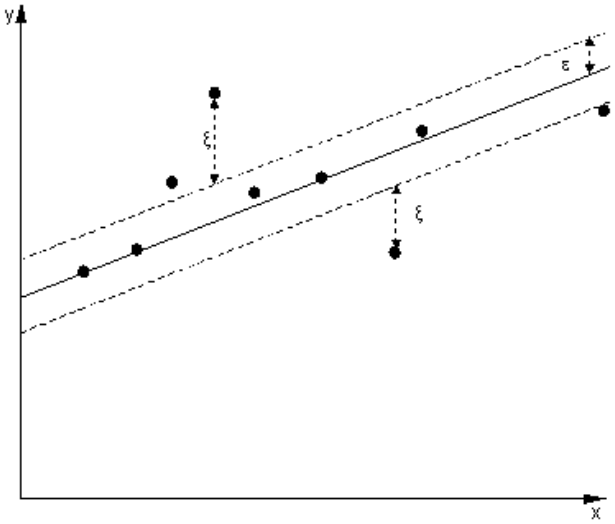
DEFINIZIONE DEL MODELLO E FLOW DI UTILIZZO

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.



La funzione risultante dall'ottimizzazione del modello è descritta genericamente come f(x) = w * x + b

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 della ***STRONG DUALITY*** più tardi.

Introduciamo a questo punto delle variabili dette *slack* per formulare la nostra *objective function,* la quale rappresenta il nostro *primal problem*:

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

Ciò che viene sommato alla norma di 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:

$$\begin{aligned} &y_i - \mathbf{w}^\top\phi(\mathbf{x}_i) - b \le \epsilon + \xi_i, \\ &b + \mathbf{w}^\top\phi(\mathbf{x}_i) - y_i \le \epsilon + \xi_i^*, \\ &\xi_i, \xi_i^* \ge 0 \end{aligned}$$

(x i-esimo rappresenta l'i-esimo input mentre y i-esimo rappresenta l'output che ci si aspetta → **supervised**)

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

$$L(\alpha, \alpha^*, \mu, \mu^*) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) + \sum_{i=1}^{m}\left(\alpha_i(y_i - \mathbf{w}^\top\phi(\mathbf{x}_i) - b - \epsilon - \xi_i) + \alpha_i^*(\mathbf{w}^\top\phi(\mathbf{x}_i) + b - y_i - \epsilon - \xi_i^*\right. \sum_{i=1}^{m}(\mu_i\xi_i + \mu_i^*\xi_i^*)$$

A causa del concetto di WEAK DUALITY qualsiasi valore del *primal problem* risulta >= 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 diventa uguaglianza!

Cerchiamo dunque di rielaborare il problema duale per lavorare con meno variabili possibili, in particolare eliminiamo dai calcoli la variabile w, b e le 2 *slack* 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:

$$\hat{\mathbf{w}} = \sum_{i=1}^{m}(\hat{\alpha}_i - \hat{\alpha}_i^*)\phi(\mathbf{x}_i)$$

Infine tramite sostituzione nella Lagrangiana definita precedentemente arriviamo ad una funzione dipendente solamente dai 2 α:

$$\max_{\alpha_i, \alpha_i^*} -\frac{1}{2}\sum_i\sum_j(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(\mathbf{x}_i, \mathbf{x}_j) - \epsilon\sum_i(\alpha_i + \alpha_i^*) + \sum_i y_i(\alpha_i - \alpha_i^*)$$

Dove K(xi,xj) rappresenta il concetto di KERNEL, sostituendo ciò che sarebbe un prodotto scalare nello spazio necessitato. Rielaboriamo un attimo 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 il kernel giusto, bisognerà svilupparne vari, magari certi funzioneranno meglio di altri (*rbf, sigmoid, etc*).

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 permetterci di attuare la seguente formula, ricavata semplicemente manipolando le formule di vincolo dichiarate precedentemente:

$$\text{Using any } \mathbf{x}_j \text{ with } \alpha_j \in (0, C):\ b = y_j - \sum_i(\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}_j)$$

Una volta trovata la funzione risultante possiamo testare la bontà del modello effettuando *prediction* su un set di test input per poi calcolare la MSE tra l'output previsto dal modello e l'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*).

DUBBI:

- Nella nostra casistica di utilizzo (dataset fornitoci dal corso ML) abbiamo una regression con un output bidimensionale. Il modello SVR sembra ragionato per lavorare con output unidimensionali, altrimenti andrebbero variati anche i concetti e vincoli relativi alle variabili *slack*. L'idea che ci risulta più immediata è quella di dividere l'output bidimensionale in 2 output unidimensionali, a quel punto creare un modello SVR per ogni output e ad ogni *prediction* dunque eseguire l'input sui 2 modelli per infine unire il risultato. Ci sta sfuggendo qualcosa? Il modello è capace di ragionare anche con più dimensioni di output?
- Durante il calcolo della formulazione del problema duale ci siamo resi conto della scomparsa delle variabili *slack*, le quali non risultano presenti nemmeno nella fase di *prediction*. Deriviamo dunque che esse siano fondamentali per la definizione del problema e dei vincoli a livello matematico, ma che effettivamente a livello implementativo risultino poi inutilizzate. Questo fatto è rinforzato dal comprendere come queste variabili *slack* siano strettamente dipendenti dalla funzione risultato e dunque come esse siano impossibili da conoscere a priori. Di nuovo, ci sta sfuggendo qualcosa?