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

Confronto tra Algoritmi di Regressione Lasso Sviluppati da Zero

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 Soft-Thresholding 3
2.3 ADMM 4
2.4 ADMM Distribuito 6

3 Implementazione **8**
3.1 Dataset 8
3.2 Preprocessing dei dati 8
3.3 Classe LassoReg 9
3.4 Script principale 9
3.5 Modularità ed estensibilità 9

4 Risultati e Confronto tra Algoritmi **10**
4.1 Configurazione parametri e Tabella comparativa 10
4.2 Grafici delle Performance 11
4.3 Discussione dei Risultati 12
4.4 Conclusioni 13

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.

Con la crescente complessità degli algoritmi, il rischio di overfitting, ossia la tendenza di un modello a modellare troppo fedelmente i dati di addestramento includendo anche il rumore, è diventato sempre più rilevante. Il metodo di regolarizzazione LASSO affronta tale problema introducendo una penalizzazione sulla somma dei valori assoluti dei coefficienti delle variabili nel modello. L'intensità di questa penalità è regolata da un iperparametro.

La caratteristica distintiva della regressione LASSO è proprio l'utilizzo della regolarizzazione di tipo $L1$, che incentiva la sparsità nei coefficienti stimati. In pratica, ciò comporta che LASSO tende ad azzerare molti coefficienti, selezionando automaticamente solo le variabili più significative per la previsione.

Questa sua proprietà la differenzia da altre tecniche come la regressione Ridge, che usa una penalità $L2$, e la rende particolarmente utile quando si ha a che fare con dataset che hanno un numero elevato di feature.

2 Analisi Teorica

In questo capitolo si propone un'analisi dettagliata del problema dal punto di vista sia teorico che matematico, con l'obiettivo di approfondire la comprensione degli algoritmi che verranno implementati in seguito.

Viene presentata una descrizione esaustiva della regressione LASSO, introducendo il concetto della regolarizzazione L1 e discutendo le soluzioni teoriche relative al problema di ottimizzazione collegato.

Si prendono inoltre in considerazione le principali varianti dell'algoritmo, ovvero ISTA, ADMM e ADMM distribuito, illustrandone le basi teoriche necessarie per comprendere appieno le applicazioni pratiche che saranno affrontate nei capitoli successivi.

2.1 LASSO

Il LASSO (Least Absolute Shrinkage and Selection Operator) è un metodo di regressione che applica una penalizzazione di tipo $L1$ alla funzione di ottimizzazione. Tale penalità, proporzionale alla somma dei valori assoluti dei coefficienti, 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.

Tale risultato è ottenuto risolvendo un problema di ottimizzazione che coinvolge la minimizzazione di una funzione di costo, la quale comprende sia il termine di regressione che la penalità $L1$.

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

L'espressione riportata descrive il problema di ottimizzazione relativo alla regressione LASSO. Nell'equazione, $\mathbf{x}$ indica il vettore dei coefficienti del modello, $\mathbf{y}$ rappresenta il vettore delle variabili dipendenti, A corrisponde alla matrice delle variabili indipendenti (features) e α è l'iperparametro che controlla l'intensità della regolarizzazione.

2.2 Soft-Thresholding

Nell'algoritmo ISTA (Iterative Shrinkage Thresholding Algorithm), la regressione LASSO viene risolta utilizzando un approccio iterativo basato sul Gradient Descent.

In questo contesto, il Gradient Descent viene impiegato per affrontare il problema di ottimizzazione precedentemente illustrato, aggiornando progressivamente i pesi del modello nella direzione opposta al gradiente della funzione costo, fino al raggiungimento della convergenza.

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

Poiché la norma L_1 non è differenziabile, si ricorre al concetto di sub-gradiente attraverso l'impiego dell'operatore di Soft-Thresholding.

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

Nel contesto della regressione LASSO, questo si traduce nella seguente espressione:

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

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

Questa operazione, nota come Soft-Thresholding, fornisce una soluzione analitica per l'aggiornamento dei coefficienti nel caso di penalità $L1$, dove il gradiente classico non è applicabile.

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:

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

$$\text{s.t.} \quad \mathbf{x} - \mathbf{z} = 0$$

Il framework ADMM scompone 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:

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

$$\mathbf{z}^{(k+1)} = \underset{\mathbf{z}}{\operatorname{argmin}} \left\{ \alpha\|\mathbf{z}\|_1 + \frac{\rho}{2}\|\mathbf{x}^{(k+1)} - \mathbf{z} + \mathbf{u}^{(k)}\|_2^2 \right\}$$

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(0)} + \sum_{i=1}^{k+1} \|\mathbf{x}^{(i)} - \mathbf{z}^{(i)}\|_2^2$$

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

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

$$A^T(A\mathbf{x} - \mathbf{y}) + \rho(\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{u}^{(k)}) = 0$$

$$(A^T A + \rho I)\mathbf{x} = A^T\mathbf{y} + \rho(\mathbf{z}^{(k)} - \mathbf{u}^{(k)})$$

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

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:

$$\mathbf{x}^{(k+1)} = (A^T A + \rho I)^{-1}(A^T\mathbf{y} + \rho(\mathbf{z}^{(k)} - \mathbf{u}^{(k)}))$$

$$\mathbf{z}^{(k+1)} = S_{\frac{\alpha}{\rho}}(\mathbf{x}^{(k+1)} + \mathbf{u}^{(k)})$$

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

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.

Un ulteriore vantaggio decisivo è la sua capacità di essere parallelizzato e implementato in modo distribuito, che lo rende particolarmente adatto per scalare su problemi di grandi dimensioni.

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 (o agenti). L'ottimizzazione del consenso viene implementata partizionando il dataset tra gli agenti, ciascuno dei quali mantiene una copia locale delle variabili di ottimizzazione. In questo modo, il problema globale viene scomposto rispetto ai dati e risolto in maniera collaborativa attraverso lo scambio di informazioni tra i nodi.

Questo approccio è possibile quando la funzione obiettivo presenta una struttura additiva rispetto ai dati, il che permette di scomporre il problema di ottimizzazione in sottoproblemi indipendenti che possono essere risolti in parallelo.

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

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

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

Il problema Lasso è intrinsicamente distribuibile grazie alla natura separabile della sua funzione obiettivo. Infatti, sia le osservazioni y che la matrice A possono essere partizionati 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:

$$\min_{(x_1,\dots,x_N),\mathbf{z}} \sum_{i=1}^{N} \|\mathbf{y}_i - A_i\mathbf{x}_i\|_2^2 + \lambda\|\mathbf{z}\|_1$$
$$\text{s.t.} \quad \mathbf{x}_i - \mathbf{z} = \mathbf{0}, \quad i = 1,\dots,N$$

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

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

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

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

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.

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 sono definite dalle seguenti equazioni:

$$\mathbf{x}_i^{(k+1)} = (A_i^T A_i + \frac{\rho}{2} I)^{-1}(A_i^T \mathbf{y_i} + \frac{\rho}{2}(\mathbf{z}_i^{(k)} - \mathbf{u}_i^{(k)}))$$

$$\mathbf{z}^{(k+1)} = \operatorname{Prox}_{\frac{\lambda}{N\rho}\|\cdot\|_1} \left\{ \hat{\mathbf{x}}^{(k+1)} + \hat{\mathbf{u}}^{(k)} \right\} = S_{\frac{\lambda}{N\rho}}(\hat{\mathbf{x}}^{(k+1)} + \hat{\mathbf{u}}^{(k)})$$

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

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 Dataset, un insieme di dati pubblici ampiamente utilizzato nella letteratura di machine learning. Questo dataset 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 del blocco;
- **latitude**: latitudine del blocco;
- **housing_median_age**: età mediana delle abitazioni nel blocco;
- **total_rooms**: numero totale di stanze nel blocco;
- **total_bedrooms**: numero totale di camere da letto nel blocco;
- **population**: popolazione nel blocco;
- **households**: numero di nuclei familiari nel blocco;
- **median_income**: reddito mediano delle famiglie nel blocco;
- **median_house_value**: valore mediano delle abitazioni nel blocco;

3.2 Preprocessing dei dati

La fase iniziale prevede la pulizia e la normalizzazione del dataset California Housing. Lo script `preprocess_california_housing.m` si occupa di:

- Caricare il dataset originale e gestire eventuali valori mancanti, sostituendo quelli presenti nella colonna *total_bedrooms* con la mediana.
- Escludere la colonna *ocean_proximity*, che rappresenta una variabile categorica, per focalizzare il modello su variabili numeriche.

- Selezionare le feature principali e la variabile target.
- Normalizzare le feature nell'intervallo $[0, 1]$ per garantire una migliore convergenza degli algoritmi di ottimizzazione.
- Salvare il dataset preprocessato in un nuovo file CSV, pronto per l'analisi.

3.3 Classe LassoReg

Gli algoritmi descritti in precedenza sono stati implementati all'interno della classe `LassoReg`, 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 mette inoltre a disposizione metodi per la predizione sui nuovi dati, il calcolo della funzione di loss, e la visualizzazione della convergenza degli algoritmi di ottimizzazione.

3.4 Script principale

Il file `main.m` rappresenta il cuore dell'esperimento, orchestrando tutte le fasi di addestramento e valutazione:

- Carica il dataset preprocessato.
- Suddivide i dati in training e test set.
- Istanzia e addestra i modelli LASSO secondo i tre approcci.
- Valuta le performance tramite la metrica R^2 e visualizza graficamente sia le predizioni sia l'andamento della funzione obiettivo e dei residui.
- Consente di confrontare in modo diretto efficienza, rapidità di convergenza e accuratezza dei diversi 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	0.5588	12.876	50000
ADMM	0.5839	0.0010	4
ADMM-Dist	0.5726	0.0376	170

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

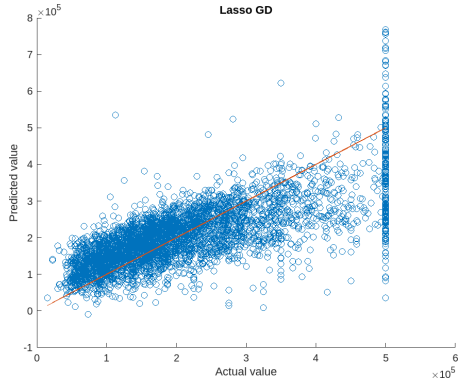

Figura 1: Predizioni ottenute con ISTA.

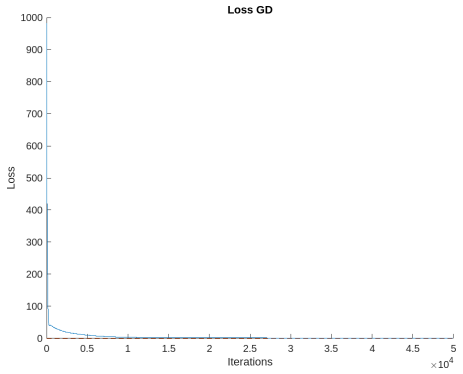

Figura 2: Convergenza di ISTA.

ADMM

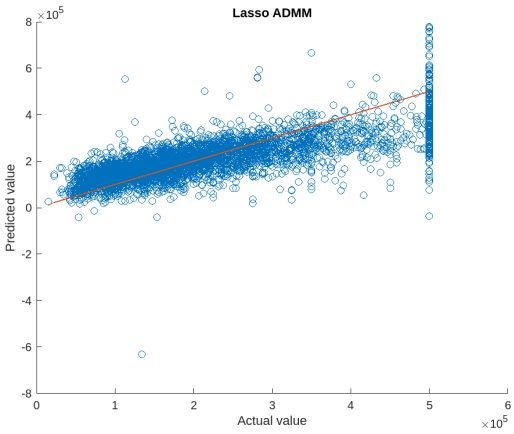

Figura 3: Predizioni ottenute con ADMM.

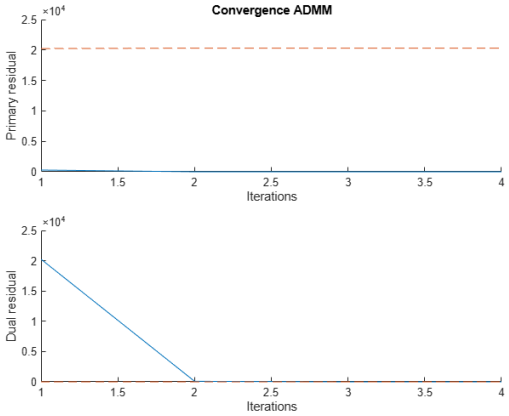

Figura 4: Convergenza di ADMM.

ADMM Distribuito

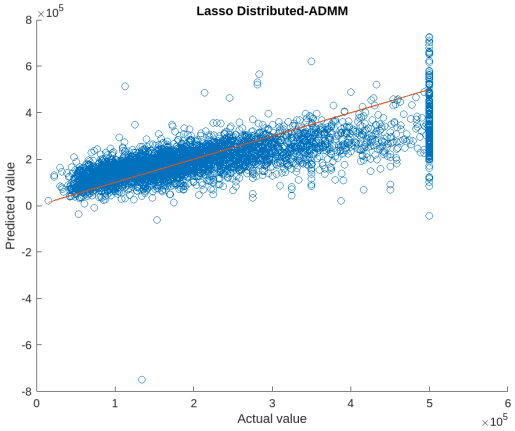

Figura 5: Predizioni ottenute con ADMM distribuito.

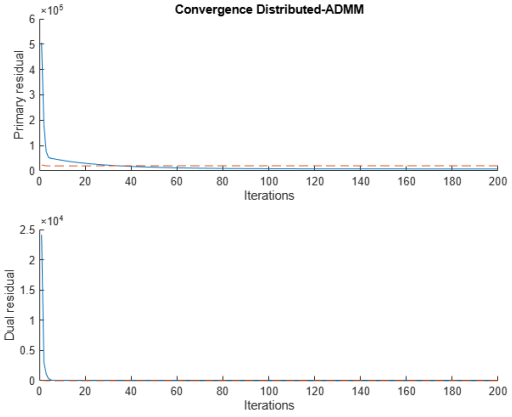

Figura 6: Convergenza di ADMM distribuito.

4.3 Discussione dei Risultati

Precisione (R^2)

- **ADMM** ottiene il valore di R^2 più alto (0.5839), indicando una migliore capacità predittiva rispetto agli altri algoritmi.
- **ADMM Distribuito** raggiunge un R^2 leggermente inferiore (0.5726), ma comunque superiore rispetto a ISTA.
- **ISTA** si attesta su un valore di R^2 di 0.5588, mostrando una performance comunque valida ma inferiore agli altri metodi.

Tempo di esecuzione

- **ADMM** si distingue nettamente per la rapidità: solo 0.001 secondi per raggiungere la convergenza, grazie all'efficienza della soluzione analitica dei sottoproblemi e alla rapida decrescita dei residui.
- **ADMM Distribuito** impiega più tempo (0.0376s) 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 (12.876s), 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 (170), 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 essendo leggermente meno accurato e più lento del classico ADMM, rappresenta una valida soluzione in contesti distribuiti o quando i dati sono suddivisi tra più nodi.

L'algoritmo ISTA, pur essendo semplice da implementare, soffre di una convergenza lenta e di una minore accuratezza, risultando meno 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 prestazioni, Distributed ADMM per scenari distribuiti, mentre ISTA può essere considerato per semplicità su dataset piccoli o come baseline.