Support Vector Regression using Deflected Subgradient Methods

FearEP
Lord Gugger

January 25, 2021

A project presented for the
Computational Mathematics for Learning and Data Analysis
course

University of Pisa
Artificial Intelligence
A.Y. 2020/2021

Abstract

Project aim is developing the implementation of a model which follows an SVR-type approach including various different kernels. The implementation uses as optimization algorithm a dual approach with appropriate choices of the constraints to be dualized, where the Lagrangian Dual is solved by an algorithm of the class of deflected subgradient methods.

1 Introduction

Per affrontare questo problema di regressione ci vogliamo affidare ad un modello di apprendimento supervisionato che è il Support Vector Regression. SVR ha come obiettivo trovare una funzione tale per cui ogni record assegnatoci per il training non devii da essa più di ε (per questo ogni valore all'interno del cosiddetto ε-tube non viene considerato come errore nella fase di ottimizzazione, rendendo la loss del modello ε-insensitive). Per fare ciò abbiamo bisogno di un certo parametro C (per capire il livello di regolarizzazione che desideriamo) ed un valore ε (per esprimere l'errore che accettiamo), oltre ad eventuali parametri necessari ad attuare i kernel (e.g. gamma per quanto riguarda il kernel RBF). Parte fondante del modello, oltre a ciò sopra descritto riguardo l' ε-tube, è dare allo stesso tempo importanza al mantenere la funzione *as flat as possible*, per evitare overfitting ed avere dunque un modello che sia un corretto tradeoff tra accuratezza e generalità.

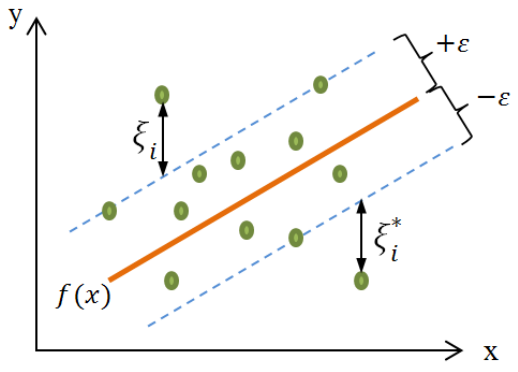

Figure 1: a generic svr

La funzione risultante dall'ottimizzazione del modello è descritta genericamente come:

$$f(x) = wx + b \tag{1}$$

Obiettivo dell'ottimizzazione è dunque fare in modo che la curva sia, di nuovo, *as flat as possible*, ma questo è equivalente ad un problema di ottimizzazione dove vogliamo avere $\|w\|$ minima. Per comodità di formulazione del problema possiamo minimizzare $\|w\|^2$ senza cambiare il significato. Questo ci permette di portarci in un problema di ottimizzazione quadratico, grazie al quale potremo approfittare del concetto di **strong duality** più tardi. Introduciamo a questo punto delle variabili dette *slack* per formulare la *dual objective function*, la quale rappresenta il nostro *primal problem*:

$$\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2}\|w\|^2 + C\sum_i(\xi_i + \xi_i^*) \tag{2}$$

Ciò che viene sommato a $\|w\|$ è un elemento che ci permette di regolare l'errore, e di conseguenza la penalità, dovuti alla possibile presenza di elementi che non rimangono all'interno dell' ε-tube. Vediamo dunque come C funga da regolarizzatore in una metodica simile a L1. I vari ξ vengono detti *slack variables* e ci permettono di definire i vincoli del problema per qualsiasi i-esimo dato:

$$\begin{aligned} &y_i - w^T\phi(x_i) - b \leq \epsilon + \xi_i, \\ &b + w^T\phi(x_i) - y_i \leq \epsilon + \xi_i, \\ &\xi_i, \xi_i^* \geq 0 \end{aligned} \tag{3}$$

x i-esimo input, y i-esimo output

Essendo in un problema di ottimizzazione quadratico, la soluzione al *dual problem* risulta equivalente a quella del *primal problem*. In particolare in questa casistica risulta più facile la risoluzione del *dual problem* vista la possibile applicazione del concetto di kernel. Costruiamo dunque il *dual problem* definendo la relativa Lagrangiana (equivalente del *primal problem* al quale aggiungiamo i vincoli sommandoli o sottraendoli):

$$
\begin{aligned}
L(\alpha, \alpha^*, \mu, \mu^*) = \frac{1}{2} \|w\|^2 & \\
& + C \sum_{i=1}^{m} (\xi_i + \xi_i^*) \\
& + \sum_{i=1}^{m} (\alpha_i (y_i - w^T \phi(x_i) - b - \epsilon - \xi_i)) \\
& + \sum_{i=1}^{m} (\alpha_i^* (w^T \phi(x_i) + b - y_i - \epsilon - \xi_i^*)) \\
\sum_{i=1}^{m} (\mu_i \xi_i + \mu_i^* \xi_i^*) &
\end{aligned}
\tag{4}
$$

A causa del concetto di ***weak duality*** qualsiasi valore del *primal problem* risulta maggiore (o uguale) del *dual problem*. Da questa considerazione deriviamo il fatto che il punto di massima vicinanza tra i 2 problemi è dove il *primal problem* ha minimo e il *dual problem* ha massimo. Nella casistica di *strong duality* questa vicinanza si tramuta uguaglianza. Cerchiamo dunque di rielaborare il *dual problem* per lavorare con meno variabili possibili, in particolare eliminiamo dai calcoli la variabile w, b e le 2 *slack* ξ e ξ^*. In che modo? Stiamo cercando un valore ottimo che tra l'altro sarà unico data la casistica quadratica, dunque la derivata parziale relativa ad ognuna di queste variabili va posta a 0. Questo ci porta a poter ridefinire w:

$$
w = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) \phi(x_i) \tag{5}
$$

Infine tramite sostituzione nella Lagrangiana definita precedentemente arriviamo ad una funzione dipendente solamente da α e α^*:

$$
\begin{aligned}
\max_{\alpha_i, \alpha_i^*} & - \frac{1}{2} \sum_i \sum_j (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i, x_j) \\
& - \epsilon \sum_i (\alpha_i + \alpha_i^*) \\
& + \sum_i y_i (\alpha_i - \alpha_i^*)
\end{aligned}
\tag{6}
$$

Notiamo che le uniche variabili rimaste sono i moltiplicatori lagrangiani α e α^*, i quali sono sottoposti ai seguenti vincoli:

$$\begin{aligned} &\forall i\ \alpha_i, \alpha_i^* \geq 0 && (KKT condition) \\ &\forall i\ \alpha_i, \alpha_i^* \in [0, C] && (from\ deriving\ (6)) \\ &\forall i\ \sum(\alpha_i - \alpha_i^*) = 0 && (from\ deriving\ (6)) \\ &\forall i\ \alpha_i \alpha_i^* = 0 && (from\ model\ construction) \end{aligned} \tag{7}$$

Restando ancora su (6), approfondiamo l'elemento $K(x_i, x_j)$, ovvero ciò rappresenta il concetto di **kernel**, sostituendo ciò che sarebbe un prodotto scalare nello spazio necessitato. Rielaboriamo dunque la precedente frase: certi problemi di regressione non possono essere adeguatamente descritti da modelli lineari. Risulta dunque comodo cambiare spazio di visualizzazione, per portarci in un nuovo spazio (con ogni probabilità con più dimensioni rispetto allo spazio originale) dove il problema diventa linearmente affrontabile. Il cambio di base risulta essere incredibilmente dispendioso per il training del modello ed è proprio in questo caso che il kernel diventa il fulcro dell'efficienza di SVC/SVR. Ci permette infatti di eseguire il *dot-product* nello spazio attuale ma avere come risultato il *dot-product* nello spazio richiesto. Ci permette di risparmiare molte computazioni e di poterle riutilizzare salvandoci i valori risultanti in una *kernel matrix* (riutilizzabile ad ogni ciclo di ottimizzazione). Ovviamente non c'è la certezza di aver preso in considerazione il kernel corretto per la casistica, bisognerà dunque svilupparne vari, magari certi funzioneranno meglio di altri (*rbf, sigmoid, etc*).
In particolare questa implementazione prevede il possibile utilizzo di diversi kernel:

- linear: $\langle x, x' \rangle$
- polynomial: $(\gamma \langle x, x' \rangle + r)^d$
- rbf: $\exp(-\frac{\|x\ x'\|^2}{2\sigma^2})$
- sigmoid: $\tanh(\gamma \langle x, x' \rangle + r)$

Definito il problema possiamo a questo punto cercare il massimo del *dual problem*. La task da noi scelta utilizzerà **deflected subgradient methods**

per raggiungere l'obiettivo. Una volta raggiunto il massimo avremo a nostra disposizione i corretti α e α^* necessari per il calcolo di w.

Per completare la funzione risolutrice del problema ci basta trovare il parametro b. Ricavarlo risulta semplice una volta preso in considerazione un qualsiasi elemento del nostro insieme di input tale per cui la relativa predizione (y i-esimo) sia al limite dell' ε-tube o al di fuori di esso. Proprio quell'insieme di valori sarà infatti l'unico ad avere come vincoli attivi almeno uno tra i vari α e α^*, ovvero un α o α^* diversi da 0. Essendo dunque al margine di un vincolo potremo, derivando da (1) e (5):

$$x_j \; with \; \alpha_j \in (0, C) : b = y_j - \sum_i (\alpha_i - \alpha_i^*) K(x_i, x_j)$$

Una volta trovata la funzione risultante possiamo testare la bontà del modello effettuando *prediction* su un set di test input per poi calcolare la metrica MSE tra output previsto dal modello e output effettivo del set di test input.
Questo fatto è particolarmente importante per il confronto tra modelli e dunque per la ricerca dei parametri migliori possibili (*grid search*).

2 Structures for implementation

- kernel matrix: presi m record di input sarà una matrice simmetrica $K(x, x') \in \mathbb{R}^{m,m}$
- alpha matrix: presi m record di input sarà un vettore $A \in \mathbb{R}^m$ tale per cui $A_i \neq 0$ solo se il relativo input genera un output che contribuisce alla $loss$
- alpha* matrix: analogo ad alpha matrix