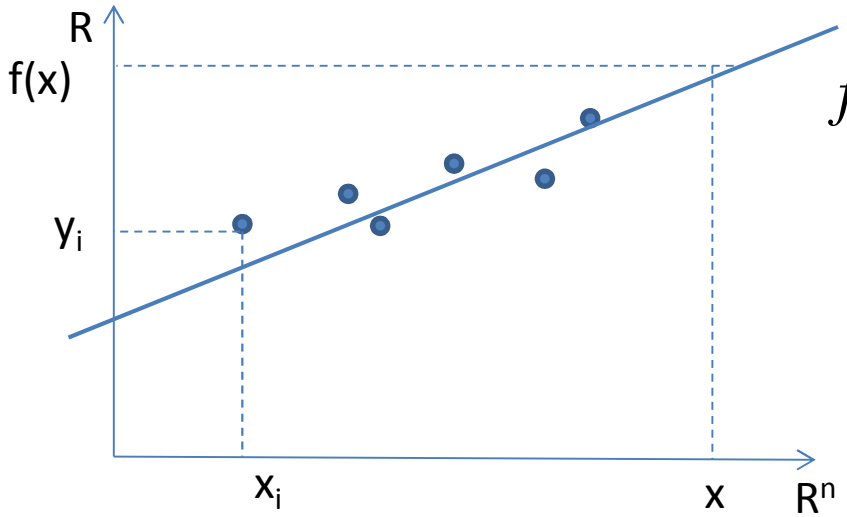
SVRégréssion – Principe et formulation



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

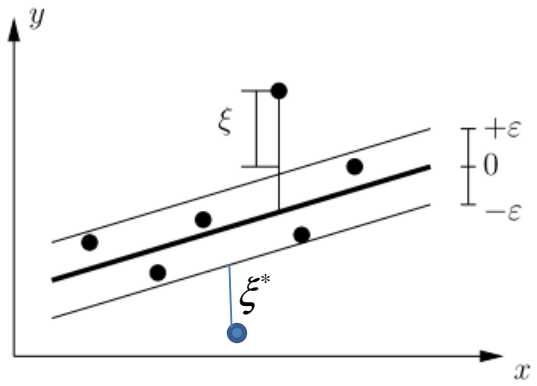
On dispose de p points x_i de R^n (échantillon d'apprentissage)
Par exemple p résultats de runs d'un simulateur

On cherche une fonction de régression (d'approximation, d'évaluation ….) qui « approche au mieux » les points, à un sens qu'on va définir

Dans un premier temps: on cherche f(x) sous la forme d'un hyperplan P(x) (droite sur le dessin)

Puis avec le *kernel trick* qu'on connait déjà, on pourra « tordre » cette « droite » à volonté

SVR – Principe et formulation



- On fixe une marge (// y) de demi-hauteur ε autour de P
- (cf dessin)
- On ne peut pas exiger que tous les points soient
- à l'intérieur de cette marge aussi introduit-on des *variables d'écart positives* vers le haut (ξ_i) et vers le bas (ξ^*_i)

- On veut
- P(x) le plus horizontal possible (robustesse)
  ξ_i, ξ^*_i les plus petits possible
- Pour chaque point $-\varepsilon - \xi^*_i \leq y_i - f(x_i) \leq \varepsilon + \xi_i$

- Problème Primal : on cherche w, ξ_i, ξ^*_i réalisant

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \ ,$$

sous les contraintes

$$\begin{aligned} y_i - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - b &\leq \varepsilon + \xi_i \\ \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b - y_i &\leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0 \ , \ i = 1, \ldots, \end{aligned}$$

p

Lagrangien

$$L(w,b,\xi,\xi^*, \quad \alpha,\alpha^*,\eta,\eta_i^*)$$

Variables primales Variables duales

$$L = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}(\eta_i\xi_i + \eta_i^*\xi_i^*)$$
$$- \sum_{i=1}^{N}\alpha_i(\varepsilon + \xi_i - y_i + \langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b)$$
$$- \sum_{i=1}^{N}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle \boldsymbol{w}, \boldsymbol{x}_i\rangle - b) \ ,$$

Fonction duale

$$H(\alpha,\alpha^*,\eta,\eta_i^*) = \underset{w,b,\xi,\xi^*}{Min}\ L(w,b,\xi,\xi^*, \quad \alpha,\alpha^*,\eta,\eta_i^*)$$

On annule les dérivées de L / variables primales (Min L)

$$\begin{aligned}
\frac{\partial L}{\partial b} &= \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0 \\
\frac{\partial L}{\partial \boldsymbol{w}} &= \boldsymbol{w} - \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)\boldsymbol{x}_i = 0 \\
\frac{\partial L}{\partial \xi_i} &= C - \alpha_i - \eta_i = 0 \\
\frac{\partial L}{\partial \xi_i^*} &= C - \alpha_i^* - \eta_i^* = 0 \ .
\end{aligned}$$

On ré-injecte la valeur de w dans L
H in fine ne dépend que de α

$$\begin{aligned}
L_D = &-\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle \\
&- \varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) \ ,
\end{aligned}$$

sc H(α) = traintes

$$\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \quad \text{et} \quad \alpha_i, \alpha_i^* \in [0, C] \ .$$

On note $\tilde{\alpha}$ le vecteur de $\mathrm{R}^{2p}\begin{pmatrix}\alpha\\ \alpha^*\end{pmatrix}$, pareil pour $\tilde{\eta}, \tilde{\xi}$

$$A = \begin{pmatrix} B & -B \\ -B & B \end{pmatrix}$$

$$\max_{\alpha \in R^{2p}}(H(\tilde{\alpha})) = Max(\tilde{\alpha}^T A\tilde{\alpha} - u_1{}^T\tilde{\alpha} + u_2{}^T\tilde{\alpha})$$

$$u_3^T\tilde{\alpha} = 0, \qquad \tilde{\alpha} \in [0, C]^{2p}$$

$$u_1 = \varepsilon\begin{pmatrix}1\\ \dots\\ 1\end{pmatrix}, u_2 = \begin{pmatrix}Y\\ -Y\end{pmatrix}, u_3 = \begin{pmatrix}1\\ \dots\\ -1\\ \dots\end{pmatrix}$$

L'algorithme d'UZAWA se déroule de la même façon que pour les SVM
Différences: on est dans R^{2p} , et il y a deux termes linéaires dans H au lieu d'un seul

$$\nabla H(\tilde{\alpha}) = A\tilde{\alpha} - u_1 + u_2$$

Algorithme de gradient à pas constant avec projection d'abord sur l'hyperplan $u_3^T\alpha=0$
Puis sur la boite « α dans $[0,C]^{2p}$ »

Détermination de b : comme pour les SVM, on se sert des points supports:

On cherche les indices tels que $\tilde{\alpha}(i) > 0 \quad (> epsilon)$

Suivant que l'indice est plus petit ou plus grand que 2p, on a pour les points supports l'annulation de la contrainte

$$(\varepsilon + \xi_i - y_i + \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) = 0 \qquad \text{ou} \qquad (\varepsilon + \xi_i^* + y_i - \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - b)$$

ξ nous « gène » pour déterminer b , mais on remarque que si $\tilde{\alpha}(i) \neq C$ Alors $\tilde{\eta}(i) \neq 0$

Et donc $\tilde{\xi}(i) = 0$ $b =$

Et pour ces indices là, on peut calculer b, différemment suivant la place de l'indice

et le modèle s'écrit donc

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + b \ .$$

Le kernel trick marche de la même façon que pour les svm, à utiliser dans le calcul de A et de b

On pourra stocker le matériel pour calculer la fonction f, dans un fichier, de façon à ne pas avoir à refaire tourner les SVM pour accéder à f.

On construira une fonction f(x) qui charge ce fichier et calcule f(x). C'est de cette fonction Dont on se servira pour faire de l'optimisation (recherche de la position optimale des antennes)