Optimization Methods
Confronto tra SMO e DCD per l'addestramento di SVMs

Carlo Lucchesi
Matricola: 7176837

Febbraio 2025

1 Obbiettivo dell'Elaborato

Questo elaborato contiene i risultati del Laboratorio di Optimization Methods in merito allo studio di due algoritmi di ottimizzazione utilizzati per l'addestramento di SVM lineari: il Sequential Minimal Optimization (SMO) il Dual Coordinate Descent (DCD).

L'obbiettivo consiste nell'implementazione degli algoritmi, e la loro validazione con un modello di confronto. Stabilita la correttezza, eseguire dei test con dataset di dimensioni e tolleranze diverse al fine di confrontarne le performance.

1.1 Strumenti utilizzati

Per la realizzazione del codice sono stati utilizzati i seguenti strumenti:

- **numpy**: libreria che offre supporto per operazioni di algebra lineare.
- **scipy**: i dataset recuperati con la funzione *load_svmlight_file()* restituiscono una matrice sparsa (della libreria scipy), che deve essere convertita in una array bidimensionale denso di **numpy**.
- **scikit-learn**: da questa libreria sono stati utilizzati il modello LinearSVC come implementazione di riferimento, e altre funzioni per la gestione di training e test set.
- **pandas**: questa libreria è stata utilizzata per analizzare velocemente le caratteristiche salienti dei dataset utilizzati.
- **matplotlib**: questa libreria è stata utilizzata per la generazione di grafici.
- **jupyter**: si è scelto di implementare il codice in un Notebook Jupyter per renderlo più chiaro e interattivo.

Per evitare conflitti con le librerie installate localmente, si è scelto si creare un ambiente virtuale mediante **conda**. Questo è stato realizzato con i comandi:

```
conda create --name OPT numpy scikit-learn jupyter pandas\
    conda-forge::matplotlib
conda activate OPT
```

2 Algoritmi implementati

Nel contesto del machine learning, l'efficienza degli algoritmi diventa cruciale data la crescere dimensione dei dataset utilizzati. Nel caso delle SVM, la formulazione del problema duale è diventata la principale data la maggiore semplicità del problema, e per la possibilità di realizzare il *Kernel Trick*.

Nel contesto di dati con un numero elevato di features, l'utilizzo del Kernel Trick diventa irrilevante. Infatti, un ulteriore aumento della dimensionalità potrebbe far perdere la capacità di generalizzazione del modello finale, e portare al problema dalla *Curse of Dimensionality*.

Limitandosi quindi ai kernel lineari, è possibile realizzare una formulazione semplificata del problema duale che richiede meno tempo per l'esecuzione di una singola iterazione.

2.1 Sequential Minimal Optimization (SMO)

Questo algoritmo permette di risolvere il problema duale di una SVM, ovvero:

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{1}$$

$$0 \leq \alpha \leq C \tag{2}$$

$$\alpha^T y = 0 \tag{3}$$

Sia $X \in \mathbf{R}^{n\times p}$ il dataset, e $Y \in \{-1, 1\}^n$ le classi di appartenenza, si ottiene che $Q_{ij} = y^i y^j K(x^i, x^j)$, dove K è una Kernel Function valida. Noi vedremo il caso lineare dove $Q_{ij} = y^i y^j {x^i}^T x^j$.

Al crescere della dimensione del dataset calcolare esplicitamente Q potrebbe non essere possibile, per questo motivo spesso vengono calcolate le singole colonne della matrice quando necessario. Infatti, per un numero elevato di variabili $\alpha_i = 0$, rendendo molte colonne inutili. Per via nelle limitate capacità di calcolo a disposizione, sono stati usati dataset abbastanza piccoli, tali da permette un calcolo esplicito di Q. I tempi sono stati confrontati con una implementazione che calcola le singole colonne (definizione *lazy* della matrice Q), e non si sono notate significative differenze. Con n più grandi (come si vedrà in alcuni test) potrebbe non essere possibile.

Dato l'elevato numero di variabili, è conveniente ottimizzare il problema con una tecnica di decomposizione. In questo caso, dato il vincolo di eguaglianza all'equazione (3), è necessario utilizzare un working set $W = \{i, j | i, j\{1, 2, \ldots, n\}, i \neq j\}$, cosa che permette di individuare la soluzione ottima del sotto problema in forma chiusa.

Oltre a questo, bisogna associare a W una direzione che sia: ammissibile, di discesa, e che rispetti l'equazione (3). Sia $\alpha' = \alpha + d$ con d direzione dello spostamento, il mantenimento del vincolo (3) sia ha con:

$$d_i = \frac{1}{y^i}, d_j = -\frac{1}{y^j}, d_k = 0 \ \forall k \neq i, j \tag{4}$$

Data questa forma della direzione, perché sia ammissibile si deve restringere la selezione a $W = \{i, j\}$ tale per cui:

$$\begin{aligned} i \in R(\alpha) &= \{i | (\alpha_i = 0 \ \wedge \ y^i = 1) \vee (\alpha_i = C \ \wedge \ y^i = -1) \ \vee (\alpha_i \in (0, C))\} \\ j \in S(\alpha) &= \{j | (\alpha_j = 0 \ \wedge \ y^j = -1) \vee (\alpha_j = C \ \wedge \ y^j = 1) \ \vee (\alpha_j \in (0, C))\} \end{aligned} \tag{5}$$

Infine, per garantire una direzione di discesa si seleziona, la coppia $W = \{i, j\}$ che rispetti l'equazione (6):

$$\frac{\nabla_i f(\alpha)}{y_i} < \frac{\nabla_j f(\alpha)}{y_j} \tag{6}$$

Per garantire una condizione di terminazione finita nel numero di passi, si sceglie la coppia che realizza la massima distanza, detta anche la Most Violating Pair (MVP). Questa è definita dall'equazione (7):

$$i \in arg \max_{h \in R(\alpha)} \{-\frac{\nabla_h f(\alpha)}{y^h}\}, \ j \in arg \min_{h \in S(\alpha)} \{-\frac{\nabla_h f(\alpha)}{y^h}\} \tag{7}$$

Data questa scelta specifica dalle variabili, sfruttando le condizioni KKT si può dimostrare (sezione 7.2.2 [1]) che $\forall \epsilon$ la condizione (8) viene soddisfatta in un numero finito di

passi:

$$\max_{h \in R(\alpha)} \{-\frac{\nabla_h f(\alpha)}{y^h}\} + \epsilon > \min_{h \in S(\alpha)} \{-\frac{\nabla_h f(\alpha)}{y^h}\} \tag{8}$$

2.2 DCD

Se ci limitiamo al caso di una SVM con Kernel lineare (ovvero dove $K(x^i, x^j) = {x^i}^T x^j$), è possibile riformulare il problema di ottimizzazione senza il bias b, aggiungendo a ogni elemento una feature fittizia sempre uguale ad 1:

$$\min_w \frac{1}{2}||w||^2 + C \sum_i^n \max\{0, 1 - y^i(w^T x^i)\} \tag{9}$$

Oppure, in modo analogo, si può passare alla formulazione:

$$\min_w \frac{1}{2}||w||^2 + C \sum_i^n \max\{0, 1 - y^i(w^T x^i)\}^2 \tag{10}$$

Da queste si può derivare la formulazione del duale:

$$\begin{gathered} \min_\alpha \frac{1}{2}\alpha^T \bar{Q}\alpha - e^T\alpha \\ 0 \leq \alpha \leq U \end{gathered} \tag{11}$$

dove $\bar{Q} = Q + D$, con D matrice diagonale. Da primale (9) si ottiene che $U = C$ e $D_{ii} = 0$, mentre da (10) che $U = \infty$ e $D_{ii} = \frac{1}{2C}$. In ogni caso, è stato rimosso il vincolo di uguaglianza (3), permettendo di lavorare con working set di una singola variabile.

In questo elaborato, si è esplorata soltanto la formulazione derivata da (9) in modo da poterne confrontare il duale con quello di (1).

Questa formulazione semplificata riduce il costo per singolo passo, a discapito di più iterazioni.

La scelta della variabile è indifferente, purché vengano scelte ciclicamente. Come condizione di terminazione, è stata utilizzata quella descritta nella sezione 3.3 dell'articolo [2].

2.3 Analisi di Complessità

Analizzando le operazioni richieste per eseguire un singolo passo si ottengono i risultati riportati nella tabella 1. Quindi, in questi esperimenti si vorrà verificare se il ridotto costo

Aggiornamento	SMO	DCD (caso lineare)
Variabili	$O(1)$	$O(p)$
Gradiente	$O(np)$	NA

Table 1: Costo asintotico delle operazioni di aggiornamento

di aggiornamento di DCD rispetto a SMO compensa il numero maggiore di operazioni eseguite.

Come detto in sezione 2.1, la matrice Q viene calcolata esplicitamente, comportando un costo asintotico complessivo di $O(pn^2)$. Questo non può essere confrontato direttamente

con il costo di una singola iterazione, in quanto il calcolo della matrice avviene una sola volta all'inizio dell'algoritmo. Inoltre, si è notato che fintanto sia possibile mantenere Q interamente in memoria, non c'è nessun peggioramento delle performance rispetto a una valutazione singola delle colonne necessarie.

3 Dataset Utilizzati

In entrambi i dataset è stata aggiunta una feature fittizia con valore sempre uguale a 1 nei casi di test con DCD, cosa richiesta dalla formulazione del problema.

Per entrambi i dataset è stato necessario accertarsi che le classi fossero identificate con {-1, 1}, dato che questi valori vengono utilizzati nelle operazioni aritmetiche degli algoritmi.

3.1 Breast Cancer

Questo dataset è stato scelto per via della sua ridotta dimensione, così da verificare la correttezza delle implementazioni sotto analisi. Tra le versioni disponibili è stata scelta quella le cui feature erano state "scalate" tra [-1, 1], in modo da semplificare l'addestramento.

Di questo dataset non è stato possibile recuperare ulteriori informazioni.

3.2 Covtype

Questo dataset è stato utilizzato per i test di performance. Data l'elevata dimensione (581.012 elementi per 54 feature ciascuno), non è stato possibile utilizzarlo interamente per le limitate capacità di calcolo a disposizione. Quindi, per ovviare a questo problema, ne sono stati estratti dei campioni più piccoli.

Analizzando svariati dataset disponibili su
https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/, escludendo quelli di dimensioni inadeguati, o con un numero di feature eccessive o poche, con una preliminare analisi con i DataFrame di **pandas**, si è notato che molti di essi avessero features con le stesse metriche di tendenza centrale. Questo fenomeno suggeriva una scarsa qualità del dataset. Questo dataset non risultava affetto dal fenomeno, e per perciò è stato scelto per i test.

Maggiori informazioni sul dataset sono disponibili a
https://archive.ics.uci.edu/dataset/31/covertype.

Notare che il dataset che è stato precedentemente convertito per realizzare una classificazione binaria, e non multi classe come originariamente era stato pensato.

4 Risultati

Inizialmente si è verificato che le implementazioni fossero corrette, andando ad analizzare la accuratezza ottenuta sia da SMO che DCD, confrontandola con quella del modello LinearSVC della libreria **scikit-learn**.

Una volta accertata la correttezza, i test si sono focalizzati sulle performance dei due algoritmi al variare della dimensione del dataset. Data la scarsità di risorse computazionali, e di un numero di dataset comparabili, si è scelto di eseguire i vari test con sotto insiemi del dataset covertype descritto nella sezione 3.2.

Oltre a questi test, si è confrontata la massima accuracy che gli algoritmi raggiungevano in un determinato lasso temporale, cercando di confrontare gli algoritmi nelle stese condizioni.

Infine, si è misurato il comportamento degli algoritmi al cambiamento del numero di features.

In tutti i test condotti, è stato estratto un test set pari al 20% del campione proveniente dal dataset originario. Nei grafici, quando si fa riferimento alla grandezza del dataset utilizzato per l'addestramento, questo corrisponde all'80% dei dati che sono stati effettivamente usati per questo scopo.

4.1 Correttezza

Come detto, per prima cosa si è verificata la correttezza degli algoritmi implementati. Per questo scopo si è utilizzato il dataset descritto nella sezione 3.1.

Classe	Precision			Recall			F1		
	LinearSVC	SMO	DCD	LinearSVC	SMO	DCD	LinearSVC	SMO	DCD
-1	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
1	0.97	0.97	0.97	0.97	0.97	0.7	0.97	0.97	0.97
	LinearSVC			SMO			DCD		
accuracy	0.97			0.96			0.97		

Table 2: Risultati dei modelli ottenuti dai vari algoritmi sul training set.

I risultati riportati in 2 sono stati ottenuti ponendo il coefficiente di regolarizzazione $C = 10$.

Dalla tabella 2 si può notare come il modelli siano praticamente identici. In questo caso, i risultati fanno riferimento al training set in quanto ci interessava misurare quanto fossero simili, piuttosto che le loro capacità di generalizzazione.

4.2 Cambiamento della dimensione

Gli esperimenti successivi si sono concentrati sul comportamento degli algoritmi al variare della dimensione del dataset.

Come si può notare dalle figure in 1, il tempo richiesto da DCD cresce più lentamente rispetto a SMO. In generale, la complessità di DCD è lineare rispetto alla dimensione del dataset, mentre per SMO quadratica, cosa che sembra in linea con il tempo per il calcolo della matrice Q.

In generale si può notare come la differenza tra i due algoritmi sembra venir meno al crescere della precisione richiesta nella condizione di stop. Questo fenomeno è più facilmente descritto dalla figura 2.

Per SMO, un aumento di una cifre significative della tolleranza sulla condizione d'arresto sembra far crescere linearmente il tempo di addestramento. Questo non vale per DCD, la cui relazione sembra esponenziale. Detto questo, è importante notare come la condizione di terminazione non sia strettamente correlata alla precisione ottenuta su training e test set, come mostrato in figura 3.

Fatta eccezione di condizioni terminanti troppo piccole, l'accuratezza sul training set è raggiunta anche con tolleranze piuttosto piccole, ammesso di utilizzare dataset di adeguate dimensioni.

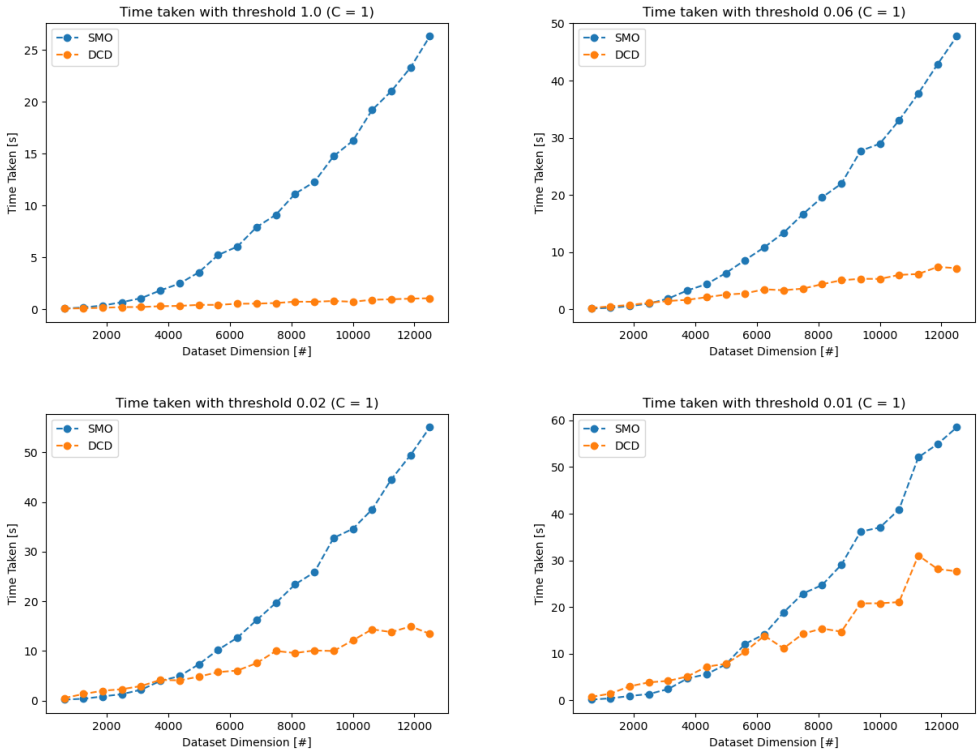

Figure 1: Andamento del tempo di addestramento al cambiare della dimensione del dataset.

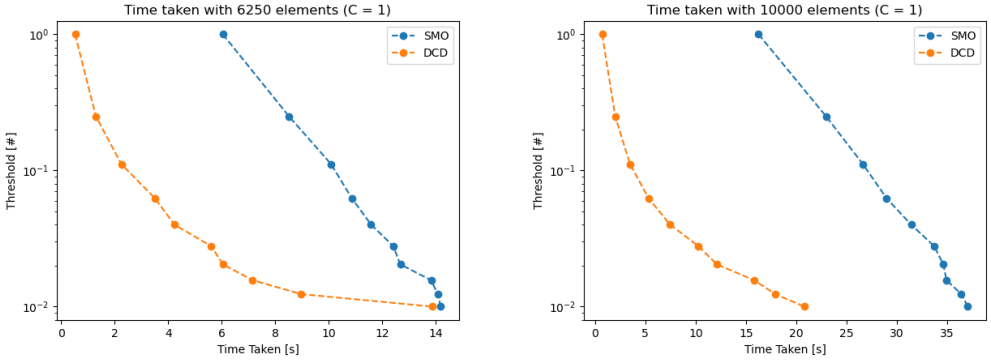

Figure 2: Rapporto tra tolleranza della condizione terminante e il tempo.

Infatti, come mostrato in figura 4, sembra che la precisione sul test set dipenda principalmente dalla dimensione del dataset utilizzato, piuttosto che dalla tolleranza sulla condizione di terminazione.

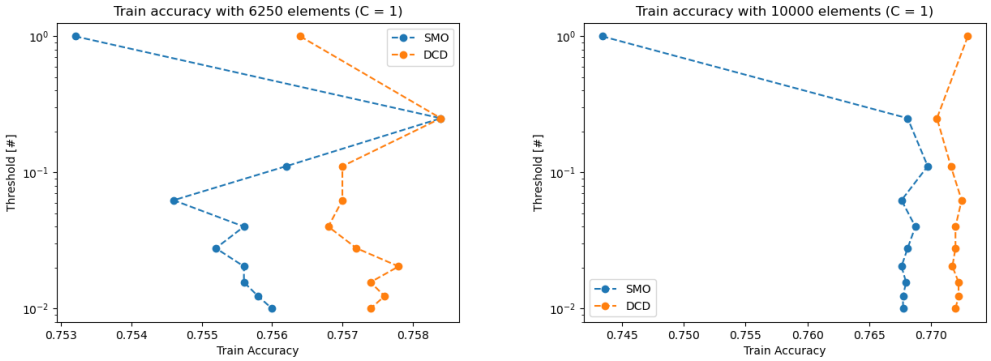

Figure 3: Rapporto tra tolleranza della condizione terminante e precisione su training set.

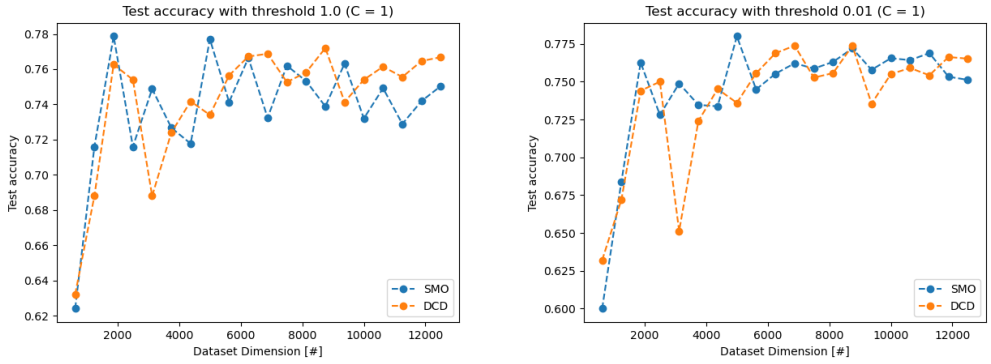

Figure 4: Rapporto tra dimensione del training set e precisione su test set.

4.3 Accuratezza dei modelli rispetto al Tempo

Dato che le condizioni di arresto dei due algoritmi sono diverse e scarsamente comparabili, si sono eseguiti dei test in cui si limitava il tempo massimo per l'esecuzione, per poi misurare le accuracy ottenute. Questi sono riportati in figura 5.

In questi test è stata utilizzata la versione *lazy* di SMO, in quanto il calcolo esplicito della matrice Q portava a una saturazione della memoria.

Il tempo minimo misurato è di 0.5 secondi. In entrambi gli esperimenti DCD raggiunge la massima accuracy sul test set già nel primo caso. Invece, SMO richiede molto più tempo per il raggiungimento di performance analoghe. Questa differenza, come era prevedibile, aumenta al crescere della dimensione del dataset.

Come ultimo test, si è verificata la possibilità di addestrare un SVM sull'intero dataset da 580000 elementi utilizzando DCD. Di questi ne sono stati estratti casualmente il 20% come test set. Il risultato, riportato in figura 6, mostra come in appena 20s si raggiunga la massima accuracy possibile (posto il $C = 1$). Questo addestramento non sarebbe stato possibile con SMO in tempi compatibili con questo laboratorio.

I risultati mostrano la netta superiorità di DCD nella risoluzione di problemi quadratici volti all'addestramento di SVM lineari.

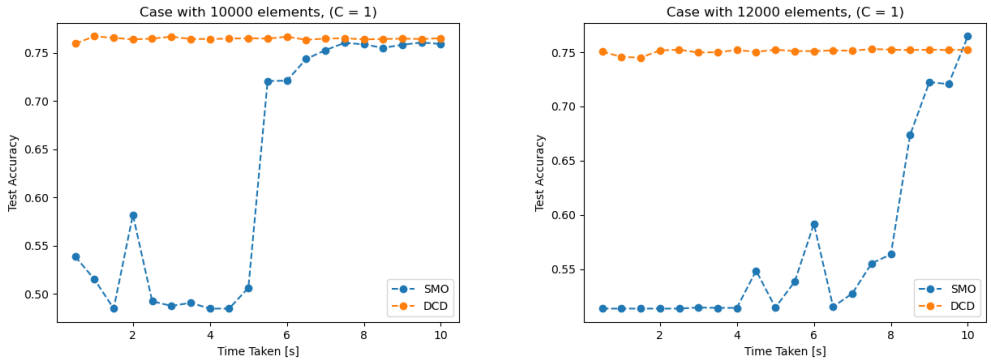

Figure 5: Risultato sul test set sotto vincoli temporali.

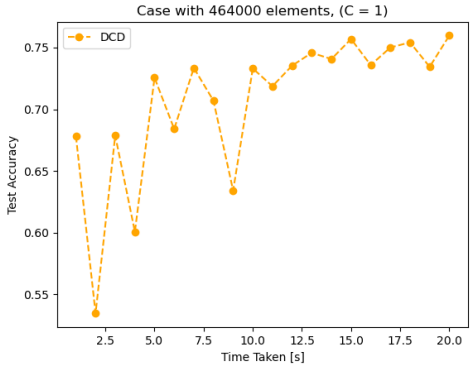

Figure 6: Addestramento utilizzando DCD sull'intero dataset.

4.4 Cambiamento del numero di features

Sono stati condotti anche dei test che verificavano il tempo impiegato per l'addestramento al variare del numero di features utilizzate. In ogni caso di test ci si è assicurati che entrambi gli algoritmi utilizzassero lo stesso sottoinsieme di features. Questo era volto a evitare che ricevessero insiemi che discriminavano diversamente il dataset, cosa che avrebbe falsato il confronto.

I risultati riportati in figura 7, mostrano un andamento lineare del tempo di addestramento rispetto al numero di features. Questo è in linea con le aspettative teoriche sul costo asintotico per operazione.

5 Conclusioni

In questo laboratorio è stato possibile apprezzare come la maggior efficienza delle singole iterazioni di DCD ne compensi il numero maggiore.

In particolare, si è notato un comportamento asintotico per l'addestramento di DCD pari a $O(pn)$, mentre per SMO di $O(pn^2)$. Notare che le aspettative teoriche riportate

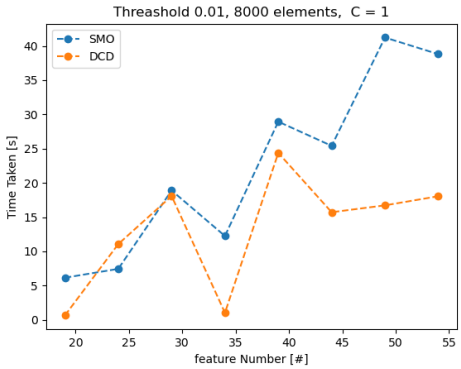

Figure 7: Tempo di addestramento in funzione delle feature.

in 2.3 sono compatibili con i risultati. Questi indicano il costo computazionale per una singola iterazione, mentre il tempo di addestramento è influenzato dal numero complessivo di iterazioni. Sembra quindi ragionevole assumere che il numero di iterazioni richiesto cresca linearmente con il numero di elementi nel dataset.

Oltre a questo, si conferma che l'utilizzo di una tolleranza più piccola aumenti il tempo complessivo di addestramento. Gli esperimenti mostrano come DCD ne sia maggiormente influenzato rispetto a SMO, mantenendo però un andamento lineare della complessità rispetto alla grandezza del dataset. Quindi, al netto di errori di calcolo numerico che impediscono di raggiungere elevate precisioni, DCD rimane più efficiente a livello asintotico. Inoltre, come mostrato dalla figura 3, per ottenere un modello adeguato è sufficiente l'utilizzo di tolleranze piuttosto piccole.

Invece, confrontando l'accuracy sul test set con il tempo di addestramento, si nota come DCD sia nettamente superiore a SMO. In entrambi i casi DCD raggiunge la massima accuracy già nel primo caso di test, mentre SMO richiede un tempo nettamente maggiore. Questa notevole differenza suggerisce come DCD possa essere impiegato su dataset con svariati ordini di grandezza superiore a quelli che SMO è capace di gestire.

Infine, come mostrato dalla figura 4, la formulazione senza bias non ha comportato un peggioramento del modello, rendendo DCD una valida opzione per l'addestramento di SVM lineari su dataset di grandi dimensioni.

References

1. Lapucci, M. *Lecture Notes* (2024).
2. Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S. & Sundararajan, S. *A dual coordinate descent method for large-scale linear SVM* in *Proceedings of the 25th international conference on Machine learning* (2008), 408–415.