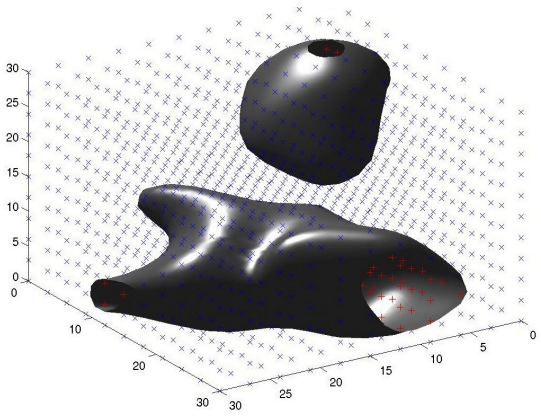
Cours GMM

Introduction au Data Mining

1.0

DT_GMM3



Décembre 2009

Légende

Table des matières

Introduction

Ce support de cours est un complément au cours magistral et n'est en aucun cas auto-suffisant.

Méthodologie générale

I

A. Data-Mining ?

Domaines d'application

Le data-mining, au même titre que les statistiques, entre dans à peu près tous les domaines :

- l'infini petit : la génomique
- l'infini grand : l'astrophysique
- le quotidien : la relation client
- le singulier : aide au pilotage aéronautique
- l'ouvert : le e-commerce
- le sécuritaire : détection de fraude de carte bancaire ou sim, prévention du terrorisme
- théorique : enquêtes en sciences humaines, études biologiques, médicales...
- industriel : contrôle qualité, pilotage de production
- alimentaire : études agronomiques
- divertissement : prédiction d'audience TV

Finalités

Les finalités sont :

- compréhension et modélisation de phénomènes
- mais surtout : aide à la décision (parfois en temps réel) en limitant la subjectivité humaine

Définition : Data-Mining

C'est l'ensemble des méthodes et techniques destinées à l'exploration et l'analyse de (souvent grandes) bases de données informatiques, de façon automatique ou semi-automatique en vue de détecter dans ces données des règles, des associations, des tendances inconnues ou cachées, des structures particulières restituant l'essentiel de l'information utile tout en réduisant la quantité de données.

Différents types

Data-mining descriptif :

Ce que l'on appelle **l'exploration de données** : mettre en évidence des informations présentes mais cachées par le volume.

- Techniques de classification ou d'association (*clustering*).

Data-mining prédictif :

Ce que l'on appelle **l'explication de données** : extrapoler de nouvelles informations à partir de celles présentes.

- Qualitatif : Techniques de classement ou discrimination (*classification*) ou de score (*scoring*)
- Quantitatif : Techniques de prédiction ou régression (*regression*).

B. Les données

Définition : Terminologie

- variable : toute caractéristique d'une entité
- mesure : expression par une valeur numérique d'une variable
- attribut : expression par un code d'une variable
- modalité d'une variable : ensemble des valeurs que peut prendre une variable
- individu : l'entité étudiée (personne, objet, événement,...), aussi appelée observation

L'organisation

- en base de données relationnelle
- en base de données multi-dimensionnelle (OLAP)
- en base de données géographique
- en fichier plat type matrice : les lignes correspondent aux individus, les colonnes aux variables

Constitution de la base d'analyse prédictive :

- variable explicative : qui décrit, est une cause
- variable à expliquer : cible de l'analyse, est une conséquence : aussi appelée l'étiquette (*label*)

Exemple : Exemple de mise en forme de données

Nom	Genre	Taille	...	Enfants	Client fidèle
M. X	homme	1m80	...	2 enfants	oui
M. Y	homme	1m70	...	0 enfant	non
...	...	...	...	...	...

Tableau 1 Exemple de mise en forme

Pour utilisation, la mise en forme deviendra :

$$data = \begin{bmatrix} 1 & 1.7 & ... & 2 \\ 1 & 1.8 & ... & 0 \\ ... & ... & ... & ... \end{bmatrix} \quad label = \begin{bmatrix} 1 \\ -1 \\ ... \end{bmatrix}$$

Les données : exploration et préparation

- fiabiliser, remplacer ou supprimer les données incorrectes

- créer des indicateurs pertinents
- réduire le nombre de dimensions

Complément : Fiabiliser les données

Exemples de problèmes à détecter :

- individu avec trop de valeurs manquantes
- individu avec des valeurs aberrantes (*outlier*)
- variable en anomalie pour de nombreux individus

Exemple de traitement :

- suppression d'un individu
- correction de valeurs
- suppression de variable pour l'ensemble de la base

Complément : Créer des indicateurs

A partir des données brutes :

- remplacer des grandeurs absolues par des ratios
- normaliser
- calculer des évolutions temporelles (rapport entre la moyenne sur une période récente et la moyenne sur une période antérieure)
- combiner linéairement des variables
- composer des variables avec des fonctions (type logarithme)
- recoder une variable ("faible, moyen,fort" devient "1,2,3")
- remplacer les dates par des durées
- remplacer les lieux par des coordonnées

Complément : Réduire la dimension

Sur le nombre d'individus

- au cours de la fiabilisation
- au cours de l'échantillonnage (dans la suite du cours)

Sur le nombre de variables

- détecter les variables très corrélées
- détecter les variables non pertinentes pour le problème posé
- utiliser des analyses factorielles pour combiner les variables

Sur le nombre de modalités

- pour les variables discrètes et qualitatives, regrouper les modalités trop nombreuses ou bien presque vides
- discrétiser des variables continues

C. Les techniques

1. Terminologie

Méthodes descriptives

- modèles géométriques (ACP, nuées dynamiques, centres mobiles, méthodes hiérarchiques, cartes de Kohonen...)
- modèles combinatoires (classification relationnelle)

- modèles à base de règles logiques (recherche d'associations, de séquences similaires)

Méthodes prédictives

- modèles à base de règles logiques (arbres de décision)
- modèles à base de fonctions mathématiques (réseaux de neurones, SVM, régression linéaire, régression logistique, analyse discriminante...)
- prédiction sans modèle (k-plus proches voisins)

D. L'évaluation

Qualités attendues d'une technique de classement ou prédiction

- La précision: utilisation de la courbe ROC ou de l'indice de Gini pour évaluer la précision
- La robustesse : le modèle doit dépendre le moins possible de l'échantillon d'apprentissage utilisé, varier peu en cas de valeurs manquantes...
- La concision : ou **parcimonie** : modèle le plus simple s'appuyant sur le moins de choses possibles
- Des résultats explicites : pouvoir interpréter le résultat
- La diversité des types de données manipulées : possibilité de traiter des données discrètes ou manquantes
- La rapidité de calcul du modèle : possibilité de mieux régler le modèle
- Les possibilité de paramètrage : pondération des erreurs par exemple

Définition : Pouvoir de généralisation

La généralisation, c'est la capacité d'un modèle à être efficace sur des données qui n'ont pas servi à sa conception.

Groupes de données

On distingue en général trois groupes de données :

- les données d'apprentissage : servent à calculer le modèle
- les données de validation : servent à vérifier la généralisation du modèle pour les paramètres courants
- les données de test : ne sont utilisées qu'une fois les paramètres définitivement fixés (équivalent des données d'application).

Toute donnée utilisée une fois en test ne devrait jamais resservir dans ce groupe : elle devient automatiquement une donnée de validation. Il est conseillé d'isoler physiquement les données de test tout au long de la conception ou d'une étude de data-mining pour ne pas risquer une utilisation malencontreuse, qui fausserait l'évaluation de la capacité réelle de généralisation.

La mauvaise généralisation

Peut être due :

- Si le taux d'erreur sur les données d'apprentissage est faible ou nul :
  - à un nombre insuffisant de données d'apprentissage : les données ne sont pas représentatives du problème à traiter
  - au sur-apprentissage : le modèle apprend **par-coeur** les données d'apprentissage sans parvenir à en tirer de l'information (il faut alors changer le réglage du modèle). Exemple simple de sur-apprentissage : l'interpolation exacte par polynôme de Lagrange...

- Si le taux d'erreur sur les données est plutôt élevé
  - à un mauvais réglage des paramètres
  - à un mauvais choix de représentation des données
  - à l'absence de l'information recherchée dans les données

La validation : comment régler efficacement un modèle ?

L'ensemble de validation sert à évaluer la performance probable d'un modèle.

Méthode naïve

```
Xapp ← données d'apprentissage
Xval ← données de validation
parametres ← plage de n valeurs
pour i ←1 à n faire
    calculer le modèle pour paramètres(i) et Xapp
    erreurApp(i) ← calcul de l'erreur sur Xapp
    erreurVal(i) ← calcul de l'erreur sur Xval
finpour
choisir les paramètres qui minimisent erreurVal
vérifier que l'ordre de grandeur de l'erreur en apprentissage est similaire à celle de l'erreur en validation
```

Algorithme de validation naïve

Attention : Danger de la méthode

Le choix des données de l'ensemble de validation est critique : si cet ensemble n'est pas représentatif, alors le modèle choisi risque de ne pas bien généraliser. D'une certaine façon, on risque de sur-apprendre l'ensemble de validation.

Validation Croisée

Validation croisée (*k-fold cross validation*)

```
parametres ← plage de n valeurs
découper les données en k groupes de taille équivalente
pour j ←1 à k faire
    Xapp ← données des groupes 1 :j-1 et j+1 :k
    Xval ← données du groupe j
    pour i ←1 à n faire
        calculer le modèle pour paramètres(i) et Xapp
        erreurApp(j,i) ← calcul de l'erreur sur Xapp
        erreurVal(j,i) ← calcul de l'erreur sur Xval
    finpour
finpour
choisir les paramètres qui minimisent erreurVal en moyenne sur les différents découpages des données
vérifier que l'ordre de grandeur de l'erreur en apprentissage est similaire à celle de l'erreur en validation
```

Algorithme de validation croisée

Effets de la validation croisée

- Le moyennage de l'erreur en validation sur différents groupes de données permets de limiter l'impact de données non représentative.
- La méthode convient lorsqu'il y a peu de données, car toutes servent à la fois à apprendre et à valider
- La méthode du *Leave-One-Out* pousse le concept à l'extrême en posant k = nombre d'exemples disponibles. Très utile pour les bases d'apprentissage vraiment petites.
- Méthode gourmande en temps!

Les outils de test

- la matrice de confusion
- la courbe ROC

La matrice de confusion

La matrice de confusion permet de mesurer le taux d'erreur (ou taux de mauvais classement) par classe.

Prédit / Réalisé	Classe 1	Classe 2
Classe 1	300	2
Classe 2	40	250

Tableau 2 Exemple de matrice de confusion

La matrice de confusion permet de facilement visualiser quelles classes sont bien apprises et lesquelles posent plus de problèmes.

La courbe ROC

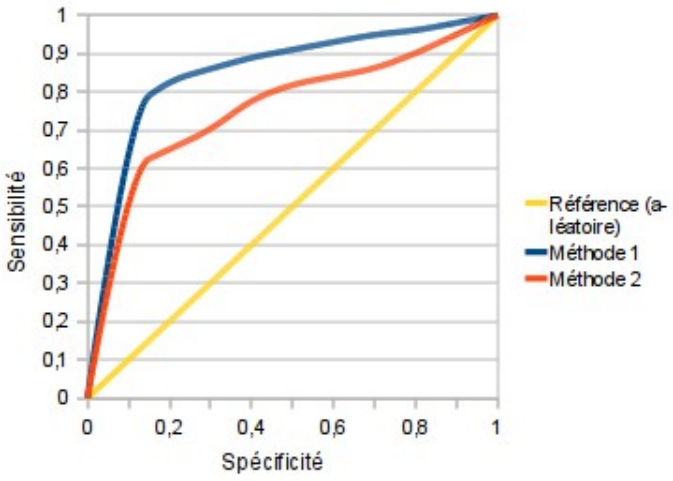
La courbe ROC (*Receiver Operating Characteritics*) est une technique issue du traitement du signal. Elle représente la proportion d'événements (classe c_1 prédits c_1) détectés comme tels en fonction de la proportion de faux événements (classe c_2 prédits c_1). Les détections se font en fonction d'un seuil s que l'on fait varier.

Plus précisément, on définit deux fonctions de s

- la sensibilité $\alpha(s) = \mathbb{P}(score(x) \geq s \quad | \quad y = c_1)$
- la spécificité $\beta(s) = \mathbb{P}(score(x) < s \quad | \quad y = c_2)$

et l'on peut dire que la proportion de faux événements parmi les non-événements est $1 - \beta(s) = \mathbb{P}(score(s) \geq s \quad | \quad y = c_2)$

La courbe ROC représente donc $\alpha(s)$ en fonction de $1 - \beta(s)$ pour des valeurs de s allant du maximum (on considère tous les individus comme des non-événement, d'où $\alpha(s) = 1 - \beta(s) = 0$) au minimum (on considère tous les individus comme événement, d'où $\alpha(s) = 1 - \beta(s) = 1$.



courbe ROC

Le calcul de la courbe ROC se sert de la matrice de confusion à chaque valeur du seuil. On peut facilement calculer la sensibilité et la spécificité à partir de la matrice de confusion :

P/R	oui	non
oui	M_{11}	M_{12}
non	M_{21}	M_{22}

Tableau 3 Matrice de confusion au seuil s

On a $\alpha(s) = M_{11}/(M_{11} + M_{12})$ et $1 - \beta(s) = 1 - (M_{22}/(M_{21} + M_{22}))$.

La courbe ROC permet :

- De comparer des modèles différents avec un critère neutre
- De comparer les comportements globaux et locaux des méthodes : une méthode peut-être globalement moins bonne mais localement meilleure (et vice-versa).

L'aire sous la courbe (AUC : *Air Under Curve*) permet de comparer les modèles avec une seule valeur. Un AUC de 0.5 correspond en général au hasard, un AUC de 1 serait un modèle parfait (qui ne fait jamais aucune erreur quelque soit le seuil).

Méthodes d'apprentissage

II

Objectifs

Nous allons voir la séparation linéaire de deux classes, avec en particulier l'algorithme du perceptron et la séparation linéaire à vastes marges. A partir de là, nous allons regarder deux méthodes permettant de traiter des données plus complexes, c'est-à-dire non linéairement séparable. D'une part, nous verrons les réseaux de neurones qui peuvent être vus comme un extension du perceptron (le perceptron multi-couche), dans lesquels on introduit la non-linéarité dans l'ajout de couches et l'utilisation de fonctions d'activation non linéaires (on complexifie le modèle). D'autre part, nous étudierons le SVM, méthode dont le principe est de séparer linéairement les données projetées dans un espace de grande dimension (espace bien choisi dans lequel les données se trouvent linéairement séparables), via l'astuce du noyau.

A. Le perceptron

Le plus simple de réseaux de neurones

Un perceptron est un réseau de neurone sans couche cachée. Il se résume à un neurone par variable sur la couche d'entrée plus un autre dont la valeur d'entrée sera constante à 1 (afin d'obtenir un biais) et à la couche de sortie.

La règle de décision est linéaire :

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

L'algorithme d'apprentissage

Les poids du perceptron sont initialisés aléatoirement.

Les individus sont présentés succéssivement au perceptron. Si la sortie effective du perceptron correspond à l'étiquette, les poids ne sont pas modifiés.

Dans le cas contraire on applique la mise à jour suivante :

$$w^{k+1} = w^k - \alpha(y - f(x))x \quad \text{avec} \quad 0 < \alpha < 1$$

α est le taux d'apprentissage et est un paramètre de la méthode.

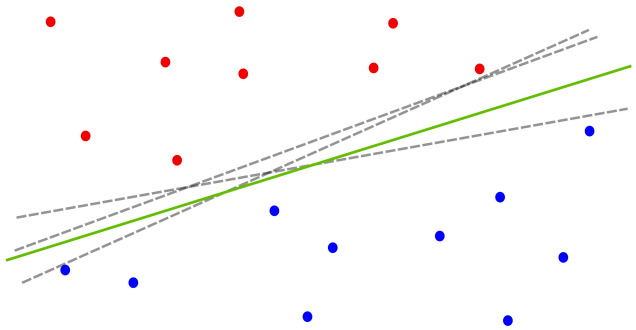
B. Séparateurs à Vaste Marge Linéaire

Séparation linéaire

On s'intéresse dans un premier temps au cas de deux classes linéairement séparables, c'est-à-dire qu'il existe au moins une droite (en général, une infinité) expliquant la totalité des données. Dans ce contexte, on se pose la question de trouver **la meilleure droite**, qui sera définie comme celle qui aura le meilleur **pouvoir de généralisation.**

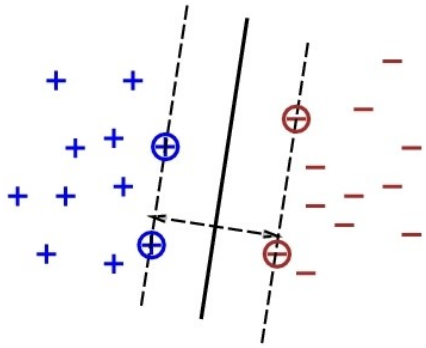
Le principe des **vastes marges** permet d'ajouter des contraintes sur la définition de la droite de telle façon que la solution du problème **devient unique**. Intuitivement, cela consiste à dire que la meilleure droite parmi toutes celles qui séparent les deux classes est celle qui se trouve le plus loin possible de tous les exemples d'apprentissage (ou encore "au milieu").

Définition : Vaste marge



Différentes solutions linéaires

Dans les méthodes à vaste marge, on recherche parmi tous les hyperplans discriminants celui qui **maximise la marge**, c'est-à-dire celui qui se trouve à égale distance des deux classes à séparer. Cette marge se calcule par rapport au points les plus proches de l'hyper plan, ils sont appelés points **supports**.



SVM Linéaire et marge

L'hyperplan recherché est alors défini par une fonction $f(x) = 0$ telle que $f(x) = a^{\top} x + b$ et telle que les points support $x_s upp$ répondent à la contrainte $f(x_s upp) = y_s upp$ (donc $+1$ ou -1 en fonction de leur classe). Ainsi, on demande à ce que tous les points d'apprentissage se trouve à une distance minimum de 1 de la frontière. Plus formellement, on pose le système à résoudre (problème quadratique donc la solution est unique si elle existe) :

$$\begin{cases} \min\limits_{a} & \frac{1}{2}\|a\|^2 \\ tq & (a^\top x_i + b)y_i \geq 1 \quad i \in [1..n] \end{cases}$$

Dans le cas dit **non séparable**, on introduit une variable de relâchement des contraintes qui permet de tolérer les écarts. En pratique, cela revient à admettre que certains points se trouvent du mauvais côté de leur marge, mais au prix d'une constante C qui pénalise la minimisation (cette constante devient un paramètre de la méthode). Le système quadratique à résoudre devient alors :

$$\begin{cases} \min\limits_{a} & \frac{1}{2}\|a\|^2 + C\sum_{i=1}^{n}\xi_i & \\ tq & (a^\top x_i + b)y_i \geq 1 - \xi_i & i \in [1..n] \\ et & \xi_i > 0 & i \in [1..n] \end{cases}$$

C. Les réseaux de neurones

Historique et généralités

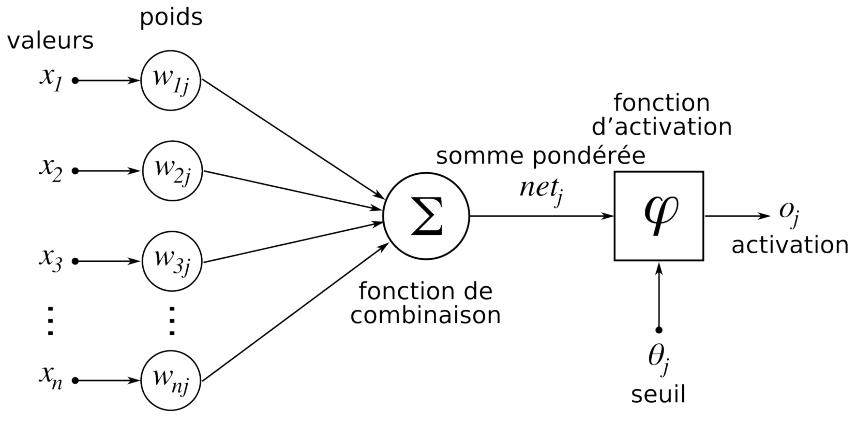
Le **neurone formel** a été définit par Mc Culloch et Pitts en 1943. Le premier réseau de neurone date de 1958, c'est le **perceptron** de Rosenblatt.

Les réseaux de neurones ont connus leur essor dans les années 80 et sont largement utilisés en milieu industriel depuis les années 90.

1. Le réseau de neurones

Le neurone formel

Il s'agit d'un neurone binaire, c'est-à-dire dont la sortie vaut 0 ou 1. Pour calculer cette sortie, le neurone effectue une somme pondérée de ses entrées (qui, en tant que sorties d'autres neurones formels, valent aussi 0 ou 1) puis applique une fonction d'activation à seuil : si la somme pondérée dépasse une certaine valeur, la sortie du neurone est 1, sinon elle vaut 0 (cf les sections suivantes).

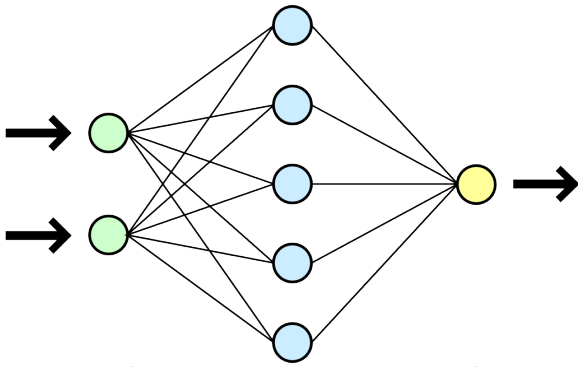


Neurone

Structure

Un réseau de neurones est **un ensemble de neurones formels connectés entre eux** (par des synapses). Chaque variable en entrée correspond à un neurone : c'est

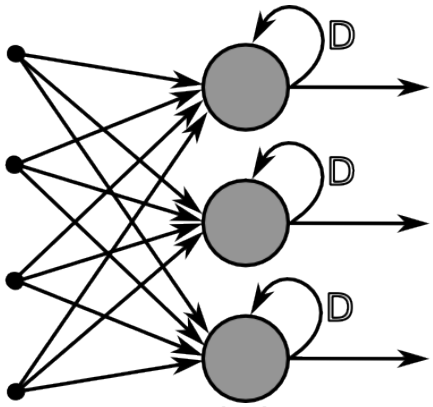
la **couche d'entrée** ainsi que chaque variable à expliquer : c'est la **couche de sortie**. Entre ces deux couches, il peut éventuellement y avoir d'autres couches dites **cachées**.



Réseau de neurone simplifié

Un réseau de neurone particulier est caractérisé par :

- la définition des neurones formels le constituant (type de fonction de combinaison ou d'activation)
- sa structure (couches cachées ou non), nombre de connexions
- son mode d'apprentissage (utilise plusieurs fois les données ou non par exemple)
- son mode : supervisé (modèle prédictif) ou non supervisé (modèle descriptif)



Réseau de neurone à rétro-propagation

Dimensionnement

Choisir le dimensionnement d'un réseau de neurones peut s'avérer complexe.

Quelques règles empiriques :

- Réseau à rétro-propagation : au moins 5 à 10 individus pour ajuster chaque poids
- En général : une couche cachée pour un réseau RBF, deux maximum pour un perceptron multicouches

D. Le perceptron multicouches

MLP : Multi Layer Perceptron

Le perceptron multicouches (*Multi Layer Perceptron*) généralise le perceptron afin

d'apprendre des modèles plus complexes, non linéaires.

Définition : Structure du MLP

Un MLP est constitué :

- d'une couche d'entrée (un neurone par variable plus un pour le biais)
- une ou plusieurs couches cachées (avec un nombre arbitraire de neurones)
- une couche de sortie (un neurone pour la regréssion ou la discrimination, un neurone par classe dans le multiclasse)
- il est acyclique
- il est complètement connecté

Les couches cachées

L'activation de chaque neurone se fait selon le calcul suivant : $a_j = \sum_{i=1}^{p} w_{ji}^{(l)} x_i$ avec l désignant la couche et p désignant le nombre de neurones de la couche précédente. $x_0 = 1$ pour le biais.
La fonction d'activation de chaque neurone est non linéaire : $z_j = f(a_j)$.

La couche de sortie

L'activation d'un neurone en sortie se fait selon le calcul suivant : $a_k = \sum_{j=0}^{h} w_{kj}^{(l)} z_j$ avec k désignant le nombre de neurones de la couche de sortie et h désignant le nombre de neurones de la couche précédente. $x_0 = 1$ pour le biais.
La fonction d'activation de chaque neurone est en général différente de celle des couches cachées : $y_k = g(a_k)$.

Remarque : Fonction analytique à optimiser

Exemple dans le cas d'un MLP à une seule couche cachée :

$$y_k = g\left(\sum_{j=0}^{h} w_{kj}^{2} f\left(\sum_{i=0}^{p} w_{ji}^{1} x_i\right)\right)$$

Apprentissage

L'apprentissage d'un MLP passe par une technique d'optimisation de minimisation de l'erreur quadratique, telle que la descente de gradient, les gradients conjugués...
Toutefois, la technique la plus utilisée n'est pas une méthode d'apprentissage mais la rétro-propagation du gradient.

Méthode : Rétro-propagation

Principe : calculer l'erreur en sortie pour corriger le poids de la couche précédente, puis propager jusqu'à la couche d'entrée. La correction utlise la dérivée de l'erreur par rapport à chacun des poids de la couche précédente.

Initialiser les poids du réseau

Initialiser $\Delta w_{kj} \leftarrow 0, \Delta w_{ji} \leftarrow 0$

Répéter jusqu'à terminaison

Pour chaque exemple d'apprentissage faire

1. *Appliquer le réseau et calculer les sorties*
2. *Calculer et cumuler les deltas*
   - *Pour chaque unité de sortie* k
     Calculer δ_k; $\Delta w_{kj} \leftarrow \Delta w_{kj} - \eta \delta_k z_j$
   - *Pour chaque unité cachée* j
     Calculer δ_j; $\Delta w_{ji} \leftarrow \Delta w_{ji} - \eta \delta_j x_i$

Ajuster les poids

- $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$
- $w_{kj} \leftarrow w_{kj} + \Delta w_{kj}$

Algorithme de rétro-propagation du gradient

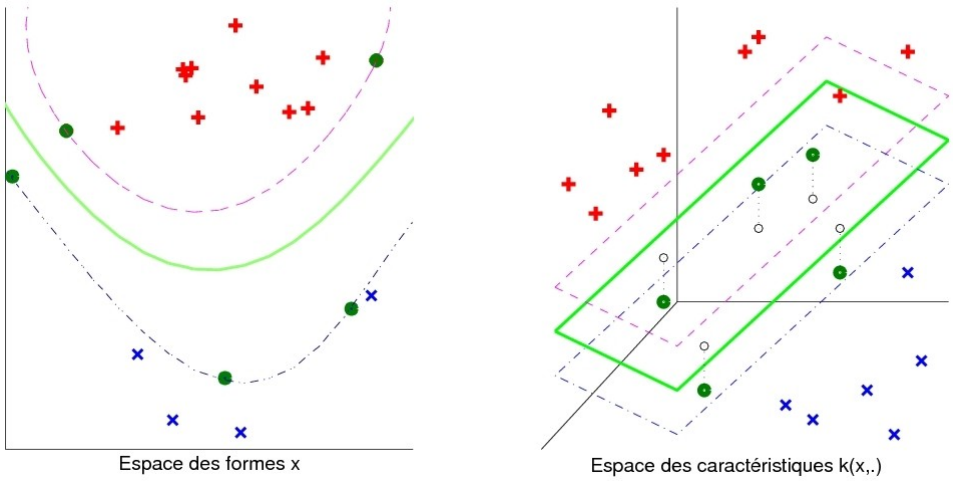
E. Les noyaux

Les noyaux d'un point de vue fonctionnel...

La page disponible en téléchargement ci-dessous donne les aspects fonctionnels des noyaux.

D'un point de vue plus intuitif, un noyau est une fonction qui prends deux instances en entrée et renvoie un scalaire qui représente une notion de distance entre les deux instances. Cette distance est le produit sclaire des deux instances dans un espace de représentation qui dépend du type de noyau choisit.

On peut également voir le noyau comme un moyen implicite de projeter les poins de l'espace de départ vers un espace de représentation dans lequel les classes sont linéairement séparables.



Espace des formes, espace d'apprentissage

F. Les Séparateurs à Vaste Marge

Quand le problème devient non linéaire.

Les noyaux permettent aux méthodes linéaires de proposer des solutions non linéaires. Dans le cas des SVM, l'introduction des noyaux conduit au problème d'optimisation quadratique suivant :

$$\begin{cases} \min\limits_{a} & \frac{1}{2}\|f\|^2 + C\sum\limits_{i=1}^{n}\xi_i & \\ tq & (f(x_i)+b)y_i \geq 1-\xi_i & i \in [1..n] \\ et & \xi_i > 0 & i \in [1..n] \end{cases}$$

où f est une combinaison linéaire de noyaux $f(.) = \sum_i \alpha_i k(x_i, .)$ dans l'espace de Hilbert à noyaux reproduisants construit à partir de la fonction noyau choisie.

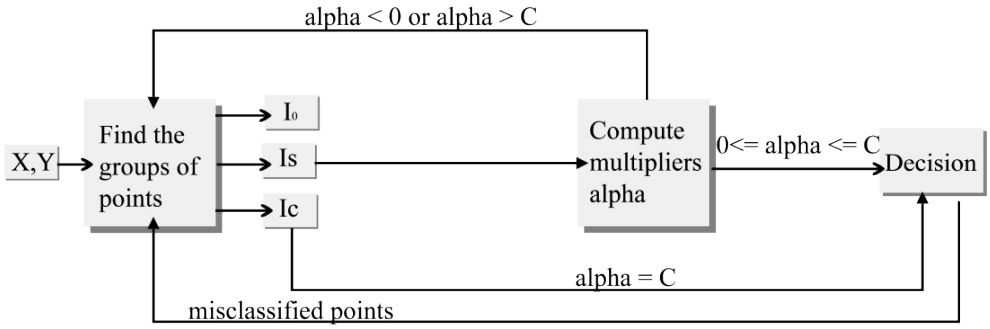
L'algorithme de résolution de ce problème passe classiquement par la résolution du problème dual (obtenu par le Lagrangien), dans lequel on ne cherche plus une fonction mais un vecteur de multiplicateur de Lagrange de taille n.

$$\begin{cases} \max\limits_{\alpha} & -\frac{1}{2}\alpha^\top G\alpha + 1^\top\alpha & \\ tq & \alpha^\top y = 0 & \\ et & 0 \leq \alpha_i \leq C & i \in [1..n] \end{cases}$$

Les conditions d'optimalité de ce système, appelée conditions KKT (Kuhn Karuch et Tucker) sont les suivantes :

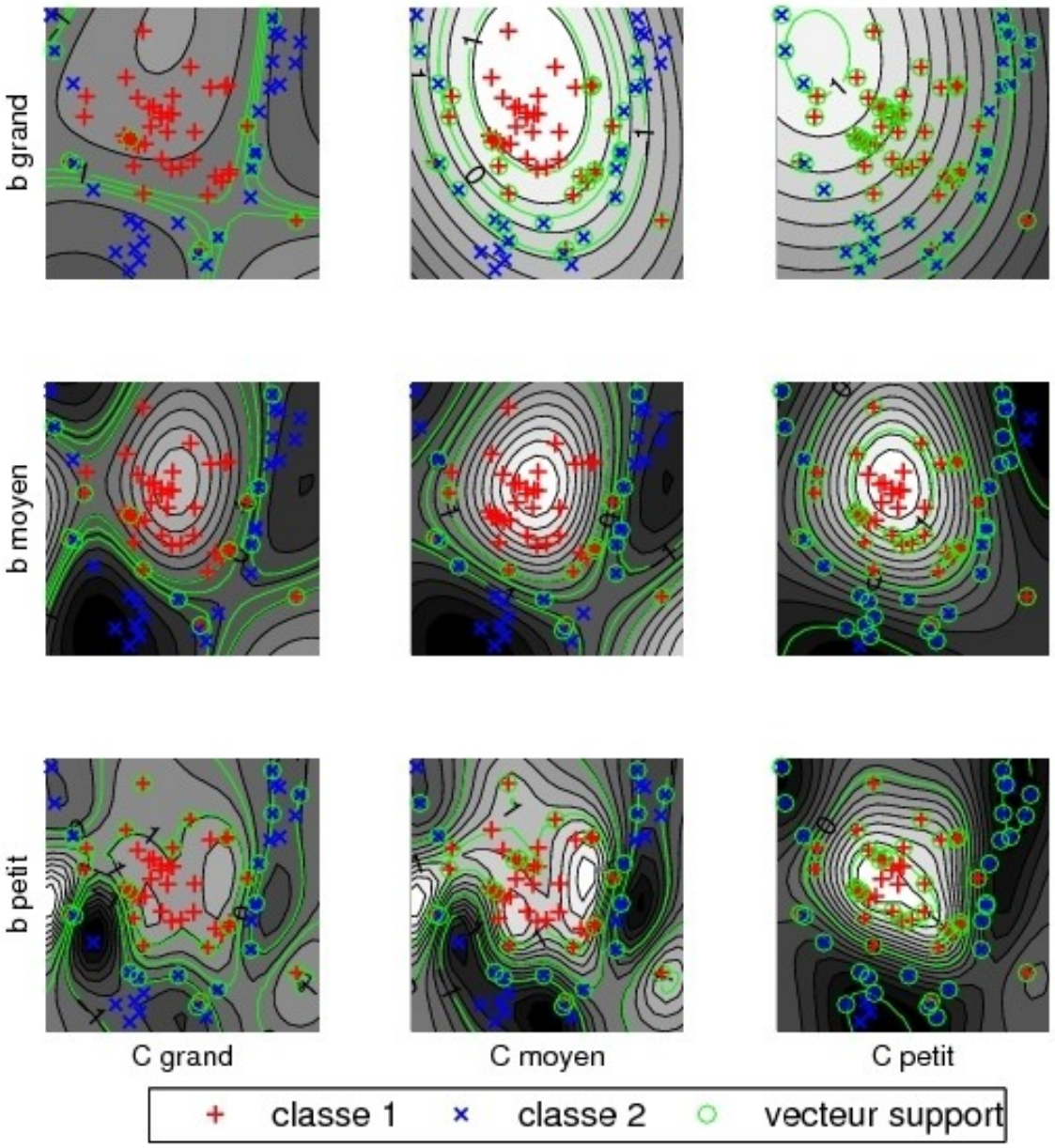
$$\begin{cases} \alpha_i = 0 & \rightarrow & (f(x_i)+b)y_i > 1 & (I_0) \\ 0 < \alpha_i < C & \rightarrow & (f(x_i)+b)y_i = 1 & (I_w) \\ \alpha_i = C & \rightarrow & (f(x_i)+b)y_i < 1 & (I_c) \end{cases}$$

La résolution de ce problème ce fait en général par une méthode de décomposition appelée SMO. Les méthode d'*active set* sont aussi particulièrement adaptées. Il faut remarquer que selon les conditions d'optimalité, seuls les points se trouvant sur la marge ou mal classés ont un multiplicateur de Lagrange associé non nul. Parmi les points actifs dans la solution, seuls ceux se trouvant exactement sur la marge nécéssitent le calcul de leur α. Ainsi, l'algorithme peut se résumer à la recherche des trois groupes de points (I_0, I_w et I_c) et on ne résoud le problème quadratique pour pour le groupe I_w (appelé aussi *working set*)



Algorithme simple de résolution de SVM

La figure ci-après illustre les effets des paramètres sur la forme de la solution d'un SVM. On fait varier ici la largeur de bande (paramètre du noyau gaussien) et la constante de pénalisation des erreurs C.



Exemples SVM en fonction des paramètres