Module 1- cours 2 : Introduction à la classification supervisée

Florence d'Alché-Buc,
florence.dalche@telecom-paristech.fr

Telecom Evolution, Paris, France

Outline

Objectifs

- Donner un premier panorama de quelques modèles et méthodes d'apprentissage associées pour la classification supervisée
- Un peu de théorie vient plus tard (les 8 et 9 Juin)

Objectif: détecteur de spam

Apprendre à classer des messages

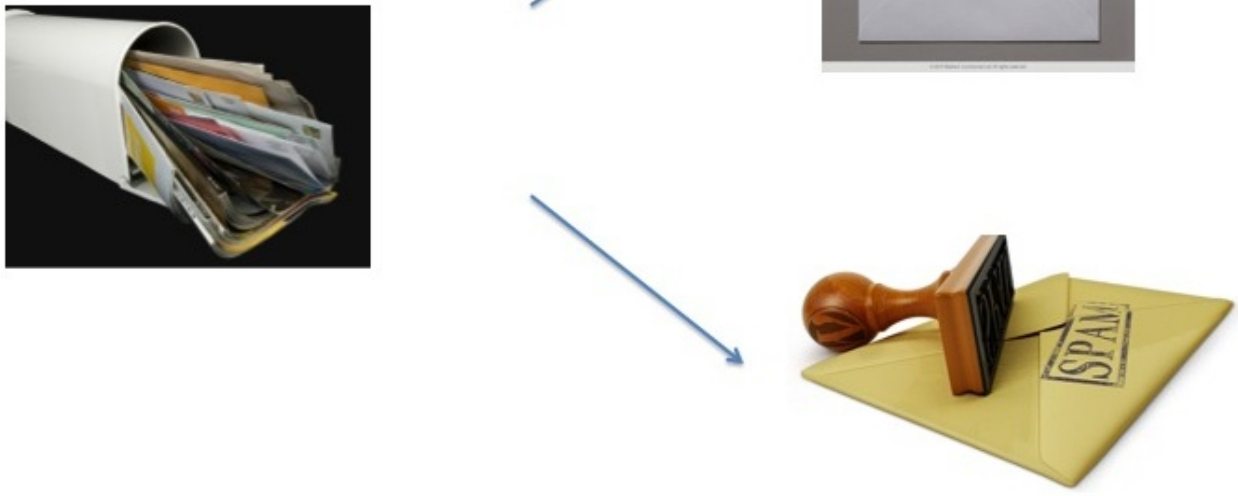
Ensemble d'apprentissage

Algorithme d'apprentissage

Classifieur h

Evaluer le détecteur de SPAM

- Mesurer le nombre d'erreurs commises par h sur un ensemble de messages jamais vus par l'algorithme d'apprentissage



Classification binaire supervisée

Cadre probabiliste et statistique

Soit X un vecteur aléatoire de $\mathcal{X} = \mathbb{R}^p$
X décrit ici les caractristiques ("features") d'un message
Y une variable aléatoire discrète $\mathcal{Y} = \{-1, 1\}$
Soit P la loi de probabilité jointe de (X,Y), loi fixée mais inconnue
Supposons que $S_{app} = \{(x_i, y_i), i = 1, \ldots, n\}$ soit un échantillon i.i.d. tiré de la loi P

Classification binaire supervisée

Cadre probabiliste et statistique

- A partir de S_{app}, déterminer la fonction $h \in$ qui minimise $R(h) = \mathbb{E}_P[\ell(X, Y, h(X)]$
- ℓétant une fonction de coût local qui mesure à quel point la vraie classe et la classe prédite du classifieur sont différentes
- Par exemple: $\ell(, y, h(\mathbf{x})) = 0$ if $h(\mathbf{x}) = y$ and 1 otherwise

Risque $R(h)$

$$R(h) = \sum_{y=-1,1} P(Y = y) \int_{\mathbb{R}^p} \ell(h, x, y) p(x|Y = y) dx$$

Pb: la loi jointe n'est pas connue

Dans un monde idéal ...

Si je connaissais toutes les probabilités en jeu, je pourrai calculer :

- $R(h) = \mathbb{E}_P[\ell(X, Y, h(X))]$
- Utiliser les probabilités conditionnelles aux classes pour définir le classifieur de Bayes:
- $\mathrm{h}_{Bayes}(\mathbf{x}) = \arg\max_{y \in \{-1,+1\}} P(Y = y|\mathbf{x})$

Formule de Bayes

- $P(Y = k|x) = \frac{p(x|Y=k)P(Y=k)}{p(x|Y=-1).P(Y=-1)+p(x|Y=1).P(Y=1)}$

Classifieur bayésien

Definition

$$h_{Bayes}(x) = argmax_{k=1,-1} P(Y = k|x)$$

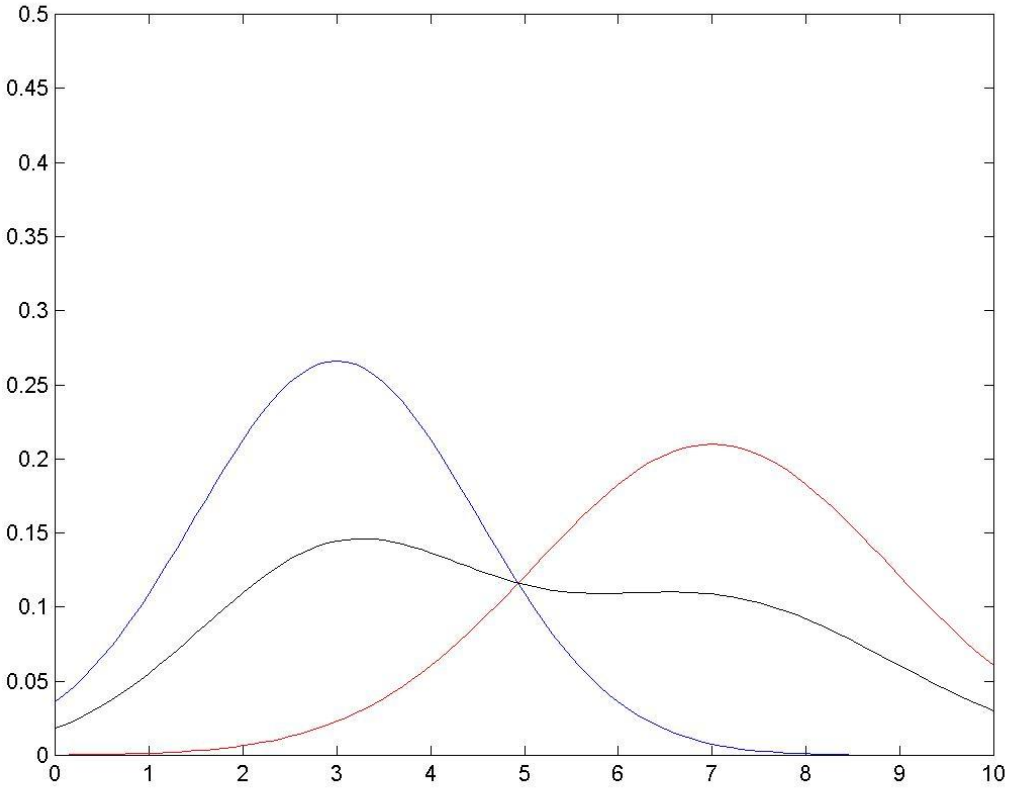
.

Risque bayesien

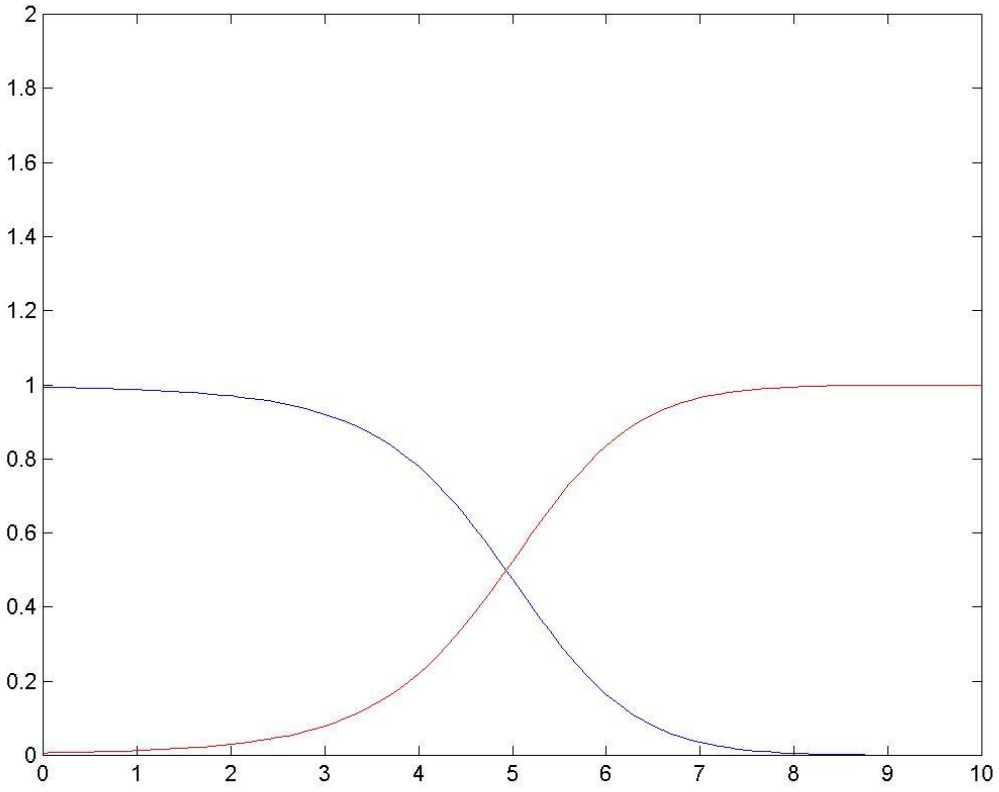
$$\begin{aligned} R(h_{bay}) &= \int_{R_1} P(h_{bay}(x) \neq 1)p(x)dx + \int_{R_{-1}} P(h_{bay}(x) \neq -1)p(x)dx \quad (1) \\ &= \int_{R_1} P(y = -1|x)p(x)dx + \int_{R_{-1}} P(y = 1|x)p(x)dx \quad (2) \end{aligned}$$

On démontre qu'il s'agit du meilleur classifieur .

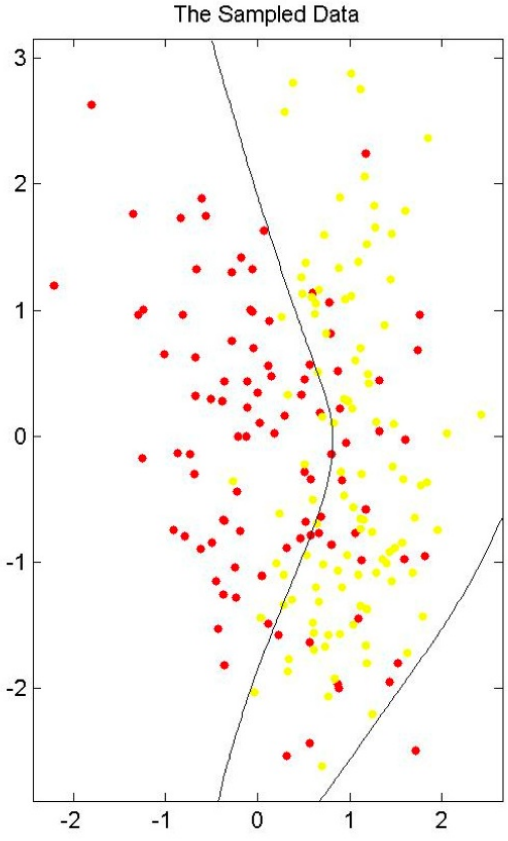
Exemple en 1D avec des lois conditionnelles normales

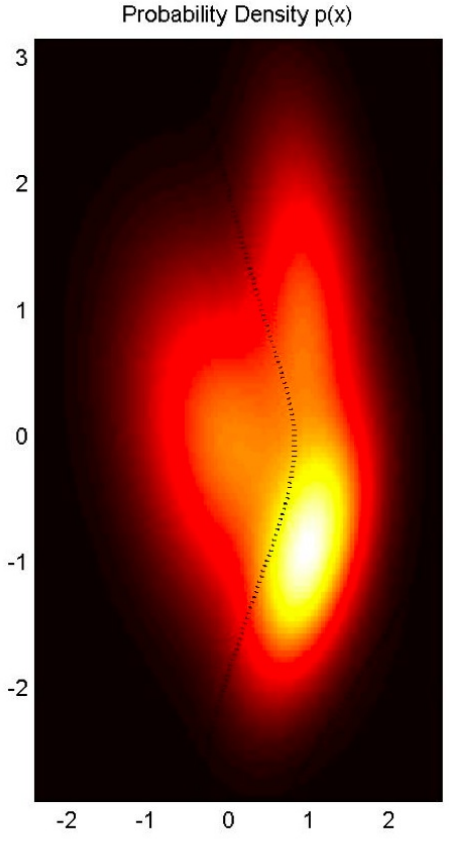


Classifieur bayesien avec des lois conditionnelles normales

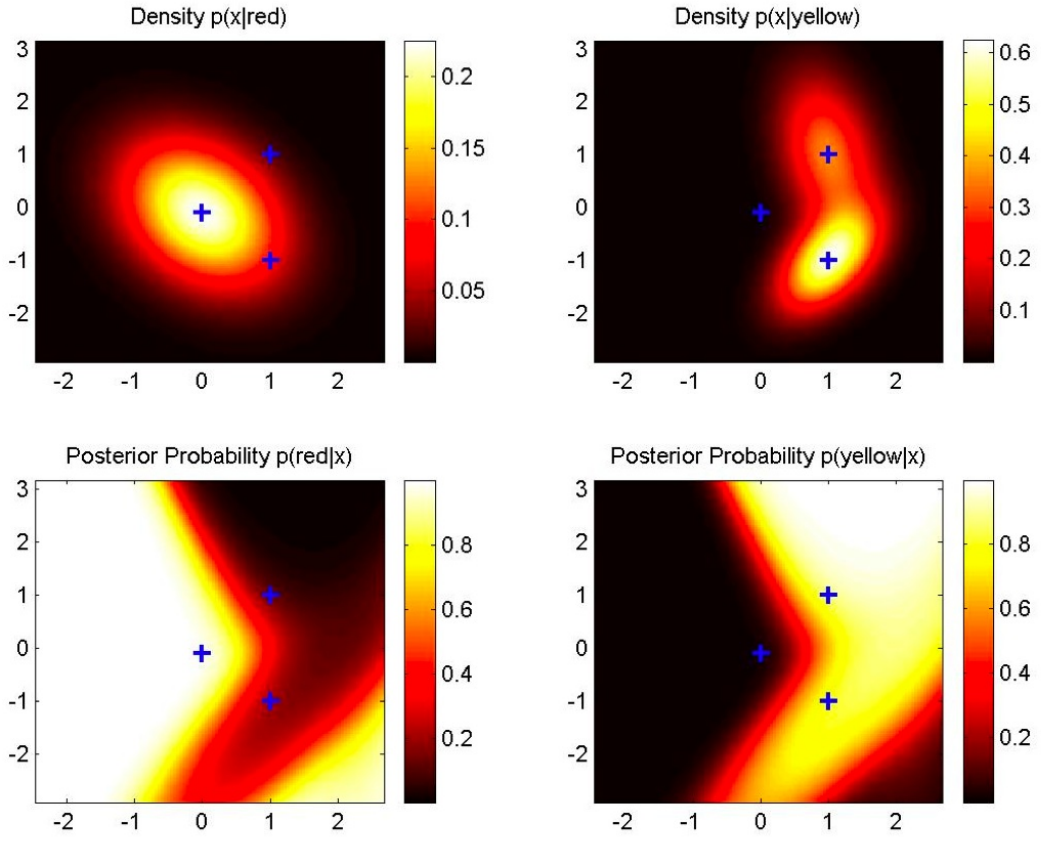


Exemple en 2D

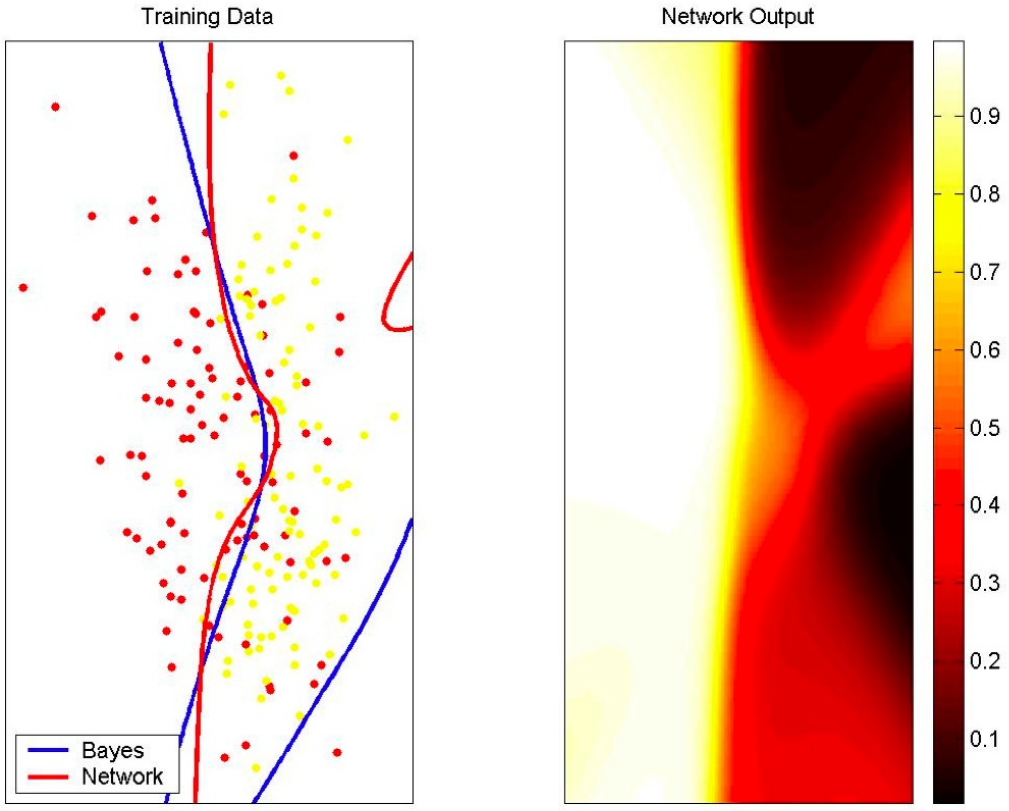




Exemple en 2D



En utilisant un ensemble d'apprentissage



Take-home message

- La fonction cible pour la perte $0-1$ en classification supervisée est le classifieur de Bayes
- On ne peut pas obtenir un risque plus petit que le risque bayesien: $R(h_{Bayes})$ qui est une caractéristique du problème
- NB : en régression, la fonction cible pour la perte quadratique est l'espérance conditionnelle $h(x) = \mathbb{E}[Y|x]$

Apprentissage statistique en pratique

On distingue deux types d'approches:

1. les approches dites "génératives": $h(\mathbf{x}) = P(Y \hat{=} 1|\mathbf{x})$ et h est fondée sur la modélisation des probabilités conditionnelles de chaque classe: $p(\mathbf{x}|Y = 1)$ et $p(\mathbf{x}|Y = -1)$
2. les approches dites "discriminantes": avec $h(\mathbf{x})$ on essaie de discriminer entre les classes sans modélisation des probabilités conditionnelles

Apprentissage statistique - en pratique

Pb: la loi jointe n'est pas connue : on ne peut pas calculer $R(h)$

Exemple de l'approche par régularisation

- A la place de $R(h)$, on minimise la somme de deux termes:
- le risque empirique $R_n(h) = \frac{1}{n}\sum_i L(x_i, y_i, h(x_i))$ et un terme régularisateur $\Omega(h)$ qui mesure la "complexité" de h.
- On cherche : $\hat{h} = \arg\min_{f \in \mathcal{F}} R_n(h) + \lambda\Omega(h)$

Apprentissage statistique - en pratique

Pb: la loi jointe n'est pas connue : on ne peut pas calculer $R(h)$

Exemple de l'approche par régularisation

- A la place de $R(h)$, on minimise la somme de deux termes:
- le risque empirique $R_n(h) = \frac{1}{n}\sum_i L(x_i, y_i, h(x_i))$ et un terme régularisateur $\Omega(h)$ qui mesure la "complexité" de h.
- On cherche : $\hat{h} = \arg\min_{f \in \mathcal{F}} R_n(h) + \lambda\Omega(h)$

NB : on cherche à obtenir un compromis entre une bonne adéquation aux données et une complexité limitée : $\Omega(h)$ est en général choisi pour renforcer la régularité de la fonction

Et en pratique, comment fait-on ?

Méthodologie pour développer une approche discriminante

- Définir
  - l'**espace de représentation** des messages

Et en pratique, comment fait-on ?

Méthodologie pour développer une approche discriminante

- Définir
  - l'**espace de représentation** des messages
  - la **classe des fonctions** de classification binaire considérées

Et en pratique, comment fait-on ?

Méthodologie pour développer une approche discriminante

- Définir
  - l'**espace de représentation** des messages
  - la **classe des fonctions** de classification binaire considérées
  - la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe

Et en pratique, comment fait-on ?

Méthodologie pour développer une approche discriminante

- Définir
  - l'**espace de représentation** des messages
  - la **classe des fonctions** de classification binaire considérées
  - la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe
  - l'**algorithme de minimisation** de cette fonction de coût

Et en pratique, comment fait-on ?

Méthodologie pour développer une approche discriminante

- Définir
  - l'**espace de représentation** des messages
  - la **classe des fonctions** de classification binaire considérées
  - la **fonction de coût** à minimiser pour obtenir le meilleur classifieur dans cette classe
  - l'**algorithme de minimisation** de cette fonction de coût
  - une **méthode de sélection de modèle** pour définir les hyperparamètres

Coder les documents

Codage Term-Frequency-Inverse Document Frequency (TF-IDF)

- une collection C de messages (documents)
- un mot $\longrightarrow$ un terme
- à définir : un dictionnaire D de p termes apparaissant dans C
- un message (document) $d \longrightarrow$ un ensemble de termes avec leur occurrence
- C: a collection of N documents
- $TF(t,d) = \frac{\text{nb d'occurrence de t dans d}}{\text{nb de termes dans d}}$
- $IDF(t,C) = \log \frac{\text{N}}{\text{nb de documents de } C \text{ où t apparaît}}$

Espace de représentation des messages

Codage TF-IDF d'un message *d*

- un vecteur **x** de dimension p
- $x_i = TF - IDF(t_i, d, C), i = 1, \ldots, p$
- On prend : C = S_{app}, documents de l'échantillon d'apprentissage

Espace de représentation des données

- $\mathcal{X} = \mathbb{R}^p$

Classe des fonctions de classification

Aujourd'hui, au programme

1. Classifieur linéaire
2. Classification non linéaire

Outline

Analyse discriminante linéaire : 2 classes

On s'intéresse au classifieur suivant:

- Soit $\mathbf{x} \in \mathbb{R}^p$,
  - $h_{LDA}(x) = 1$ si $\log\left(\frac{P(Y=+1|\mathbf{x})}{P(Y=-1|\mathbf{x})}\right) \geq 0$, -1 sinon.
- Hypothèse 1: $p(\mathbf{x}|Y=+1)$ (resp. $p(\mathbf{x}|Y=-1)$) est une densité normale $\mathcal{N}(\mu_+, \Sigma)$ (resp. $\mathcal{N}(\mu_-, \Sigma)$)
- Hypothèse 2 : La probabilité *a priori* $P(Y=+1) = p_1$.

Question: quelle est la forme du classifier ainsi construit ?

Réponse

Formule de Bayes:

$$P(Y = i|\mathbf{x}) = \frac{p(\mathbf{x}|Y = i)P(Y = i)}{p(\mathbf{x})}$$

Puis, on cherche à définir la frontière de décision induite par le classifieur LDA:

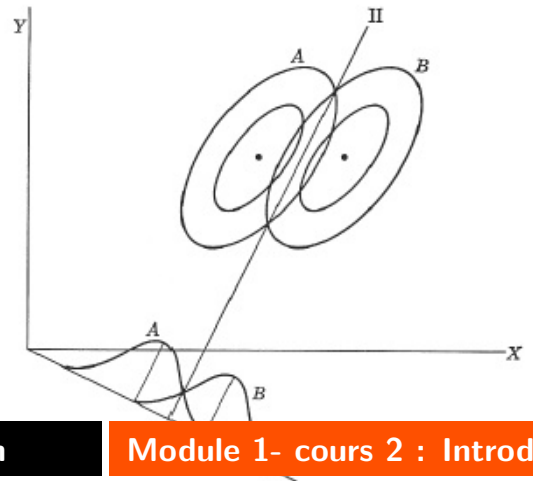
Analyse Discriminante Linéaire

$$\log\left(\frac{P(Y=+1|\mathbf{x})}{P(Y=-1|\mathbf{x})}\right)=0$$

$$\text{soit } \log\left(\frac{p(\mathbf{x}|Y=1)P(Y=1)}{p(\mathbf{x}|Y=-1)P(Y=-1)}\right)=0$$

$$\log(\frac{p_1}{1-p_1})+\log(\frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}})-\frac{1}{2}(\mathbf{x}-\mu_+)^T\Sigma^{-1}(\mathbf{x}-\mu_+)-\log(\frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}})+\frac{1}{2}(\mathbf{x}-\mu_-)^T\Sigma^{-1}(\mathbf{x}-\mu_-)=0$$

$$\mathbf{x}^T\Sigma^{-1}(\mu_+-\mu_-)+\log(\frac{p_1}{1-p_1})-\frac{1}{2}(\mu_+-\mu_-)^T\Sigma^{-1}(\mu_+-\mu_-)=0$$



Estimation des paramètres (LDA)

- Prendre les estimations empiriques définies à partir des données
- $S_+ = \{(x_i, y_i) \in S,\ s.t\ y_i = 1\}$
- $S_- = \{(x_i, y_i) \in S,\ s.t\ y_i = -1\}$

$$\hat{\mu_+} = \frac{1}{|S_+|} \sum_{x_i \in S_+} x_i$$

$$\hat{\mu_-} = \frac{1}{|S_-|} \sum_{x_i \in S_-} x_i$$

$$\hat{\Sigma} = \frac{1}{2} \left(\frac{1}{|S_+|} \sum_{x_i \in S_+} (x_i - \hat{\mu}_+)(x_i - \hat{\mu}_+)^T + \frac{1}{|S_-|} \sum_{x_i \in S_-} (x_i - \hat{\mu}_-)(x_i - \hat{\mu}_-)^T \right)$$

Outline

Classifieur linéaire

Définition

Supposons $\mathbf{x} \in \mathbb{R}^p$

$f(\mathbf{x}) = \text{signe}(\mathbf{w}^T\mathbf{x} + w_0)$

L'équation : $\mathbf{w}^T\mathbf{x} + w_0 = 0$ définit un hyperplan dans l'espace euclidien $\mathbb{R}^p$

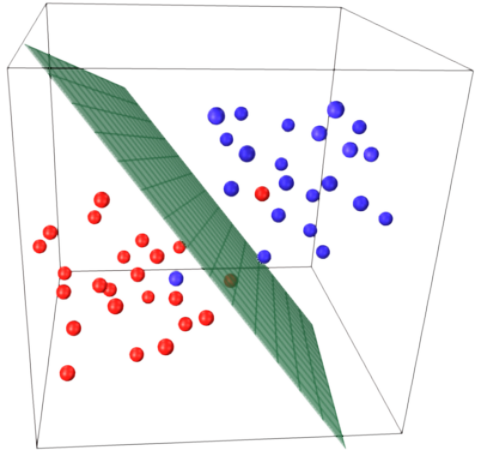


Figure en dimension 2

Comment apprendre un classifier linéaire (approche discriminante)?

- Algorithme originel du perceptron (Rosenblatt 1957, 1959), algorithme de descente de gradient pour le perceptron
- D'autres algorithmes que nous verrons plus tard (22 et 23 Juin) : hyperplan de marge optimale

A linear classifier: the formal neuron and perceptron

- First model proposed by McCullogh and Pitts (physiologists) in 1943 to model the activity of a neuron
- Input signals represented by a vector $\mathbf{x}$ is processed by a neuron whose weighted synapses are linked to the input
- The neuron computes a weighted sum of the components of the signal
- Rosenblatt proposed a learning rule in 1959

Formal neuron and perceptron

- $h_{perc}(\mathbf{x}) = sign(\mathbf{w}^T.\mathbf{x})$
- $sign(a) = 1$ if $a \geq 0$ and -1 otherwise

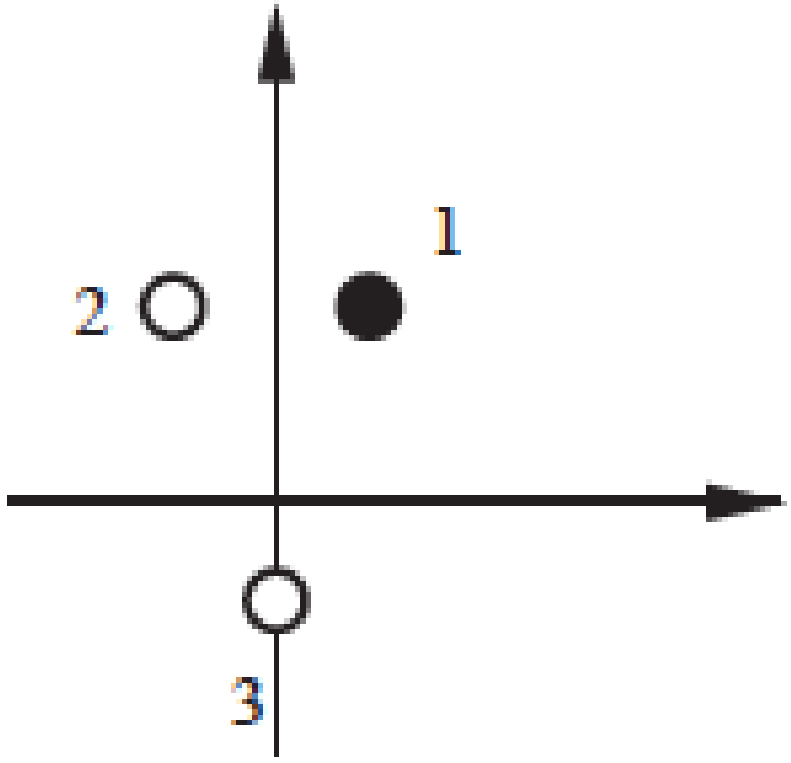
Données d'apprentissage:

- $\mathcal{S} = \{(\mathbf{x_1}, \mathbf{y_1}), ..., (\mathbf{x_n}, \mathbf{y_n})\}$
- $\mathbf{x}_i \in \mathbb{R}^{p+1}$: the 0^{th} *componentisfixedto*$1.\mathrm{y}_i \in \{-1, +1\}$

Apprendre un perceptron (algorithme classique)

- Fonction de perte :
  - $L(\mathbf{w}) = \sum_{i=1}^{n}(1 - comp(y_i, h_{perc}(\mathbf{x}_i))$
  - comp(y,y') $= 1$ si $y = y'$ et 0, sinon
  - Avec L je compte le nombre d'erreurs en apprentissage

Un classifieur linéaire



Algorithme classique du perceptron

Algorithme

- STOP = faux
- Jusqu'à ce que STOP soit vrai:
- Pour i=1 à n
  - Si $y_i \neq sign(\mathbf{w}^T \mathbf{x}_i)$, alors je corrige: w(nouveau) = w(ancien) + $y_i.\mathbf{x}_i$
- STOP = $(L(\mathbf{w}(nouveau) = \mathbf{w}(ancien))$

Convergence de l'algorithme du perceptron

Convergence

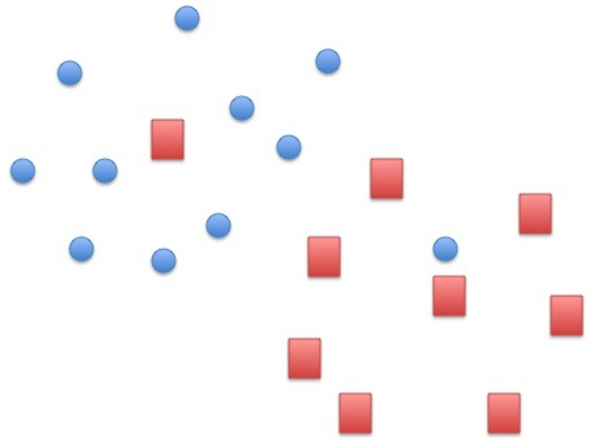
L'algorithme converge si les données sont exactement linéairement séparables

Deux types de non séparabilité:

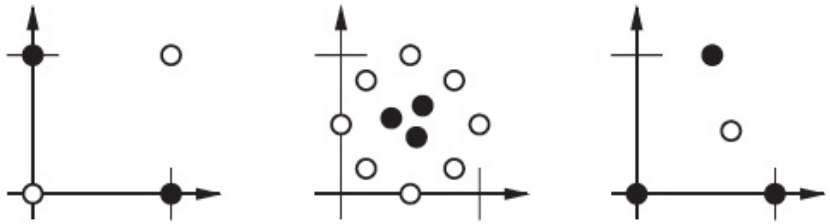
1. Données presque " linéairement séparables": bruit dans les données
2. Données séparables mais avec une frontière non linéaire
3. NB : on cumule en général les deux difficultés

Limites d'un perceptron 1

Exemple 1 de données non séparables:



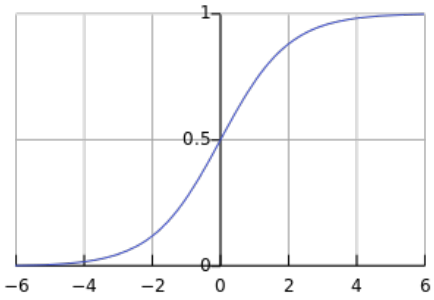
Limites d'un perceptron 2



- **Premier problème :** XOR problem: un perceptron seul ne peut implémenter une fonction XOR
- Solution :
  - Soit on rajoute une couche de neurones avant le neurone de sortie perceptron mulit-couches (algorithme d'apprentissage par rétro-propagation du gradient), Werbos 1974, Le Cun 1985, Rumelhart et al. 1986.
  - Soit on transforme les données en les plongeant dans un espace où elles sont linéairement séparables (see Practical session)

Apprendre un perceptron(vue plus générale)

- Remplacer la fonction signe par une sigmoide différentiable
  - $sigm(x) = \frac{1}{1+\exp(-\frac{1}{2}x)}$



- Définir une fonction de perte différentiable
  - $\ell_i(\mathbf{w}) = (y_i - \ sigm(\mathbf{w}^T\mathbf{x}))^2$
  - $L(\mathbf{w}) = \sum_i \ell_i(\mathbf{w})$

Apprendre un perceptron(vue plus générale)

Perceptron algorithm (gradient-like version)

- STOP = faux
- ϵ; nbIter; $j = 0$; $t = 0$
- Initialiser $\mathbf{w}_0$
- Jusqu'à ce que STOP soit vrai:
  - Pour i =1 jusqu'à n:
    - $\mathbf{w^{t+1}} = \mathbf{w^t} - \eta \nabla_{\mathbf{w}} \ell_{\mathbf{i}}(\mathbf{w})$
    - $t \rightarrow t + 1$
- $j \rightarrow j + 1$

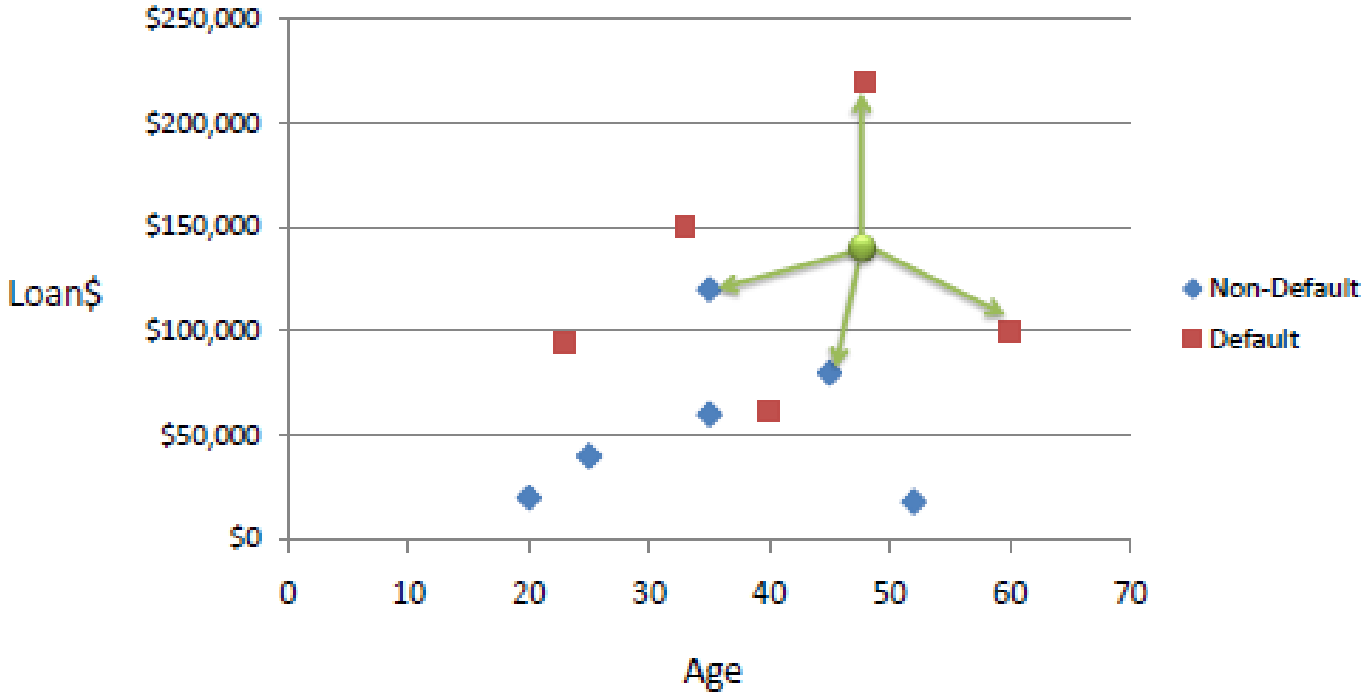
- STOP = $(L(||\mathbf{w}(nouveau) - \mathbf{w}(ancien)|| < \epsilon)$ et $(\text{nbIter} \leq nbMax)$

Perceptron

- Early stopping: arrêter avant de sur-apprendre (nbIter petit)
- Eviter le sur-apprentissage : contrôler la norme du vecteur $\mathbf{w}$ pendant l'apprentissage
  - La fonction de perte devient : $L(\mathbf{w}) = \sum_i \ell_i(\mathbf{w}) + \lambda||\mathbf{w}||^2$
- Variante intéressante (celle utilisée en pratique)
  - Descente de gradient local et stochastique (Bottou 1991: applicaiton aux réseaux de neurones)

Outline

Algorithme des K-plus-proches voisins



Algorithme des K-plus-proches voisins

K-PPV (en anglais K-Nearest neighbors: K-NN)

Cas 2 classes:

$$h_{KNN}(x) = \arg \max_{y \in \{-1,1\}} \frac{N_y^K(x)}{K},$$

avec :

- Soit K un entier strictement positif.
- Soit d une métrique définie sur $\times$
- $S = \{(x_i, y_i), i = 1, \ldots, n\}$
- Pour une donnée x, on définit σ la permutation d'indices dans $\{1, \ldots, n\}$ telle que:
  - $d(x, x_{\sigma(1)}) \leq d(x, x_{\sigma(1)}) \ldots \leq d(x, x_{\sigma(n)})$
- $S_x^K = \{x_{\sigma(1)}, \ldots, x_{\sigma(K)}\}$: K premiers voisins de x
- $N_y^K(x) = |\{x_i \in S_x^K, y_1 = y\}|$

Le paramètre de lissage K

K: trop petit : la fonction f est trop sensible aux données : large variance

K : trop large : la fonction f devient trop peu sensible aux donnes : biais important

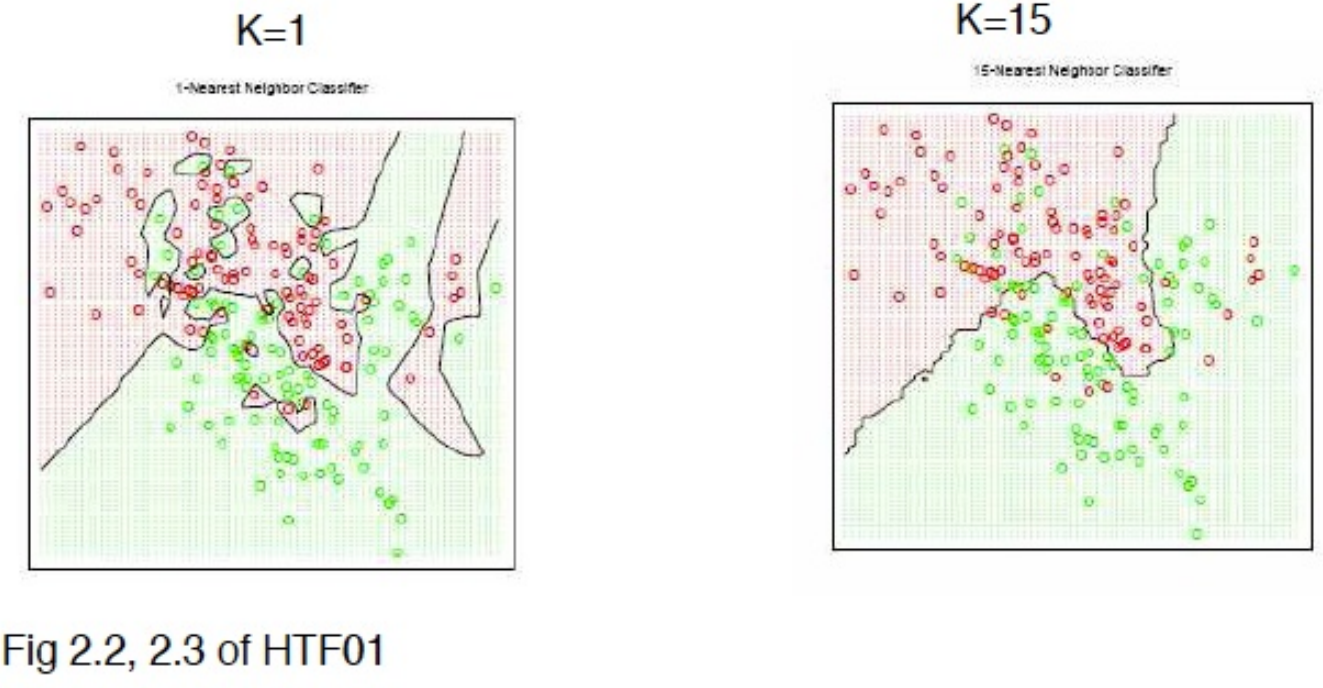
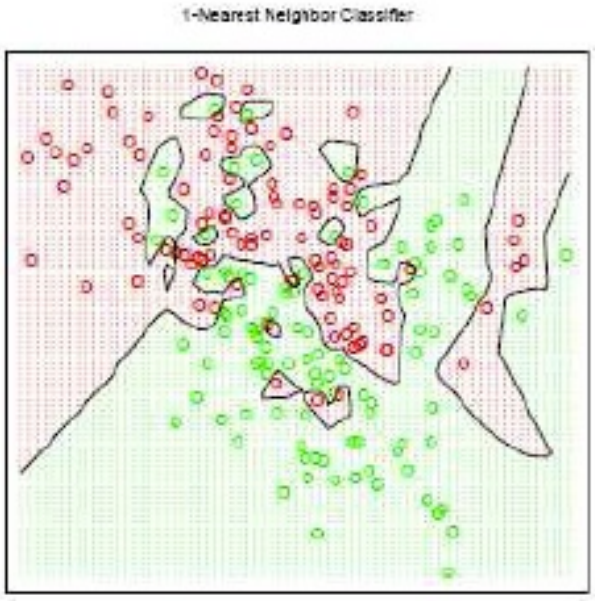


Fig 2.2, 2.3 of HTF01

Book of Hastie, Tibshirani and Friedman (The elements of statistical learning, Springer)

Le paramètre de lissage K

K=1



K=15

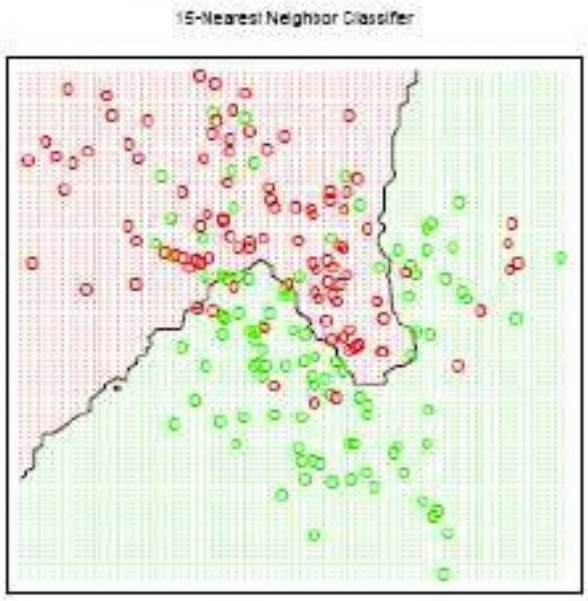


Fig 2.2, 2.3 of HTF01

Book of Hastie, Tibshirani and Friedman (The elements of statistical learning,

Erreur de test en fonction de $\frac{n}{K}$