

## SVM, noyaux, régularisation quelques idées récentes en Machine Learning

30 mai 2005

Olivier Bousquet

olivier.bousquet@pertinence.com

### > Plan de l'exposé

- **Contexte**
- **Illustration en 2 dimensions**
- **Anciennes idées**
- **Nouvelles idées**
  - Optimisation convexe
  - Régularisation
  - Espaces Hilbertiens et noyaux
  - SVM / Boosting

### > Contexte

- **Simplifions le problème à l'extrême**
- **Données**
  - Exemples  $x_1, x_2, \dots, x_n \in X$
  - Etiquettes  $y_1, y_2, \dots, y_n \in \{-1, +1\}$
- **Résultat attendu**
  - Fonction  $f: X \rightarrow \{-1, +1\}$
  - Telle que  $f(x)$  approche  $y$ , au moins sur les données, mais si possible aussi « en dehors »

### > Fondements

- **Si on veut espérer faire quelque chose, il faut des hypothèses**
- **Les données futures sont « similaires » aux données d'apprentissage**
- **Deux points « proches » sont susceptibles d'avoir la même classe**
- **Sans ces hypothèses, on ne peut rien faire**

## > Apprentissage local

Étant donnée une distance  $d$  « pertinente »

- On utilise les  $k$  plus proches voisins
- Ou l'apprentissage local :

- fonction de similarité

$$k(x, x') = e^{-d^2(x, x')/\sigma^2}$$

- vote de tous les points en fonction de leur distance au nouveau point

$$f(x) = \sum_{i=1}^n y_i k(x, x_i)$$

- rééquilibrage des classes

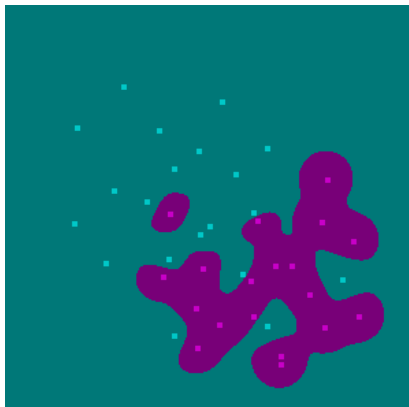
$$f(x) = \frac{1}{n_+} \sum_{i \in +} k(x, x_i) - \frac{1}{n_-} \sum_{i \in -} k(x, x_i)$$

## > Normalisation

- Seul le signe compte, on peut aussi choisir un décalage
- moyenne sur les positifs = -moyenne sur les négatifs = 1

$$g(x) = \frac{f(x) + b}{a} = \frac{f(x) - \frac{1}{2} \left( \frac{1}{n^+} \sum_{i \in +} f(x_i) + \frac{1}{n^-} \sum_{i \in -} f(x_i) \right)}{\frac{1}{2} \left( \frac{1}{n^+} \sum_{i \in +} f(x_i) - \frac{1}{n^-} \sum_{i \in -} f(x_i) \right)}$$

## > Exemple



## > Sigma très petit

- $k(x, x') = 1$  si  $x = x'$
- $k(x, x') = 0$  sinon

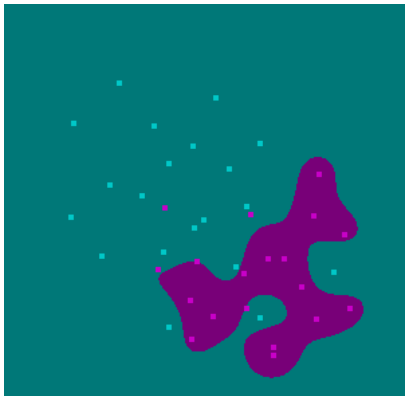
$$f(x_i) = y_i / n_{\pm}$$

$$f(x) = 0$$

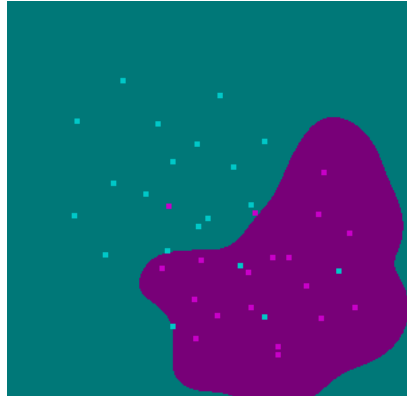
$$g(x_i) = y_i$$

$$g(x) = 0$$

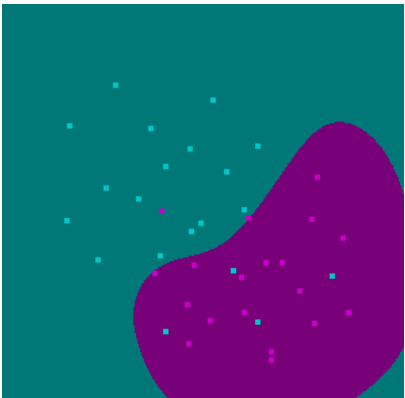
> Sigma augmente



>



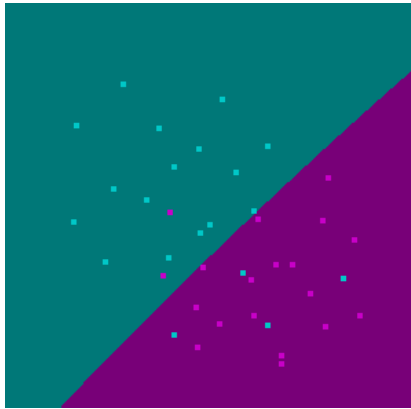
>



>



## > Sigma très grand



## > Sigma très grand

$$k(x, x') \approx 1 - \frac{d^2(x, x')}{\sigma^2}$$

$$f(x) \approx 0$$

$$g(x) \approx \frac{d^2(x, x_-) - d^2(x, x_+)}{d^2(x_+, x_-)}$$

- **g(x) = classifieur linéaire entre les centres**

## > Conclusion de l'exemple

- **Le paramètre sigma permet de choisir un niveau de « localité »**
- **Théorie: pour sigma fixé, pas de consistance**
- **pour sigma qui décroît vers 0 (pas trop vite), consistance**
- **Sigma doit être adapté à la densité des points**

## > Poids sur les exemples

- **réécriture : alpha = 1/n+ ou 1/n-**

$$f(x) = \sum_{i=1}^n \alpha_i y_i k(x_i, x)$$

$$g(x) = \frac{\sum_{i=1}^n \alpha_i y_i k(x_i, x) - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j k(x_i, x_j)}{\frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j k(x_i, x_j)}$$

## > Choix des poids

- Poids qui somment à 1 sur chaque classe

$$\sum_{i=1}^n \alpha_i = 2 \quad \sum_{i=1}^n y_i \alpha_i = 0$$

- Idée: on donne plus de poids aux exemples mal classés

## > Utilisation des poids



>



>





## > Algorithme possible

- Exemple mal classés

$$y_i f(x_i) \leq 0$$

- Algorithme

$$\min \sum_{i=1}^n \alpha_i y_i f_\alpha(x_i)$$

$$\sum_{i=1}^n \alpha_i = 2 \quad \sum_{i=1}^n y_i \alpha_i = 0$$



## > Limitation des poids

- On peut introduire une contrainte sur les poids
- Contrainte maximale: poids égaux
- Contrainte minimale: des poids vont pouvoir s'annuler au profit d'autres



## > Pourquoi cela fonctionne-t-il ?

- **Tout est dans la distance (hypothèse fondamentale)**
- **et dans la souplesse offerte par les paramètres.**
- **La localité peut être adaptée (en fonction de la qualité de la distance et du nombre de points)**
- **On peut trouver le juste compromis entre adaptation aux données et complexité (poids)**

## > Contexte

- **Simplifions le problème à l'extrême**
- **Données**
  - Exemples  $x_1, x_2, \dots, x_n \in X$
  - Etiquettes  $y_1, y_2, \dots, y_n \in \{-1, +1\}$
- **Résultat attendu**
  - Fonction  $f: X \rightarrow \{-1, +1\}$
  - Telle que  $f(x)$  approche  $y$ , au moins sur les données, mais si possible aussi « en dehors »

## > Anciennes idées (1)

- **Il est clair qu'il faut restreindre l'espace de recherche de  $f$**
- **Donc apprendre c'est faire une recherche dans un espace en minimisant l'erreur sur les données**
- **On peut introduire un biais d'apprentissage (sous la forme d'heuristiques)**
- **La recherche est algorithmiquement complexe**

## > Anciennes idées (2)

- **Remplacer l'optimisation combinatoire par une optimisation régulière: réseaux de neurones**

$$\min \sum_{i=1}^n (f(x_i) - y_i)^2$$

$$f(x) = s \left( \sum_{j=1}^d \beta_j s(\dots \phi(x)) \right)$$

- **Mais la combinatoire subsiste (optimas locaux)**

## > Nouvelles idées (1)

- **Optimisation : quels sont les problèmes faisables ?**
  - Ce ne sont pas seulement les problèmes linéaires ou les problèmes avec peu de variables
- **Ce sont les problèmes convexes**
- **Un problème est convexe si son objectif est une fonction convexe pour le bon paramétrage et l'espace de recherche est convexe**

## > Nouvelles idées (2)

- **Se ramener à un problème convexe**
- **Relaxation d'un problème combinatoire ou régulier mais non convexe**

## > Avantages

- **On ne parle plus d'heuristiques mais de critères**
- **Pas de questions sur l'algorithme**
- **Propriétés de dualité**

## > Fonction d'erreur convexe

- **Problème initial (combinatoire)**
- **On remplace la fonction indicatrice par une fonction convexe plus grande**

$$\min_{f \in F} \sum_{i=1}^n I_{\{y_i f(x_i) \leq 0\}}$$

$$\min_{f \in F} \sum_{i=1}^n (1 - y_i f(x_i))_+$$

## > Espace de recherche convexe

- Pour que le problème soit convexe, il faut aussi que l'espace de recherche soit convexe
- On peut prendre un espace vectoriel

$$F = \left\{ \sum_{j=1}^m \beta_j \phi_j(x) \right\}$$

## > Paramétrage convexe

- Il faut aussi que le paramétrage soit adapté
- Convexité par rapport à la variable (qui ici est une fonction)
- On peut en général considérer que  $f(x)$  est une fonction linéaire en  $f$

## > Problème linéaire inverse

- On doit résoudre quelque chose comme
  - $(f(x_1), f(x_2), \dots, f(x_n)) = (y_1, y_2, \dots, y_n)$
  - que l'on peut représenter comme
- Af = Y**
- avec A une matrice, f et Y des vecteurs (f représentée par les coefficients beta).
  - C'est un problème linéaire inverse: on voudrait inverser A (mais ce n'est pas toujours possible)

## > Nouvelles idées (3)

- Régularisation: technique pour résoudre les problèmes inverses mal posés
- Deux solutions possibles
  - Pénalisation:  $\min \|Af - Y\| + \|f\|$
  - Itération:  $\min \|Af - Y\|$  dans un sous espace de faible dimension

## > Interprétation

- **Régularisation: introduction de régularité, donc généralisation**
- **Permet de compenser le manque de données**
- **Besoin d'un paramètre pour faire un compromis entre régularité et adaptation aux données**
  - Ce paramètre doit s'adapter à la quantité de données

## > Régularisation

- **Minimisation de l'erreur pénalisée**

$$\min_{f \in F} \sum_{i=1}^n (1 - y_i f(x_i))_+ + \lambda \|f\|^2$$

- **Intuitivement (formulation précise plus tard)**

$$f(x) = \langle f | \delta_x \rangle$$

- **Formulation équivalente (cf calculs de sous-gradients)**

$$f(x) = \sum_{i=1}^n \alpha_i y_i \delta_i \quad \max \sum_{i=1}^n \alpha_i - \sum_{i,j=1}^n \alpha_i y_i \alpha_j y_j \langle \delta_i | \delta_j \rangle$$

## > Régularisation

- **Supposons que**

$$\langle \delta_i | \delta_j \rangle = k(x_i, x_j)$$

- **on retrouve (à la normalisation près) l'algorithme présenté en exemple**
- **Donc la dualité permet de faire le lien entre**
  - un algorithme local avec pondération des exemples proches de la frontière
  - un algorithme de minimisation d'une erreur convexe régularisée dans un espace de fonctions

## > Régularisation

- **On a utilisé la dualité**

- on change d'espace : primal (F) dual ( $\mathbb{R}^n$ )
- on change de paramétrisation : au lieu de fonctions on a un vecteur de paramètres
- mais le problème reste le même

- **On gagne (potentiellement) en nombre de variables**

- Une variable duale par contrainte (par exemple d'apprentissage)

## > Espaces de fonctions

- **Existe-t-il des espaces de fonctions qui se comportent comme des espaces vectoriels euclidiens ?**
  - Oui: espaces de Hilbert (munis d'un produit scalaire)
  - Exemple: L2
- **Dans lesquels on peut identifier  $f(x)$  avec un produit scalaire ?**
  - Impossible dans L2
- **Dans lesquels le produit scalaire de deux évaluations est une fonction de similarité ?**

## > Exemple

- **Exemple: S1**
- **Considérons l'ensemble des fonctions absolument continues sur  $[0,1]$  (nulles en 0)**
- **muni du produit scalaire**

$$\langle f | g \rangle = \int_0^1 f'(t)g'(t)dt$$

- **sa norme est une notion de régularité (une fonction qui varie peu a une petite norme)**
- **on peut représenter l'évaluation**

$$\langle f | \min(x, \cdot) \rangle = \int_0^1 f'(t) \min(x, t) dt = f(x)$$

## > Propriétés

- **Si on note  $k(x, x') = \min(x, x')$**
- **on a**

$$f(x) = \langle k(x, \cdot) | f \rangle$$

$$\langle k(x, \cdot) | k(x', \cdot) \rangle = k(x, x')$$

## > Retour sur l'algorithme

- **Quand est-ce que le problème est convexe ?**

- Primal: quand on a une norme et que l'évaluation est « dérivable »
- Dual: quand la dérivée seconde est positive. Dans le cas vectoriel, il faut un Hessien (matrice des dérivées secondes) défini positif. La condition est donc que la matrice

- **Noyau défini positif**

- Une fonction symétrique de deux variables est définie positive si pour chaque ensemble de points et de coefficients on a

$$\sum_{i,j=1}^n \alpha_i \alpha_j k(x_i, x_j) \geq 0$$

- Ceci est équivalent au fait que la matrice Hessienne soit définie positive donc le problème convexe
- Ceci est équivalent au fait qu'il existe un espace associé avec une évaluation « dérivable » (qui correspond à un produit scalaire)

## > Espaces de Hilbert à noyau reproduisant

- **Etant donné un noyau défini positif, on peut définir un espace de fonction en formant toutes les combinaisons linéaires**
- **Cet espace peut être muni d'un produit scalaire**
- **C'est un espace de Hilbert à noyau reproduisant**
- **La norme est généralement une mesure de la régularité**

## > Feature space

- **Une autre interprétation possible:**
  - Si  $k$  est un noyau défini positif,
  - il existe une transformation telle que  $k(x, x') = \langle \phi(x) | \phi(x') \rangle$
- **Mais la transformation n'est pas explicite**
- **La bonne vision est la vision fonctionnelle (régularisation)**
- **ou la vision duale où on pondère les exemples**
- **En effet les solutions sont des combinaisons linéaires de noyaux calculés aux points d'apprentissage**

## > Boosting

- **Le boosting présente les mêmes ingrédients:**
  - Fonction d'erreur convexe (généralement exponentielle)
  - Régularisation (mais en norme L1 au lieu de L2)
  - Feature space: l'image d'un point par tous les apprenant faibles (au lieu de la similarité avec tous les autres points)
  - Dualité: poids sur les points plus importants sur les points frontière
  - Régularisation par arrêt prématuré (dans Adaboost) plutôt que par pénalisation

## > Messages

- **Optimisation convexe: algorithmes simples**
- **Régularisation: nécessaire pour la généralisation**
- **SVM: somme pondérée de noyaux en chaque point, avec un point plus important sur les points proches de la frontière**
- **Noyau: mesure de similarité**
- **Noyau défini positif: condition suffisante pour la convexité du problème d'optimisation**

## > Messages pratiques

- **Tout est dans le noyau (i.e. dans la représentation des données)**
- **Deux façons d'envisager le traitement d'un problème:**
  - Construire une représentation (vectorielle) des points (individuels)
  - Construire une distance (ou un noyau) entre paires de points

## > Avantages et inconvénients

- **Représentation vectorielle**
  - On peut utiliser quasiment tous les algorithmes
  - On peut utiliser des algorithmes qui sélectionnent naturellement les attributs (e.g. moindres généralisés, arbres...)
- **Utilisation de distance**
  - On peut utiliser tous les algorithmes à base de distance
  - Parfois plus naturel que de construire les vecteurs
  - Parfois plus efficace à calculer que les vecteurs
  - Dans les bons cas on a un noyau défini positif ce qui donne des propriétés de convexité et on peut alors utiliser tous les algorithmes à base de noyaux (classification, régression,...)
  - Mais il est souvent difficile qu'une fonction de similarité vérifie la condition ad hoc.
  - Le résultat est difficilement interprétable (pas de sélection d'attributs mais plutôt combinaison)

## > Noyaux vs vecteurs

- **Pour toute représentation vectorielle, on peut construire un noyau (produit scalaire des vecteurs)**
- **Pour toute fonction de similarité définie positive, il existe des vecteurs associés**