i. Questi sotto-problemi possono essere risolti in parallelo su processori diversi o da agenti autonomi senza necessità di comunicazione diretta tra loro.

La variabile globale di consenso viene poi aggiornata in modo centralizzato (o non) come:

$$z^{(k+1)} := \arg\min_{z \in \mathbb{R}^{n_{\text{glob}}}} \left(N g(z) + \frac{\rho}{2} \sum_{i=1}^{N} \| A_i x_i^{(k+1)} - z + u_i^{(k)} \|_2^2 \right).$$

Quando $g(z) = 0$ (assenza di regolarizzazione globale) e $A_i = I$ per ogni i, questo problema ha soluzione in forma chiusa:

$$z^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} (x_i^{(k+1)} + u_i^{(k)}),$$

che corrisponde semplicemente alla media aritmetica delle stime locali aggiustate.

Infine, le variabili duali si aggiornano localmente secondo:

$$u_i^{(k+1)} := u_i^{(k)} + (A_i x_i^{(k+1)} - z^{(k+1)}), \quad i = 1, \ldots, N.$$

Lo schema, noto come *Consensus ADMM*, consente di risolvere in modo parallelo e scalabile problemi convessi di grandi dimensioni mantenendo un unico punto di convergenza globale condiviso tra tutti gli agenti. La struttura di comunicazione richiesta è minimale: ogni agente comunica solo con un coordinatore centrale (o con i propri vicini in una rete) per sincronizzare la variabile di consenso z.

2.2.5 ADMM per SVM distribuita secondo i campioni (Split-by-Examples)

Il problema SVM in forma primale può essere scritto come:

$$\min_{\mathbf{x}} \left\{ \sum_{i=1}^{n} \max(0, 1 - \mathbf{a}_i^\top \mathbf{x}) + \lambda \|\mathbf{x}\|_2^2 \right\},$$

dove ogni termine $\ell_i(\mathbf{x}) = \max(0, 1 - \mathbf{a}_i^\top \mathbf{x})$ rappresenta la *Hinge Loss* associata al singolo esempio. Per una formulazione distribuita, i campioni vengono raggruppati in N blocchi, ciascuno di dimensione n_j, con $\sum_{j=1}^{N} n_j = n$. Indicando con $\mathbf{A}_j$ e $\mathbf{x}_j$ rispettivamente la matrice dei dati e la variabile locale del j-esimo agente, il problema diventa:

$$\begin{aligned} &\min_{\{\mathbf{x}_j\}} \sum_{j=1}^{N} \left(\frac{1}{n_j} \mathbf{1}_{n_j}^\top \max\left(\mathbf{0}_{n_j},\ \mathbf{1}_{n_j} - \mathbf{A}_j \mathbf{x}_j\right) + \lambda \|\mathbf{C}\mathbf{z}\|_2^2 \right) \\ &\text{s.t. } \mathbf{A}_j \mathbf{x}_j - \mathbf{z} = \mathbf{0}, \quad j = 1, \ldots, N, \end{aligned}$$

dove $\mathbf{z}$ è una variabile di consenso che garantisce coerenza tra le soluzioni locali. I passi iterativi di ADMM risultano quindi:

1) Aggiornamento di $\mathbf{x}_j$ (SVM locale)

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \left\{ \sum_{j=1}^{n_i} \max(\mathbf{0},\ \mathbf{1} - \mathbf{a}_{i_j}^\top \mathbf{x}_i) + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}^k + \mathbf{u}^k\|_2^2 \right\}.$$

Poiché la funzione hinge non è differenziabile, non esiste una forma chiusa di soluzione; tuttavia il problema può essere risolto tramite un solver convesso (es. `CVX`) come una **SVM locale** sui dati del i-esimo agente.

2) Aggiornamento della variabile di consenso

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \left\{ \lambda \|\mathbf{C}\mathbf{z}\|_2^2 + \frac{N\rho}{2} \|\bar{\mathbf{x}}^{k+1} - \mathbf{z} + \mathbf{u}^k\|_2^2 \right\},$$

dove

$$\bar{\mathbf{x}}^{k+1} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{x}_j^{k+1}$$

Questo passo corrisponde al **fusion center** che aggrega le soluzioni locali.

3) Aggiornamento delle variabili duali

$$\mathbf{u}_j^{k+1} = \mathbf{u}_j^k - \mathbf{z}^{k+1} + \mathbf{x}_j^{k+1}.$$

In sintesi:

- Ogni agente risolve una **SVM locale** sui propri dati;
- Il *fusion center* calcola una media pesata;
- Le variabili duali vengono aggiornate per ridurre il disaccordo tra le soluzioni.

Questo schema realizza un approccio **split-by-examples**, in cui la decomposizione avviene sui campioni, mentre la coerenza globale è garantita dalla variabile di consenso $\mathbf{z}$.

2.2.6 ADMM per SVM distribuita secondo le feature (Split-by-Features)

Nel caso **split-by-features**, la distribuzione riguarda le variabili del problema, ossia le feature di input, e non i campioni. Sia $\mathbf{x} \in \mathbb{R}^n$ il vettore dei pesi complessivo, suddiviso in N blocchi:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(N)} \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2)} & \cdots & \mathbf{A}^{(N)} \end{bmatrix},$$

tali che

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^{N} \mathbf{A}^{(i)}\mathbf{x}^{(i)}.$$

Introducendo per ogni agente i una variabile locale $\mathbf{z}_i$ e la variabile di consenso media:

$$\bar{\mathbf{z}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{z}_i,$$

si ottiene la formulazione distribuita:

$$\begin{aligned} &\min_{\{\mathbf{x}_i\},\bar{\mathbf{z}}} \{\lambda \sum_{i=1}^{N} \|\mathbf{x}_i\|_2^2 + \mathbf{1}_n^\top \max\big(\mathbf{0}_n,\ \mathbf{1}_n - N\bar{\mathbf{z}}\big)\} \\ &\text{s.t. } \mathbf{A}^{(i)}\mathbf{x}^{(i)} - \mathbf{z}_i = \mathbf{0}. \end{aligned}$$

si ottengono i seguenti aggiornamenti iterativi di ADMM:

1) Aggiornamento di $\mathbf{x}^{(i)}$ (ridge locale)

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} \{\lambda\|\mathbf{x}_i\|_2^2 + \frac{\rho}{2}\|\mathbf{A}^{(i)}\mathbf{x}_i - \mathbf{s}^k + \mathbf{u}^k\|_2^2\},$$

dove

$$\mathbf{s}^k = \mathbf{A}^{(i)}\mathbf{x}^k + \bar{\mathbf{z}}^k - \overline{\mathbf{A}\mathbf{x}}^k\|_2^2\},$$

che rappresenta un classico problema di **ridge regression** risolvibile in forma chiusa:

$$\mathbf{x}_i^{k+1} = \left(\mathbf{A}^{\top(i)}\mathbf{A}^{(i)} + \tfrac{2\rho}{\lambda}\mathbf{I}\right)^{-1}\mathbf{A}^{\top(i)}\big(\mathbf{s}^k - \mathbf{u}^k\big).$$

2) Aggiornamento di $\bar{\mathbf{z}}$

$$\bar{\mathbf{z}}^{k+1} = \arg\min_{\bar{\mathbf{z}}} \Big(\mathbf{1}_n^\top \max(\mathbf{0}_n,\ \mathbf{1}_n - N\bar{\mathbf{z}}) + \frac{\rho N}{2} \|\bar{\mathbf{z}} - \overline{\mathbf{Ax}}^{k+1} - \mathbf{u}^k\|_2^2 \Big),$$

che risulta separabile per componente $l = 1, \ldots, n$. La soluzione è data da un operatore di **biased thresholding operator**:

$$\bar{z}_l^{k+1} = \begin{cases} c_l, & (1 - \mathbf{N}\mathbf{c}_l < 0), \quad \mathbf{c}_l > \frac{1}{N} \\ c_l + \frac{1}{\rho}, & (1 - \mathbf{N}\mathbf{c}_l - \frac{N}{\rho} > 0), \quad \mathbf{c}_l < \frac{1}{N} - \frac{1}{\rho} \\ \frac{1}{N}, & \mathbf{c}_l \in [\frac{1}{N} - \frac{1}{\rho}, \frac{1}{N}\,] \end{cases}$$

3) Aggiornamento della variabile duale

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \overline{\mathbf{Ax}}^{k+1} - N\,\bar{\mathbf{z}}^{k+1}.$$

In sintesi:

- **Step** x_i: risoluzione indipendente di N problemi di ridge regression locale;
- **Step** $\bar{z}$: aggiornamento *component-wise* tramite biased thresholding operator;
- **Step** u: aggiornamento duale di consenso globale.

2.2.7 Utilizzo della PCA come estensione dello schema Split-by-Features

In aggiunta alle tre formulazioni principali di Support Vector Machine, è stata introdotta una fase sperimentale basata sull'utilizzo **Principal Component Analysis (PCA)**. L'obiettivo è valutare come la riduzione della dimensionalità influenzi la velocità di convergenza e le prestazioni di SVM distribuito secondo le feature (Split-by-Features), mantenendo un livello di accuratezza comparabile alla versione completa.

Principal Component Analysis (PCA)

PCA è una tecnica di riduzione dimensionale che trasforma i dati in un nuovo sistema di coordinate ottenuto massimizzando la varianza proiettata. Dato un dataset, sotto l'ipotesi che i dati siano normalizzati $\mathbf{X} \in \mathbb{R}^{m \times d}$, la matrice di covarianza è:

$$\mathbf{C} = \frac{1}{m} \mathbf{X}^\top \mathbf{X},$$

e le componenti principali si ottengono risolvendo:

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \qquad i = 1, \ldots, d.$$

La proiezione dei dati nello spazio ridotto di dimensione k è:

$$\mathbf{X}_{(k)} = \mathbf{X}\mathbf{V}_{1:k},$$

dove $\mathbf{V}_{1:k}$ contiene gli autovettori associati ai k autovalori maggiori.

Uso di PCA nel contesto ADMM

Nella versione split-by-features, ogni nodo risolve in ogni iterazione un problema locale che richiede l'inversione di una matrice delle dimensioni del numero di feature assegnate. Diminuendo il numero totale di feature, tali inversioni diventano più piccole e computazionalmente meno onerose. PCA permette di ridurre la dimensionalità mantenendo la maggior parte dell'informazione utile, offrendo quindi un buon compromesso tra efficienza e accuratezza.

Modalità di utilizzo nel progetto

PCA è stata applicata una sola volta all'intero dataset prima di suddividere le feature tra i nodi ADMM. Sono stati considerati valori di k compresi tra 1 e 13 (il numero di feature che caratterizzano il problema trattato), ottenendo matrici ridotte $\mathbf{X}_{(k)}$ da cui sono stati ricavati i corrispondenti set di train e test. Per ciascun valore di k è stata eseguita l'ottimizzazione ADMM split-by-features, misurando:

- accuratezza sul test set,
- tempo di esecuzione totale,
- numero di iterazioni necessarie alla convergenza.

Risultati sperimentali con PCA

La varianza spiegata cumulativa mostra che:

- le prime 5 componenti spiegano circa il 62% della varianza,
- con 8 componenti si supera l'85%,
- con 12 componenti si arriva al 98%.

L'accuratezza aumenta monotonicamente con il numero di componenti:

- 1 componente: 77.81%,
- 3 componenti: 87.31%,
- 9 componenti: 88.56%,
- 10–12 componenti: tra 89.01% e 89.24%, equivalenti alla versione completa (89.09%).

I tempi di esecuzione risultano molto più contenuti rispetto alla versione senza PCA: per $k \leq 5$ si ottengono tempi tra 0.03 e 0.05 secondi, contro tempi dell'ordine del secondo nella versione completa. Il numero di iterazioni resta stabile a circa 30, evidenziando che la riduzione di tempo è dovuta principalmente a una minore complessità dei problemi locali.

2.3 Metodologia e Implementazione

Il codice è organizzato in blocchi funzionali indipendenti: caricamento e preprocessing dei dati, definizione dei parametri, addestramento SVM centralizzato, imple-

mentazione dei due schemi ADMM distribuiti, sezione PCA e le relative ottimizzazioni per ogni k, salvataggio degli storici necessari alla generazione dei grafici e infine produzione dei plot di confronto.

2.3.1 Struttura del Codice MATLAB

La struttura complessiva del codice può essere riassunta come segue:

1. **Data Preparation**: lettura del dataset, normalizzazione delle feature, dummizzazione delle variabili categoriche, definizione della partizione train/test mediante.
2. **Definizione Parametri**: impostazione dei parametri SVM e dei parametri ADMM, in particolare il numero di nodi N, il parametro di penalità ρ, il numero massimo di iterazioni e le tolleranze di convergenza.
3. **SVM Centralizzata**: risoluzione della formulazione primale del problema tramite CVX, ottenendo il modello di riferimento rispetto a cui confrontare le metodologie distribuite.
4. **ADMM Split-by-Examples**: suddivisione del dataset per gruppi di campioni, risoluzione locale tramite CVX, aggiornamento della variabile di consenso z e delle variabili duali secondo lo schema ADMM.
5. **ADMM Split-by-Features**: suddivisione del vettore dei pesi tra i nodi, aggiornamento locale tramite problemi di ridge regression risolvibili in forma chiusa, aggiornamento del consenso nello spazio degli score e aggiornamento delle variabili duali.
6. **PCA**: applicazione della PCA al dataset e valutazione del comportamento di ADMM per diversi numeri di componenti.
7. **Generazione dei Plot**: creazione di grafici relativi all'evoluzione dell'accuracy per iterazione, al confronto delle accuratezze finali e agli effetti di PCA.

2.4 Risultati ottenuti

I tre metodi producono accuratezze praticamente identiche:

- **Centralized SVM**: **88.94%**
- **ADMM Split-by-Examples**: **89.04%**
- **ADMM Split-by-Features**: **88.93%**

La differenza massima è inferiore allo 0.1%, confermando che entrambe le formulazioni distribuite convergono alla stessa soluzione del problema centralizzato.

Evoluzione del disagreement

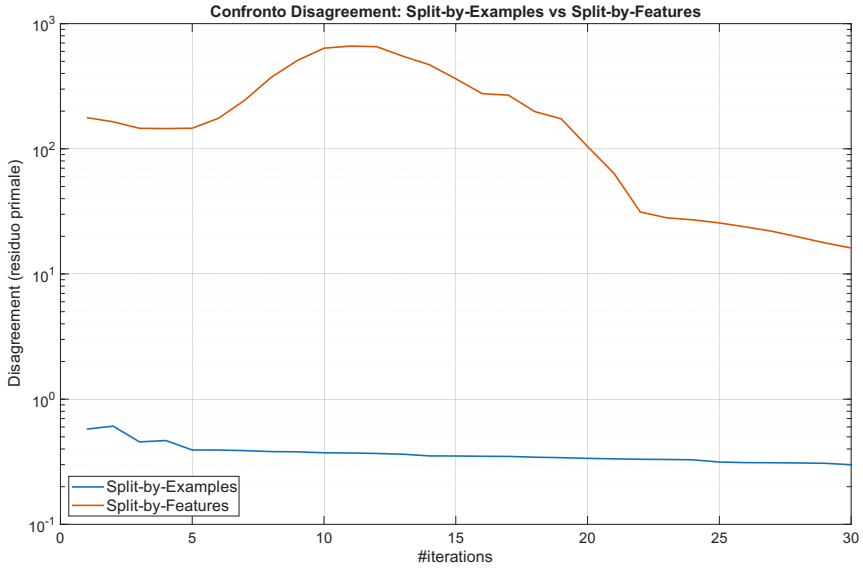

Figura 2.1: Disagreement in realzione al numero di iterazioni

Nel caso Split-by-Examples il disagreement è quasi costante: i modelli locali w_j risultano molto simili già dalla prima iterazione mentre nel caso Split-by-Features si osserva un picco iniziale seguito da una rapida convergenza.

Accuracy di classificazione per iterazione

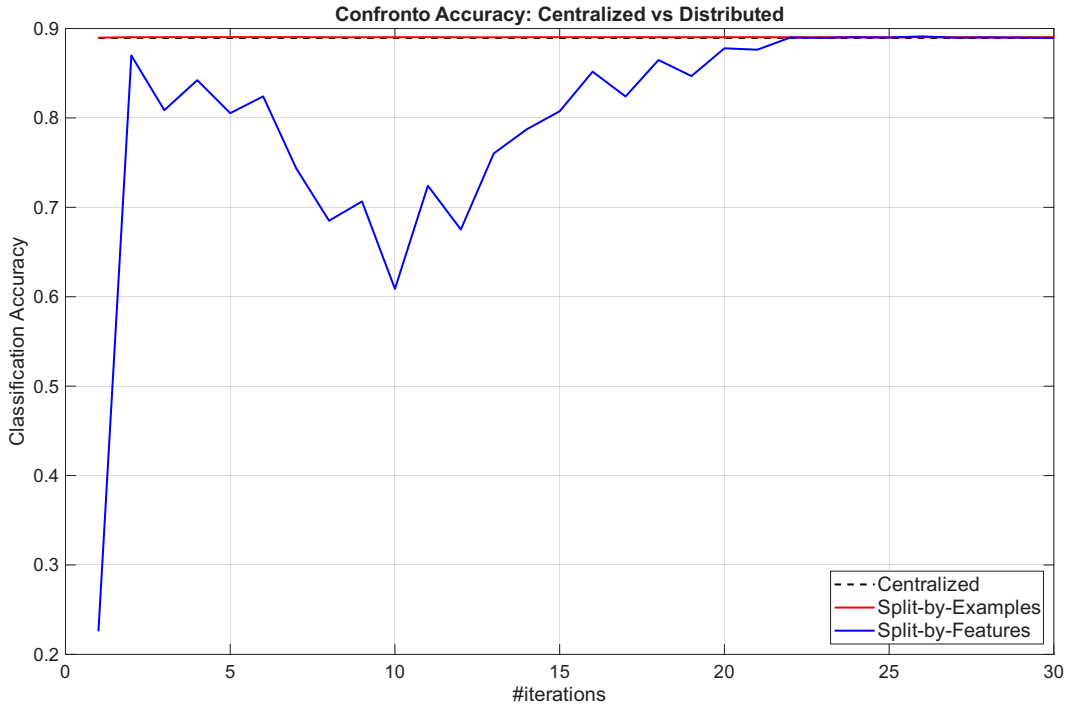

Figura 2.2: Accuracy in realzione al numero di iterazioni

Il metodo Split-by-Examples converge quasi immediatamente sin dalla prima iterazione mentre il metodo Split-by-Features mostra una crescita graduale ma stabile fino ad allinearsi perfettamente con la versione centralizzata.

Effetto di PCA

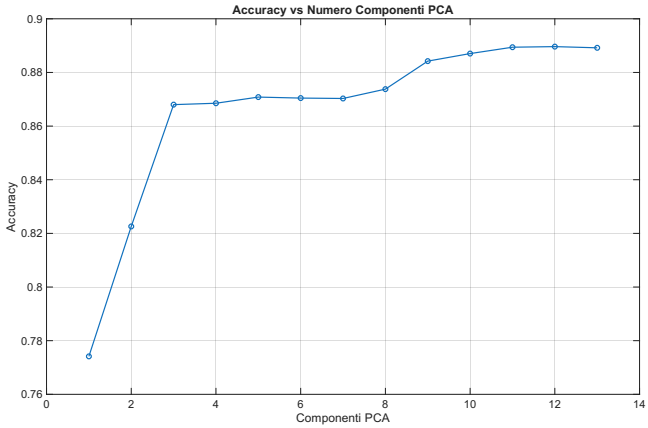

Figura 2.3: Accuracy in realzione al numero componenti PCA mantenute

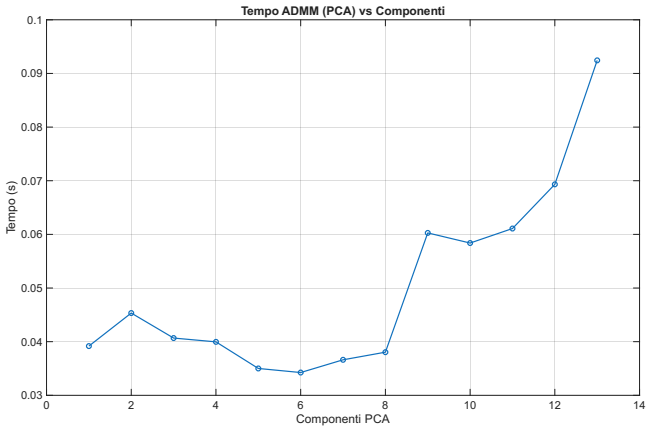

Figura 2.4: Tempi di addestramento in relazione al numero di componenti PCA mantenute

PCA permette di valutare l'impatto della riduzione dimensionale sulle performance dello schema Split-by-Features. I risultati mostrano che:

- 1 componente: $\sim$78% di accuratezza,
- 3 componenti: $\sim$87%,
- 9–12 componenti: $\sim$89%, equivalenti al modello completo.

I tempi di esecuzione rimangono molto ridotti per tutti i valori di $k \leq 10$, con un incremento graduale per dimensioni maggiori.

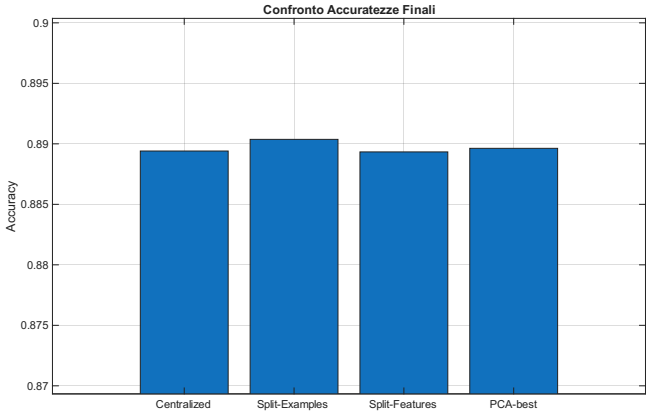

Figura 2.5: Accuracy ottenute a confronto

2.5 Conclusioni

Tutte le versioni dell'algoritmo mostrano risultati sovrapponibili, incluso il modello PCA con 11 componenti che presenta il miglior trade-off tra accuracy e tempo di addestramento. Dai risultati sperimentali emergono alcune conclusioni fondamentali:

- **Corretta convergenza**: entrambe le versioni ADMM raggiungono la soluzione centralizzata, come previsto dalla teoria del consenso.
- **Efficienza computazionale**: lo schema Split-by-Features è estremamente più rapido, grazie alla possibilità di aggiornamenti in forma chiusa e alla mancanza di problemi CVX locali.
- **Stabilità**: il metodo Split-by-Examples soffre maggiormente la complessità numerica e mostra residui più irregolari.
- **PCA come acceleratore**: con 11 componenti si mantiene la stessa accuratezza del modello completo, riducendo la dimensionalità e riducendo i tempi di calcolo.

Il progetto ha dimostrato che ADMM rappresenta un approccio efficace per la risoluzione distribuita di Support Vector Machine. In particolare:

- entrambe le varianti distribuite convergono alla stessa soluzione della versione centralizzata;
- la formulazione Split-by-Features offre il miglior compromesso tra accuratezza e velocità, risultando di due ordini di grandezza più rapida rispetto allo Split-by-Examples;
- l'aggiunta della PCA permette di ridurre ulteriormente la complessità mantenendo invariata l'accuratezza finale;
- i grafici dei residui confermano la corretta convergenza numerica dello schema ADMM.

Nel complesso, si dimostra come l'uso combinato di ADMM, decomposizione per

feature e PCA possa costituire un approccio estremamente scalabile ed efficiente per la classificazione su dataset di grandi dimensioni.