SVM non-linéaire, et méthodes à noyaux

29 novembre 2019

Rappel SVM linéaire

Le modèle

- Données d'apprentissage $\{\mathbf{x}_i, y_i\}$
- Formulation du problème primal

$$\begin{array}{lll} \min_{w,b,\{\xi_i\}} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i & \\ \text{s.c.} & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i & \forall i = 1, \cdots, n \\ & \xi_i \geq 0 & \forall i = 1, \ldots, n \end{array}$$

- Formulation du problème dual

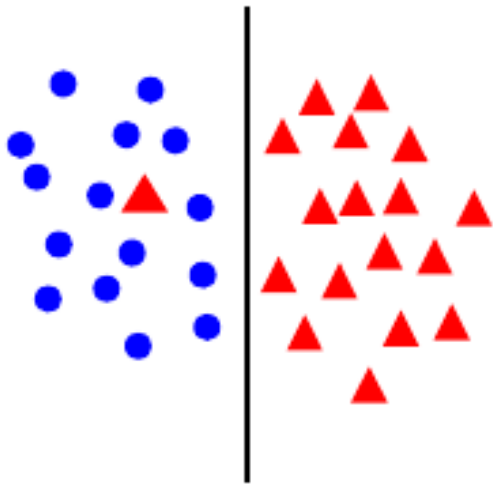
$$\begin{array}{ll} \max\limits_{\{\alpha_i\}} & \sum\limits_{i=1}^n \alpha_i - \frac{1}{2}\sum\limits_{i,j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.c.} & 0 \leq \alpha_i \leq C, \quad \forall\, i = 1, \cdots, n \\ & \sum\limits_{i=1}^n \alpha_i y_i = 0 \end{array}$$

- La fonction de décision $f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}$

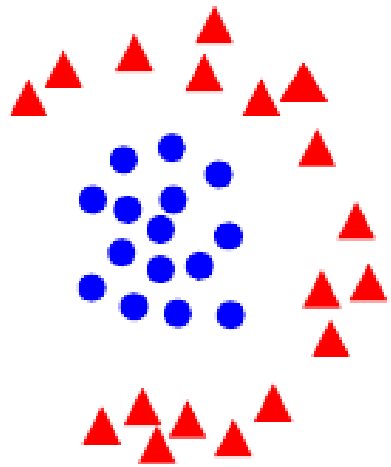
Non-linéarité ?

Exemples de cas d'usage des SVM

- problème linéaire non-séparable



- problème séparable mais non-linéaire ?

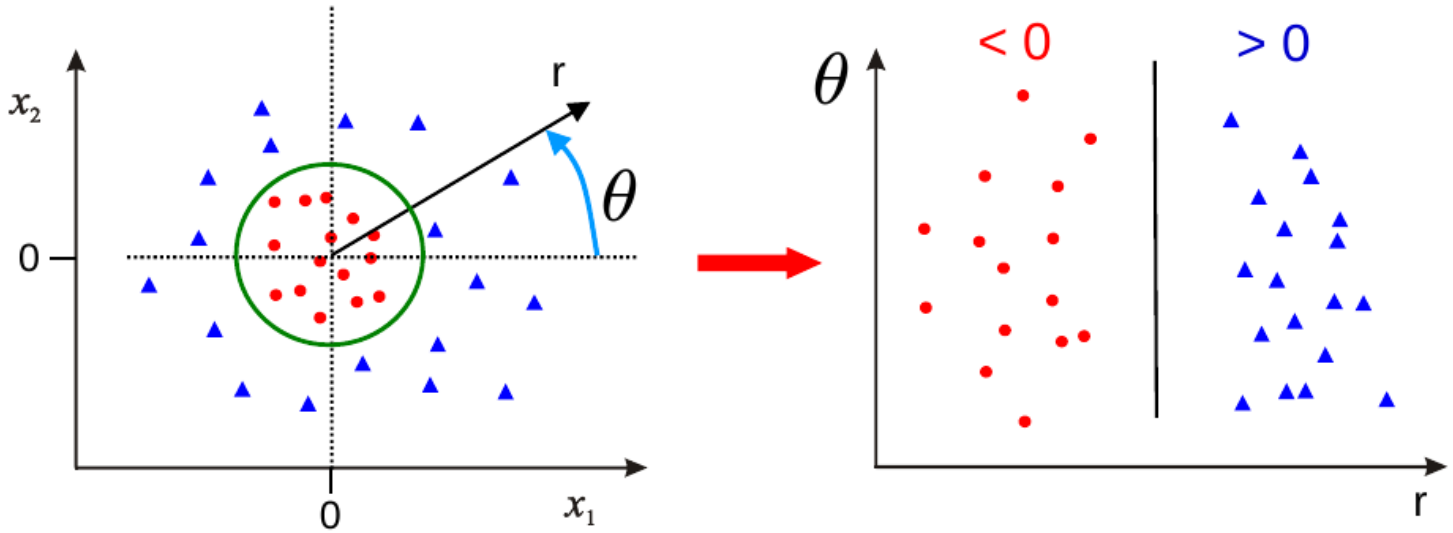


Quelles solutions ?

- transformation non-linéaire

Exemples de transformation non-linéaire

Coordonnées polaires

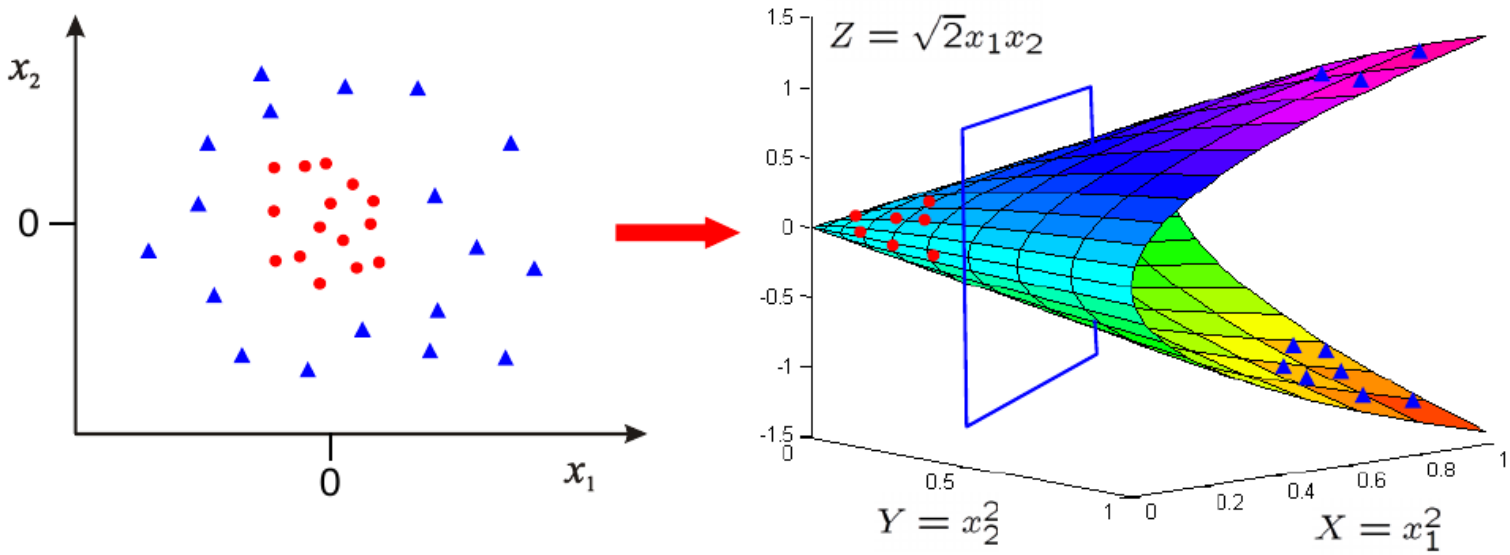


- Les données sont linéairement séparables dans l'espace en coordonnées polaire
- agit comme une transformation non-linéaire de l'espace original

$$\Phi(\mathbf{x}) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix}$$

Exemples de transformation non-linéaire

Projection dans un espace de plus haute dimension

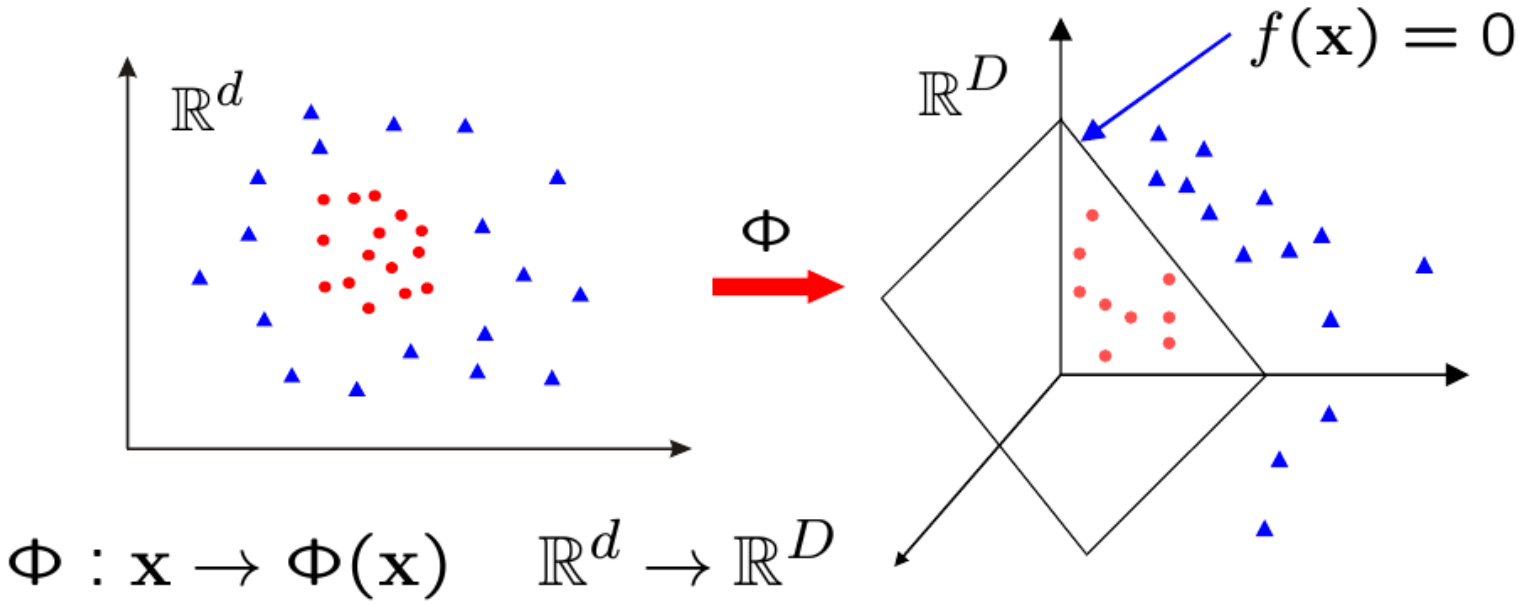


- les données sont séparables dans un espace 3D.

$$\Phi(\mathbf{x}) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

- on peut utiliser un SVM linéaire dans un autre espace.

SVM dans un espace transformé



- Apprendre un classifieur linéaire dans le nouvel espace $\mathbb{R}^D$

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$$

où $\Phi(\mathbf{x})$ est la fonction de transformation des données avec $\Phi : \mathbb{R}^d \mapsto \mathcal{F}$ (dans l'exemple $\mathcal{F} = \mathbb{R}^D$)

Principes de la non-linéarisation

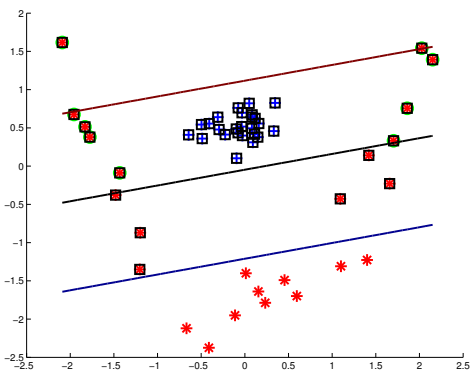
- On projette les données x grâce à une transformation Φ dans un espace $\mathcal{F}$. La fonction de décision devient $f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$

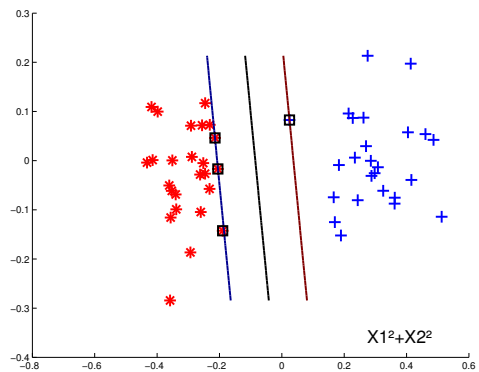
- On applique l'algorithme linéaire dans l'espace $\mathcal{F}$.

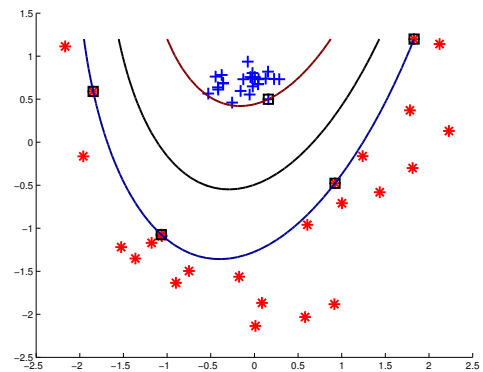
$$\begin{array}{lll} \min_{w,b,\{\xi_i\}} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i & \\ \text{s.c.} & y_i(\mathbf{w}^\top \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i & \forall i = 1, \cdots, n \\ & \xi_i \geq 0 & \forall i = 1, \ldots, n \end{array}$$

avec $\mathbf{w} \in \mathcal{F}$.

- La fonction de transfert obtenue est non-linéaire dans l'espace original.







Problème dual et fonction de décision transformée

Fonction de décision duale

La fonction de décision pour les SVMs

$$f(x) = \sum_{i=1}^{n} \alpha_i x^\top x_i + b \quad \Rightarrow \quad f(x) = \sum_{i=1}^{n} \alpha_i \Phi(x)^\top \Phi(x_i) + b$$

Problème dual

$$\begin{array}{ll} \max_{\{\alpha_i\}} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(x_i)^\top \Phi(\mathbf{x}_j) \\ \text{s.c.} & 0 \leq \alpha_i \leq C, \quad \forall\, i = 1, \cdots, n \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{array}$$

Astuce du noyau

Constat

- La fonction $\Phi(\mathbf{x})$ intervient toujours sous la forme $\Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$
- dans le problème dual, une fois que tout les produits scalaires $\Phi(\mathbf{x}_i)^\top (\Phi(\mathbf{x}_j)$ ont été calculés, on n'a besoin que résoudre le problème dual en $\alpha \in \mathbb{R}^n$.
- On n'a pas besoin de calculer $\mathbf{w} \in \mathbb{R}^D$. c'est avantageux si D est très grand.

Transformation implicite par un noyau $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$

- Fonction de décision

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

- problème dual

$$\begin{array}{ll} \max_{\{\alpha_i\}} & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j k(x_i, \mathbf{x}_j) \\ \text{s.c.} & 0 \leq \alpha_i \leq C, \quad \forall\, i = 1, \cdots, n \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{array}$$

Kernel Ridge Regression

Formulation regression ridge

- $\mathcal{D} = \{(x_i, y_i)\} \in \mathcal{X} \times \mathbb{R}, \quad i = 1 \cdots n$: ensemble de points étiquetés .
- Modele linéaire : $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$
- Cout ℓ_2 penalité ℓ_2

Apprentissage du modèle

- Optimisation

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- Condition d'optimalité

$$-\mathbf{X}^\top \mathbf{y} + \mathbf{X}^\top \mathbf{X}\mathbf{w}^\star + \lambda I \mathbf{w}^\star = 0$$

- Solution

$$\mathbf{w}^\star = (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1}(\mathbf{X}^\top \mathbf{y})$$

Ici $\mathbf{X}^\top \mathbf{X}$ représente une matrice de covariance

Kernel Ridge regression : reformulation

- La condition d'optimalité se réécrit :

$$\mathbf{w}^\star = \frac{1}{\lambda}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}^\star)$$

- on peut donc dire qu'il existe un vecteur $\alpha \in \mathbb{R}^n$ tel que

$$\mathbf{w}^\star = \mathbf{X}^\top\alpha = \sum_i \mathbf{x}_i\alpha_i$$

- Dans ce contexte, on a $f(\mathbf{x}) = \sum_i \alpha_i \mathbf{x}_i^\top \mathbf{x}$
- les variables α s'obtiennent par $\alpha = \frac{1}{\lambda}(\mathbf{y} - \mathbf{X}\mathbf{w}^\star) = \frac{1}{\lambda}(\mathbf{y} - \mathbf{X}\mathbf{X}^\top\alpha)$ donc

$$\alpha = (\lambda I + \mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{y}$$

ici, $(\mathbf{X}\mathbf{X}^\top)_{i,j} = \mathbf{x}_i^\top \mathbf{x}_j$

Kernel Ridge Regression : reformulation

- la fonction de décision :

$$f(\mathbf{x}) = \sum_i \alpha_i \mathbf{x}_i^\top \mathbf{x}$$

- le problème d'apprentissage

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{X}^\top \alpha\|_2^2 + \frac{\lambda}{2}\alpha^\top \mathbf{X}\mathbf{X}^\top \alpha$$

Kernelized

- l'ensemble du problème se reformule en fonction des produits scalaires $\mathbf{x}_i^\top \mathbf{x}_j$

Kernel Kmeans

Principe

- remplacer la distance euclidienne par la distance dans l'espace transformée

$$d(\mathbf{x}_i, \mu_k)^2 = \|\Phi(\mathbf{x}_i) - \Phi(\mu_k)\|^2 = k(\mathbf{x}_i, \mathbf{x}_i) + k(\mu_k, \mu_k) - 2k(\mathbf{x}_i, \mu_k)$$

où $\Phi(\cdot)$ est la transformation implicite et $k(\cdot, \cdot)$ le produit scalaire dans l'espace transformée

Détail

- μ_k moyenne des $\Phi(x_i)$ du cluster
- $k(x_i, \mu_k) = \frac{1}{|J|} \sum_{jinJ} \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$