

Prévision des Séries Chronologiques à l'aide de Support Vector Machines



Hassan OUKHOUYA



Encadrement: Pr.Khalid EL HIMDI

Université Mohammed V
Faculté des Sciences – Rabat
Département de Mathématiques
Laboratoire de Mathématiques, Statistique et Applications



Université Mohammed V
Faculté des Sciences
Rabat



Plan

- 1 Introduction
- 2 SVM pour la classification binaire
 - Éléments de théorie pour les SVMs
 - Minimisation du Risque Empirique (ERM)
 - Théorie de l'Apprentissage de Vapnik
- 3 Formalisation de Support Vector Machines (SVMs)
 - SVM linéaire pour des données séparables
 - SVM linéaire pour des données non séparables
 - SVM non linéaire : astuce du noyau
- 4 SVM pour la régression (SVR)
 - SVR linéaire
 - SVR non linéaire
- 5 La méthode LS-SVM
- 6 La méthode DLS-SVM
- 7 Sources



Qu'est-ce que l'apprentissage ?

En psychologie

Acquisition d'un nouveau comportement à la suite d'un entraînement : habituation, conditionnement...

En neurobiologie

Modifications synaptiques dans des circuits neuronaux : règle de Hebb, règle de Rescorla et Wagner...

Apprentissage Automatique

C'est le processus de **construire un modèle général** à partir de **données** (observations) **particulières** du monde réel.

Ainsi, le **but** est double :

- **Prédire** un comportement face à une nouvelle donnée.
- **Approximer** une fonction ou une densité de probabilité.



Types des problèmes d'apprentissage



- Apprentissage Supervisé.
- Apprentissage par Renforcement.
- Apprentissage Non-Supervisé.
- Apprentissage Semi-Supervisé.

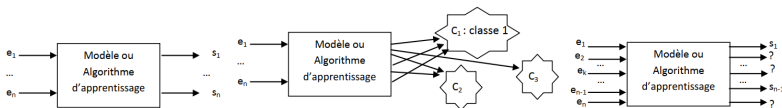



Figure – Schémas pour différents types d'apprentissage

N. B. : Dans notre cas, nous nous concentrerons sur le type d'apprentissage à savoir **supervisé**.



L'interface de **libsvm** dans le package **e1071**

 Récemment, une nouvelle théorie de l'apprentissage statistique, à savoir. Les SVMs sont actuellement un sujet brûlant dans la communauté de l'apprentissage automatique, représentent pourtant une technique puissante pour **la classification** générale (non linéaire), **la régression** et **la détection** des valeurs aberrantes avec une représentation intuitive du modèle.

Le package **e1071** offre une interface vers le $C++$ mise en œuvre par *Chih-Chung Chang* et *Chih-Jen Lin*, **libsvm**, avec :

- ▶ C - et ν -classification.
- ▶ Classification en une seule classe (détection de nouveauté).
- ▶ ϵ et ν -régression.



L'historique de la découverte du SVM

📖 Les SVMs ont été développés par **Vapnik** et ses collègues des laboratoires AT & T Bell en 1995.

📍 Initialement, les SVMs ont été conçus pour résoudre des problèmes de classification de motifs, tels que la reconnaissance optique de caractères, l'identification de visage et la classification de texte, etc. Mais ils ont rapidement trouvé de larges applications dans d'autres domaines, tels que la fonction problèmes d'approximation, d'estimation de régression et de **prévision de séries chronologiques.**

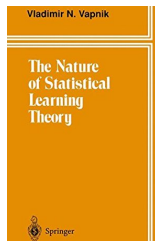


Figure – Vladimir Naumovitch Vapnik, The Nature of Stat. Lea. The.

L'approche du SVM

⚡ L'approche SVM basé en principe sur trois scénarios :

- Séparation des classes.
- Classes qui se chevauchent(Overlapping Classes).
- Non-linéarité.

⇒ Solution du problème.

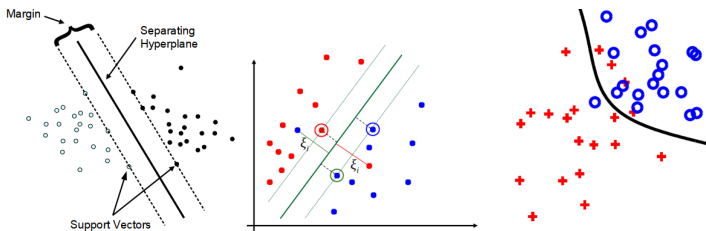


Figure – Classifications



Formalisme probabiliste pour la classification binaire

- Soit un ensemble d'apprentissage $S = \{(x_i, y_i)\}_{1, \dots, N}$ tel que ;

- ▶ $x_i \in X \subseteq \mathbb{R}^n$ **observations**.
- ▶ $y_i \in D = \{0, 1\}$ ou $\{-1, +1\}$ **classe** (ou label ou étiquette) binaire.
 - si $D = \mathbb{R}$ on a un problème de régression.

Dont les éléments obéissent à la loi jointe tel que :

$$P(x, y) = P(x)P(y|x) \quad (X, D) \text{ sont i.i.d}$$

On cherche à approcher une loi sous-jacente $f : X \rightarrow D$ telle que $y_i = f(x_i)$ par une hypothèse $h_w(x)$ aussi proche que possible, Où les w sont les paramètres du système d'apprentissage.

Mais que veut-on dire par : “ aussi proche que possible ” ?



Remarques

- - Si $f()$ est **discrète** (catégorie) comme rouge et bleu ou maladie et pas de maladie..., on parle de **classification**.
- - Si $f()$ est une **fonction continue** (valeur réelle) tel que dollars ou poids..., on parle alors de **régression**.

!!! Pour les **séries chronologiques** peuvent être formulées comme un **apprentissage supervisé**. Nous pouvons le faire en utilisant les pas de temps précédents comme variables d'entrée(X) et en utilisant le pas de temps suivant comme variable de sortie(D).



Calcul du risque

Pour mesurer la qualité d'une hypothèse h_w on considère généralement une **fonction de coût** ou **perte** (loss function)

$$L(y_t, f(x_t, w)) = \begin{cases} 0 & \text{Si } y_t = f(x_t, w) \\ 1 & \text{Sinon.} \end{cases}$$

On souhaite à minimiser cette distance L entre la cible y_t et la prédiction $f(x_t)$.

Exemples de fonction de coût

- **Coût 0/1** : Ce type de coût est utilisé en cas de la classification.
- **Erreur quadratique** : $(y_t - f(x_t))^2$ Elle est utilisée particulièrement en régression.

Ainsi, on cherche à **minimiser** la fonction **risque** (ou **d'erreur**) :

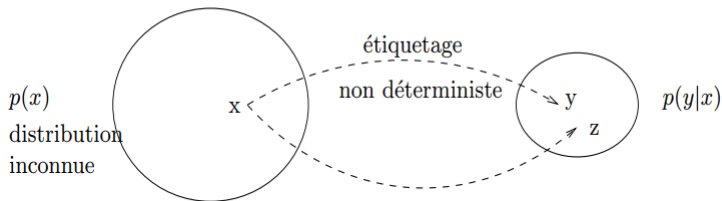
$$\begin{aligned} R(f) &= \int L(y_t, f(x_t, w)) dP(x_t, y_t) = \int_{y_t \neq f(x_t)} dP(x_t, y_t) \\ &= P(y_t \neq f(x_t)) \end{aligned}$$



Récapitulatif

X : domaine des descriptions

Y : ensemble des classes



$S = \{(x_1, y_1), \dots, (x_l, y_l)\}$ tiré selon $p(x, y) = p(x)p(y|x)$

Objectif : trouver $f : X \rightarrow Y$ dont l'erreur $R(f) = P(y \neq f(x))$ soit la plus petite possible.

Figure – Schéma d'illustration



Principe de Minimisation du Risque Empirique (ERM)

Étant donnée la distribution de probabilité $P(x, y)$ des données est **inconnue** et le risque attendu. On ne peut pas accéder directement à cette valeur.

Ce problème critique motive Vapnik à suggérer le principe de minimisation des risques empiriques (ERM). On construit donc le risque empirique d'une fonction f sur l'échantillon $S = \{(x_i, y_i)\}_{i=1}^N$ est la **moyenne de la fonction de perte** calculée sur S et qui mesure les **erreurs d'apprentissage** réalisées par le modèle :

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i, w)) \quad (1)$$

♠ $R_{emp}(f)$ est une **estimation** du risque réel $R(f)$ de f .



Principe de Minimisation du Risque Empirique (ERM)

- ★ **En Classification** : $R_{emp}(f)$ est la moyenne du nombre d'erreurs de prédiction de f sur les éléments de S :

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^N \mathbb{I}_{[f(x_i) \neq y_i]} = \frac{\text{Card}\{i | f(x_i) \neq y_i\}}{N}$$

- ★ **En Régression** : $R_{emp}(f)$ est la moyenne des carrés des écarts à la moyenne de f sur S :


$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2$$

- ★ **En Estimation de densité** : $R_{emp}(f)$ est l'opposé de la log-vraisemblance de S :

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^N -\log f(x_i) = -\frac{1}{N} \log \prod_{i=1}^N f(x_i)$$



Principe de Minimisation du Risque Empirique (ERM)

 **Problème de prévision** : Dorénavant, on peut reformuler le problème de prévision comme suit :

Il s'agit de trouver, à l'aide des données S , un prédicteur $f \in \mathcal{F}^1$ tel que son risque $R_{emp}(f)$ est minimal.

On dit que le principe ERM est consistant dans la classe \mathcal{F} si $R(f)$ tend vers $R_{emp}(f)$ lorsque la taille de S tend vers l'infini.

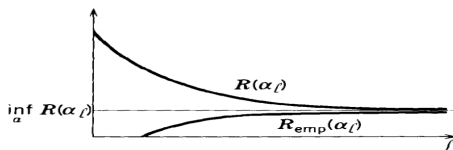


Figure – le risque $R(f)$ et le risque $R_{emp}(f)$ convergent vers la valeur minimale possible du risque.

1. Les classes d'estimateurs ou la famille de fonctions de décision ou l'ensemble des prédicteurs $\mathcal{F} = \{f(x, w)\}$.



Inconvénients de l'ERM

Inconvénients de l'ERM

- Algorithmique : problème NP-difficile.
- Contrôle de la **complexité** : propriété de Glivenko-Cantelli pour éviter le sur-apprentissage (**overfitting**).
- La non consistance et la difficulté du principe ERM ou plus généralement (le principe choisi pour approcher une fonction optimale dans \mathcal{F}).

La théorie de Vapnik-Chervonenkis (VC) permet d'anticiper, d'étudier et de réguler les phénomènes liés au sur-apprentissage.



Dimension de Vapnik-Chervonenkis (VC)

Dans la théorie de l'apprentissage automatique, la dimension - VC est une mesure de la capacité d'un algorithme de classification statistique ; elle est définie comme le **cardinal du plus grand ensemble de points** que l'algorithme peut **pulvériser**.

Donc mathématiquement,

$$h = \max\{|X| \mid \forall b \in \{-1, 1\}^{|X|}, \exists f \in \mathcal{F} \text{ tq } f(x_i) = b_i\} \quad (2)$$

Sachant que, $X \subseteq \mathbb{R}^n$, $\forall x_i \in X$ ($1 \leq i \leq N$).

C'est un concept défini par Vladimir Vapnik et Alexey Chervonenkis, il est central dans la théorie de Vapnik-Chervonenkis.



Il y a 2^N différents problèmes d'apprentissage qui peuvent être définis, car N points peuvent être étiquetés dans 2^N manières positives ou négatives. par exemple, pour trois points, il y a 24 étiquettes différentes et 8 différentes limites de classification qui peuvent être apprises.

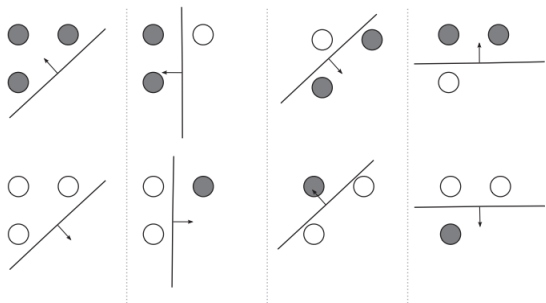


Figure – différentes limites de classification qui peuvent être apprises



Théorie de l'Apprentissage de Vapnik (1995)

Minimisation Structurale du Risque (SRM)

Théorème

Soit \mathcal{F} classes d'estimateurs de dimension VC ($d_{VC} = h$). Alors $\forall \{x_i, y_i\}_{i=1}^N$, avec $x_i \in X \subseteq \mathbb{R}^n$, $y_i \in D \subseteq \mathbb{R}$ avec une probabilité au moins égale à $1 - \eta$ ($0 \leq \eta \leq 1$) :

$$R(f) \leq R_{emp}(f) + \sqrt{\frac{h(\ln(\frac{2N}{h}) + 1) - \ln(\frac{\eta}{4})}{N}} \quad (3)$$

♣ *Le second terme est l'intervalle de confiance VC et $1 - \eta$ est appelé le niveau de confiance.*

Attention



Cette formule n'est valide que lorsque $d_{VC} = h < N$.



La minimisation du risque dépend :

- * Du **risque empirique**.
- * De **Risque Structural de Minimisation** lié au terme d_{VC} (dimension - VC) qui dépend de la complexité du modèle h choisi.

Conditions de Construction

Ainsi, pour construire un bon modèle d'apprentissage, il est nécessaire de :

- ◇ **Minimiser les erreurs** sur la base d'apprentissage, c'est le principe d'induction naïf utilisé, par exemple, dans les réseaux de neurones.
- ◇ Construire un système, dont la capacité de **généralisation** (qualité de **prévision**) est la plus grande possible.



La validation croisée (cross-validation)

La méthode la plus simple pour éviter le phénomène liés au **sur-apprentissage** est de partager la population d'apprentissage en deux sous-ensembles. Le premier sert à l'apprentissage et le deuxième sert à l'évaluation de l'apprentissage. Tant que l'erreur obtenue sur le deuxième ensemble diminue, on peut continuer l'apprentissage, sinon on l'arrête.

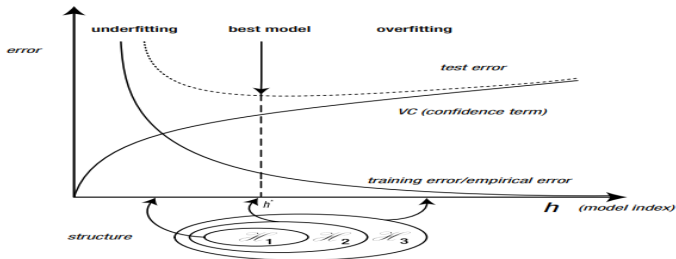


Figure – Erreur de prédiction sur les ensembles d'apprentissage et de test, en fonction de la complexité du modèle.



Notions Élémentaires

* **Produit scalaire** entre deux vecteurs sur \mathbb{R}^n :

$$\forall u, v \in \mathbb{R}^n, \quad \langle u, v \rangle = \langle v, u \rangle = u^T v = \sum_{i=1}^n u_i v_i,$$

Où $u = (u_1, \dots, u_n)^T$ et $v = (v_1, \dots, v_n)^T$.

|| **Norme euclidienne** sur \mathbb{R}^n :

$$\forall u \in \mathbb{R}^n, \quad \|u\| = \sqrt{\langle u, u \rangle} = \sqrt{u^T u} = \sqrt{\sum_{i=1}^n u_i^2}.$$

⚡ ||| **Distance euclidienne** :

$$\forall u, v \in \mathbb{R}^n, \quad d(u, v) = \|u - v\| = \sqrt{\langle u, u \rangle + \langle v, v \rangle - 2 \langle u, v \rangle}$$



Quelques propriétés

↗ **Transformation non-linéaire** : $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^n$ avec $n > d$.

ℙ **Fonction Noyau** : $k(x, x') = \langle \phi(x), \phi(x') \rangle$.

d_ϕ **Distance image** :

$$d_\phi(x, x') = \|\phi(x) - \phi(x')\| = \sqrt{k(x, x) + k(x', x') - 2k(x, x')}$$

⇒ La distance induite par ϕ ne fait intervenir que le noyau.

⊥ $\forall w \in \mathbb{R}^n$, $w^* = \frac{w}{\|w\|}$ est le vecteur normal à \mathcal{H} .

⊥ $\forall x_0 \in \mathcal{H}$, $\langle w, x_0 \rangle = -b$.

↔ **la distance signée** (éventuellement négative !) d'un point $x \in \mathbb{R}^n$ à \mathcal{H} est donnée par :

$$d(x, \mathcal{H}) = \langle w^*, x - x_0 \rangle = \frac{1}{\|w\|} (\langle w, x \rangle + b)$$

Où $x_0 \in \mathcal{H}$, $b \in \mathbb{R}$.



Séparateurs linéaires

- ✓ **Forme des fonctions de décision :**

$$f(x, w, b) = w^T x + b = w \cdot x + b,$$

Où $b \in \mathbb{R}$, $x, w \in \mathbb{R}^n$.

- ✓ L'équation $f(x) = 0$ définit un **hyperplan séparateur** \mathcal{H} dans \mathbb{R}^n .
- ✓ **Classifieur associé** : $g_f : \mathbb{R}^n \rightarrow \{-1, +1\}$

$$\forall x \in \mathbb{R}^n, \quad g_f(x) = \begin{cases} +1 & \text{Si } f(x) > 0 \\ -1 & \text{Si } f(x) \leq 0 \end{cases}$$

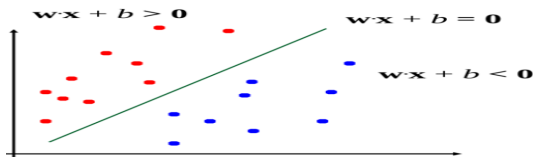


Figure – Classifieur Linéaire simple



Mais, puisqu'il existe de nombreux choix possibles pour w et b :

$$g_f(x) = \text{sign}(w \cdot x + b) = \text{sign}(kw \cdot x + k \cdot b) \quad \forall k \in \mathbb{R}.$$

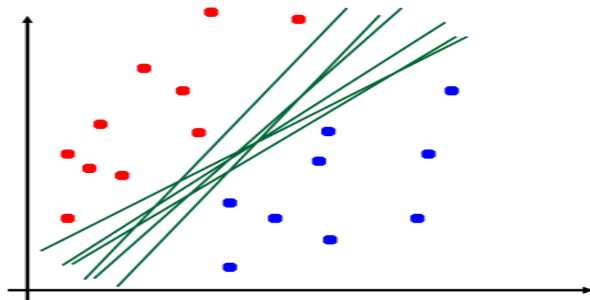


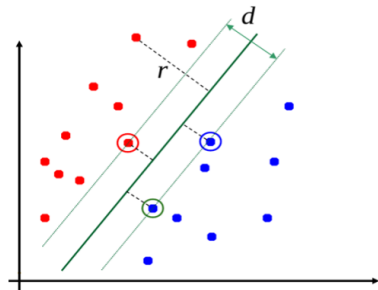
Figure – Infinité des hyperplans séparateurs linéaires.

Problème de choix : Quel est le meilleur classifieur qui "généralise" bien ?



Hyperplan séparateur, notion de marge :

Dans le cas linéairement séparable, on va considérer les **points les plus près de l'hyperplan séparateur** (Optimal) appelés : **vecteurs supports** (support vectors). Pour tout point de l'espace des exemples, la distance entre un point x et l'hyperplan séparateur est donnée par :



$$r = \frac{|w \cdot x + b|}{\|w\|} = \frac{|f(x)|}{\|w\|} \quad (4)$$


Ainsi,

On appelle **marge** $d = d_+ + d_-$ la distance entre les 2 classes.

C'est cette distance **d** qu'on souhaiterait **maximiser**.



Quantification de la Marge dure (hard-margin)

 Pour limiter l'espace des "*possibles*", on ne considère que les points les plus proches, ceux qui sont situés sur les **hyperplans canoniques** donnés par :

$$w^T x + b = \pm 1 \quad (5)$$

Dans ce cas, la marge est :

$$d = d_+ + d_- = \frac{1}{\|w\|} + \frac{1}{\|w\|} = \frac{2}{\|w\|} \quad (6)$$

Ainsi, les conditions d'une bonne classification seront :

$$\begin{cases} w^T x + b \geq 1 & \text{Si } y_i = 1 \\ w^T x + b < 1 & \text{Si } y_i = -1 \end{cases}$$



Maximisation de la marge d'un classifieur

SVM vise à trouver $w \in \mathbb{R}^n$ et $b \in \mathbb{R}$ tels que le problème se réécrit comme un d'optimisation sous contraintes alors :

$$\left\{ \begin{array}{l} \max_{w,b} d = \frac{2}{\|w\|} \\ \text{Avec } y_i(w^\top x_i + b) \geq 1 \quad \forall i \in [1, n] \end{array} \right.$$

La formulation « classique » des SVMs s'obtient alors en minimisant $\|w\|^2$ au lieu de maximiser l'inverse de la norme, ce qui donne le **problème primal** suivant :

Problème d'optimisation avec contraintes

$$(\mathcal{P}) \quad \left\{ \begin{array}{l} \min_{w,b} J(w) = \frac{1}{2} \|w\|^2 \\ \text{Sous les contraintes : } y_i(w^\top x_i + b) \geq 1, \quad \forall i \in [1, n]. \end{array} \right.$$



○ De manière équivalent, pour résoudre le problème précédent, revient à résoudre le **problème dual** qui est aussi un programme quadratique :

Problème d'optimisation quadratique (QPP)

Ce problème d'optimisation sous contraintes est un programme quadratique de la forme :

$$\min_z \frac{1}{2} z^T A z - d^T z \quad (7)$$

Avec $Bz \leq e$

Où $z = (w, b)^T \in \mathbb{R}^{n+1}$, $d = (0, \dots, 0)^T \in \mathbb{R}^{n+1}$, $A = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$,

I est la matrice identité de \mathbb{R}^n , $B = -[\text{diag}(y)X, y]$,

$e = -(1, \dots, 1)^T \in \mathbb{R}^n$, $y \in \mathbb{R}^n$ le vecteur des signes des observations et X la matrice $n \times d$ des observations dont la ligne i est le vecteur x_i^T .

😊 Ce problème est **convexe** (A semi défini positive). Il admet donc une **solution unique** (optimum global).



Conditions d'optimalité et vecteurs supports

La minimisation de (\mathcal{P}) est possible sous les conditions dites de "Karush-Kuhn-Tucker (KKT)" :

Méthode des multiplicateurs de Lagrange, Kuhn-Tucker (1951)

Dans le cas des SVM le lagrangien \mathcal{L} s'écrit :

$$\mathcal{L}(w, b, \lambda) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \lambda_i [y_i(w \cdot x_i + b) - 1] \quad (8)$$

Où les $\lambda_i \geq 0$ sont les multiplicateurs de Lagrange associés aux contraintes. Les conditions de KKT sont alors :

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \ ; \ \frac{\partial \mathcal{L}}{\partial b} = 0 \ ; \ \frac{\partial \mathcal{L}}{\partial \lambda_i} \geq 0 \quad \text{Avec } \lambda_i \geq 0,$$

$$\lambda_i [y_i(w \cdot x_i + b) - 1] = 0, \quad \forall i \in \{1, \dots, n\}.$$



- ≡ Les *vecteurs support* sont les vecteurs x_i pour lesquels la contrainte est active (ou saturée), c'est-à-dire les plus proches du plan, et vérifiant donc :

$$y_i(w \cdot x_i + b) = 1, \text{ le } \lambda_i \text{ est nul.}$$

- ⊗ Les conditions d'annulation des dérivées partielles du lagrangien permettent d'écrire les relations que vérifient le plan optimal, avec les λ_i^* non nuls seulement pour les points supports :

$$\begin{cases} \frac{\partial \mathcal{L}(w, b, \lambda)}{\partial w} = w - \sum_{i=1}^n \lambda_i y_i x_i = 0 \Leftrightarrow w^* = \sum_{i=1}^n \lambda_i^* y_i x_i. \\ \frac{\partial \mathcal{L}(w, b, \lambda)}{\partial b} = - \sum_{i=1}^n \lambda_i y_i = 0 \Leftrightarrow \sum_{i=1}^n \lambda_i^* y_i = 0. \end{cases}$$

- ☞ Ces contraintes d'égalité permettent d'exprimer la formule duale du lagrangien :

$$W(\lambda) = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j=1}^n \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle.$$



Le problème primal & Le problème dual de SVM

La solution du **problème d'optimisation primal** est donnée par :

$$\spadesuit \quad w^* = \sum_{i=1}^n \lambda_i^* y_i x_i$$

$$\spadesuit \quad b^* = \frac{-1}{2} \{ \min_{y_i=1} \langle w^*, x_i \rangle + \min_{y_i=-1} \langle w^*, x_i \rangle \}$$

λ^* est la solution du **problème d'optimisation dual** de SVM est :

$$\begin{aligned} & \max_{\lambda} W(\lambda) \\ \text{s.c.} \quad & \sum_{i=1}^n \lambda_i y_i = 0, \quad \lambda_i \geq 0 \quad \forall i. \end{aligned} \quad (9)$$

☺ La résolution de ce problème dual, fournit La fonction de décision de l'hyperplan optimal est donnée par l'équation :

$$y(x) = \text{sgn} \left(\sum_{i=1}^n \lambda_i^* y_i \langle x, x_i \rangle + b^* \right).$$

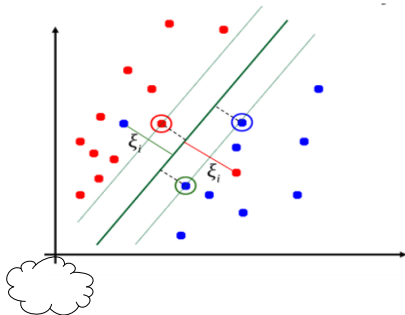
🦄 Pour une nouvelle observation x non apprise présentée au modèle, est classée dans l'une des deux classes selon le signe de $y(x)$.



Cas non séparable

Dans le cas où :

- ⚡ Méthode précédente non applicable si les données **ne sont pas linéairement séparables**.
- ⚡ Méthode très sensible aux " **outliers** ".



L'IDÉE est d'**ajouter** des **variables d'ajustement** ou les **variables ressort (slacks)** ξ_i dans la formulation pour prendre en compte les erreurs de classification ou le bruit.



Classification à marge souple (marge relaxée) :

Il est possible de suivre la même démarche adoptée dans le cas séparable au prix de l'introduction de variables d'écart ξ_i , qui en contrôlent le dépassement :

$$y_i < w, x_i > +b \geq 1 - \xi_i \quad \text{avec } \xi_i \geq 0, \forall i \in \{1, \dots, n\}$$

Problème d'optimisation primal

Le problème de minimisation est réécrit en introduisant une **pénalisation** ($C > 0$, fixé) par le dépassement de la contrainte :

$$\begin{cases} \min J(w, b, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \\ \text{Avec } \forall i, y_i < w, x_i > +b \geq 1 - \xi_i \\ \forall i \quad \xi_i \geq 0 \end{cases}$$

Comme nous l'avons fait précédemment, La solution est fournie par un point-selle $(w^*, b^*, \alpha^*, \beta^*)$ du lagrangien : $\mathcal{L}(w, b, \alpha, \beta)$

$$= \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i (y_i (w^\top x_i + b) - 1 + \xi_i) - \sum_{i=1}^n \beta_i \xi_i.$$


⦿ Avec des multiplicateurs de Lagrange $\alpha_i \geq 0$ et $\beta_i \geq 0$. Le calcul des gradients est identique par rapport à w et b . la dérivée partielle du lagrangien par rapport aux variables d'écart s'écrit : $\partial_{\xi_i} \mathcal{L}(w, b, \alpha, \beta) = C - \alpha_i - \beta_i$. Cette condition supplémentaire de stationnarité permet d'éliminer les β car :

$$\beta_i \geq 0 \text{ et } C - \alpha_i - \beta_i = 0 \Rightarrow \alpha_i \leq C.$$

Problème d'optimisation dual

Sous la forme dual, la formulation SVM à marge souple est :

$$\begin{cases} \max_{\alpha} \left(\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i x_j \right) \\ \text{s.c. } \sum_{i=1}^n \alpha_i y_i = 0 \text{ et } 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, n \end{cases}$$

Remarque : Le problème dual à marge souple est équivalent au problème dual à marge dure, sauf que la variable dual est délimitée par le paramètre de régularisation C .

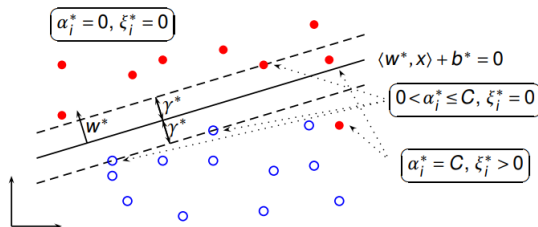


Représentation des vecteurs supports

Les x_i tels que $\alpha_i^* > 0$ sont les vecteurs supports.

Υ Deux types de vecteurs supports :

- 1 Les vecteurs correspondant à des variables ressort nulles. Ils sont situés sur les frontières de la région définissant la marge.
- 2 Les vecteurs correspondant à des variables ressort non nulles : $\xi_i^* > 0$ et dans ce cas $\alpha_i^* = C$.



≡ Les vecteurs qui ne sont pas supports vérifient $\alpha_i^* = 0$ et $\xi_i^* = 0$.



Séparateur non linéaire

◁ Exemple de données difficiles à discriminer linéairement :

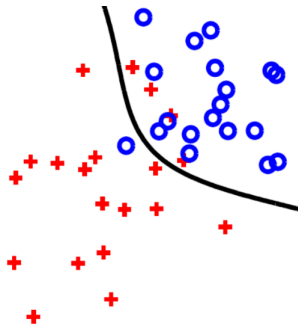


Figure – Classification non linéaire

↪ Une SVM linéaire donnera une très mauvaise discrimination avec un nombre de vecteurs supports très élevé \Rightarrow SVM non linéaire ?



Du non-linéaire au linéaire

✓ **L'idée** (Boser, Guyon, Vapnik, 1992) : L'espace des données peut toujours être plongé dans un espace de plus grande dimension dans lequel les données peuvent être séparées linéairement.

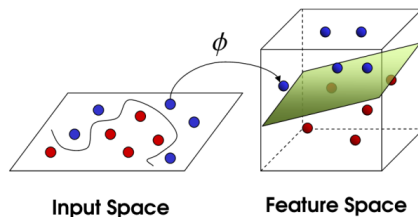


Figure – Rôle de l'espace intermédiaire dans la séparation des données

- La fonction ϕ est appelée la **fonction de représentation (feature function)**.
- L'espace \mathcal{H} est appelé l'**espace de représentation (feature space)**.



Mais il y a un problème...

- ⚡ La transformation vers un espace de dimension supérieure peut nécessiter beaucoup de calculs ???
- ⚡ Comment peut-on choisir ϕ et \mathcal{H} ?

Le " **kernel trick** "

Soit \mathcal{X} l'espace où vivent les observations.



Le noyau est une fonction k qui associe à tout couple d'observations (x, x') est défini de manière générale comme une fonction de deux variables sur \mathbb{R} :

$$k : \mathcal{X}, \mathcal{X} \rightarrow \mathbb{R} \quad (x, x') \mapsto k(x, x')$$

😊 la connaissance seule de la fonction k définie par :

$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$ permet de lancer SVM dans \mathcal{H} , sans déterminer explicitement ϕ et \mathcal{H} .



Comment savoir si K est un noyau ?

Noyau défini positif (n.d.p.)

Une fonction $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ est un **n.d.p.** si et seulement si :

- ☐ k est **symétrique** : $k(x, x') = k(x', x)$, $\forall x, x' \in \mathcal{X}$
- ☐ k est **positive** :

$$\sum_{i=1}^n \sum_{j=1}^n c_i c_j k(x_i, x_j) \geq 0, \quad \forall c_i, c_j \in \mathbb{R}, \quad \forall x_i, x_j \in \mathcal{X}, \quad \forall n \geq 1$$

Théorème de Mercer (1909)

Pour tout noyau positif k sur \mathcal{X} il existe un espace de Hilbert \mathcal{H} et une application ϕ tels que :

$$k(x, x') = \langle \phi(x), \phi(x') \rangle, \quad \forall x, x' \in \mathcal{X}$$

Où \langle, \rangle représente le produit scalaire sur \mathcal{H} .



Construire un noyau



Le théorème de Mercer est non constructif : il ne fournit ni \mathcal{H} , ni ϕ .

Matrice de Gram

La matrice de Gram du noyau $k(\cdot, \cdot)$ pour les observations $\{x_i\}_{i=1}^n$ est la matrice **carrée** K de taille n et de terme général $K_{ij} = k(x_i, x_j)$. Elle est **symétrique** et **semi-définie positive** pour un noyau de Mercer.

Par exemple, si l'on considère $x = (x_1, x_2)$ dans \mathbb{R}^2 et $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$ est explicite. Dans ce cas, \mathcal{H} est de dimension 3 et le produit scalaire dans \mathbb{R}^3 s'écrit :

$$\begin{aligned} \langle \phi(x), \phi(x') \rangle &= x_1^2 x_1'^2 + 2x_1 x_2 x_1' x_2' + x_2^2 x_2'^2 = (x_1 x_1' + x_2 x_2')^2 \\ &= (x^\top x')^2 = (\langle x, x' \rangle)^2 = k(x, x') \end{aligned}$$

\Rightarrow Noyau polynomial de degré 2.



Quelques noyaux classiques pour $\mathcal{H} = \mathbb{R}^n$



Ci-dessous, nous présentons quelques noyaux bien connus utilisés dans la littérature SVM :

Type	Nom	$k(x, y)$
Projectif	Polynômial	$(x^\top y)^p$
Projectif	Linéaire	$x^\top y$
Projectif	Affine	$(x^\top y + \sigma)^p$
Projectif	Corrélation	$\exp(\frac{x^\top y}{\ x\ \ y\ } - \sigma)$
Radial (RBF)	Gaussien	$\exp(-\frac{\ x-y\ ^2}{2\sigma^2})$
Radial (RBF)	Laplacien	$\exp(-\frac{\ x-y\ }{\sigma})$
Radial (RBF)	Rationnel	$1 - \frac{\ x-y\ ^2}{\ x-y\ ^2 + \sigma}$
Non Stationnaire	χ^2	$\exp(-(\sum_k \frac{(x_k - y_k)^2}{x_k + y_k}))$



SVM non-linéaire : formulation

📖 La grande flexibilité dans la définition des **noyaux**, permettant de définir un noyau plus ou moins exotique et adapté à une problématique posée, à condition bien sûr de construire et tester le bon noyau. D'où apparaît encore l'importance de correctement évaluer des erreurs de **prévision** par exemple par validation croisée.

La formulation primal du noyau SVM est :

- $\min_{w, \xi} \frac{1}{2} w^T w + C \sum_{i=1}^n \xi_i,$
- Avec $\forall i, y_i(w^T \phi(x_i) + b) \geq +1 - \xi_i$ et $\xi_i \geq 0$
- Où $\phi(x_i)$ tel que $K(x_i, x_j) = \phi(x_i) \phi(x_j).$

🔁 Encore une fois, la solution SVM doit satisfaire aux conditions KKT, comme suit :



- $w = \sum_{i=1}^n \alpha_i y_i \phi(x_i).$
- ◻ $\sum_{i=1}^n \alpha_i y_i.$
- ◼ $C - \alpha_i - \beta_i = 0, \forall i = 1, \dots, n.$
- ◼◼ $\alpha_i [y_i(w^\top \phi(x_i) + b) - 1 + \xi_i], \forall i = 1, \dots, n.$
- ◼◼ $\beta_i \xi_i = 0, \forall i = 1, \dots, n.$
- ◼◼◼ $\beta_i, \xi_i > 0, \forall i = 1, \dots, n.$

Comme mentionné précédemment, la formulation dual de ce problème est plus efficace à résoudre et utilisée dans la plupart des implémentations de SVM :

Problème dual

$$\max_{\alpha} \left(\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j) \right),$$

$$\text{s.t. } \sum_{i=1}^n \alpha_i y_i = 0 \text{ et } 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, n.$$

La solution est de la forme :

$$w^* = \sum_i \alpha_i y_i \phi(x_i) \text{ et } f(x) = w^* \cdot \phi(x) + b^* = \sum_i \alpha_i y_i K(x_i, x) + b^*.$$



Présentation de SVR



Lorsque les SVMs sont utilisés dans des problèmes de régression pour **prédire des valeurs réelles** $y_i \in \mathbb{R}$, on parle des **SVR** (Support Vector Regression). Toutes les concepts et les autres variables auront la même signification comme en cas de problèmes de classification. Dans le cas non linéaire, le **principe** consiste à **rechercher une estimation (marge de tolérance (ε)) de la fonction** par sa décomposition sur une base fonctionnelle.

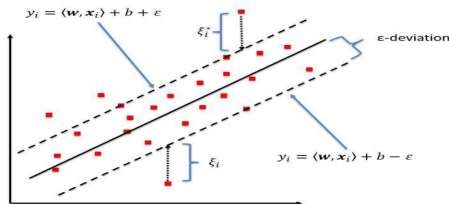


Figure – SVR linéaire unidimensionnel



💎 L'idée principale est toujours la même : **minimiser l'erreur** entre la valeur prédite de la fonction pour une entrée donnée et la sortie réelle, **individualiser l'hyperplan qui maximise la marge**, en gardant à l'esprit qu'une partie de **l'erreur est tolérée**.

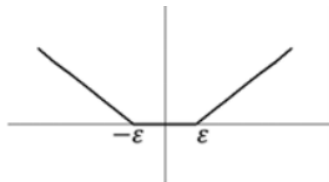
Afin d'obtenir de la **parcimonie** (et espérer un petit nombre de vecteurs support), il faut modifier le terme d'attache aux données.

Parmi toutes les solutions envisageables, celle retenue dans le cadre de la support vector regression,

SVR adopte une **fonction de perte**

ε -*insensible* (rend le modèle plus **robuste**), pénalisant les prédictions plus éloignées qu' ε de la sortie souhaitée. Vapnik a utilisé la fonction de perte linéaire ε -*insensible* définie par :

$$L_{\varepsilon}(y_t, f(x_t, w)) = \begin{cases} 0 & \text{Si } |y_t - f(x_t, w)| \leq \varepsilon \\ |y_t - f(x_t, w)| - \varepsilon & \text{Sinon.} \end{cases}$$



SVR linéaire (unidimensionnel)

Ensuite, comme d'habitude, le risque empirique à minimiser est donné par :

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^N L_{\varepsilon}(y_i, f(x_i, w)) \quad (10)$$

🔒 Le QPP associé peut être écrit comme suit :

$$\left\{ \begin{array}{ll} \min J(w, \xi, \xi^*) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i + \xi_i^* & \\ \text{Contraintes : } y_i - w^{\top} x_i \leq \varepsilon + \xi_i^* & \forall i = 1, \dots, N, \\ w^{\top} x_i - y_i \leq \varepsilon + \xi_i & \forall i = 1, \dots, N, \\ \xi_i, \xi_i^* \geq 0, & \forall i = 1, \dots, N \end{array} \right.$$



Par suite,

$$\mathcal{L}(w, \xi^*, \xi, \lambda, \lambda^*, \alpha, \alpha^*) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*)$$

$$+ \sum_{i=1}^N \alpha_i (y_i - w^\top x_i - \varepsilon - \xi_i^*) + \sum_{i=1}^N \alpha_i^* (-y_i + w^\top x_i - \varepsilon - \xi_i) - \sum_{i=1}^N (\lambda_i \xi_i + \lambda_i^* \xi_i^*).$$

Avec les multiplicateurs de Lagrange $\lambda, \alpha \geq 0$. Sur la base des conditions de Karush-Kuhn-Tucker (KKT). Les dérivées partielles par rapport aux multiplicateurs de Lagrange renvoient les contraintes, qui doivent être inférieures ou égales à zéro :

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w} &= w - \sum_{i=1}^N (\alpha_i^* - \alpha_i) x_i = 0 \text{ et } \frac{\partial \mathcal{L}}{\partial \xi_i^*} = C - \lambda_i^* - \alpha_i = 0 \quad \forall i \\ \frac{\partial \mathcal{L}}{\partial \xi_i} &= C - \lambda_i - \alpha_i^* = 0 \text{ et } \frac{\partial \mathcal{L}}{\partial \lambda_i} = \sum_{i=1}^N \xi_i \leq 0 \text{ et } \frac{\partial \mathcal{L}}{\partial \lambda_i^*} = \sum_{i=1}^N \xi_i^* \leq 0 \\ \frac{\partial \mathcal{L}}{\partial \alpha_i} &= y_i - w^\top x_i - \varepsilon - \xi_i^* \leq 0 \text{ et } \frac{\partial \mathcal{L}}{\partial \alpha_i^*} = -y_i + w^\top x_i - \varepsilon - \xi_i \leq 0 \quad \forall i. \end{aligned}$$

↪ La condition KKT finale stipule que le produit des multiplicateurs de Lagrange avec les contraintes est égal à zéro :

$$\begin{cases} \alpha(y_i - w^\top x_i - \varepsilon - \xi_i^*) = 0 \text{ et } \alpha^*(-y_i + w^\top x_i - \varepsilon - \xi_i) = 0 & \forall i \\ \lambda_i \xi_i = 0 \text{ et } \lambda_i^* \xi_i^* = 0 \end{cases}$$

Par conséquent, $0 \leq \alpha_i, \alpha_i^* \leq C$ Pour les vecteurs de support.

Ainsi, $w = \sum_{i=1}^{N_{SV}} (\alpha_i^* - \alpha_i) x_i$

Enfin, l'hyperplan de décision optimal est obtenu comme

$$y = f(x) = \langle w, x \rangle + b = \sum_{i=1}^N w_i x_i + b$$

$$f(x) = \sum_{i=1}^{N_{SV}} (\alpha_i^* - \alpha_i) x_i \cdot x + b = \sum_{i=1}^{N_{SV}} (\alpha_i^* - \alpha_i) \langle x_i, x \rangle + b, \forall \alpha_i, \alpha_i^* \in [0, C].$$



SVR du noyau

La section précédente a traité pour les fonctions linéaires. Pour les fonctions non linéaires, les données peuvent être projetés dans un espace de dimension supérieure pour permettre d'effectuer la séparation linéaire. On parle donc aux **fonctions du noyau transformant** $\phi(x)$.

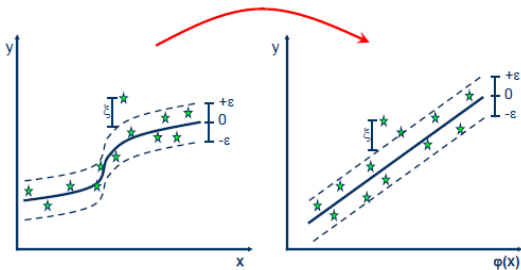


Figure – Transformation une fonction non séparable linéairement en une fonction séparable linéairement.



la formulation primal du noyau SVR :

$$(\star) \quad \begin{cases} \min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i + \xi_i^* \\ \text{Contraintes : } y_i - w^\top \phi(x_i) \leq \varepsilon + \xi_i^* & \forall i = 1, \dots, N, \\ w^\top \phi(x_i) - y_i \leq \varepsilon + \xi_i & \forall i = 1, \dots, N, \\ \xi_i, \xi_i^* \geq 0, & \forall i = 1, \dots, N \end{cases}$$

Pour obtenir la solution de (★) deux ensembles de multiplicateurs de Lagrange sont utilisés qui sont : α_i, α_i^*

Avec $0 \leq \alpha_i, \alpha_i^* \leq C$. Pour les vecteurs de support

$0 \leq \alpha_i, \alpha_i^* \leq C$. Finalement, l'hyperplan de décision optimal est obtenu comme :

$$f(x) = \langle w, x \rangle + b = \sum_{i=1}^{N_{SV}} (\alpha_i^* - \alpha_i) \langle \phi(x_i), \phi(x) \rangle + b$$

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

$$\text{Avec } k(x_i, x) = \phi(x_i) \cdot \phi(x) \text{ et } w = \sum_{i=1}^{N_{SV}} (\alpha_i^* - \alpha_i) \phi(x_i).$$



La méthode LS-SVM & DLS-SVM



Pendant les dernières années, grands travaux de recherche ont été menés en vue de l'application de SVM pour la [prévision des séries chronologiques](#). En conséquence, de nombreuses prévisions SVM différentes des algorithmes ont été dérivées. Certains d'entre eux sont l'**algorithme CSVM** (Critical Support Vector Machine), l'**algorithme LS-SVM** (Least Square Support Vector Machine) et **DLS-SVM** (Dynamic Least Squares Support Vector Machine), etc. Nous allons maintenant présenter le célèbre algorithme DLS-SVM développé par Y. Fan et al., 2006 pour la prévision des séries chronologiques. Avant cela, nous donnerons un aperçu de la technique LS-SVM.



La méthode LS-SVM



La prévision des séries chronologiques est devenue un domaine de recherche chaud avec une grande valeur théorique et une grande valeur d'application. En tant que type d'extension de machine à vecteur de support le moins carré (LS-SVM), le LS-SVM récurrent est proposé et appliqué à la **prédiction** de la **pollution atmosphérique** et de **séries chronologiques chaotiques**.

- ◆ Modèle LS-SVM comme représentation de l'espace d'entité :

$$y(x) = w^T \phi(x) + b \quad \text{Avec } x \in \mathbb{R}^n, y \in \mathbb{R}.$$

$\phi(x)$ une transformation non-linéaire vers un espace de dimension supérieure.

Soit l'ensemble d'apprentissage : $\{x_i, y_i\}_{i=1}^N$

- ◆ Problème d'optimisation :



$$(\wp) \begin{cases} \min_{w,b,e} \mathcal{J}(w,e) = \frac{1}{2}w^\top w + \frac{1}{2}\gamma \sum_{i=1}^N e_i^2 \\ \text{Contrainte : } y_i = w^\top \phi(x_i) + b + e_i, \quad \forall i = 1, \dots, N, \end{cases}$$

Où γ est le paramètre de régularisation.

\Rightarrow Ceci est une forme de **régression de ridge** (voir Saunders, 1998)

◆ le lagrangien pour le problème (\wp) est donné par :

$$\mathcal{L}(w, b, e, \alpha) = \mathcal{J}(w, e) - \sum_{i=1}^N \alpha_i \{w^\top \phi(x_i) + b + e_i - y_i\}$$

Avec $\alpha_i \geq 0$, $\forall i = 1, \dots, N$ sont les multiplicateurs de Lagrange.

◆ Appliquer les conditions d'optimalité :



$$\left\{ \begin{array}{ll} \frac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^N \alpha_i \phi(x_i). \\ \frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^N \alpha_i = 0. \\ \frac{\partial \mathcal{L}}{\partial e_i} = 0 \Rightarrow \alpha_i = \gamma e_i, & \forall i = 1, \dots, N. \\ \frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \Rightarrow w^\top \phi(x_i) + b + e_i - y_i = 0, & \forall i = 1, \dots, N. \end{array} \right.$$

- ◆ 😊 L'élimination de w et e donnera un système linéaire au lieu d'un problème de programmation quadratique, d'où la solution :

$$(SL) \begin{bmatrix} 0 & 1_N^T \\ 1_N & \Omega + \gamma^{-1} \mathbb{I}_N \end{bmatrix}_{(N+1) \times (N+1)} \begin{bmatrix} b \\ \alpha \end{bmatrix}_{(N+1) \times 1} = \begin{bmatrix} 0 \\ Y \end{bmatrix}_{(N+1) \times 1} \quad (11)$$

Avec $Y = [y_1, \dots, y_N]^T$, $1_N = [1, \dots, 1]^T$ et $\alpha = [\alpha_1, \dots, \alpha_N]^T$. Ici, I_N est une matrice identité $N \times N$, et $\Omega \in \mathbb{R}^{N \times N}$ est la matrice du noyau définie par : $\Omega_{i,j} = \phi(x_i)^\top \phi(x_j) = K(x_i, x_j)$.



- ◆ Modèle LS-SVM résultant pour l'estimation de la fonction de décision est donc donnée par :

$$y(x) = \sum_{i=1}^N \alpha_i k(x, x_i) + b$$

Ici α , b sont les solutions du système linéaire (SL).



Le principal avantage de la technique LS-SVM est qu'elle transforme le QPP traditionnel en un problème de système linéaire simultané, garantissant ainsi la simplicité des calculs, une convergence rapide et haute précision.



La méthode DLS-SVM



Basé sur la narration de la théorie de LS-SVM, le DLS-SVM convient à la reconnaissance de systèmes en temps réel et à la prédiction de séries chronologiques. Chaque fois qu'une nouvelle observation est obtenue, la méthode se débarrasse du premier vecteur et le remplace par le nouveau vecteur d'entrée. Autrement dit, cet algorithme peut ajuster le modèle pour suivre la dynamique du système non linéaire variant dans le temps.



Nous avons prouvé que le DLS-SVM devait contenir trois phases : **augmentation des échantillons**, **détermination des échantillons à retirer** et **retrait des échantillons**.



Augmentation des échantillons



En gardant compte tenu de la formulation LS-SVM dont nous venons de parler, considérons que

$$Q_N = \begin{bmatrix} 0 & 1_N^T \\ 1_N & \Omega + \gamma^{-1} \mathbb{I}_N \end{bmatrix}_{(N+1) \times (N+1)} \quad (12)$$

Maintenant, pour résoudre (11) ou (SL) nous avons besoin de Q_N^{-1} qui doit être calculé efficacement. Lorsqu'une nouvelle observation doit être ajoutée à l'ensemble d'apprentissage existant, alors Q_N devient :

$$Q_{N+1} = \begin{bmatrix} Q_N & k_{N+1} \\ k_{N+1}^T & k_{N+1}^* \end{bmatrix} \in \mathbb{R}^{(N+1) \times (N+1)} \quad (13)$$

Où $k_{N+1} = [K(x_{N+1}, x_1), \dots, K(x_{N+1}, x_N)]^T$, $k_{N+1}^* = K(x_{N+1}, x_{N+1}) + \gamma^{-1} = 1 + \gamma^{-1}$.



Pour gagner du temps de calcul, en utilisant le lemme d'inversion de matrice, est appliqué pour calculer Q_{N+1}^{-1}

$$Q_{N+1}^{-1} = \begin{bmatrix} Q_N & k_{N+1} \\ k_{N+1}^T & k_{N+1}^* \end{bmatrix} = \begin{bmatrix} Q_N^{-1} + Q_N^{-1} k_{N+1} k_{N+1}^T Q_N^{-1} \beta^{-1} & -Q_N^{-1} k_{N+1} \beta^{-1} \\ -k_{N+1} Q_N^{-1} \beta^{-1} & \beta^{-1} \end{bmatrix} \quad (14)$$

Où $\beta = k_{N+1}^* - k_{N+1}^T Q_N^{-1} k_{N+1}$.

Avec l'équation de récursion ci-dessus, toutes les inversions de matrice directes sont éliminés. Pour utiliser la relation (14) il suffit de connaître Q_N^{-1} , qui est obtenu lors de la solution de (11).



Déterminer les échantillons à retirer, nous utilisons la stratégie heuristique basée sur la description du champ de données des vecteurs de support pour déterminer les échantillons à retirer. La description du champ de données du vecteur de support a été proposée par *Tax* et d'autres en s'inspirant de la machine à vecteur de support.



Déterminer les échantillons à retirer.



Son but est de rechercher l'hypersphère avec un rayon minimum possible pour couvrir tous ou la plupart des échantillons dans un espace d'entités de grande dimension. La question peut être exprimée avec une programmation quadratique comme suit :

$$\max \mathcal{Q}(\beta) = \sum_{i=1}^N K(x_i, x_i) \beta_i - \sum_{i,j=1}^N \beta_i \beta_j K(x_i, x_j) \quad (15)$$

Avec $0 \leq \beta_i \leq C$, $i = 1, \dots, N$

Où β_i est le coefficient de Lagrange et se rencontre :

$$\sum_{i=1}^N \beta_i = 1.$$

Solution de QPP précédente pour obtenir le rayon d'hypersphère R dans l'espace d'entités et $D(x_i)$, qui est la distance entre le point d'échantillonnage et le centre d'hypersphère.





La distance peut être obtenue en utilisant l'équation suivante :

$$D(x_i) = \sqrt{\sum_{i,j=1}^N \beta_i \beta_j K(x_i, x_j) + K(x_i, x_j) - 2 \sum_{i,j=1}^N K(x_j, x_i) \beta_j}$$

La taille de $D(x_i)$ reflète l'écart d'un échantillon par rapport à sa population de classe, lorsque $D(x_i)$ est petit, l'échantillon x_i est proche de la droite de régression, dans le cas contraire, x_i est loin de la droite. Par conséquent, chaque fois que des échantillons sont ajoutés. L'échantillon (x_d, y_d) avec le plus grand $D(x_i)$ sera retiré.



Pour faciliter la solution de l'inverse Q_{N+1} pendant le prélèvement d'échantillon, Q_{N+1} dans la formule (13) est soumis à une transformation élémentaire $r_d \leftrightarrow r_1$, $c_d \leftrightarrow c_1$, avec la matrice inverse correspondante \hat{Q}_{N+1}^{-1} obtenu :

$$\hat{Q}_{N+1} = E(d, 1)Q_{N+1}E(d, 1) \Rightarrow \hat{Q}_{N+1}^{-1} = E(d, 1)Q_{N+1}^{-1}E(d, 1) \quad (16)$$



Retrait d'échantillon

☰ En supposant que le nouveau point de données venait d'être ajouté et que \hat{Q}_{N+1}^{-1} était déjà connu à l'aide de la formule (16), \hat{Q}_{N+1}^{-1} est ré-exprimé comme suit :

$$\hat{Q}_{N+1} = \begin{bmatrix} k_1^* & k_1^T \\ k_1 & \hat{Q}_1 \end{bmatrix} \quad (17)$$

Ici $k_1^* = K(x_1, x_1) + \gamma^{-1} = 1 + \gamma^{-1}$, $k_1 = [k(x_1, x_1), \dots, k(x_1, x_{N+1})]^T$

$$\hat{Q}_N = \hat{\Omega}_N + \gamma^{-1} \mathbb{I} = \begin{bmatrix} 0 & 1_N^T \\ 1_N & \hat{\Omega} + \gamma^{-1} \mathbb{I}_N \end{bmatrix}_{(N+1) \times (N+1)}$$

Avec $\hat{\Omega}_{(i-1)(j-1)} = K(x_i, x_j)$, $\forall i, j = 2, \dots, N+1$

☐ On remarque que la différence entre Q_{N+1} et \hat{Q}_{N+1} n'est que l'ordre des lignes et des colonnes, de même pour Q_{N+1}^{-1} et \hat{Q}_{N+1}^{-1} .




Ainsi, en ajustant les positions des éléments de Q_{N+1}^{-1} . Nous avons utilisé le lemme d'inversion de matrice pour obtenir :

$$\hat{Q}_{N+1}^{-1} = \begin{bmatrix} \beta^{-1} & -k_1 \hat{Q}_N^{-1} \beta^{-1} \\ -\hat{Q}_N^{-1} k_1 \beta^{-1} & \hat{Q}_N^{-1} (\mathbb{I} + k_1 k_1^T \hat{Q}_N^{-1} \beta^{-1}) \end{bmatrix} = \begin{bmatrix} q & P^T \\ P_{(N+1) \times 1} & Q_{(N+1)(N+1)}^* \end{bmatrix} \quad (18)$$

Où $\beta = k_1^* - k_1^T \hat{Q}_N^{-1} k_1$, $\hat{Q} \in \mathbb{R}^{N \times N}$, $P \in \mathbb{R}^N$

Selon la formule (18) on a \hat{Q}_N^{-1} comme :

$$\hat{Q}_N^{-1} = Q^* - \frac{P P^T}{q} \quad (19)$$

 Après avoir ajouté un nouvel échantillon (x_{N+1}, y_{N+1}) , échangez l'échantillon à retirer et le premier échantillon, puis du premier échantillon (x_1, y_1) , retirez ; la matrice du noyau du nouveau modèle est $\hat{\Omega}_N$; le \hat{Q}_N^{-1} correspondant est obtenu à partir de la formule (19) ; mettez à jour a et b de (11) pour obtenir le nouveau modèle de fonction.



Récapitulatif

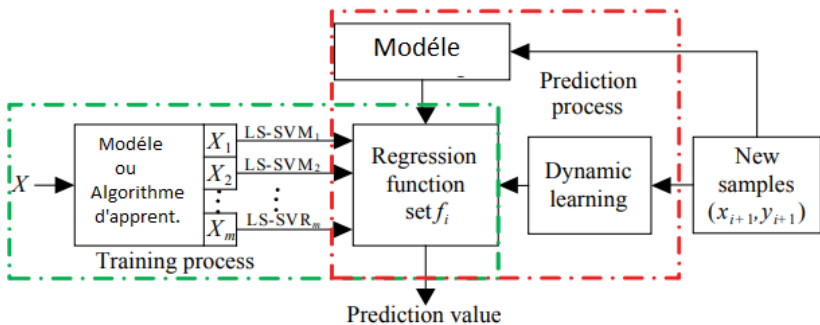








Figure – Diagramme de structure d'algorithme










Sources

-  Mokhtar TAFFAR, *INITIATION A L'APPRENTISSAGE AUTOMATIQUE*, Université de Jijel, 5-44.
-  Ratnadip Adhikari, R. K. Agrawal, *An Introductory Study on Time Series Modeling and Forecasting*, 31-41.
-  Jason Brownlee, *Introduction to Time Series Forecasting with Python*, Edition : v1.9, 2020, 15-16.
-  Hachem Kadri, *Data Science, Fondements de l'apprentissage statistique*, 2018-2019, 4-20.
-  Mariette Awad, Rahul Khanna, *Efficient Learning Machines*, 39-80.
-  Sylvain Arlot, *Fondamentaux de l'apprentissage statistique*, 2017, 3-49.





Sources


-  Vladimir N.Vapnik, *STATISTICAL LEARNING THEORY*, JOHN WILEY & SONS, INC, 1998, 79-80.
-  Magalie Fromont, *Apprentissage Statistique*, Master de Statistique appliquée, Université Rennes 2 , 2016 - 2017.
-  Johan Suykens, *Least Squares Support Vector Machines*, NATO-ASI Learning Theory and Practice Leuven July 2002, 5-83.
-  Autres ouvrages de référence :
 -  <https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-svm.pdf>
 -  <https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-svm-old.pdf>
 -  http://www.saedsayad.com/support_vector_machine_reg.htm



Sources

 https://en.wikipedia.org/wiki/Least-squares_support-vector_machine

 C. Saunders, A. Gammerman and V. Vovk, *Ridge Regression Learning Algorithm in Dual Variables*, Royal Holloway, University of London Egham, Surrey, TW20 0EX, UK, 1998

 X.L.ZHU^a, J.LING^a, Y.J.HE^b, B.WANG, *SOFT-SENSING MODELLING METHOD FOR MARINE BIOLOGICAL ENZYME FERMENTATION PROCESS*, School of Electrical and Information Engineering, Jiangsu University, 212 013 Zhenjiang, China, 2016, 3727-3730.

