



A.D. 1308
unipg

DIPARTIMENTO
DI INGEGNERIA

Progetto di
Signal Processing and Optimization for Big Data
Corso di Laurea in Ingegneria Informatica e Robotica
Curriculum Data Science
DIPARTIMENTO DI INGEGNERIA

docente
Prof. Paolo BANELLI

Studio Comparativo degli Algoritmi di Regressione Lasso in MATLAB

363433 **Gian Marco Ferri** gianmarco.ferri@studenti.unipg.it

Indice

1	Introduzione	2
1.1	Contesto Generale	2
2	Analisi Teorica	3
2.1	LASSO	3
2.2	ISTA	3
2.3	ADMM	5
2.4	ADMM Distribuito	6
3	Implementazione	9
3.1	Dataset	9
3.2	Preprocessing dei dati	10
3.3	Classe LassoReg	10
3.4	Script principale	10
3.5	Modularità ed estensibilità	10
4	Risultati e Confronto tra Algoritmi	11
4.1	Configurazione parametri e Tabella comparativa	11
4.2	Grafici delle Performance	12
4.3	Discussione dei Risultati	13
4.4	Conclusioni	14

1 Introduzione

Questo progetto affronta la risoluzione del problema della regressione con regolarizzazione Lasso tramite l'implementazione e il confronto di diversi algoritmi. In particolare, sono stati sviluppati in Matlab tre approcci distinti: ISTA, ADMM e una versione simulata di ADMM distribuito su più agenti.

Nella parte finale della documentazione vengono presentati dei confronti tra i tre algoritmi in termini di tempi di calcolo, numero di iterazioni e grafici che illustrano l'andamento della convergenza durante le iterazioni.

1.1 Contesto Generale

La regressione LASSO (Least Absolute Shrinkage and Selection Operator) è una tecnica fondamentale nel campo della modellistica statistica e del machine learning.

Questa metodologia permette di affrontare problemi come l'overfitting e la selezione delle variabili, offrendo una strategia efficace per sviluppare modelli affidabili e generalizzabili.

Il metodo di regolarizzazione LASSO affronta il problema dell'overfitting, ossia la tendenza di un modello a modellare troppo fedelmente i dati di addestramento includendo anche il rumore, introducendo una penalizzazione sulla somma dei valori assoluti dei coefficienti delle variabili nel modello. L'intensità di questa penalità è regolata da un iperparametro.

Questa penalità, di tipo L_1 , induce sparsità nel vettore dei parametri, portando molti di essi ad annullarsi. Questo meccanismo consente non solo di selezionare automaticamente le variabili più rilevanti, ma anche di ottenere un modello più semplice, riducendo efficacemente il fenomeno dell'overfitting.

Proprio per questa sua capacità di produrre modelli sparsi, il LASSO si differenzia da altre tecniche di regolarizzazione come la regressione Ridge, che utilizza una penalità L_2 , risultando particolarmente vantaggioso nell'analisi di dataset con un numero elevato di feature.

2 Analisi Teorica

Questo capitolo presenta la formalizzazione matematica del problema LASSO e degli algoritmi per risolverlo.

Viene descritto il problema di ottimizzazione alla base del metodo, evidenziando le proprietà della regolarizzazione $L1$ e la sua non-differenziabilità.

Vengono quindi analizzate le basi matematiche dei tre algoritmi implementati, ISTA, ADMM e ADMM distribuito, esaminandone i meccanismi di funzionamento e convergenza, in preparazione dell'implementazione pratica che seguirà nei prossimi capitoli.

2.1 LASSO

Il problema di regressione LASSO è formalizzato come il seguente problema di ottimizzazione convessa:

$$\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \alpha \|\mathbf{x}\|_1$$

dove:

- $\mathbf{x} \in \mathbb{R}^n$ indica il vettore dei coefficienti del modello,
- $\mathbf{y} \in \mathbb{R}^m$ rappresenta il vettore delle osservazioni,
- $A \in \mathbb{R}^{m \times n}$ corrisponde alla matrice delle variabili indipendenti (features),
- $\alpha > 0$ è l'iperparametro che controlla l'intensità della regolarizzazione.

La funzione obiettivo del LASSO combina due componenti fondamentali: un termine di fitting dei dati, che misura l'errore quadratico tra le previsioni del modello e i valori osservati, e un termine di regolarizzazione $L1$, che penalizza la somma dei valori assoluti dei coefficienti.

2.2 ISTA

Nell'algoritmo ISTA (Iterative Shrinkage Thresholding Algorithm), la regressione LASSO viene risolta utilizzando un approccio di Proximal Gradient, particolarmente adatto per funzioni obiettivo che combinano una componente derivabile con una non derivabile, in questo caso la penalità $L1$.

Questo metodo estende il classico Gradient Descent affrontando il problema di ottimizzazione attraverso aggiornamenti iterativi che separano il trattamento della componente derivabile (applicando il gradiente) da quella non derivabile (applicando l'operatore di proximal), fino al raggiungimento della convergenza.

L'aggiornamento iterativo del Proximal Gradient per il problema LASSO assume quindi la seguente forma generale:

$$\mathbf{x}^{(k+1)} = \text{Prox}_{\gamma\alpha\|\cdot\|_1} \left(\mathbf{x}^{(k)} - \gamma \nabla \left[\frac{1}{2} \|\mathbf{y} - A\mathbf{x}^{(k)}\|_2^2 \right] \right)$$

dove

$$\text{Prox}_{\gamma\alpha\|\cdot\|_1}(\mathbf{x}) = \underset{\mathbf{u}}{\operatorname{argmin}} \left\{ \gamma\alpha\|\mathbf{u}\|_1 + \frac{1}{2}\|\mathbf{u} - \mathbf{x}\|_2^2 \right\}$$

Poiché la norma L_1 non è differenziabile, si ricorre al concetto di sub-gradiente:

$$[\text{sub}\nabla f(x_0)]_i = \begin{cases} \frac{\partial f(x_0)}{\partial x_i}, & \text{se differenziabile in } x_0 \\ \left[\frac{\partial f(x_0^-)}{\partial x_i}, \frac{\partial f(x_0^+)}{\partial x_i} \right], & \text{se non differenziabile in } x_0 \end{cases}$$

Applicando questa definizione al caso in esame, si ottiene:

$$\text{Prox}_{\gamma\alpha\|\cdot\|_1}(\mathbf{x}) = \underset{\mathbf{u}}{\operatorname{argmin}} \underbrace{\left\{ \gamma\alpha \sum_{i=1}^n |u_i| + \frac{1}{2} \sum_{i=1}^n (u_i - x_i)^2 \right\}}_{\Phi(u_1, \dots, u_n)}$$

Per risolvere questo problema di minimizzazione, si applicano le condizioni di sub-gradiente alla funzione Φ :

$$\text{sub}\nabla\Phi = \begin{cases} \gamma\alpha + (u_j - x_j) & \text{se } u_j > 0 \\ -\gamma\alpha + (u_j - x_j) & \text{se } u_j < 0 \\ [-\gamma\alpha + (u_j - x_j), \gamma\alpha + (u_j - x_j)] & \text{se } u_j = 0 \end{cases}$$

Imponendo l'annullamento del sub-gradiente, si ottiene la soluzione in forma chiusa:

$$\Rightarrow \begin{cases} u_j = x_j - \gamma\alpha & \text{se } x_j > \gamma\alpha \\ u_j = x_j + \gamma\alpha & \text{se } x_j < -\gamma\alpha \\ u_j = 0 & \text{se } -\gamma\alpha < x_j < \gamma\alpha \end{cases}$$

Questa operazione è nota come Soft-Thresholding e viene indicata con $S_{\gamma\alpha}(\cdot)$. Essa fornisce una soluzione analitica per l'aggiornamento dei coefficienti nel caso di penalità L_1 , dove il gradiente classico non è applicabile.

Quindi, la soluzione al problema LASSO è la seguente:

$$\mathbf{x}^{(k+1)} = S_{\gamma\alpha} \left(\mathbf{x}^{(k)} - \gamma \nabla \left[\frac{1}{2} \|\mathbf{y} - A\mathbf{x}^{(k)}\|_2^2 \right] \right)$$

Espandendo il termine del gradiente, si ottiene la forma equivalente:

$$\mathbf{x}^{(k+1)} = S_{\gamma\alpha} \left((I - \gamma A^T A) \mathbf{x}^{(k)} + \gamma A^T \mathbf{y} \right)$$

Il processo iterativo prosegue fino a quando la norma della differenza tra i coefficienti di due iterazioni successive diventa inferiore a una soglia di tolleranza prestabilita, indicando il raggiungimento della convergenza.

2.3 ADMM

L'Alternating Direction Method of Multipliers (ADMM) è un algoritmo di ottimizzazione primal-dual che risolve problemi complessi scomponendoli in una serie di sottoproblemi più semplici e gestibili, che vengono risolti in modo iterativo attraverso l'aggiornamento alternato delle variabili primali e duali.

Per applicare l'ADMM al problema LASSO, è necessario riformularlo introducendo una variabile ausiliaria (o *slack variable*) \mathbf{z} . Questa scomposizione permette di separare l'ottimizzazione della funzione di loss originale dalla penalità $L1$, facilitando la risoluzione attraverso sottoproblemi indipendenti.

La formulazione risultante è la seguente:

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}} & \frac{1}{2} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \alpha \|\mathbf{z}\|_1 \\ \text{s.t. } & \mathbf{x} - \mathbf{z} = 0 \end{aligned}$$

Il framework ADMM scomponete il problema di ottimizzazione in tre distinti sottoproblemi, le cui soluzioni vengono calcolate in modo alternato e iterativo fino al soddisfacimento di un criterio di convergenza.

Di seguito viene presentata la formulazione standard in forma scalata dell'algoritmo:

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \operatorname{argmin}_{\mathbf{x}} \left\{ \frac{1}{2} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{u}^{(k)}\|_2^2 \right\} \\ \mathbf{z}^{(k+1)} &= \operatorname{argmin}_{\mathbf{z}} \left\{ \alpha \|\mathbf{z}\|_1 + \frac{\rho}{2} \|\mathbf{x}^{(k+1)} - \mathbf{z} + \mathbf{u}^{(k)}\|_2^2 \right\} \\ \mathbf{u}^{(k+1)} &= \mathbf{u}^{(0)} + \sum_{i=1}^{k+1} \|\mathbf{x}^{(i)} - \mathbf{z}^{(i)}\|_2^2 \end{aligned}$$

Il primo step ammette una soluzione in forma chiusa, ottenuta ponendo a zero la derivata della funzione obiettivo:

$$\nabla_{\mathbf{x}} \left(\frac{1}{2} \|A\mathbf{x} - \mathbf{y}\|_2^2 + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{u}^{(k)}\|_2^2 \right) = 0$$

$$\begin{aligned}
A^T(A\mathbf{x} - \mathbf{y}) + \rho(\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{u}^{(k)}) &= 0 \\
(A^T A + \rho I)\mathbf{x} &= A^T \mathbf{y} + \rho(\mathbf{z}^{(k)} - \mathbf{u}^{(k)}) \\
\rightarrow \mathbf{x}^{(k+1)} &= (A^T A + \rho I)^{-1}(A^T \mathbf{y} + \rho(\mathbf{z}^{(k)} - \mathbf{u}^{(k)}))
\end{aligned}$$

Per il secondo sottoproblema, la funzione obiettivo non è differenziabile a causa della norma $L1$. La sua soluzione, tuttavia, può essere ottenuta in forma chiusa imponendo le condizioni di sub-gradiente, le quali, per la norma $L1$, si traducono nell'applicazione dell'operatore di soft-thresholding.

In conclusione, la formulazione del problema LASSO mediante ADMM in forma scalata è la seguente:

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= (A^T A + \rho I)^{-1}(A^T \mathbf{y} + \rho(\mathbf{z}^{(k)} - \mathbf{u}^{(k)})) \\
\mathbf{z}^{(k+1)} &= S_{\frac{\alpha}{\rho}}(\mathbf{x}^{(k+1)} + \mathbf{u}^{(k)}) \\
\mathbf{u}^{(k+1)} &= \mathbf{u}^{(k)} + \mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)}
\end{aligned}$$

I parametri ρ e α rivestono un ruolo critico nell'algoritmo, poiché la loro scelta influenza fondamentalmente il bilanciamento tra velocità di convergenza e robustezza della soluzione.

Nonostante questa sensibilità ai parametri, l'utilizzo dell'ADMM per il problema LASSO offre notevoli vantaggi.

Il suo vantaggio principale deriva dalla scomposizione del problema originale in sottoproblemi più semplici e gestibili, spesso risolvibili in forma chiusa. Questa architettura garantisce all'algoritmo una notevole stabilità e consente di raggiungere la soluzione ottimale in un numero di iterazioni generalmente contenuto.

2.4 ADMM Distribuito

La versione distribuita dell'ADMM estende il framework classico per gestire problemi su larga scala distribuendo il carico computazionale tra diversi nodi. Quando la funzione obiettivo è additiva rispetto ai dati, è possibile scomporla in sottoproblemi indipendenti, ciascuno gestito da un agente. Tuttavia, per garantire una soluzione comune, si ricorre alla *Consensus Optimization*: ogni agente mantiene una copia locale \mathbf{x}_i della variabile globale, con vincoli che impongono il consenso verso un valore comune.

La formulazione standard del problema di ottimizzazione vincolata del consenso, in cui N agenti devono minimizzare una funzione obiettivo globale scomponibile, è la seguente:

$$\min_{(x_1, \dots, x_N), \mathbf{z}} \sum_{i=1}^N f_i(\mathbf{x}) + g(\mathbf{z})$$

$$\text{s.t. } \mathbf{x}_i - \mathbf{z} = \mathbf{0}, \quad i = 1, \dots, N$$

Il problema Lasso è intrinsecamente distribuibile grazie alla natura separabile della sua funzione obiettivo. Infatti, sia le osservazioni \mathbf{y} che la matrice A possono essere partizionate in N blocchi, rendendo possibile la scomposizione del problema originale in sottoproblemi indipendenti.

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}, \quad A\mathbf{x} = \begin{bmatrix} A_1 \\ \vdots \\ A_N \end{bmatrix} \mathbf{x} = \begin{bmatrix} A_1\mathbf{x} \\ \vdots \\ A_N\mathbf{x} \end{bmatrix}$$

Andando a riformulare il problema nella forma di un'ottimizzazione al consenso, si ottiene la seguente rappresentazione:

$$\min_{(\mathbf{x}_1, \dots, \mathbf{x}_N), \mathbf{z}} \sum_{i=1}^N \|\mathbf{y}_i - A_i \mathbf{x}_i\|_2^2 + \lambda \|\mathbf{z}\|_1$$

$$\text{s.t. } \mathbf{x}_i - \mathbf{z} = \mathbf{0}, \quad i = 1, \dots, N$$

La soluzione viene ottenuta applicando la versione scalata dell'ADMM, la cui struttura è la seguente:

$$\mathbf{x}_i^{(k+1)} = \operatorname{argmin}_{\mathbf{x}_i} \left\{ \|\mathbf{y}_i - A_i \mathbf{x}_i\|_2^2 + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}^{(k)} + \mathbf{u}_i^{(k)}\|_2^2 \right\}$$

$$\mathbf{z}^{(k+1)} = \operatorname{argmin}_{\mathbf{z}} \left\{ g(\mathbf{z}) + \frac{\rho}{2} \sum_{i=1}^N \|\mathbf{x}_i^{(k+1)} - \mathbf{z} + \mathbf{u}_i^{(k)}\|_2^2 \right\} \quad i = 1, \dots, N$$

$$\mathbf{u}_i^{(k+1)} = \mathbf{u}_i^{(k)} + (\mathbf{x}_i^{(k+1)} - \mathbf{z}^{(k+1)}) \quad i = 1, \dots, N$$

Il primo step si compone di N sottoproblemi indipendenti, ciascuno assegnabile a un agente distinto. Come nella formulazione centralizzata, questi ammettono una soluzione in forma chiusa e possono essere quindi calcolati in parallelo.

Il secondo step richiede la raccolta di tutte le variabili primali e duali locali per il calcolo di un valore di consenso globale. Questa operazione, non distribuibile, viene affidata a un fusion center. In assenza di un fusion center, qualora i nodi fossero collegati in una rete connessa, sarebbe possibile calcolare il consenso globale attraverso un altro algoritmo al consenso, che richiederebbe solamente una comunicazione tra nodi connessi.

Il terzo step può essere eseguito in modo completamente autonomo e parallelo da ciascun agente sulla base dei nuovi valori di \mathbf{x}_i e \mathbf{z} .

Di conseguenza, le iterazioni dell'algoritmo per ogni agente i e per il fusion center (o, in alternativa, per la rete nel caso di implementazione completamente distribuita) sono definite dalle seguenti equazioni:

$$\begin{aligned}\mathbf{x}_i^{(k+1)} &= (A_i^T A_i + \frac{\rho}{2} I)^{-1} (A_i^T \mathbf{y}_i + \frac{\rho}{2} (\mathbf{z}^{(k)} - \mathbf{u}_i^{(k)})) \\ \mathbf{z}^{(k+1)} &= \text{Prox}_{\frac{\lambda}{N\rho} \|\cdot\|_1} \left\{ \hat{\mathbf{x}}^{(k+1)} + \hat{\mathbf{u}}^{(k)} \right\} = S_{\frac{\lambda}{N\rho}} (\hat{\mathbf{x}}^{(k+1)} + \hat{\mathbf{u}}^{(k)}) \\ \mathbf{u}_i^{(k+1)} &= \mathbf{u}_i^{(k)} + \mathbf{x}_i^{(k+1)} - \mathbf{z}^{(k+1)} \quad i = 1, \dots, N\end{aligned}$$

dove $\hat{\mathbf{x}}^{(k+1)}$ e $\hat{\mathbf{u}}^{(k)}$ sono le medie delle variabili primarie e duali al passo $k+1$ e k rispettivamente.

Questa tecnica è particolarmente adatta per l'elaborazione di dataset di grandi dimensioni e per scenari in cui i dati sono intrinsecamente distribuiti su più nodi computazionali.

3 Implementazione

L'implementazione del confronto tra diversi algoritmi di regressione LASSO è stata realizzata interamente in MATLAB, sfruttando un approccio modulare e facilmente estendibile. Il progetto si compone di diversi script e classi che gestiscono la preparazione dei dati, la definizione e l'addestramento dei modelli, nonché la valutazione delle relative performance.

3.1 Dataset

Il dataset utilizzato in questo studio è il California Housing Prices Dataset, che contiene informazioni statistiche aggregate sul mercato immobiliare californiano, dove ogni osservazione rappresenta un blocco residenziale.

Le feature disponibili includono sia variabili demografiche che proprietà fisiche delle abitazioni, in particolare:

- **longitude**: Longitudine della posizione del blocco residenziale.
- **latitude**: Latitudine della posizione del blocco residenziale.
- **housing_median_age**: Età mediana delle abitazioni nel blocco residenziale.
- **total_rooms**: Numero totale di stanze in tutte le abitazioni del blocco residenziale.
- **total_bedrooms**: Numero totale di camere da letto in tutte le abitazioni del blocco residenziale.
- **population**: Popolazione residente nel blocco.
- **households**: Numero di nuclei familiari (households) nel blocco.
- **median_income**: Reddito mediano delle famiglie nel blocco (in decine di migliaia di dollari).
- **median_house_value**: Valore mediano delle abitazioni nel blocco (in dollari USA).
- **ocean_proximity**: Prossimità all'oceano, valore categorico (ad es. NEAR BAY, INLAND, < 1H OCEAN, ecc.).

3.2 Preprocessing dei dati

La fase iniziale prevede la pulizia e la normalizzazione del dataset California Housing Prices attraverso lo script `preprocess_california_housing.m`.

Lo script carica il dataset originale, gestisce i valori mancanti nella colonna `total_bedrooms` sostituendoli con la mediana, seleziona le feature principali e la variabile target `median_house_value`, ed esclude la variabile categorica `ocean_proximity` per focalizzare l'analisi sulle feature numeriche. Le feature vengono normalizzate nell'intervallo $[0, 1]$ per garantire una migliore convergenza degli algoritmi, e il dataset preprocessato viene salvato in `california_housing_processed.csv`.

3.3 Classe LassoReg

Gli algoritmi descritti in precedenza sono stati implementati all'interno della classe `LassoReg.m`, presente nel file `src/LassoReg.m`.

In fase di istanziazione, è possibile specificare diversi parametri, tra cui: l'algoritmo di ottimizzazione da utilizzare per la fase di training, il valore dello step-size, la tolleranza per il criterio di convergenza, il numero massimo di iterazioni e il coefficiente di penalizzazione $L1$.

La classe fornisce metodi per l'addestramento del modello, la predizione su nuovi dati, il calcolo della funzione di loss e la visualizzazione degli andamenti di convergenza.

3.4 Script principale

Il file `main.m` coordina le operazioni principali per l'analisi e il confronto degli algoritmi LASSO: carica il dataset preprocessato, suddivide i dati in training e test set (80%-20%), istanza addestra i tre modelli LASSO secondo i diversi approcci, valuta le performance tramite la metrica R^2 e visualizza graficamente sia le predizioni sia l'andamento della funzione obiettivo e dei residui, consentendo un confronto diretto tra efficienza, velocità e accuratezza degli algoritmi.

3.5 Modularità ed estensibilità

La struttura del codice è stata pensata per favorire la modularità: è possibile aggiungere nuovi algoritmi di ottimizzazione o modificare quelli esistenti con interventi minimi. L'uso di classi e funzioni dedicate permette di mantenere il codice chiaro e riutilizzabile.

4 Risultati e Confronto tra Algoritmi

4.1 Configurazione parametri e Tabella comparativa

I parametri configurati per il confronto tra i tre algoritmi sono i seguenti:

- Iterazioni massime = 50000;
- Step-size = 0.01;
- Penalità $L1 = 1$;
- Tolleranza = 1e-4;
- Agenti = 8 (ADMM distribuito).

La Tabella 1 riassume il confronto delle prestazioni ottenute.

	R2	time (s)	iterazioni
ISTA	0.5339	4.3333	50000
ADMM	0.5793	0.0004	4
ADMM-Dist	0.5794	0.0363	216

Tabella 1: Comparazione algoritmi

È importante sottolineare che i valori riportati sono stati ottenuti effettuando una suddivisione casuale del dataset in training set e test set. Pertanto, esecuzioni successive potrebbero produrre risultati leggermente differenti.

4.2 Grafici delle Performance

A seguire viene illustrato l'andamento dei criteri di convergenza durante le iterazioni degli algoritmi, insieme a dei grafici che confrontano le previsioni con i valori reali.

ISTA

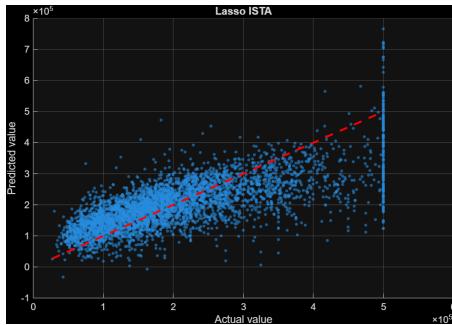


Figura 1: Predizioni ottenute con ISTA.

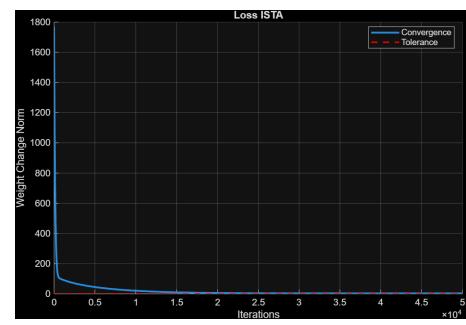


Figura 2: Convergenza di ISTA.

ADMM

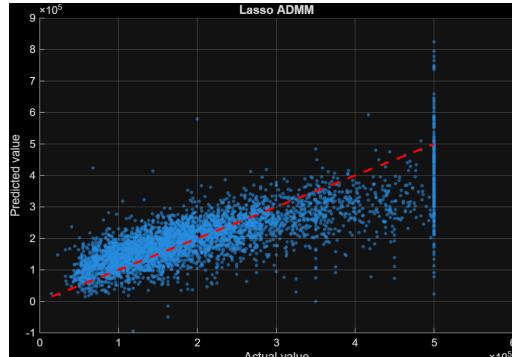


Figura 3: Predizioni ottenute con ADMM.

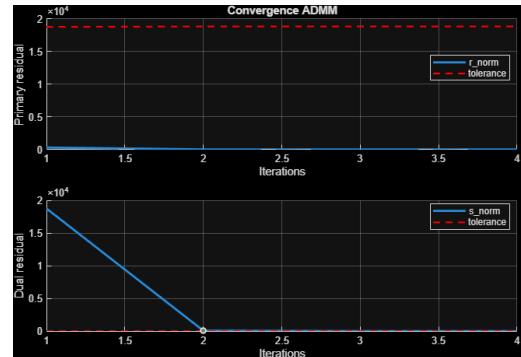


Figura 4: Convergenza di ADMM.

ADMM Distribuito

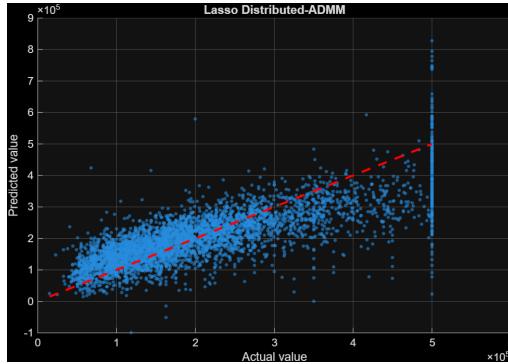


Figura 5: Predizioni ottenute con ADMM distribuito.

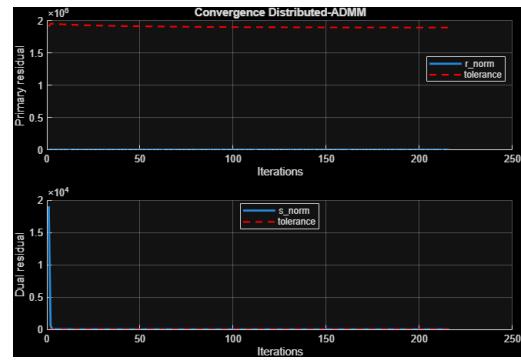


Figura 6: Convergenza di ADMM distribuito.

4.3 Discussione dei Risultati

Precisione (R^2)

- **ADMM** ottiene il valore di R^2 più alto (0.5793), indicando una migliore capacità predittiva rispetto agli altri algoritmi.
- **ADMM Distribuito** raggiunge un R^2 praticamente identico (0.5794), dimostrando di preservare l'accuratezza nonostante la distribuzione del calcolo.
- **ISTA** si attesta su un valore di R^2 di 0.5339, mostrando una performance inferiore agli altri metodi a causa della mancata convergenza entro il limite massimo di iterazioni.

Tempo di esecuzione

- **ADMM** si distingue nettamente per la rapidità: solo 0.0004 secondi per raggiungere la convergenza, grazie all'efficienza della soluzione analitica dei sottoproblemi e alla rapida decrescita dei residui.
- **ADMM Distribuito** impiega più tempo (0.0363s) rispetto all'ADMM classico, principalmente a causa della suddivisione del problema tra più agenti e della necessità di sincronizzazione, ma resta comunque molto più veloce di ISTA.
- **ISTA** è significativamente più lento (4.3333s), dovendo iterare il processo gradualmente a causa della natura dell'ottimizzazione tramite gradient descent e del trattamento della regolarizzazione $L1$.

Numero di iterazioni

- **ADMM** converge in pochissime iterazioni (4), dimostrando efficienza nella risoluzione del problema Lasso.
- **ADMM Distribuito** richiede più iterazioni (216), a causa del processo di coordinamento tra gli agenti, ma rimane estremamente efficiente rispetto a ISTA.
- **ISTA** ha raggiunto il numero massimo di iterazioni impostato (50000) senza riuscire a convergere, evidenziando una convergenza molto più lenta rispetto agli altri algoritmi.

4.4 Conclusioni

L'analisi dei risultati mostra chiaramente che l'algoritmo ADMM è nettamente superiore sia in termini di accuratezza che di efficienza computazionale, risultando ideale quando si dispone di risorse centralizzate e la dimensione del problema lo consente.

Il Distributed ADMM, pur richiedendo più tempo e iterazioni rispetto alla versione centralizzata, raggiunge la medesima accuratezza e rappresenta una valida soluzione in contesti distribuiti o quando i dati sono intrinsecamente suddivisi tra più nodi computazionali.

L'algoritmo ISTA, pur essendo concettualmente semplice da implementare, soffre di una convergenza estremamente lenta e di una minore accuratezza, risultando poco competitivo per problemi di dimensioni medio-grandi.

In sintesi, la scelta dell'algoritmo più adatto dipende dal contesto applicativo e dai vincoli computazionali: ADMM è preferibile per massimizzare le prestazioni in ambienti centralizzati, Distributed ADMM per scenari distribuiti, mentre ISTA può essere considerato per semplicità su dataset piccoli o come baseline di confronto.