1. Introduzione

In questo report verrà presentato il *Progetto* 16, che ha come tema principale l'applicazione e l'analisi del Support Vector Regressor (SVR) per la stima e l'approssimazione di funzioni. L' SVR rappresenta una delle metodologie più rilevanti nell'ambito dell'apprendimento supervisionato, basandosi sui principi teorici dei vettori di supporto introdotti da *Vladimir Vapnik*. Questa tecnica si è dimostrata particolarmente efficace per risolvere problemi di regressione grazie alla sua capacità di trovare un compromesso ottimale tra complessità del modello e accuratezza nella previsione.

L'obiettivo principale del Support Vector Regressor è quello di individuare una funzione che sia in grado di prevedere un valore continuo a partire da input noti, minimizzando l'errore di previsione entro un margine di tolleranza prefissato (denominato *epsilon-insensitive zone*). Il modello si propone di costruire un iperpiano in uno spazio multidimensionale tale che la maggior parte dei dati si trovi all'interno di una fascia di errore accettabile. La funzione risultante cerca di ridurre al minimo la deviazione tra i valori osservati e quelli stimati, penalizzando solo gli errori che superano questa soglia di tolleranza.

(M)

1.1 Costruzione di una funzione lineare

Dal punto di vista matematico, l'obiettivo è la costruzione di una funzione lineare f(x) che può essere espressa nella forma:

$f(x) = \langle w, x \rangle + b$

dove:

- w è il vettore dei pesi,
- x rappresenta il vettore delle caratteristiche in input,
- b è un termine di bias.

NOTA : In formulazione duale,la funzione predittiva non viene costruita esplicitamente tramite un vettore dei pesi w, ma piuttosto come combinazione lineare di prodotti kernel centrati sui dati di training . Più precisamente, si ha:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)\, k(x_i, x) + b$$

1.2 funzione di perdita

Vogliamo che la nostra previsione $f(x)$ sia vicina ad y, ma senza punire l'errore finchè sia piccolo, quindi all'interno di un errore ε.

$$L\varepsilon(y, f(x)) = \begin{cases} 0 & se\ |y - f(x)| \leq \varepsilon \\ |y - f(x)| - \varepsilon & altrimenti \end{cases}$$

Si prevede che la previsione f(x) si avvicini il più possibile al valore reale y, senza tuttavia penalizzare gli errori di piccola entità, ossia quelli compresi entro una soglia ε ; quindi se l'errore rientra nella soglia di tolleranza ε, esso non viene considerato significativo e la funzione di perdita assume valore nullo; viceversa, vengono penalizzati soltanto gli scostamenti che superano tale intervallo.

A differenza della *classificazione*, dove l'obiettivo è separare i dati mediante un margine ottimale, nella *regressione* l'obiettivo è stimare valori numerici continui. Per mantenere un concetto simile di margine, nello SVR si introduce una *zona di insensibilità* (epsilon-insensitive zone), che definisce un intervallo attorno alla funzione f(x) entro cui gli errori non vengono penalizzati. In questo modo, il modello si concentra esclusivamente sugli errori più significativi, ignorando le deviazioni minori che possono derivare da rumore nei dati.

L'obiettivo è quindi individuare una funzione predittiva sufficientemente semplice, ovvero caratterizzata da un vettore dei pesi w di norma contenuta, capace di approssimare i dati osservati entro la tolleranza ε. Un valore piccolo della norma $\| w \|$, più la funzione di regressione risulta "regolare" e meno complessa, riducendo il rischio di overfitting e migliorando la generalizzazione.

NOTA: nel presente lavoro, la funzione di perdita epsilon- insensitive non viene introdotta esplicitamente in forma funzionale all'interno dell'algoritmo, poiché si lavora direttamente con la formulazione duale del problema . Tuttavia , l'effetto della perdita è pienamente recepito nei vincoli e nelle condizioni di ottimalità (KKT). In particolare, il fatto che vengano penalizzati solo gli errori maggiori di ε è garantito del calcolo del gradiente parziale utilizzato nella verifica delle condizioni KKT. Solo quando l'errore di predizione |f(x)-y| eccede la soglia ε, la corrispondente variabile duale viene considerata per l'aggiornamento.

1.3 Funzione di costo

Per poter minimizzare la funzione di perdita e ottenere una stima dei parametri ottimale, si introduce una funzione di costo specifica:

$$C \cdot \sum_{i=1}^{n} L\varepsilon(y_i, f(x_i)) + \frac{1}{2} ||w||^2$$

$L\varepsilon\ (y_i, f(x_i))$: è la perdita epsilon - intensive

∑: somma della perdita su tutti i dati di addestramento

C: è un iper parametro che controlla il compromesso tra errore e semplicità del modello

$\frac{1}{2} ||w||^2$: è la regolarizzazione

La finalità è quella di integrare la minimizzazione della perdita stessa con una penalizzazione della complessità del modello, solitamente espressa in funzione della norma del vettore dei pesi w. Questa funzione di costo, dunque, non si limita a ridurre gli errori di previsione oltre la soglia epsilon, ma incorpora anche un termine di regolarizzazione volto a limitare l'ampiezza dei coefficienti del modello, con l'obiettivo di prevenire fenomeni di overfitting e garantire una migliore capacità predittiva su dati non visti.
Per rendere trattabile dal punto di vista dell'ottimizzazione la struttura non differenziabile della perdita epsilon-insensitive, è utile introdurre una formulazione equivalente basata su variabili ausiliarie, dette *slack variables.*

In particolare, per ciascun dato (x_i, y_i), si definiscono due variabili non negative ξ_i e ξ_i^* che misurano rispettivamente:

- lo scarto oltre ε in caso di errore positivo (quando $f(x_i) > y_i + \varepsilon$) ;
- lo scarto oltre ε in caso di errore negativo (quando $f(x_i) < y_i - \varepsilon$)

se $f(x_i)$ è dentro l'intervallo $[y_i - \varepsilon, y_i + \varepsilon]$, allora ξi = ξi*=0

se invece:

- $f(x_i) > y_i + \varepsilon$ (errore sopra) , $\xi_i > 0$
- $f(x_i) < y_i - \varepsilon$ (errore sotto), $\xi_i^* > 0$

In questo modo, la funzione di perdita diventa *lineare* rispetto alle slack, permettendo di formulare il problema come un problema di ottimizzazione vincolata più facilmente risolvibile.
Queste variabili assumono valore nullo se la previsione rientra nella fascia di tolleranza $[y_i - \varepsilon, y_i + \varepsilon]$, e crescono linearmente al di fuori di essa.
L'introduzione delle slack permette di esprimere il problema in forma vincolata, come segue:

$$min\left(\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)\right)$$

aggiungendo questi vincoli:

$$y_i - f(x_i) \leq \varepsilon + \xi_i$$
$$f(x_i) - y_i \leq \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0$$

In questa formulazione, la funzione di perdita viene assorbita nei vincoli e nei termini lineari sulle slack, rendendo il problema adatto a tecniche di ottimizzazione quadratica (QP) e, in particolare, a metodi efficienti come SMO.

(A1)

2. Metodo Active Set

I metodi active set rappresentano una famiglia di algoritmi per la risoluzione di problemi di ottimizzazione vincolata, in particolare quelli di programmazione quadratica (QP).
L'idea centrale consiste nel mantenere un sottoinsieme di vincoli attivi (cioè saturi, quelli che stanno influenzando il risultato), come ad esempio $\alpha_i=0$ e $\alpha_i=C$, e aggiornare dinamicamente tale insieme per identificare progressivamente la soluzione ottimale.
Essi permettono di concentrarsi solo sui vincoli effettivamente rilevanti in ogni iterazione, consentono di rilevare e scartare i support vectors non attivi. Evitano il costo computazionale della risoluzione di un problema QP (ottimizzazione quadratica) a piena dimensione.
Questo approccio è particolarmente efficace nella *formulazione duale* dello SVR, dove i vincoli sulle variabili duali α_i, α_i* delimitano il dominio ammissibile.

2.1 FORMULAZIONE PRIMALE

Come discusso nella sezione precedente, la funzione di costo con slack variables consente di riformulare il problema di regressione epsilon-insensitive in una forma ottimizzabile.
Questa rappresentazione costituisce, a tutti gli effetti, la formulazione primale dell' SVR, in cui il problema viene espresso come un ottimizzazione vincolata.

La funzione obiettivo, già introdotta in precedenza, è composta da due elementi principali:

- La complessità del modello, penalizzata tramite il termine di regolarizzazione $||w||^2$,che favorisce funzioni semplici e poco flessibili;
- Le violazioni della soglia epsilon, misurate dalla somma delle variabili slack $\sum(\xi_i + \xi_i^*)$, che riflettono gli errori oltre il margine di tolleranza (ε).

I vincoli associati alle slack variables, anch'essi già presentati, garantiscono che il modello penalizzi soltanto le deviazioni superiori a epsilon e ne misuri l'entità in modo lineare.

2.2 FORMULAZIONE DUALE

Come conseguenza dell'introduzione delle slack variables nella formulazione primale, è possibile derivare la formulazione duale del problema di Support Vector Regression (SVR) mediante la costruzione della Lagrangiana e l'applicazione delle condizioni di KKT (Karush-Kuhn-Tucker). La duale consente di esprimere il problema interamente in termini dei moltiplicatori di Lagrange $\alpha_i e\, \alpha_i^*$, associati rispettivamente ai vincoli superiori e inferiori sulle previsioni rispetto alla zona di tolleranza ε.
Nel caso specifico del kernel lineare, la formulazione duale risulta la seguente:

$$max_{\alpha_i, \alpha_i^*} \; -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\,K(x_i, x_j) - \varepsilon\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*)$$

$$\text{soggetta a } \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$$

$$\alpha_i, \alpha_i^* \in [0, C]\,\forall_i = 1, \ldots, n$$

Dove:

- $\alpha_i\, e\, \alpha_i^*$ sono i *moltiplicatori di Lagrange* ;
- $K(x_i, x_j) = x^T_i\, xj$ è il kernel lineare, cioè il prodotto scalare diretto tra i vettori di input;
- Il vincolo $\Sigma(\alpha_i - \alpha_i^*) = 0$ garantisce il bilanciamento dei residui e deriva dalle condizioni di stazionarietà della Lagrangiana rispetto a b.

Questa formulazione rappresenta una trasformazione matematica equivalente alla primale, ma è più adatta per l'ottimizzazione numerica quando si utilizzano tecniche basate sui kernel, come avverrà nel nostro progetto.
Nel problema duale:

- La funzione da massimizzare è quadratica concava (quindi ben posta per metodi numerici);
- Tutte le dipendenze dai dati si esprimono tramite i prodotti scalari $x^T_i\, xj$ che in generale possono essere sostituiti da un kernel;
- Le soluzioni ottimali (cioè i coefficienti $\alpha_i\, e\, \alpha_i^*$) permettono di ricostruire la funzione di regressione senza conoscere direttamente i parametri w della primale.

Nella formulazione duale, i moltiplicatori $\alpha_i - \alpha_i^*$ sono associati ai vincoli sulle deviazioni positive e negative rispetto alla zona di insensibilità ε. Tuttavia, la presenza dei vincolo globale $\sum_i (\alpha_i - \alpha_i^*) = 0$ introduce un ulteriore moltiplicatore μ, che deve essere considerato nelle condizioni di ottimalità (KKT). questo spiega perché, nella verifica delle KKT, la derivata della Lagrangiana rispetto a α_i include anche il termine μ , ovvero :

$$\frac{\partial L_D}{\partial \alpha_i} + \mu = 0 \; se \; 0 < \alpha_i < C$$

Nel nostro progetto si è scelto di lavorare sulla formulazione duale, per due motivi principali:

1. Permette di applicare metodi di ottimizzazione efficienti, come il *Sequential Minimal Optimization* (SMO) e i solver QP generici, che lavorano direttamente su questo tipo di struttura.

2. Abilita l'uso del *kernel trick*, ovvero la possibilità di estendere il modello a spazi non lineari (anche se, per semplicità analitica, nel progetto è stato adottato il kernel lineare).

2.3 KERNEL

Dalla formulazione primale dello SVR, è possibile derivare la formulazione *duale*, che risulta particolarmente vantaggiosa per l'implementazione pratica, soprattutto nel caso in cui si voglia sfruttare il cosiddetto kernel trick.
La formulazione duale consente infatti di lavorare in uno spazio delle caratteristiche implicito, evitando la trasformazione esplicita dei dati tramite ϕ(x). Ciò viene ottenuto attraverso una funzione di kernel.

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

Il kernel calcola direttamente il prodotto scalare nello spazio trasformato.

In questo progetto si è scelto di utilizzare il kernel lineare:

$$K(x_i, x_j) = x_i^T x_j$$

Tale scelta permette di mantenere semplicità analitica, non introducendo parametri aggiuntivi (come γ nel caso RBF), e rende più trasparente la valutazione delle prestazioni dei due approcci di ottimizzazione considerati: il metodo SMO (A1) e il solver generico (A2).

2.4 SMO come metodo active-set per SVR

L' algoritmo *Sequential Minimal Optimization* (SMO) è un metodo di tipo active set progettato per risolvere in modo efficiente problemi di programmazione quadratica vincolata, come quelli che emergono dalla formulazione duale della Support Vector Regression. La sua efficienza deriva dalla strategia di ottimizzare iterativamente solo due variabili alla volta, mantenendo fisse tutte le altre. Questo approccio riduce

drasticamente la complessità computazionale rispetto a metodi che risolvono il problema duale nella sua interezza.

Nel contesto della SVR, l'algoritmo SMO si adatta naturalmente alla struttura del problema duale e consente di mantenere attivo solo un sottoinsieme di vincoli rilevanti, aggiornandoli dinamicamente secondo le condizioni KKT.

Nel metodo SMO per SVR, l'idea principale è la seguente:

1. *Selezione attiva di una coppia di variabili*

Ad ogni iterazione, l'algoritmo seleziona una coppia di indici (i,j) le cui variabili duali (α_i,α_i*) violano maggiormente le condizioni di *Karush-Kuhn-Tucker* (KKT). Questi rappresentano i candidati più promettenti per il miglioramento della funzione obiettivo.

2. *risoluzione del sottoproblema*

Una volta scelta la coppia (i, j), si definisce:

$$\delta = (\alpha_i - \alpha_i^*) - (\alpha_j - \alpha_j^*)$$

Questa espressione rappresenta la differenza tra le componenti duali dei due punti selezionati.

Tuttavia, nella formulazione duale, c'è un vincolo di uguaglianza:

$$\sum_i (\alpha_i - \alpha_i^*) = 0$$

quindi :

$$(\alpha_i - \alpha_i^*) + (\alpha_j - \alpha_j^*) = \text{costante}$$

Questo vuol dire che:

Se una variabile sale, l'altra deve scendere per mantenere il vincolo soddisfatto. Ciò permette di risolvere il sottoproblema a una sola variabile, con un'analisi analitica.

3. *verifica delle condizioni* KKT

Nel problema duale della SVR, ogni punto i è associato a due moltiplicatori di Lagrange $\alpha_i - \alpha_i^*$, che rappresentano rispettivamente le penalizzazioni per l'errore sopra e sotto la soglia di intensità ε. La verifica dell'ottimalità si basa sulle condizioni di Karush - Kuhan - Tucker (KKT). A differenza di problemi con solo vincoli box, la presenza del vincolo globale:

$$\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0$$

introduce un ulteriore moltiplicatore μ nella Lagrangiana . Questo influisce sulle condizioni di stazionarietà:

$$\frac{\partial L}{\partial \alpha_i} = \nabla_i + \mu = 0 \text{ se } 0 < \alpha_i < C$$

Per tutti gli altri casi, le condizioni KKT diventano

- se $\alpha_i = 0$, allora $\nabla_i + \mu \geq 0$
- se $\alpha_i = C$, allora $\nabla_i + \mu \leq 0$
- lo stesso vale per α_i^*.

L'algoritmo SMO aggiorna solo due variabili per volta , in modo da mantenere automaticamente il vincolo di uguaglianza. Confrontando le derivate relative a due variabili (α_i, α_i***) ,** il termine μ si cancella.

$(\nabla_i L + \mu) - (\nabla_j L + \mu) = \nabla_i L - \nabla_j L$

Questo rende possibile procedere senza conoscere μ, purché si lavori solo sulle differenze tra i gradienti .

4. *Aggiornamento e iterazione*

Dopo l'aggiornamento dei valori α_i, α_i***** si aggiorna anche il *bias b* secondo una formula chiusa che tiene conto della variazione dell'output predetto .

Il processo si ripete finché tutte le condizioni KKT sono soddisfatte, entro una soglia di tolleranza τ.

2.5 strategia di selezione delle variabili

Nel contesto SVR, la presenza di due moltiplicatori duali per ciascun punto (α_i, α_i *) rende la selezione delle variabili più complessa rispetto al caso della SVM per classificazione. SMO adotta una strategia in due passaggi:

1. *Scelta del primo indice i*

Viene scelto tra le variabili che violano maggiormente le condizioni KKT. In genere, si preferiscono variabili non sature (cioè $0 < \alpha_i < C$) e con errore di predizione elevato.

2. *Scelta del secondo indice j*:

$$j = arg\,max_j |E_i - E_j|$$

Una volta fissato i, si seleziona j in modo da massimizzare la differenza $|E_i - E_j|$ dove E_i è l'errore di predizione sul punto x_i. L'obiettivo è ottenere il massimo impatto possibile sull'ottimizzazione duale.

Questa strategia mira a selezionare le due variabili la cui ottimizzazione porti al miglioramento più rapido della funzione obiettivo, nel rispetto dei vincoli.

Questo produce un aggiornamento più grande dei moltiplicatori duali (cioè α_i,α_j) e quindi un miglioramento più rapido nella funzione obiettivo.

2.5. Aggiornamento dei moltiplicatori duali

Una volta selezionata la coppia (i,j), il problema ridotto a due variabili viene risolto in modo analitico, mantenendo tutte le altre α_k fisse.

Le variabili da aggiornare sono (α_i, α_i^*) *e* (α_j, α_j^*), ma nel calcolo concreto , si aggiornano solo le differenze tra $\alpha_i - \alpha_i^*$, perché la funzione obiettivo duale dipende solo da : $(\alpha_i - \alpha_i^*) - (\alpha_j - \alpha_j^*)$.

Quindi spesso si lavora con la quantità $\delta = \alpha_i - \alpha_i^*$.

a. I vincoli " box" di intervallo da rispettare :

$$0 \le \alpha_i, \alpha_i^*, \alpha_j, \alpha_j^* \le C$$

Questo serve a garantire che i moltiplicatori non escano dal range valido.

b. vincoli di somma :

$$\sum_k (\alpha_i - \alpha_i^*) = 0$$

quindi se si modifica $\alpha i - \alpha i^*$, bisogna modificare anche $\alpha_j - \alpha_j^*$.

L'idea di base è che cambiando due variabili , in modo che la somma resti invariata.

Per aggiornare analiticamente , introduciamo una quantità :

$$\eta = K(x_i, x_i) + K(x_j, x_j) - 2K(x_i, x_j)$$

Questa misura quanto sono diversi i due dati x_i e x_j rispetto al kernel; serve a calcolare quanto deve cambiare la variabile , con effetto sulla funzione obiettivo.

Il calcolo dell'aggiornamento avviene tramite una " *regola chiusa*" (calcolare direttamente il risultato, senza dover risolvere equazioni numeriche, iterazioni o sistemi lineari) :

$$(a_i - \alpha_i^*)^{new} = (a_i - \alpha_i^*)^{old} + \frac{E_j - E_i}{\eta}$$

$E_i = f(x_i) - y_i$: è l'errore di previsione per il punto i,

$f(x_i)$: è la predizione attuale del modello,

y_i: è il valore vero .

quindi se il modello ha predetto troppo alto o troppo basso , questo errore viene usato per aggiornare i moltiplicatori e migliorare il modello nel prossimo passo.

Quindi per calcolare l'errore , serve conoscere il modello corrente , cioè tutti gli α:

$$E_k = f(x_k) - y_k = \sum_l (\alpha_l - \alpha_l^*) K(x_l, x_k) + b - y_k$$

2.6 Clipping:

Dopo l'aggiornamento, è possibile avere $\alpha^{new}_i > C$ o < 0, ma i vincoli sono esplicitati come :

$$0 \le \alpha_i, \alpha_i^* \le C$$

quindi se $\alpha > C$, si riporta a C

se $\alpha < 0$, si riporta a 0

questo è detto clipping : forzare il valore a restare nei limiti.

Dopo aver aggiornato α_i, α_i^* anche f(x) cambia; per mantenere l'equilibrio del modello, serve aggiornare anche il termine bias *b*, cioè lo spostamento verticale della funzione.

Il calcolo del *bias* si basa sull'idea che, per i punti i con i moltiplicatori α_i, α_i^* , strettamente compresi tra 0 e C, il corrispondente residuo dell'output è esattamente sul margine ε.

Questo consente di risolvere b in modo chiuso usando $f(x_i)$.
Quando uno dei due moltiplicatori α_i *o* α_i^* è strettamente compreso tra 0 e C, si assume che il punto (x_i, y_i) sia sul bordo del margine epsilon, e questo consente di calcolare direttamente il bias.

- caso 1 : $0 < \alpha_i < C$

  useremo :

$$b = y_i - \varepsilon - \sum_{k=1}^{n} (\alpha_k - \alpha_k^*) K(x_k, x_i)$$

  con questa formula indichiamo che il punto $f(x_i) = y_i - \varepsilon$ sia sul margine inferiore.
  la somma rappresenta la parte del modello appresa finora , quindi b corregge il resto.

- caso 2 : $0 < \alpha_i^* < C$

  useremo:

$$b = y_i + \varepsilon - \sum_{k=1}^{n} (\alpha_k - \alpha_k^*) K(x_k, x_i)$$

  in questo caso il punto è sul margine superiore, quindi $f(x_i)=y_i+\varepsilon$.

Nel caso in cui sia α_i che α_i^* sono = 0 o C, non è possibile applicare le formule sopra ,ma useremo la media dei bias calcolati da entrambi i punti :

$$b = \frac{bi + bj}{2}$$

Quest'ultimo approccio non è perfetto, ma potrebbe essere un compromesso ragionevole .

2.7 Criteri di Arresto (Stop Criteria)

Per determinare la condizione di arresto nel metodo SMO, si verifica se tutte le condizioni KKT (Karush-Kuhn-Tucker) sono soddisfatte entro una certa soglia di tolleranza τ.

Nel problema duale della SVR, le variabili duali α_i **e** α_i^* sono soggette a vincoli box $0 < \alpha_i, \alpha_i^* < C$. Le condizioni KKT stabiliscono che il gradiente della funzione obiettivo rispetto a ciascuna variabile duale deve essere compatibile con questi vincoli.

Nel contesto dell'ottimizzazione con vincoli, in teoria si dovrebbe considerare la proiezione del gradiente sul cono tangente definito dai vincoli attivi.

Il cono tangente è l'insieme delle direzioni ammissibili lungo cui ci si può muovere a partire da un punto , rimanendo nei vincoli. In pratica , nelle condizioni KKT, si richiede che il gradiente della funzione obiettivo sia proiettato all'interno di questo cono, perchè solo così esisterà una direzione che rispetta i vincoli e consente di migliorare la funzione. Nel nostro caso , questo si traduce nel verificare che il gradiente della funzione duale rispetti la condizione di stazionarietà compatibilmente con i vincoli :

- se $\alpha_i = 0$, allora non è ammissibile un gradiente che spingerebbe verso valori negativi (cioè gradiente negativo);
- $\alpha_i = C$, non è ammissibile un gradiente positivo ;

- $0< \alpha_i <C$, allora il gradiente deve essere prossimo a zero.

Questa logica dell'algoritmo può essere rispettata in forma condizionale , senza calcolare esplicitamente un vettore proiettato. La verifica della soddisfazione delle KKT viene fatta attraverso un controllo sul segno e sul valore del gradiente, con condizioni *if* che discriminano se una violazione è significativa o meno rispetto alla soglia di tolleranza τ. Questo approccio è equivalente alla proiezione esplicita, ed è comunemente adottato negli algoritmi SMO.

Per ciascun moltiplicatore α_k(o α_k *), si calcola il gradiente parziale:

$$\nabla_k = \frac{\partial L}{\partial \alpha_k}$$

e la violazione KKT è definita in base al valore di α_k:

- $0< \alpha_k <C$, la condizione ottimale richiede $\nabla_k = 0$ (entro τ);
- se $\alpha_k = 0$, la violazione è max (0, - ∇_k);
- se $\alpha_k = C$, la violazione max (0, ∇_k).

L'algoritmo SMO continua ad aggiornare le variabili finchè la massima violazione tra tutte le variabili (considerando sia α_k che α_k *) è inferiore alla tolleranza τ, segnalando così che tutte le condizioni KKT sono soddisfatte entro l'accuratezza desiderata.

2.8 Criteri ausiliari

Oltre alla tolleranza sulle condizioni KKT, è prassi comune definire anche criteri di arresto ausiliari, come:

- un numero massimo di iterazioni,
- una soglia di miglioramente minimo sulla funzione obiettivo

(A2)

3. Scipy.optimize.minimize

Per affrontare il punto (A2) del progetto, ovvero la risoluzione della formulazione duale della SVR mediante un risolutore generico, introduciamo la funzione ***scipy.optimize.minimize()*** della libreria SciPy. Questa scelta è motivata dal fatto che, pur non essendo specializzata per problemi di programmazione quadratica (QP), ***minimize*** è un metodo *general-purpose* pienamente ammesso secondo i vincoli del progetto. In particolare, il metodo **SLSQP** (Sequential Least Squares programming) consente la gestione di vincoli di uguaglianza e disuguaglianza, nonché vincoli di tipo box.

La formulazione duale dell SVR da risolvere è una funzione obiettivo quadratica con

- un vincolo di uguaglianza:

$$\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0$$

- e i vincoli box : $0 \le \alpha_i, \alpha_i^* \le C$

A livello implementativo, il problema verrà formulato come segue:

- la funzione obiettivo rappresenta il problema duale da minimizzare;
- i vincoli vengono inseriti o attraverso il modo classico con “def” o con il metodo compatto detta “funzione anonima” ;
- i bound vengono applicati direttamente su ciascun parametro α_i e α_i*.

Questo approccio permette di valutare le prestazioni di un solver generico numerico rispetto al metodo SMO (A1), sia in termini di accuratezza che efficienza computazionale . Si ottiene così un confronto completo tra approccio specializzato iterativo (SMO) e uno numerico general-purpose (SLSQP).

4. Convergenza

Nel contesto dell’ ottimizzazione matematica, le proprietà di convessità e concavità giocano un ruolo fondamentale nella determinazione dell’unicità e della qualità della soluzione.
Una funzione si definisce convessa se, per ogni coppia di punti nel dominio, il segmento che li collega giace sopra o sulla curva della funzione. Una funzione concava se lo stesso segmento giace sotto la curva.

Matematicamente , una funzione f(x) è :

- *Convessa* se:

  $$f(\lambda x_1 + (1 - \lambda) x_2) \leq \lambda f(x_1) + (1 - \lambda) f(x_2), \quad \forall \lambda \in [0, 1]$$

- *Concava* se vale l’opposto:

  $$f(\lambda x_1 + (1 - \lambda) x_2) \geq \lambda f(x_1) + (1 - \lambda) f(x_2)$$

Un problema di ottimizzazione si dice convesso se la funzione obiettivo è convessa (nel caso di minimizzazione) i vincoli definiscono un insieme convesso. In questi casi, ogni punto stazionario che soddisfa le condizioni KKT è anche soluzione globale ottima.

Nel caso del Support Vector Regression, la formulazione duale del problema porta una funzione obiettivo concava , del tipo:

$$max_{\alpha_i, \alpha_i^*} \; -\frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i, x_j) - \varepsilon \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) + \sum_{i=1}^{n} y_i (\alpha_i - \alpha_i^*)$$

Questa funzione è *concava* perché il termine quadratico principale, che coinvolge il kernel $K(x_i, x_j)$, è negativo definito. Inoltre, i vincoli imposti (box $0 \leq \alpha_i, \alpha_i^* \leq C$ e la somma $\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0$) sono tutti lineari, quindi definiscono un dominio *convesso chiuso e limitato*.

Come dimostrato da [Takahashi, Guo & Nishi 2008] l’algoritmo SMO applicato alla formulazione duale del Support Vector Regression converge globalmente a una soluzione ottima in numero finito di iterazioni. La dimostrazione poggia sulle seguenti ipotesi:

- il kernel K(x_i, y_i) sia positivo definito garantendo la concavità della funzione obiettivo,
- I dati di training siano finiti e fissi,
- La strategia di selezione delle coppie di indici da aggiornare garantisce una riduzione sufficiente delle violazioni delle condizioni KKT a ogni iterazione.

L'algoritmo SMO procede ottimizzando iterativamente una coppia di variabili (α_i, α_j *) alla volta, risolvendo un sottoproblema di ottimizzazione quadratica in due dimensioni soggetto a :

- vincoli box $0 \leq \alpha_i, \alpha_i* \leq C$
- vincoli di somma $\sum_{i=1}^{n} (\alpha_i - \alpha_i*) = 0$

Ogni aggiornamento dell'algoritmo assicura che la funzione obiettivo duale non diminuisca .. Poichè la funzione è limitata superiormente (perché è concava e il dominio è chiuso e limitato), l'algoritmo non può oscillare indefinitamente. La dimostrazione si basa su una combinazione di argomenti geometrici (riduzione delle violazioni delle condizioni KKT) e analisi delle proprietà di discesa del metodo SMO: si dimostra infatti che il numero di configurazioni distinte delle violazioni di KKT è finito, e che ciascun aggiornamento riduce strettamente almeno una violazione attiva. Questo garantisce che, dopo un numero finito di iterazioni , tutte le condizioni KKT saranno soddisfatte con tolleranza arbitrariamente piccola , e quindi l'algoritmo termina su una soluzione ottima globale del problema duale SVR.

4.1 Velocità di convergenza del metodo SMO

Sebbene [Takahashi, Guo & Nishi 2008] garantiscano la convergenza in un numero finito di iterazioni, non viene fornito un tasso di convergenza esplicito (es. lineare o sublineare). Questo accade perché la velocità pratica di convergenza dell'algoritmo SMO dipende fortemente da vari fattori, tra cui:

- la strategia di selezione delle variabili da aggiornare,
- la distribuzione dei dati e la struttura del kernel,
- il valore di ε e del parametro di penalità C,
- eventuali ridondanze nei dati o vicinanza al margine di molti punti.

Secondo [Smola & Schölkopf, 2003], una scelta efficace delle coppie (i,j) da aggiornare, ad esempio selezionando quelle che violano maggiormente le condizioni di KKT, può migliorare sensibilmente la velocità empirica di convergenza. Tuttavia, nel caso generale non abbiamo trovato una dimostrazione che garantisca un tasso di convergenza asintotico deterministico (ad esempio, O(1/k)) per SMO in SVR.

5. Analisi dei Risultati

Nel progetto abbiamo confrontato due approcci per risolvere il problema duale dello SVR (Support Vector Regression):

- SMO (Sequetial Minomal Optimization) – aggiornamenti a coppie di moltiplicatori duali.
- SLSQP (Sequential Least Squares Programming) – un solver di ottimizzazione numerica "globale", usato come riferimento.

Abbiamo condotto esperimenti su dataset sintetici da 20, 50, 100 e 200 campioni.
Per ogni esperimento, stampa:

- Tempi di esecuzione di SMO e SLSQP

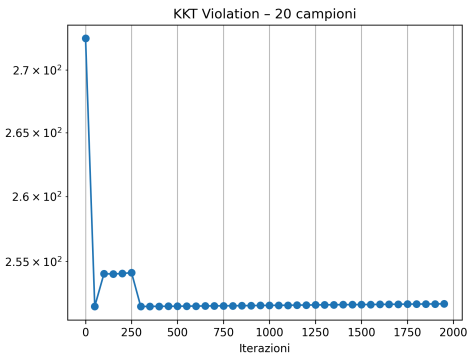
- Valori del dual objective ad ogni step (SMO)
- Dual gap fra SMO e SLSQP
- Eventuali messaggi di convergenza (o mancata convergenza)

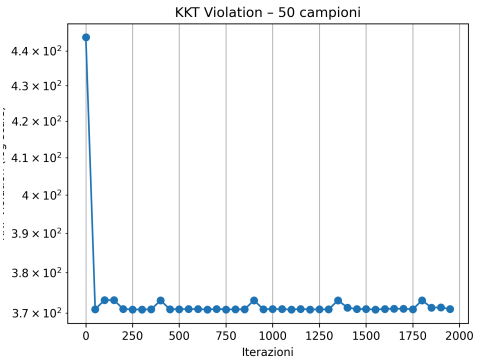
	20 campioni	50 campioni	100 campioni	200 campioni
tempo di esecuzione SMO	7.552678 s	74.218073 s	300.833707 s	1278.576192 s
tempo di esecuzione SLSQP	0.025041 s	0.184937 s	1.884574 s	1278.576192 s
obiettivo SMO	-1529.505045	-4066.100412	-7817.008568	-5451.602514
obiettivo SLSQP	-1638.147520	-4116.624679	-7918.460746	-5612.361130
dual gap assoluto	1.086425e+02	5.052427e+01	1.014522e+02	1.607586e+02
dual gap relativo	6.632032e-02	1.227323e-02	1.281211e-02	2.864367e-02

SMO è molto più lento di SLSQP, da pochi secondi (20 campioni) fino a oltre 20 minuti (200 campioni).

SLSQP, invece, impiega frazioni di secondo per dataset piccoli e pochi secondi per 200 campioni. SLSQP converge quasi sempre (stampando “Convergenza OK”), ma in alcuni casi (es. 50 e 100 campioni) compare “NON CONVERGE: Positive directional derivative for linesearch”; questo indica che il solver non ha trovato la soluzione esatta, ma si è fermato in una situazione “quasi ottima”.

Dual gap relativo finale (quanto SMO è distante da SLSQP), da 6.6e-2 (20 campioni, gap alto) a circa 1e-2 (100–200 campioni). SMO si ferma prima di raggiungere la precisione di SLSQP.





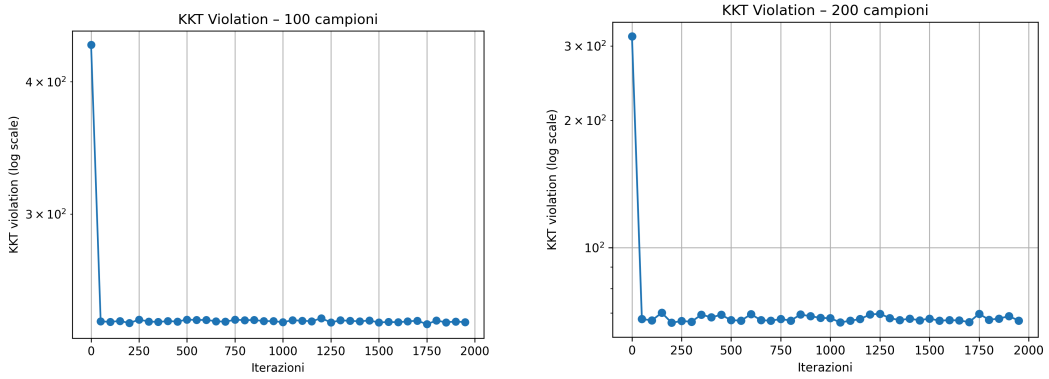
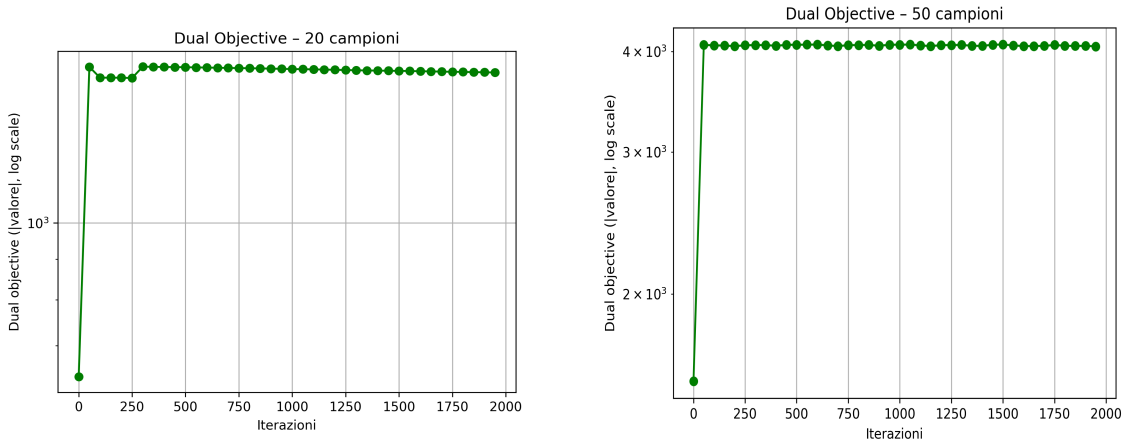


figura 1: kkt violation

Con i grafici che indicano le violazioni KKT vs iterazioni in scala logaritmica ci permette di analizzare come i valori scendono molto velocemente e se il pregresso si ferma.

In generale in tutti i grafici si vede un calo fortissimo della KKT violation nelle prime 50-100 iterazioni; dopo quel crollo iniziale, la curva si stabilizza e diventa quasi piatta , ciò significa che SMO non riesce più a migliorare. All'aumentare dei campioni SMO fatica con più dati , infatti vediamo uno stallo dopo poche iterazioni ; in particolare con 200 campioni la linea sembra " rumorosa" ,questo può accadere perchè SMO corregge alcune α, ma senza un reale progresso verso la soluzione .

Dopo il primo calo, l'algoritmo si blocca , le condizioni KKT restano violate, questo andamento spiega perché il dual gap relativo resta alto .



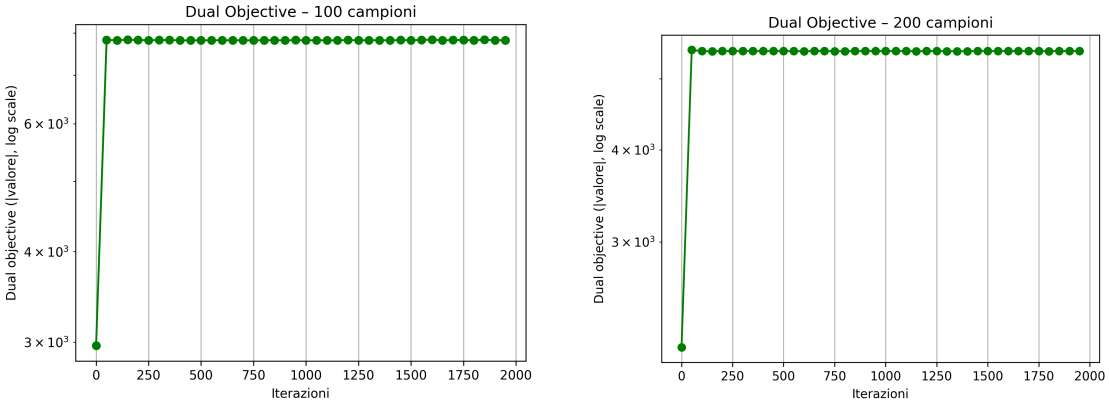


figura 2 : dual objective

il Dual objective è la funzione che stiamo massimizzando con SMO, l'andamento comune di tutti i campioni è che c'è un salto nelle prime 50 iterazioni e dopo questo salto , il grafico diventa piatto ,quindi SMO non aumenta più il dual objective in modo significativo . Questo spiega perché i risultati numerici mostrano dual gap elevati e perché SLSQP riesce a ottenere valori dell'obiettivo più alti di SMO, anche nei casi in cui SLSQP non converge formalmente. il Dual gap è la differenza tra il dual objective ottenuto da SMO e quello ottenuto da SLSQP. Poiché SMO smette di migliorare molto presto , il suo Dual objective resta inferiore a quello di SLSQP; il risultato è che il dual gap resta bloccato su valori relativamente alti , senza ridursi ulteriormente.

L'andamento del dual gap relativo nei grafici conferma esattamente questa interpretazione. In tutti i dataset (20, 50, 100 e 200 campioni) si osserva che c'è un calo brusco nelle prime iterazioni (entro le prime 50), che corrisponde al momento in cui SMO compie la parte più consistente dell'ottimizzazione; subito dopo, la curva del dual gap si stabilizza su un valore costante, senza scendere ulteriormente.

il plateau che vediamo significa che, una volta raggiunta una soluzione "accettabile", SMO non riesce più a ridurre l'errore rispetto alla soluzione ottima. L'algoritmo smette di chiudere il gap con SLSQP: il dual gap rimane "aperto" e non tende a zero, anche dopo 2000 iterazioni.

Il dual gap rimane alto perché SMO si ferma su un ottimo locale "pratico": soddisfa (quasi) tutte le condizioni KKT, ma non riesce a migliorare la funzione obiettivo duale. SLSQP, invece, continua a "spingere" verso l'ottimo teorico, anche nei casi in cui non converge formalmente (come indicato dal messaggio Positive directional derivative for linesearch).

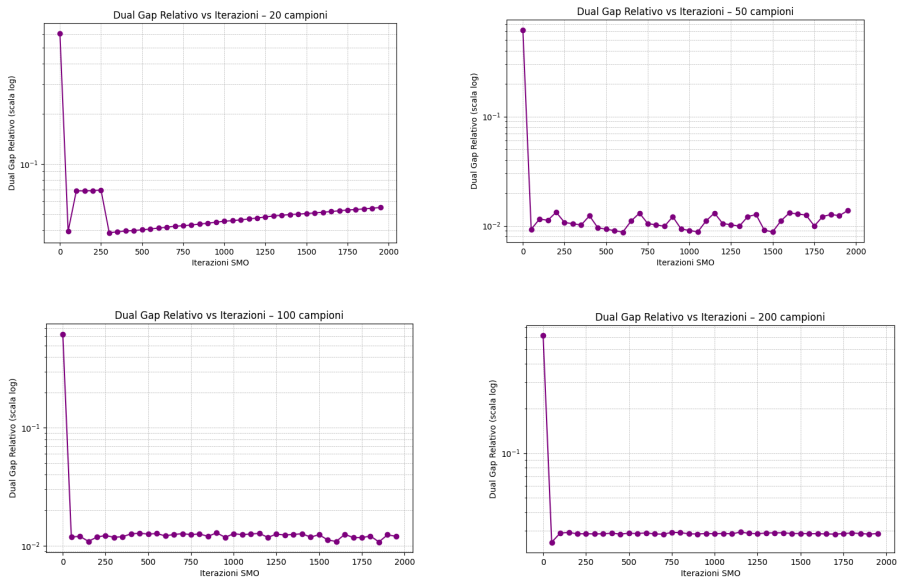


figura 3: dual gap vs iterazioni

Infine i grafici di Convergenza SMO (figura 4) mostrano l'andamento della violazione delle condizioni KKT (Karush–Kuhn–Tucker) nel corso delle iterazioni. Questa metrica indica quanto la soluzione trovata da SMO rispetti i vincoli teorici di ottimalità del problema di Support Vector Regression (SVR).

In teoria ci aspettavamo un algoritmo di ottimizzazione ben funzionante, la violazione KKT dovrebbe decrescere gradualmente fino a valori prossimi a zero, segno che l'algoritmo sta trovando una soluzione sempre più "ottima" e conforme ai vincoli.

Nei grafici osserviamo che in tutti i casi (20, 50, 100 e 200 campioni) vediamo un calo molto brusco nelle primissime interazioni (entro 50-100 step). Questo indica che SMO "sistema" velocemente le violazioni più gravi. Dopo il calo iniziale, la curva si appiattisce subito su un plateau, questo rappresenta il livello minimo di violazione che SMO riesce a raggiungere, ma da cui non riesce più a scendere.

Il plateau della violazione KKT spiega perché anche il Dual Objective smette di crescere. Se SMO si ferma presto (plateau), il Dual Gap rimane "aperto": SMO non riduce la distanza rispetto alla soluzione ottima, e il gap rimane costante nelle iterazioni finali.

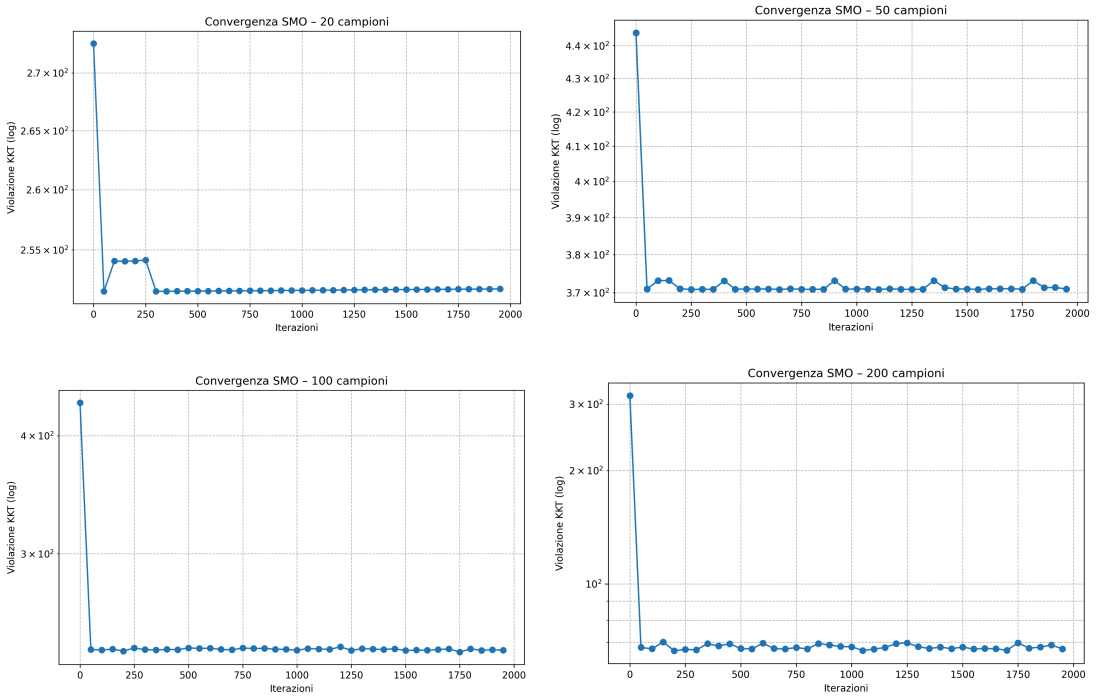


figura 4 : convergenza SMO

Questi risultati mostrano chiaramente il limite dell'algoritmo SMO nella nostra implementazione: lavora bene nelle prime iterazioni (correzioni veloci), ma si ferma troppo presto, senza spingersi fino alla piena convergenza teorica. Il risultato pratico è che SMO fornisce una soluzione decente ma non perfetta: abbastanza buona per avere predizioni ragionevoli, ma con un dual gap residuo significativo rispetto a SLSQP, come dimostrano i grafici di Dual Objective e Dual Gap. Per migliorare servono tecniche SMO più avanzate passo adattivo, scelta intelligente delle coppie da aggiornare.

REFERENZE

- https://alex.smola.org/papers/2003/SmoSch03b.pdf
- https://www.cc.okayama-u.ac.jp/imelab/pdf/ieeetnn2008accepted.pdf
- https://www.kaggle.com/code/migom6/svm-with-smo-from-scratch
- https://emilemathieu.fr/posts/2018/08/svm/
- https://lucien-east.github.io/2022/07/30/Implement-SVM-with-SMO-from-scratch/
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#scipy.optimize.minimize