Méthodes avancées en apprentissage supervisé et non-supervisé

Julien Ah-Pine (julien.ah-pine@univ-lyon2.fr)

Université Lyon 2

M2 SISE 2016/2017

Rappel du Sommaire

En quoi consiste l'apprentissage automatique ?

- De manière générale, un programme informatique tente de résoudre un problème pour lequel nous avons la solution. Par exemple : calculer la moyenne générale des étudiants, classer les étudiants selon leur moyenne...
- Pour certains problèmes, nous ne connaissons pas de solution exacte et donc nous ne pouvons pas écrire de programme informatique. Par exemple : reconnaître automatiquement des chiffres écrits à la main à partir d'une image scannée, déterminer automatiquement une typologie des clients d'une banque, jouer automatiquement aux échecs contre un humain ou un autre programme...
- En revanche, pour ces problèmes il est facile d'avoir une base de données regroupant de nombreuses instances du problème considéré.
- L'apprentissage automatique consiste alors à programmer des algorithmes permettant d'apprendre automatiquement de données et d'expériences passées, un algorithme cherchant à résoudre au mieux un problème considéré.

Un domaine pluri-disciplinaire

- L'apprentissage automatique (AA) ("**Machine Learning**") est à la croisée de plusieurs disciplines :
  - Les **statistiques** : pour l'inférence de modèles à partir de données.
  - Les **probabilités** : pour modéliser l'aspect aléatoire inhérent aux données et au problème d'apprentissage.
  - L'**intelligence artificielle** : pour étudier les tâches simples de reconnaissance de formes que font les humains (comme la reconnaissance de chiffres par exemple), et parce qu'elle fonde une branche de l'AA dite symbolique qui repose sur la logique et la représentation des connaissances.
  - L'**optimisation** : pour optimiser un critère de performance afin, soit d'estimer des paramètres d'un modèle, soit de déterminer la meilleure décision à prendre étant donné une instance d'un problème.
  - L'**informatique** : puisqu'il s'agit de programmer des algorithmes et qu'en AA ceux-ci peuvent être de grande complexité et gourmands en termes de ressources de calcul et de mémoire.

AA et matières connexes

- Quelques références et domaines d'application faisant intervenir l'AA :
  - Les **statistiques** ("Statistical Machine Learning") : modèles d'AA traités sous l'angle des statistiques [Hastie et al., 2011, Dreyfus, 2008].
  - L'**intelligence artificielle** ("Artifical Intelligence") : modèles d'AA mettant l'accent sur le raisonnement, l'inférence et la représentation des connaissances [Cornuéjols and Miclet, 2003, Mitchell, 1997, Alpaydin, 2010].
  - La **fouille de données** ("Data Mining") : lorsque les objets étudiés sont stockées dans des bases de données volumineuses [Han and Kamber, 2006].
  - La **reconnaissance de formes** ("Pattern Recognition") : lorsque les objets concernés sont de type "signal" comme les images, les vidéos ou le son [Bishop, 2006a].
  - Le **traitement automatique du langage - TAL** ("Natural Langage Processing" - NLP) : lorsque les problèmes concernent l'analyse linguistique de textes [Manning and Schütze, 1999, Clark et al., 2010].

AA et matières connexes (suite)

- Plus récemment :
  - La **science des données** ("Data science") : approche(s) pluri-disciplinaire pour l'extraction de connaissances à partir de données hétérogènes [Cleveland, 2001, Abiteboul et al., 2014].
  - Les **données massives** ("Big data") : mettant l'accent sur les problématiques "4V" (volume, variété, vélocité, véracité) et des éléments de solutions issus du stockage/calcul distribué [Leskovec et al., 2014].

  - Pour plus de ressources, consultez le site `http://www.kdnuggets.com`.

Plusieurs types de problèmes en AA

- Apprentissage automatique :
  - **Supervisé** : on dispose d'un ensemble d'objets et pour chaque objet une valeur cible associée ; il faut apprendre un modèle capable de prédire la bonne valeur cible d'un objet nouveau.
  - **Non-supervisé** : on dispose d'un ensemble d'objets sans aucune valeur cible associée ; il faut apprendre un modèle capable d'extraire les régularités présentes au sein des objets pour mieux visualiser ou appréhender la structure de l'ensemble des données.
  - **Par renforcement** : on dispose d'un ensemble de séquences de décisions (politiques ou stratégiques) dans un environnement dynamique, et pour chaque action de chaque séquence une valeur de récompense (la valeur de récompense de la séquence est alors la somme des valeurs des récompenses des actions qu'elle met en oeuvre) ; il faut apprendre un modèle capable de prédire la meilleure décision à prendre étant donné un état de l'environnement.

Plusieurs types de problèmes (suite)

- Apprentissage automatique (suite) :
  - **Semi-supervisé** : on dispose d'un petit ensemble d'objets avec pour chacun une valeur cible associée et d'un plus grand ensemble d'objets sans valeur cible ; il faut tirer profit à la fois des données avec et sans valeurs cibles pour résoudre des tâches d'apprentissage supervisé ou non-supervisé.
  - **Actif** : on dispose d'un petit ensemble d'objets avec pour chacun une valeur cible associée ; il faut intéragir avec l'utilisateur et lui demander de donner la valeur cible d'un nouvel objet afin de mieux apprendre le modèle de prédiction.
- Dans le cadre de ce cours, nous étudierons des problèmes **supervisés** dans un premier temps puis des problèmes **non-superisés**.

Rappel du Sommaire

Rappel du Sommaire

Apprentissage supervisé symbolique et numérique

- Deux familles en apprentissage supervisé [Cornuéjols and Miclet, 2003] :
  - Apprentissage supervisé **symbolique** : méthodes inspirées de l'intelligence artificielle et dont les fondements reposent beaucoup sur des modèles de logique, une représentation binaire des données (vrai/faux), et sur les méthodes de représentation des connaissances.
  - Apprentissage supervisé **numérique** : méthodes inspirées de la statistique, les données sont en général des vecteurs de réels, et les méthodes font intervenir des outils provenant des probabilités, de l'algèbre linéaire et de l'optimisation.
- Dans le cadre de ce cours, nous étudierons principalement les problèmes d'**apprentissage supervisé numérique** : le cours nécessite donc des prérequis de base dans les domaines sus-mentionnés.

Apprentissage supervisé numérique

- Il existe deux types de sous-problèmes en apprentissage supervisé numérique :
  - **Régression** ("Regression") : lorsque la valeur cible à prédire est continue.
  - **Classement**, classification ou catégorisation ("Classification") : lorsque la valeur cible à prédire est discrète.
- Par ailleurs nous supposerons également que les objets étudiés qui peuvent être complexe à l'origine (comme des données mutimédia) sont représentés dans un format numérique structuré. En d'autres termes :
  - On représente un objet X_i par un vecteur noté $\mathbf{x}_i$ défini dans un espace de description composé de plusieurs variables.
  - A chaque $\mathbf{x}_i$ on lui associe une valeur cible notée y_i.

Quelques exemples d'application

- Exemples de problèmes de régression :
  - Prédiction du montant des ventes d'une entreprise compte tenu du contexte économique.
  - Prédiction du prix de vente d'une maison en fonction de plusieurs critères.
  - Prédiction de la consommation électrique dans une ville étant donné des conditions météorologiques...
- Exemples de problèmes de catégorisation :
  - Prédiction de l'état sain/malade d'un patient par rapport à une maladie et compte tenu de différents facteurs.
  - Prédiction de l'accord ou du refus d'un crédit à un client d'une banque en fonction de ses caractéristiques.
  - Prédiction du chiffre correct à partir d'une image scannée d'un chiffre écrit à la main...

Notations

- Comme données à notre disposition nous supposerons que nous avons une table $\mathbf{X}$ avec n lignes et p colonnes et un vecteur colonne (variable cible) $\mathbf{y}$ de n éléments.

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{1p} \\ \vdots & \dots & \dots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \quad \text{et} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

- La ligne i de $\mathbf{X}$ est associée à l'objet X_i et l'ensemble des objets $\{X_1, \dots, X_n\}$ sera noté $\mathbb{O}$.
- La colonne j de $\mathbf{X}$ est associée à la variable ou attribut X^j et l'ensemble des variables $\{X^1, \dots, X^p\}$ sera noté $\mathbb{A}$.
- x_{ij} terme général de $\mathbf{X}$ est la valeur de la variable X^j pour l'objet X_i.
- A chaque objet X_i est associé une valeur y_i de la variable $Y \in \mathbb{Y}$ où $\mathbb{Y}$ est l'ensemble des valeurs que peut prendre Y.

Notations (suite)

- Chaque objet X_i est associé à un vecteur numérique $\mathbf{x}_i$ appartenant à un espace de description $\mathbb{X}$.
- Sauf mention contraire, on supposera que $\mathbb{X}$ est un espace vectoriel engendré par les variables $\{X^1, \dots, X^p\}$.
- Ainsi on notera par $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$ le vecteur colonne de taille $(p \times 1)$ des valeurs observées représentant X_i dans $\mathbb{X}$.
- On notera par $\mathbf{x}^j = (x_{1j}, \dots, x_{nj})$ le vecteur colonne de taille $(n \times 1)$ des valeurs observées sur $\mathbb{O}$ pour la variable X^j.
- $\mathbf{y} = (y_1, \dots, y_n)$ est le vecteur colonne de taille $(n \times 1)$ des valeurs observées sur $\mathbb{O}$ pour la variable cible Y.
- L'ensemble des couples observés $\mathbb{E} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_n, y_n)\}$ est appelé **ensemble d'entraînement ou d'apprentissage** (ou ensemble des données annotées ou étiquetées).
- Nous dénoterons par X un objet quelconque, $\mathbf{x}$ son vecteur représentant dans $\mathbb{X}$ et y la valeur cible associée à $\mathbf{x}$.

Formalisation du problème

- Etant donné un ensemble d'entraînement $\mathbb{E}$, on cherche à déterminer $f : \mathbb{X} \to \mathbb{Y}$ une fonction **modélisant** la relation entre les X décrits dans l'espace de représentation $\mathbb{X}$ et la variable cible Y :

$$f(X) = Y$$

- En revanche, ne connaissant pas la vraie nature de la relation entre X et Y et les données observées en $\{X^1, \dots, X^p\}$ étant soit bruitées, soit incomplètes ; il n'est pas raisonnable de supposer une relation déterministe. Aussi, il est davantage raisonnable de poser le problème en les termes suivants :

$$f(X) = Y + \epsilon$$

où ϵ est l'erreur ou le résidu.
- Autrement dit, il s'agit d'**approximer** f en commettant le **moins d'erreurs possibles** sur $\mathbb{E}$ tout en faisant de **bonnes prédictions** pour des valeurs de $\mathbb{X}$ non encore observées.

Schéma général

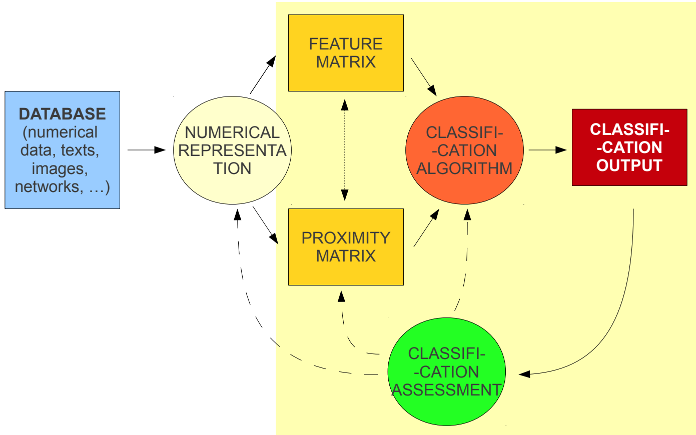


Schéma général (suite)

- Comme précisé précédemment, nous ne traitons pas dans ce cours du procédé permettant de représenter numériquement les données complexes telles que les images, vidéos, textes...
- Partant d'objets complexes comme une image par exemple, il s'agit d'extraire des variables de l'ensemble des objets permettant de représenter ceux-ci au travers d'un vecteur de nombres. On parle d'extraction d'attributs ("features extraction").
- Ces procédés sont les champs d'expertises d'autres domaines que sont l'analyse d'images et le traitement automatique du langage naturel...
- Néanmoins, des outils sont disponibles et peuvent être utilisés même par des non experts.
- Notre point de départ sera nécessairement soit une matrice de données de type table comme présenté précédemment soit une matrice carrée de dissimilarités ou de similarités entre objets (que nous étudierons ultérieurement comme pour le cas des svm).

Schéma général (suite)

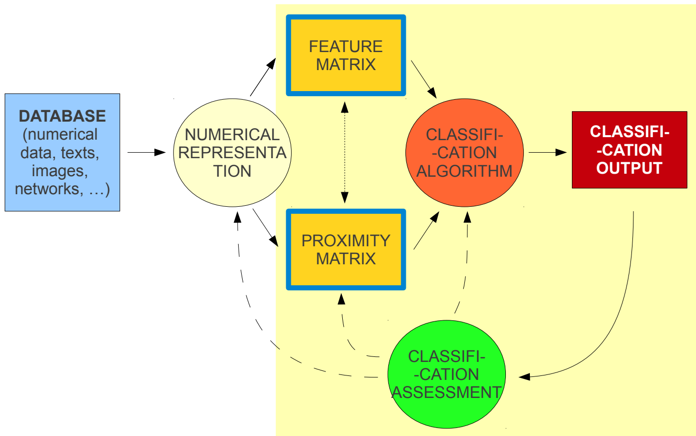
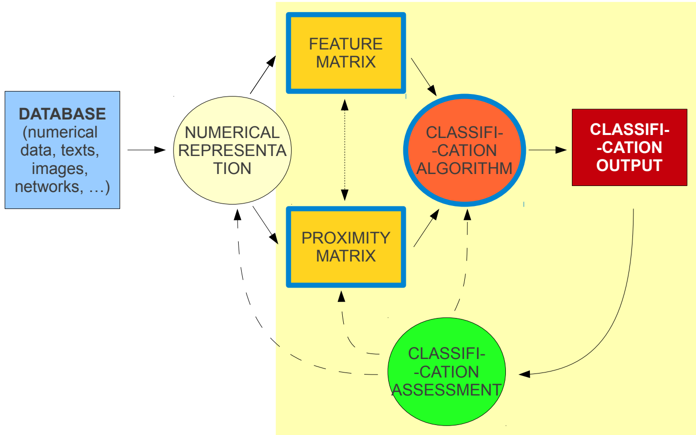


Schéma général (suite)

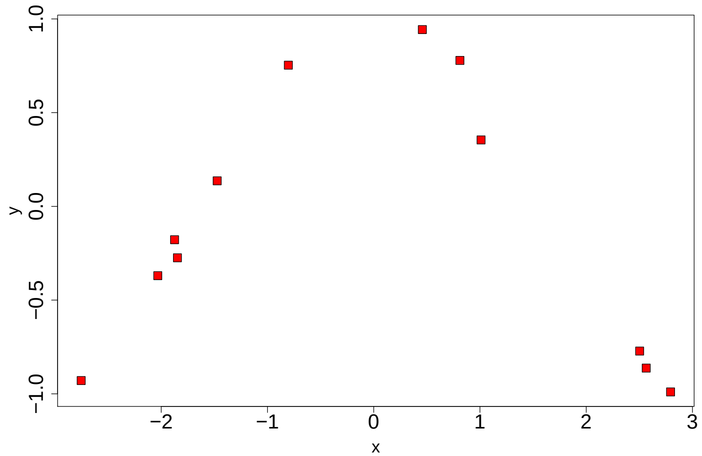
- Dans ce qui suit nous présentons des exemples simples de régression et de catégorisation qui sont à vocation pédagogique.
- Nous présentons également quelques méthodes relativement simples qui nous permettront de mettre en lumière certains concepts et sous-problèmes traités en apprentissage supervisé.

Schéma général (suite)



Exemple de problème de régression

- L'objectif est de déterminer une fonction f qui étant donné un nouveau $\mathbf{x} \in \mathbb{R}$ prédise correctement $y \in \mathbb{R}$



Régression linéaire simple

- Nous observons 12 couples de données avec en abscisse la variable X et en ordonnées la variable cible Y dont les éléments sont des réels.
- L'objectif est d'estimer une fonction $Y = f(X) + \epsilon$ qui représente la relation entre Y et X afin de prédire la valeur $\hat{y} = \hat{f}(\mathbf{x})$ pour une valeur de $\mathbf{x}$ quelconque.
- Pour un problème de régression on parlera également de **prédicteur** pour la fonction $\hat{f}$.
- En statistique une méthode très classique est donnée par les **Moindres Carrés Ordinaires (MCO)** que l'on notera par $scr(f)$ (somme des carrés des résidus ou "Residual Sum of Squares") :

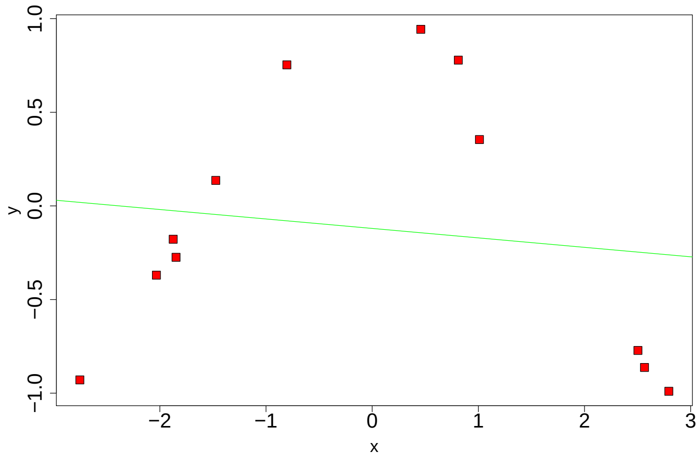
$$\begin{aligned} scr(f) &= \sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 \\ &= \sum_{i=1}^{n}(\epsilon_i)^2 \end{aligned}$$

Régression linéaire simple (suite)

- La régression linéaire simple consiste à prendre pour hypothèse que la relation f est un polynôme de degré 1 de X : $f(X) = a + bX$
- Ce qui nous donne :
$$scr(f) = scr(a, b) = \sum_{i=1}^{n}(y_i - (a + bx_i))^2$$
- $\mathbb{P} = \{a, b\}$ est l'ensemble des paramètres du modèle et on cherche les estimations $\hat{a}$ et $\hat{b}$ qui minimisent scr.
- Il faut déterminer les points critiques (ou stationnaires), solutions des équations normales (dérivées premières nulles). On obtient une solution analytique :
$$\hat{a} = \overline{\mathbf{y}} - \hat{b}\overline{\mathbf{x}} \text{ et } \hat{b} = \frac{\sum_{i=1}^{n}(x_i - \overline{\mathbf{x}})(y_i - \overline{\mathbf{y}})}{\sum_{i=1}^{n}(x_i - \overline{\mathbf{x}})^2}$$
où $\overline{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n} y_i$ est la moyenne empirique de Y.
- Le modèle de prédiction est alors donné par :
$$\hat{f}(\mathbf{x}) = \hat{a} + \hat{b}\mathbf{x}$$

Régression linéaire simple (suite)

- Régression linéaire simple



Régression linéaire multiple (polynôme de degré > 1)

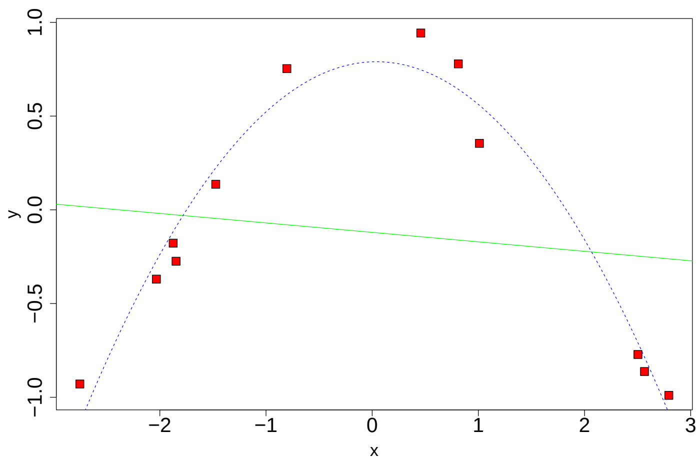
- La régression linéaire simple fait l'hypothèse que la fonction f est un polynôme de degré 1 et clairement ceci n'est pas une hypothèse raisonnable pour l'exemple traité.
- Autre type d'hypothèse : f est un polynôme de degré 2 de X : $f(X) = a + bX + cX^2$
- Dans ce cas $\mathbb{P} = \{a, b, c\}$ et on cherche à minimiser :

$$scr(f) = scr(a, b, c) = \sum_{i=1}^{n} (y_i - (a + bx_i + cx_i^2))^2$$

- Remarque : on parle de modèle linéaire car f est une fonction linéaire des paramètres $\mathbb{P}$! Les variables peuvent être tout type de fonction des variables initiales.

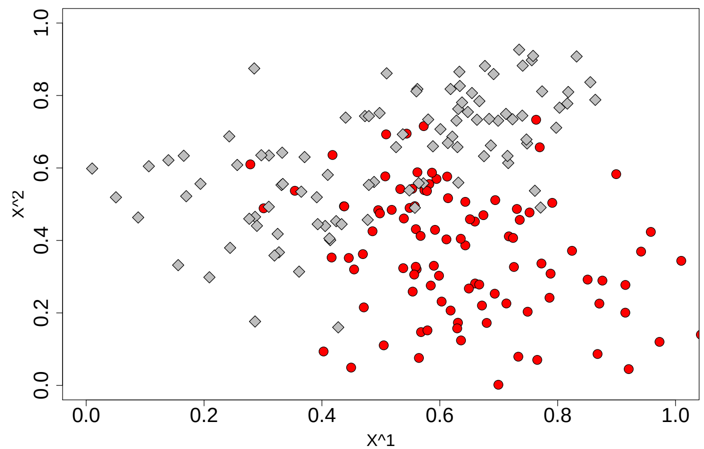
Régression linéaire multiple

- Régression linéaire multiple utilisant des fonctions de base polynômiales (jusqu'au degré 2).



Exemple de problème de catégorisation

- L'objectif est de déterminer une fonction $\hat{f}$ qui étant donné un nouveau $\mathbf{x} \in \mathbb{R}^2$ prédit correctement sa classe $y \in \{C_1, C_2\}$



Régression linéaire multiple avec variables artificielles

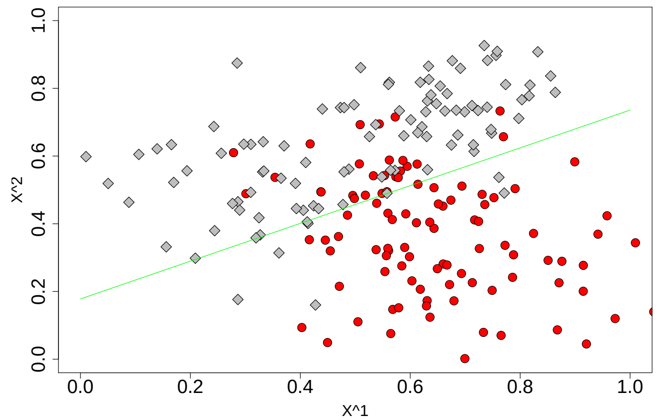
- On attribue des valeurs numériques à chacune des deux classes comme par exemple $C_1 \leftrightarrow 1$ et $C_2 \leftrightarrow -1$.
- On remplace Y variable discrète par Z une variable numérique remplie de -1 et 1.
- On traite le problème comme une régression linéaire multiple : $Z = g(X)$.
- Pour un nouveau $\mathbf{x}$ on applique la règle de décision suivante :

$$\hat{f}(\mathbf{x}) = \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) \geq 0 \\ C_2 & \text{si } \hat{g}(\mathbf{x}) < 0 \end{cases}$$

- La ligne de niveau $\{\mathbf{x} \in \mathbb{R}^2 : \hat{g}(\mathbf{x}) = 0\}$ est la **frontière de décision**.
- Pour un problème de catégorisation on parlera également de **classifieur** pour la fonction $\hat{f}$.

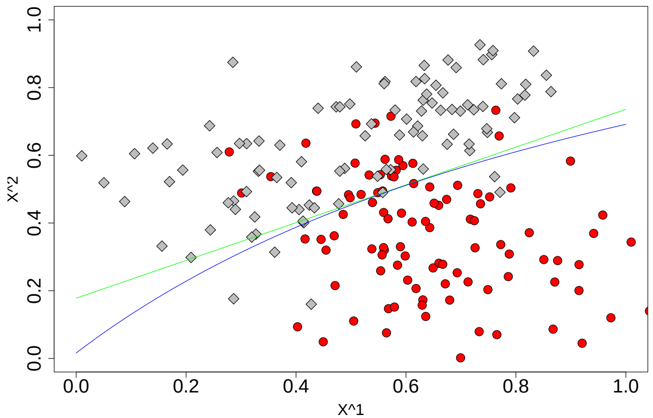
Régression linéaire multiple (suite)

- Hypothèse : $Z = g(X) = a + bX^1 + cX^2$ (polynôme de degré 1 des $\{X^j\}_{j=1}^2$)
- En vert on a tracé la frontière de décision.



Régression linéaire multiple (suite)

- Hypothèse : $Z = g(X) = a + bX^1 + cX^2 + dX^1X^2$ (polynôme de degré 2 des $\{X^j\}_{j=1}^2$).
- En bleu on a tracé la frontière de décision.



Méthode des k plus proches voisins (k-ppv)

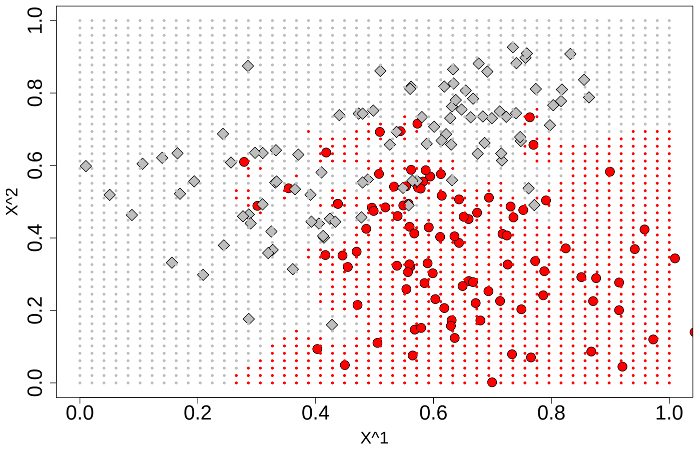
- Nous avons utilisé la régression linéaire associée à des variables artificielles pour un problème de catégorisation.
- Nous voyons une autre approche simple qui est un modèle non paramétrique : les k **plus proches voisins**.
- Etant donné un nouvel objet $\mathbf{x}$, la méthode consiste à déterminer les k plus proches objets (annotés) et d'effectuer un vote à la majorité relative afin de déterminer la classe de $\mathbf{x}$.
- Formellement nous avons la fonction de prédiction suivante :

$$\hat{f}(\mathbf{x}) = \arg \max_{C_l \in Y} \frac{|\{\mathbf{x}_i \in \mathbb{V}_k(\mathbf{x}) : y_i = C_l\}|}{k}$$

où $\mathbb{V}_k(\mathbf{x})$ est l'ensemble des k plus proches $\mathbf{x}_i$ de $\mathbf{x}$ et $\frac{|\{\mathbf{x}_i \in \mathbb{V}_k(\mathbf{x}) : y_i = C_l\}|}{k}$ est la proportion d'objets appartenant à la classe C_l parmi les k plus proches voisins.

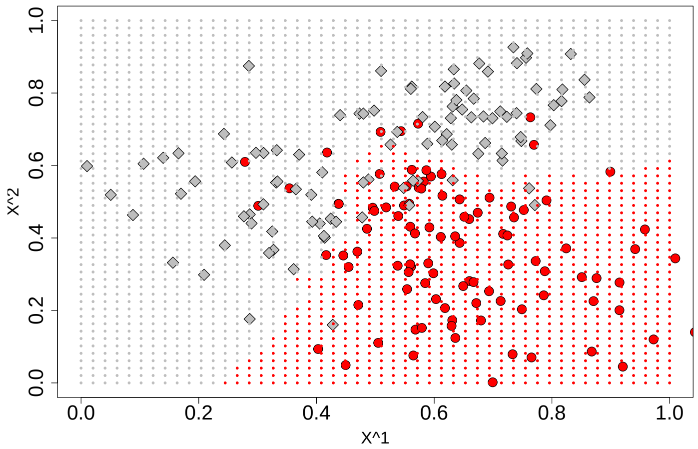
Méthode des k-ppv (suite)

- k=1



Méthode des k-ppv (suite)

- k=9



Choix de la méthode d'apprentissage

- Il existe plusieurs méthodes en apprentissage supervisé que ce soit pour la régression ou la catégorisation.
- Pour choisir une méthode, il y a deux approches complémentaires :
  - l'une relève de la **bonne compréhension des fondements des méthodes** et de ce qui permet de les distinguer afin de déterminer les modèles qui traiteraient au mieux un cas d'étude donné,
  - l'autre, résolument empirique, relève de l'**application de méthodes et critères d'évaluation et de sélection** permettant de sélectionner les algorithmes de catégorisation les plus performants étant donné un cas d'étude.

Plusieurs modèles de prédiction

- Il existe plusieurs façons d'établir par apprentissage la relation entre les $\mathbf{x}_i$ et les y_i d'entraînement afin de prédire la valeur cible pour tout $\mathbf{x} \in \mathbb{X}$. On peut distinguer :
  - Les méthodes de type **inductif** qui infèrent des données d'apprentissage une fonction de prédiction **globale** qui est estimée en optimisant un critère de performance. Les prédictions pour des nouvelles données sont déduites de la fonction de décision estimée. Il s'agit notamment des modèles paramétriques vus précédemment. Les dénominations anglo-saxonnes sont "inductive learning", "eager learning".
  - Les méthodes de type **transductif** qui ne passent pas par une phase d'inférence mais qui utilisent directement et **localement** les données d'apprentissage pour prédire la valeur cible pour les nouvelles données. Il s'agit notamment des modèles non paramétriques vus précédemment. Les dénominations anglo-saxonnes sont "transductive learning", "lazy learning" ou "instance-based learning".
- Nous verrons des méthodes de type inductif qui se distinguent les unes des autres en considérant plusieurs aspects que nous voyons ci-après.

Plusieurs types de fonction de prédiction

- Les méthodes de type inductif supposent que la fonction de prédiction f appartient à une famille de fonctions ou d'**hypothèses** $\mathbb{H}$ dont les paramètres sont dénotés par $\mathbb{P}$.
- Par exemple la régression linéaire multiple :
  $\mathbb{H} = \{f : \mathbb{X} \to \mathbb{Y} : f(X) = a_0 + \sum_{j=1}^{p} a_j X^j\}$ où
  $\mathbb{P} = \{a_0, \dots, a_p\} \in \mathbb{R}^{p+1}$.
- Le modèle linéaire dans sa forme générale peut s'écrire de la manière suivante :

$$f(X) = a_0 + a_m \sum_{m=1}^{M} g_m(X)$$

  où $g_m : \mathbb{X} \to \mathbb{R}$ sont des fonctions quelconques à valeurs dans $\mathbb{R}$ appelée **fonctions ou expansions de base** (par exemple : $f(X) = a_0 + a1X^1 + a_2X^2 + a_3X^1X^2$).
- Il existe donc plusieurs familles d'hypothèses donnant autant de résultats de prédictions différents.

Biais inductif

- Le choix d'un espace d'hypothèses $\mathbb{H}$ implique un **biais inductif** dans le processus d'apprentissage.
- En effet, il existe par exemple une infinité de façon de séparer les classes C_1 et C_2 dans l'exemple de catégorisation.
- Si on choisit la régression linéaire multiple, la forme de la fonction de prédiction est nécessairement un hyperplan.
- En d'autres termes, le biais inductif est l'ensemble des hypothèses implicites que l'on fait lorsque l'on utilise une méthode d'apprentissage supervisé pour résoudre un problème de régression ou de catégorisation.

Plusieurs types de fonction de performance

- Etant donné un espace d'hypothèses $\mathbb{H}$, pour déterminer la fonction de prédiction (une instance de $\mathbb{H}$), il faut estimer les paramètres $\mathbb{P}$ qui optimisent un critère de performance sur les données $\mathbb{E}$.
- Pour le problème de régression nous avons déjà évoqué les Moindres Carrés Ordinaires :

$$scr(f) = \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2$$

- Lorsque les données n'ont pas toutes une importance uniforme, une façon de généraliser le scr est l'utilisation de concepts issus de la **décision statistique**.

Notations

Nous nous plaçons dans un **cadre probabiliste** et dans ce cas :

- Les variables $X^1, \dots, X^p$ sont des variables aléatoires (v.a.) réelles.
- La variable Y est une v.a. prenant ses valeurs dans $\mathbb{Y}$.
- Les objets $X_1, \dots, X_n$ et X sont des vecteurs aléatoires de dimension $(p \times 1)$, chaque dimension j étant associée à la variable X^j.
- On notera par $P(X)$ la fonction de densité de probabilité (multidimensionnelle) du vecteur aléatoire X.
- On notera par $P(Y|X)$ la probabilité conditionnelle de Y sachant X.
- On notera par $\mathrm{E}_X(f(X))$ l'espérance de $f(X)$ par rapport à X.
- On notera par $\mathrm{E}_{Y|X}(f(Y)|X)$ l'espérance conditionnelle de $f(Y)$ par rapport à Y sachant X.
- $\{X_i^j\}_{i=1}^n$ et $\{Y_i\}_{i=1}^n$ sont n variables aléatoires que l'on supposera i.i.d. (indépendantes et identiquement distribuées) selon les lois mères $P(X^j)$ et $P(Y)$ respectivement.

Fonction de performance et décision statistique

- X est un vecteur aléatoire de taille $(p \times 1)$ prenant ses valeurs dans $\mathbb{X}$.
- Y est une variable aléatoire prenant ses valeurs dans $\mathbb{Y}$.
- Soient $P(X)$, $P(Y)$, les fonctions de probabilité de X et Y.
- Soit $P(X, Y)$ la fonction de probabilité jointe du couple (X, Y).
- Soit $\ell(f(X), Y)$ une **fonction de perte** ("loss") mesurant le coût de la différence entre la prédiction du modèle et l'observation.

Définition. (Espérance de la fonction de perte)

On définit l'espérance de la fonction perte associée à f de la façon suivante :

$$\mathrm{E}_{X,Y}(\ell(f(X), Y)) = \mathrm{E}_{X,Y}(\ell) = \int_{\mathbb{X}} \int_{\mathbb{Y}} \ell(f(X), Y) P(X, Y) d\mathbf{x} dy$$

Fonction de performance et décision statistique (suite)

- Etant donné $\mathbb{E}$, il s'agit de déterminer f qui minimise l'espérance empirique de la perte mesurée sur l'ensemble d'apprentissage.
- $scr(f)$ est un cas particulier puisqu'il revient à minimiser $\mathrm{E}_{X,Y}(\ell)$ en prenant :
  - une fonction de perte quadratique : $\ell_2(f(X), Y) = (Y - f(X))^2$,
  - une distribution uniforme pour le couple (X, Y),
  - une fonction de prédiction polynomiale de degré 1 des $\{X^j\}_{j=1}^p$:

$$f(X) = a_0 + \sum_{j=1}^{p} a_j X^j$$

- On peut généraliser et utiliser d'autres types de fonction de perte ou en donnant différents poids selon $P(X, Y)$.

Fonction de perte quadratique et fonction de régression

- Etudions $\mathrm{E}_{X,Y}(\ell)$ dans le cas d'une fonction de perte quadratique :

$$\min_f \mathrm{E}_{X,Y}(\ell_2) = \int_{\mathbb{X}} \int_{\mathbb{Y}} (f(\mathbf{x}) - y)^2 P(\mathbf{x}, y) d\mathbf{x} dy$$

- On montre que la solution du problème est de la forme :

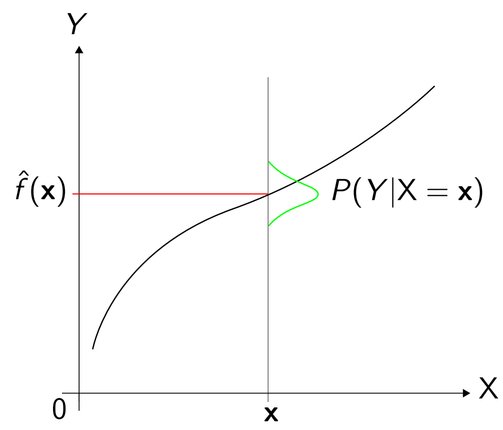
$$f^*(\mathbf{x}) = \mathrm{E}_{Y|X}(Y|X = \mathbf{x})$$

- La fonction qui minimise au point $\mathbf{x}$, $\mathrm{E}_{X,Y}(\ell_2)$, est l'espérance de Y sous la probabilité conditionnelle $P(Y|X = \mathbf{x})$. Cette solution s'appelle **fonction de régression**.
- Le cas de ℓ_2 est souvent utilisé car elle conduit à la solution simple ci-dessus mais le principe d'espérance de fonction de perte permet d'avoir plusieurs types de fonction de performance. En fait ℓ_2, est un sous-cas de la famille de fonction de perte de Minkowski :

$$\mathrm{E}_{X,Y}(\ell_r(f(X), Y)) = \int_{\mathbb{X}} \int_{\mathbb{Y}} |f(\mathbf{x}) - y|^r P(\mathbf{x}, y) d\mathbf{x} dy$$

Fonction de perte quadratique et fonction de régression (suite)

- Cas de la fonction quadratique et illustration de la fonction de régression.



Fonction de performance pour la catégorisation

- Que se passe t-il dans le cas où $\mathbb{Y}$ est discret ?
- Le principe de minimisation de l'espérance de la fonction de perte est valide mais il faut adapter la fonction de perte au cas discret.
- Un coût de la fonction de perte intervient lorsqu'on attribue à un $\mathbf{x}$ une classe qui n'est pas la bonne.
- Supposons que la classe de $\mathbf{x}$ est C_l et qu'on lui attribue par erreur la classe $C_{l'}$. Pour chaque couple $(C_l, C_{l'})$ on a le coût $\mathbf{L}(C_l, C_{l'})$ associé à une mauvaise catégorisation.
- On a la donnée d'une matrice de perte $\mathbf{L}$ de taille $(q \times q)$ (q étant le cardinal de $\mathbb{Y}$ càd le nombre de classes) dont le terme général est :

$$\mathbf{L}(C_l, C_{l'}) = \mathbf{L}_{ll'} = \text{Coût associée à une mauvaise affectation d'un objet de classe } C_l \text{ à une classe } C_{l'}$$

- $\mathbf{L}$ est d'éléments positifs ou nuls et la diagonale est remplie de 0.

Fonction de performance pour la catégorisation (suite)

- L'espérance de perte s'écrit alors :

$$\mathrm{E}_{X,Y}(\mathbf{L}) = \int_{\mathbb{X}} \sum_{C_l \in \mathbb{Y}} \mathbf{L}(C_l, f(X)) P(X, C_l) d\mathbf{x}$$

- La solution est alors :

$$\forall \mathbf{x} \in \mathbb{X} \; : f^*(\mathbf{x}) = \arg \min_{C_{l'} \in \mathbb{Y}} \sum_{C_l \in \mathbb{Y}} \mathbf{L}(C_l, C_{l'}) P(C_l | X = \mathbf{x})$$

- Comme précédemment, il existe plusieurs façons de définir la matrice de perte $\mathbf{L}$. La fonction de perte la plus simple consiste à attribuer un coût uniforme pour chaque type d'erreur.

Fonction de perte binaire et classifieur bayésien

- La fonction de perte la plus simple est associée à la matrice de perte suivante :

$$\mathbf{L}_{ll'} = \begin{cases} 1 & \text{si } l \neq l' \\ 0 & \text{si } l = l' \end{cases}$$

- Dans ce cas, nous avons :

$$\begin{aligned} \forall \mathbf{x} \in \mathbb{X} \; : f^*(\mathbf{x}) &= \arg \min_{C_{l'} \in \mathbb{Y}} \sum_{C_l \in \mathbb{Y}} \mathbf{L}_{ll'} P(C_l | X = \mathbf{x}) \\ &= \arg \min_{C_{l'} \in \mathbb{Y}} (1 - P(C_{l'} | X = \mathbf{x})) \\ &= \arg \max_{C_{l'} \in \mathbb{Y}} P(C_{l'} | X = \mathbf{x}) \end{aligned}$$

- Autrement dit, la fonction de prédiction est telle que : $f^*(\mathbf{x}) = C_l$ ssi $P(C_l | X = \mathbf{x}) = \max_{C_{l'} \in \mathbb{Y}} P(C_{l'} | X = \mathbf{x})$. Cette approche est appelée **classifieur bayésien**.

Classifieur bayésien

- Si on suppose une matrice de perte uniforme et qu'on minimise l'espérance de la perte sous $P(X, Y)$ alors on obtient le classifieur bayésien qui repose sur la probabilité conditionnelle $P(Y|X)$.
- Si on applique le théorème de Bayes on a :

$$\underbrace{P(Y|X)}_{posterior} = \frac{\overbrace{P(Y)}^{prior} \overbrace{P(X|Y)}^{likelihood}}{\underbrace{P(X)}_{evidence}}$$

- Rappelons que $X = (X^1, \ldots, X^p)$ est un vecteur aléatoire de dimension p. En utilisant successivement les probabilités conditionnelles ($P(A, B) = P(A|B)P(B)$) on a :

$$\begin{aligned} P(X|Y) &= P(X^1, \ldots, X^p | Y) \\ &= P(X^1 | Y, X^2, \ldots, X^p) P(X^2, \ldots, X^p | Y) \\ &= P(X^1 | Y, X^2, \ldots, X^p) \ldots P(X^{p-1} | Y, X^p) P(X^p | Y) \end{aligned}$$

Estimation, décision, zone de rejet

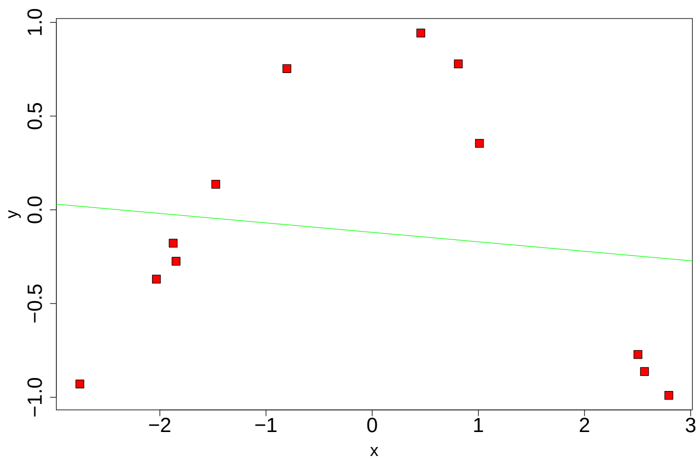
- Pour les méhodes de type inductif, il y a deux phases :
  1. une étape d'**inférence** ou d'estimation des paramètres du modèle $\mathbb{P}$,
  2. une étape de **décision** qui permet d'aboutir à la prédiction $\hat{f}(X)$.
- Il existe plusieurs façons de définir théoriquement $f(X)$ (espaces d'hypothèses $\mathbb{H}$).
- Certains modèles sont simples et conduisent à des **solutions analytiques** comme la régression linéaire multiple.
- Pour des classes d'hypothèses plus complexes on a recours à des algorithmes d'**optimisation numérique**. Il existe également plusieurs façon d'estimer les paramètres de $f(X)$ étant donné $\mathbb{E}$.
- Certains modèles permettent d'appréhender une incertitude de la prédiction donnée par $\hat{f}(X)$. Dans ce cas, on peut définir une **zone de rejet** dans la prise de décision et faire intervernir l'humain. Par exemple, dans le cas des k-ppv et d'une catégorisation binaire, si la classe majoritaire ne dépasse pas 60% on peut avoir une zone de rejet.

Généralisation, sous et sur-apprentissage

- Le choix d'une méthode revient à choisir un espace d'hypothèses, une fonction de perte et une technique d'inférence.
- Ce qu'on attend d'une bonne méthode n'est pas tant sa capacité à reproduire à l'identique le résultat des données d'entraînement mais de produire les résultats corrects sur des données de test càd non observées : c'est le principe de **généralisation**.
- Dans cette perspective il faut une bonne adéquation entre la complexité de la classe d'hypothèse choisie $\mathbb{H}$ et la véritable relation entre X et Y. Si la complexité de $\mathbb{H}$ n'est pas assez suffisante on parle de **sous-apprentissage**.
- Quand au contraire, la complexité de $\mathbb{H}$ est trop grande, il arrive que l'erreur sur $\mathbb{E}$ est proche de zéro mais l'erreur sur les données de test est grande. Dans ce cas on parle de **sur-apprentissage**.

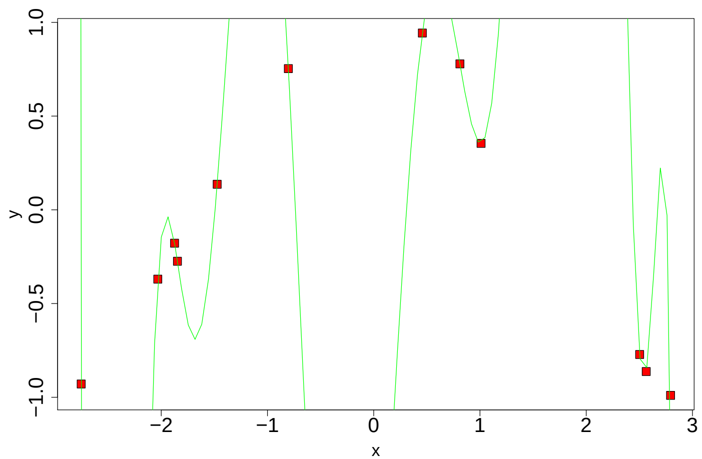
Généralisation, sous et sur-apprentissage (suite)

- Exemple de sous-apprentissage ($\mathbb{H}$=Ensemble des polynômes de degré 1 de X).



Généralisation, sous et sur-apprentissage (suite)

- Exemple de sur-apprentissage ($\mathbb{H}$=Ensemble des polynôme de degré 12 de X).



Compléxité des modèles de régression linéaire

- Dans l'exemple de régression, la complexité d'un modèle ou d'une classe d'hypothèses $\mathbb{H}$ est l'ordre du polynôme.
- Si l'ordre est trop petit, il y a sous-apprentissage et s'il est trop grand il y a sur-apprentissage.
- Pour le polynôme de degré 1 :
  - la complexité des données et celle du modèle ne coïncident pas,
  - l'erreur mesurée sur les données $\mathbb{E}$ est très grande,
  - mais la fonction de prédiction étant une droite la variance du modèle est faible (si on change $\mathbb{E}$ la "droite changera peu").
- Pour le polynôme de degré 12 :
  - la complexité des données et celle du modèle ne coïncident pas,
  - l'erreur mesurée sur les données $\mathbb{E}$ est très faible,
  - mais la fonction de prédiction est instable et donc la variance du modèle est très grande (si on change $\mathbb{E}$ la "courbe changera beaucoup").

Arbitrage biais-variance

Définition. (Décomposition Erreur quadratique - Bruit)

$$\underbrace{\mathrm{E}_{Y|X}((f(X)-Y)^2|X)}_{\mathrm{E}_{Y|X}(\ell_2|X)} = \underbrace{\mathrm{E}_{Y|X}([f(X)-\mathrm{E}_{Y|X}(Y|X)]^2|X)}_{\textbf{Erreur quadratique}} + \underbrace{\mathrm{E}_{Y|X}([\mathrm{E}_{Y|X}(Y|X)-Y]^2|X)}_{\textbf{Bruit irréductible}}$$

- Le **bruit irréductible** est intrinsèque aux données (les erreurs de mesure par exemple) et le terme associé représente l'erreur minimale que l'on peut commettre en moyenne.
- L'**erreur quadratique** est l'espérance de l'erreur entre f et la fonction de regression (que l'on a vue être la fonction optimale pour la minimisation de $\mathrm{E}_{X,Y}(\ell_2)$).

Arbitrage biais-variance (suite)

- Dans ce contexte, les méthodes d'apprentissage consistent donc à approximer $\mathrm{E}_{Y|X}(Y|X)$. Etant donné les données d'apprentissage à disposition, $\mathbb{E}$, on infère une fonction de prédiction $\hat{f}_{\mathbb{E}}(X)$.
- Si l'**on change de données d'apprentissage** on obtient une autre fonction de prédiction. Ainsi, on peut voir les données d'entraînement comme la réalisation d'un processus aléatoire et on définit ainsi : $\mathrm{E}_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X))$ qui est l'espérance de la fonction de prédiction dépendant du processus aléatoire générant les données $\mathbb{E}$.
- Etant donné un ensemble $\mathbb{E}$ et une fonction de prédiction induite $\hat{f}_{\mathbb{E}}(X)$, on peut alors décomposer l'erreur quadratique comme suit :
$$\mathrm{E}_{\mathbb{E}}\left(\left[\hat{f}_{\mathbb{E}}(X)-\mathrm{E}_{Y|X}(Y|X)\right]^2\right)$$
$$=\mathrm{E}_{\mathbb{E}}\left(\left[\hat{f}_{\mathbb{E}}(X)-\mathrm{E}_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X))\right]^2\right)+\mathrm{E}_{\mathbb{E}}\left(\left[\mathrm{E}_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X))-\mathrm{E}_{Y|X}(Y|X)\right]^2\right)$$
Puisque comme précédemment la double somme s'annule.

Arbitrage biais-variance (suite)

Définition. (Décomposition Biais - Variance)

$$\underbrace{\mathrm{E}_{\mathbb{E}}\left(\left[\hat{f}_{\mathbb{E}}(X)-\mathrm{E}_{Y|X}(Y|X)\right]^2\right)}_{\textbf{Erreur quadratique}} = \underbrace{\mathrm{E}_{\mathbb{E}}\left(\left[\hat{f}_{\mathbb{E}}(X)-\mathrm{E}_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X))\right]^2\right)}_{\textbf{Variance}} + \underbrace{\left[\mathrm{E}_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X))-\mathrm{E}_{Y|X}(Y|X)\right]^2}_{\textbf{Biais}^2}$$

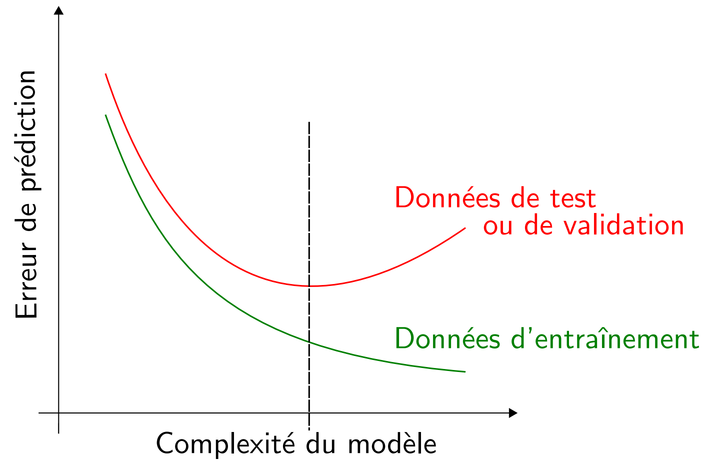
- Le **biais** indique, l'écart entre la fonction de prédiction moyenne apprise sur plusieurs jeux de données et la fonction de régression.
- La **variance** représente en moyenne, l'écart quadratique entre une fonction de prédiction apprise sur un jeux de données et la fonction de prédiction moyenne apprise sur plusieurs jeux de données.

Arbitrage biais-variance (suite)

- A erreur quadratique constante, on voit qu'il y a un **arbitrage entre biais et variance**.
- Exemple de fort biais et faible variance : régression linéaire avec polynôme de degré 1.
- Exemple de faible biais et forte variance : régression linéaire avec polynôme de degré 12.
- L'idéal est d'avoir un faible biais et une faible variance pour une meilleure généralisation mais plus facile à dire qu'à faire !
- Plus la complexité d'un modèle augmente plus le biais mesuré sur des données $\mathbb{E}$ diminue. Mais la variance augmentant également, le bon comportement du modèle estimé sur des données non observées n'est alors plus garanti.

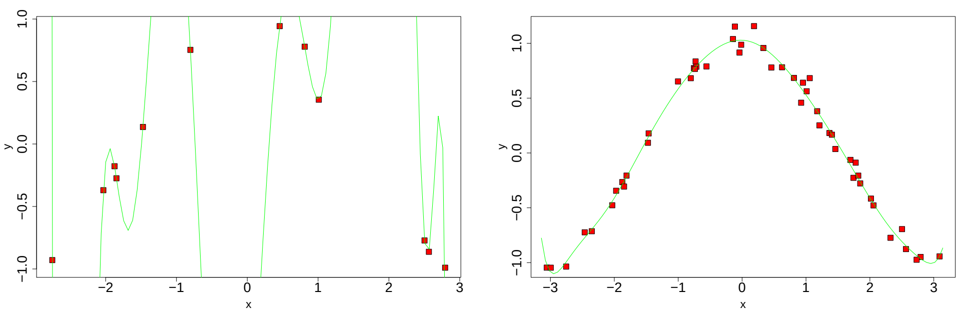
Arbitrage biais-variance (suite)

- Illustration du problème de l'arbitrage biais-variance



Impact de la taille de l'échantillon d'entraînement $\mathbb{E}$

- Même tâche que précedemment : les données sont générées par la fonction cos sur $[-\pi, \pi]$ à laquelle on ajoute un bruit $\epsilon \sim \mathcal{N}(0, 0.08)$.
- $\mathbb{H}$ =Ensemble des polynômes de X de degré 12.
- Estimation sur deux échantillons de tailles $n = 12$ et $n = 50$.



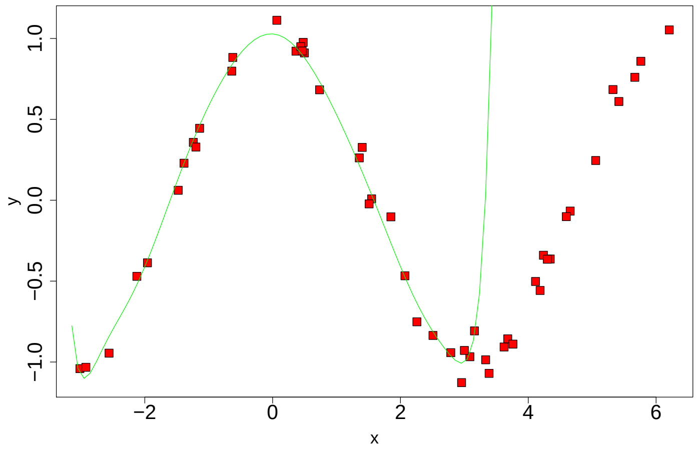
- Bien sûr plus on a de données meilleure est l'estimation !

Arbitrage triple Complexité - Données d'entraînement - Erreur en généralisation

- En apprentissage automatique à partir d'exemples d'entraînement, il y a un arbitrage entre ces trois facteurs :
  - La complexité de l'espace des hypothèses choisis $\mathbb{H}$.
  - La quantité de données d'entraînement n.
  - La qualité de généralisation de la fonction de prédiction sur des données de tests.
- Une grande complexité de $\mathbb{H}$ permet une meilleure flexibilité du modèle et implique une meilleure généralisation.
- Mais une trop grande complexité donne parfois trop de flexibilité : sur $\mathbb{E}$ l'erreur diminue mais la variance du modèle sera plus forte. Ainsi, si les données de test sortent de la région des données $\mathbb{E}$, le comportement de la fonction de prédiction risque d'être chaotique.
- Ce problème est moins fort lorsque n est grand comme on vient de le voir mais jusqu'à un certain point.

Arbitrage triple Complexité - Données d'entraînement - Erreur en généralisation (suite)

- Extrapolation du modèle appris précédemment jusqu'à 2π.



Rappel du Sommaire

Introduction

- Les méthodes de régression linéaire supposent que la fonction de régression $\mathrm{E}(Y|X)$ est une fonction linéaire des paramètres $\mathbb{P}$.
- Ce sont des méthodes développées depuis le XVIIIème siècle en statistiques et qui sont encore de nos jours très utilisées car elles sont simples et permettent une bonne interprétation de l'influence des variables explicatives sur la variable à expliquer :

$$f(X) = \sum_j a_j X^j \Rightarrow \frac{\partial f}{\partial X^j}(X) = a_j$$

- Des développements récents sont également proposés permettant d'enrichir la panoplie de ce type de méthodes. En particulier, certaines méthodes aboutissant à des frontières de décision non-linéaires sont en fait des généralisations des méthodes linéaires (au sens d'un polynôme de degré 1 des paramètres $\mathbb{P}$).
- On étudiera pour les problèmes de régression et de catégorisation, les fondements et la mise en oeuvre de méthodes de base et avancées.

Régression linéaire multiple et MCO

- Rappelons que nous souhaitons déterminer une fonction f modélisant la relation entre la variable cible Y et les variables explicatives $\{X^1, X^2, \ldots, X^p\}$ qui constituent l'espace de description des objets $\mathbb{X}$.
- Le modèle de régression linéaire est le suivant :

$$Y = f(X^1, \ldots, X^p) + \epsilon = a_0 + \sum_{j=1}^{p} a_j X^j + \epsilon$$

- On a donc $\mathbb{H} = \{f : \mathbb{R}^p \to \mathbb{R} : f(X) = a_0 + \sum_{j=1}^p a_j X^j\}$.
- Les variables explicatives peuvent être :
  - Les variables initiales.
  - Des transformations des variables initiales.
  - Des expansions de bases des variables initiales [Hastie et al., 2011].
- Le modèle reste une fonction linéaire des paramètres $\mathbb{P} = \{a_j\}_{j=0}^p$.

Régression linéaire multiple et MCO (suite)

- L'étape d'induction consiste à estimer les paramètres $\mathbb{P}$ étant données les données d'entraînement $\mathbb{E}$.
- La méthode classique est les **Moindres Carrés Ordinaires** (MCO) :

$$\begin{aligned} scr(f) &= \sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 \\ &= \sum_{i=1}^{n}(y_i - (a_0 + \sum_{j=1}^{p} a_j x_{ij}))^2 \end{aligned}$$

- Du point de vue statistique, l'utilisation de ce modèle suppose que les observations y_i sont des réalisations de v.a. Y_i i.i.d..
- Introduisons les notations suivantes :
  - $\mathbf{a}$, le vecteur colonne de taille $p+1$ contenant les paramètres.
  - $\mathbf{X}$, la matrice des données de taille $(n \times (p+1))$ à laquelle on a ajouté une 1ère colonne remplie de 1.

Régression linéaire multiple et MCO (suite)

- Nous avons :

$$\mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_p \end{pmatrix} \quad ; \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{1p} \\ \vdots & \vdots & \dots & \dots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \quad ; \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

- Notons par ailleurs $\mathbf{X}^\top$ la matrice transposée de $\mathbf{X}$.
- Nous avons alors l'écriture matricielle suivante :

$$scr(f) = (\mathbf{y} - \mathbf{Xa})^\top (\mathbf{y} - \mathbf{Xa})$$

- On cherche à déterminer les paramètres $\mathbb{P} = \{a_j\}_{j=0}^{p}$ représentés par le vecteur $\mathbf{a}$ qui minimise $scr(f)$: c'est un problème d'optimisation quadratique non contraint :

$$\hat{\mathbf{a}}_{mco} = \arg \min_{\mathbf{a} \in \mathbb{R}^{p+1}} (\mathbf{y} - \mathbf{Xa})^\top (\mathbf{y} - \mathbf{Xa})$$

- La solution s'obtient en recherchant les points $\mathbf{a}$ tel que $\nabla scr(\mathbf{a}) = \mathbf{0}$.

Rappels en calcul différentiel

- Si $f : \mathbb{R}^{p+1} \to \mathbb{R}$ est différentiable, alors la fonction ∇f défini par :

$$\nabla f(\mathbf{a}) = \begin{pmatrix} \frac{\partial f}{\partial a_0}(\mathbf{a}) \\ \frac{\partial f}{\partial a_1}(\mathbf{a}) \\ \vdots \\ \frac{\partial f}{\partial a_p}(\mathbf{a}) \end{pmatrix}$$

  est appelé **gradient** de f.
- ∇f est une fonction de $\mathbb{R}^{p+1}$ dans $\mathbb{R}^{p+1}$ et peut être vue comme un **champ de vecteurs** (fonction qui associe à tout point un vecteur).
- Quelques formules de dérivations dans le cas multivarié. La dérivée est calculée par rapport à la variable $\mathbf{x}$. $\mathbf{A}$ est une matrice de réels de taille $(m \times n)$ et $\mathbf{y}$ un vecteur de réels de taille $(m \times 1)$:
  - Si $f(\mathbf{x}) = \mathbf{y}^\top \mathbf{Ax}$ ou si $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}^\top \mathbf{y}$ alors $\nabla f(\mathbf{x}) = \mathbf{A}^\top \mathbf{y}$.
  - Si $\mathbf{A}$ est carrée et $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{Ax}$ alors $\nabla f(\mathbf{x}) = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x}$.
  - Si $\mathbf{A}$ est carrée symétrique et $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{Ax}$ alors $\nabla f(\mathbf{x}) = 2\mathbf{Ax}$.

Régression linéaire multiple et MCO (suite)

- On développe $scr(f)$ de la manière suivante :

$$\begin{aligned} scr(f) &= (\mathbf{y} - \mathbf{Xa})^\top (\mathbf{y} - \mathbf{Xa}) \\ &= \left(\mathbf{y}^\top - (\mathbf{Xa})^\top\right)(\mathbf{y} - \mathbf{Xa}) \\ &= \mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top\mathbf{Xa} - (\mathbf{Xa})^\top\mathbf{y} + (\mathbf{Xa})^\top\mathbf{Xa} \\ &= \mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top\mathbf{Xa} - \mathbf{a}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{a}^\top\mathbf{X}^\top\mathbf{Xa} \end{aligned}$$

- On a donc la solution suivante :

$$\begin{aligned} \nabla scr(f) = \mathbf{0} &\Leftrightarrow 2\mathbf{X}^\top\mathbf{Xa} - 2\mathbf{X}^\top\mathbf{y} = \mathbf{0} \\ &\Leftrightarrow \hat{\mathbf{a}}_{mco} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y} \end{aligned}$$

Régression linéaire multiple et MCO (suite)

- Une fois estimé $\hat{\mathbf{a}}_{mco}$ on peut calculer les prédictions du modèle pour un quelconque $\mathbf{x} \in \mathbb{X}$:

$$\hat{f}(\mathbf{x}) = \mathbf{x}^\top \hat{\mathbf{a}}_{mco} = \mathbf{x}^\top \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

- Pour calculer l'erreur de prédiction on regarde ce que prédit le modèle estimé pour les données $\mathbb{E}$ données par les lignes de $\mathbf{X}$:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{a}}_{mco} = \mathbf{X}\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

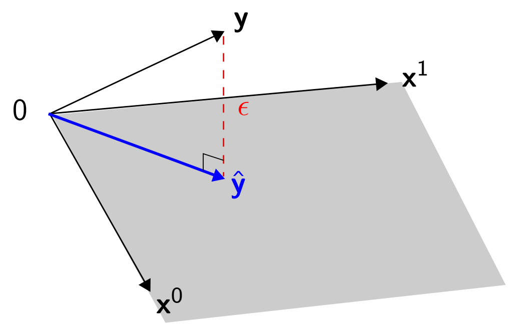
- L'erreur de prédiction est donc donnée par :

$$\begin{aligned} scr(\hat{f}) &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \end{aligned}$$

  où $\|.\|$ est la norme euclidienne.

Régression linéaire multiple et MCO (suite)

- Interprétation géométrique :



$$\hat{\mathbf{y}} = \underbrace{\mathbf{X}\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top}_{\text{Opérateur de projection}} \mathbf{y}$$

- Les MCO consistent à projeter orthogonalement $\mathbf{y}$ sur le sous-espace de $\mathbb{R}^n$ engendré par $\{\mathbf{x}^0, \ldots, \mathbf{x}^p\}$ (les $p+1$ colonnes de $\mathbf{X}$).
- Remarque : comme on cherche à minimiser $scr(f)$, on voit que la plus courte distance entre $\mathbf{y}$ et le sous-espace est donnée par la projection orthogonale.

Régression linéaire multiple et modèle gaussien

- Nous réinterprétons la régression linéaire multiple dans un cadre probabiliste. Nous avons le modèle suivant pour tout $i = 1, \ldots, n$:

$$Y_i = X_i^\top \mathbf{a} + \epsilon_i$$

- Nous faisons de plus l'hypothèse que le vecteur $\epsilon = (\epsilon_1, \ldots, \epsilon_n) \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$ où $\mathbf{I}_n$ est la matrice identité d'ordre n.
- Autrement dit les ϵ_i sont i.i.d. selon $\mathcal{N}(0, \sigma^2)$.
- On en déduit la relation suivante :

$$P(Y|X; \mathbf{a}, \sigma^2) \sim \mathcal{N}(X^\top \mathbf{a}, \sigma^2)$$

- L'étude de la régression linéaire multiple dans un cadre probabiliste nous permet d'introduire le principe d'inférence de **maximum de vraisemblance (MV)** et des propriétés statistiques des estimateurs associés.

Régression linéaire multiple et modèle gaussien (suite)

- La **vraisemblance ("likelihood")** est la probabilité d'observer l'échantillon :

$$vr(\mathbf{a}, \sigma^2) = P(Y_1, \ldots, Y_n | X_1, \ldots, X_n; \mathbf{a}, \sigma^2)$$

- Les Y_i sont supposés i.i.d. nous avons alors :

$$\begin{aligned} vr(\mathbf{a}, \sigma^2) &= \prod_{i=1}^{n} P(Y_i | X_i; \mathbf{a}, \sigma^2) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{Y_i - X_i^\top \mathbf{a}}{\sigma}\right)^2\right) \end{aligned}$$

- L'estimateur du MV est la valeur des paramètres qui maximise la probabilité d'observer l'échantillon. On résoud donc le problème :

$$\max_{(\mathbf{a}, \sigma^2) \in \mathbb{R}^{p+1} \times \mathbb{R}} \prod_{i=1}^{n} P(Y_i | X_i; \mathbf{a}, \sigma^2)$$

Régression linéaire multiple et modèle gaussien (suite)

- Il est plus commode de maximiser, de manière équivalente, le logarithme de la vraisemblance :

$$lvr(\mathbf{a},\sigma^2) = \log\left(\prod_{i=1}^{n} P(Y_i|X_i;\mathbf{a},\sigma^2)\right) = \sum_{i=1}^{n}\log(P(Y_i|X_i;\mathbf{a},\sigma^2))$$

- Dans le modèle gaussien cela se réduit à :

$$\begin{aligned} lvr(\mathbf{a},\sigma^2) &= \sum_{i=1}^{n}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2}\left(\frac{Y_i - X_i^\top\mathbf{a}}{\sigma}\right)^2\right)\right)\\ &= \sum_{i=1}^{n}\left(-\log\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2}\left(\frac{Y_i - X_i^\top\mathbf{a}}{\sigma}\right)^2\right)\\ &= -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(Y_i - X_i^\top\mathbf{a}\right)^2 \end{aligned}$$

- Nous avons la propriété suivante : $\max lvr(\mathbf{a},\sigma^2) \Leftrightarrow \min scr(f)$

Régression linéaire multiple et modèle gaussien (suite)

- Estimateur du MV :

$$\mathbf{a}_{mv} = \arg\max_{\mathbf{a}\in\mathbb{R}^{p+1}}\sum_{i=1}^{n}\log(P(Y_i|X_i;\mathbf{a},\sigma^2))$$

- On a la solution analytique :

$$\mathbf{a}_{mv} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top Y$$

- Espérance de $\mathbf{a}_{mv}$:

$$\begin{aligned} \mathrm{E}_{Y|X}(\mathbf{a}_{mv}|\mathbf{X}) &= \mathrm{E}_{Y|X}\left(\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top Y|\mathbf{X}\right)\\ &= \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathrm{E}_{Y|X}(Y|\mathbf{X})\\ &= \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{a} \quad = \quad \mathbf{a} \end{aligned}$$

- L'estimateur du MV est donc **sans biais**.

Régression linéaire multiple et modèle gaussien (suite)

- Variance de $\hat{\mathbf{a}}_{mv}$:

$$\begin{aligned} \mathrm{V}_{Y|X}(\mathbf{a}_{mv}|\mathbf{X}) &= \mathrm{V}_{Y|X}\left(\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top Y|\mathbf{X}\right)\\ &= \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathrm{V}_{Y|X}(Y|\mathbf{X})\,\mathbf{X}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\\ &= \sigma^2\left(\mathbf{X}^\top\mathbf{X}\right)^{-1} \end{aligned}$$

- **Efficacité** de l'estimateur du MV :

Théorème. (Théorème de Gauss-Markov)

Sous l'hypothèse que les résidus $\epsilon_1,\ldots,\epsilon_n$ sont i.i.d. et suivent une loi normale $\mathcal{N}(0,\sigma^2)$, l'estimateur $\mathbf{a}_{mv} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top Y$ est, parmi les estimateurs sans biais, celui de variance minimale

- On verra toutefois que d'autres estimateurs ont une **variance plus petite mais au prix d'un biais** !

Régularisation des modèles de régression linéaire

- L'estimateur du MV ou des MCO est de variance minimale parmi les estimateurs sans biais néanmoins, la variance aboutit dans certains cas à des erreurs de prédiction fortes. Dans cette situation, **on cherche des estimateurs de variance plus petite quite à avoir un léger biais**. On peut atteindre cet objectif en supprimant l'effet de certaines variables explicatives ce qui revient à leur attribuer un coefficient nul.
- Par ailleurs, dans le cas où p, le nombre de variables explicatives, est grand, l'interprétation des résultats obtenus par les MCO est parfois ardu. Ainsi, on pourra préférer un **modèle estimé avec moins de variables explicatives** afin de privilégier l'interprétation du phénomène sous-jacent aux données plutôt que la précision.
- On étudie ici des méthodes permettant de produire des estimateurs dont les valeurs sont d'amplitudes réduites. Notamment, on parle de **modèles parcimonieux** lorsque des variables ont des coefficients nuls.
- Dans ce qui suit nous verrons trois approches : la régression ridge, la régression lasso et la régression elasticnet.

Régression ridge

- Nous sommes toujours dans le même contexte que précédemment et avons l'espace des hypothèses suivant :
  $\mathbb{H} = \{f : \mathbb{R}^p \to \mathbb{R} : f(X) = a_0 + \sum_{j=1}^p a_j X^j\}$
- Soit $\mathbf{a}_{\setminus 0}$ le vecteur $(a_1, \ldots, a_p)$.
- L'estimateur ridge noté $\hat{\mathbf{a}}_{ridge}$ est défini de la manière suivante :

$$\hat{\mathbf{a}}_{ridge} = \arg \min_{\mathbf{a}\in\mathbb{R}^{p+1}} \left\{ \sum_{i=1}^n \left(y_i - (a_0 + \sum_{j=1}^p a_j x_{ij}) \right)^2 + \lambda \|\mathbf{a}_{\setminus 0}\|_{\ell_2}^2 \right\}$$

- $R(\mathbf{a}_{\setminus 0}) = \|\mathbf{a}\|_{\ell_2}^2 = \sum_{j=1}^p a_j^2$ est appelé **fonction de pénalité**.
- λ est un réel positif ou nul qui permet de contrôler l'amplitude des valeurs $\{a_j\}_{j=1}^p$ (càd la norme du vecteur $\mathbf{a}_{\setminus 0}$). On parle de **coefficient de pénalité** ou de "**shrinkage**" (rétrécissement).
- Plus λ est grand plus la valeur des coefficients se rapproche de 0 et moins la variance de l'estimateur de $\mathbf{a}$ est grande.

Régression ridge (suite)

- Une autre façon équivalente d'introduire la régression ridge est par le biais du problème d'optimisation contraint suivant :

$$\begin{array}{ll} & \min_{\mathbf{a}\in\mathbb{R}^{p+1}} \sum_{i=1}^n \left(Y_i - (a_0 + \sum_{j=1}^p a_j X^j) \right)^2 \\ slc & \sum_{j=1}^p a_j^2 \leq \tau \end{array}$$

- On montre qu'il existe une bijection entre λ et τ ce qui rend équivalent les deux problèmes.
- Cette formulation permet d'exprimer explicitement la contrainte sur l'amplitude des coefficients : on voit effectivement qu'il s'agit de minimiser $scr(f)$ avec la contrainte que $\mathbf{a}_{\setminus 0}$ appartienne à une boule de $\mathbb{R}^p$ et de rayon τ.
- Géométriquement : si $\hat{\mathbf{a}}_{\setminus 0, mco}$ appartient à la boule alors $\hat{\mathbf{a}}_{mco} = \hat{\mathbf{a}}_{ridge}$ sinon, on projette $\hat{\mathbf{a}}_{\setminus 0, mco}$ sur la boule (pour satisfaire la contrainte).

Régression ridge (suite)

- Contrairement à la régression linéaire multiple classique où on ne normalise pas nécessairement les variables, ici il est nécessaire de **réduire les variables explicatives** avant de résoudre le problème d'optimisation. En effet, si les variables sont dans des unités de mesures non commensurables le terme de pénalité (càd la contrainte) aura un impact non uniforme sur les X^j.
- En pratique, il faut également **centrer la matrice de données X** à laquelle on enlève la première colonne remplie de 1. On supposera donc par la suite que la matrice $\mathbf{X}$ est de taille $(n \times p)$ et est centrée-réduite, $\forall j = 1, \ldots, p$:

$$\frac{1}{n}\sum_{i=1}^n x_{ij} = 0 \text{ et } \frac{1}{n}\sum_{i=1}^n x_{ij}^2 = 1$$

Régression ridge (suite)

- On **centre le vecteur y** également et on suppose par la suite :

$$\frac{1}{n}\sum_{i=1}^n y_i = 0$$

- L'ordonnée à l'origine a_0 n'intervient pas dans la fonction de pénalité car ceci rendrait la fonction de prédiction dépendante d'une ordonnée à l'origine que l'on trouverait pour Y.
- On montre en fait que si $\mathbf{X}$ est centrée, on peut séparer l'estimation du modèle en deux étapes :
  1. On prend $\hat{a}_{ridge,0} = \bar{\mathbf{y}}$.
  2. On estime $(a_1, \ldots, a_p)$ en résolvant :

$$\hat{\mathbf{a}}_{ridge} = \arg \min_{\mathbf{a}\in\mathbb{R}^p} \left\{ \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij} \right)^2 + \lambda \|\mathbf{a}\|^2 \right\}$$

où $\mathbf{a} = (a_1, \ldots, a_p) \in \mathbb{R}^p$.

Régression ridge (suite)

- L'écriture matricielle de la fonction objectif devient :

$$\hat{\mathbf{a}}_{ridge} = \arg\min_{\mathbf{a}\in\mathbb{R}^p} \left\{ (\mathbf{y}-\mathbf{X}\mathbf{a})^\top(\mathbf{y}-\mathbf{X}\mathbf{a}) + \lambda\mathbf{a}^\top\mathbf{a} \right\}$$

- On a la **solution analytique** suivante :

$$\hat{\mathbf{a}}_{ridge} = \left(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_p\right)^{-1}\mathbf{X}^\top\mathbf{y}$$

- Les prédictions sont alors données par :

$$\hat{\mathbf{y}} = \underbrace{\overline{\mathbf{y}}}_{\hat{a}_{ridge,0}}\mathbf{1}_n + \mathbf{X}\hat{\mathbf{a}}_{ridge} \quad \text{et} \quad \hat{f}(\mathbf{x}) = \underbrace{\overline{\mathbf{y}}}_{\hat{a}_{ridge,0}} + \mathbf{x}^\top\hat{\mathbf{a}}_{ridge}$$

où $\mathbf{x}$ est un vecteur quelconque de taille $(p\times 1)$ et $\mathbf{x}$ est de terme général : $x_j = \frac{x_j-\overline{\mathbf{x}}^j}{\hat{\sigma}_{\mathbf{x}^j}}$.

Régression ridge (suite)

- Espérance de $\hat{\mathbf{a}}_{ridge}$:

$$\mathrm{E}_{Y|X}(\hat{\mathbf{a}}_{ridge}|\mathbf{X}) = \mathbf{a} - \underbrace{\lambda\left(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_p\right)^{-1}\mathbf{a}}_{biais}$$

- L'estimateur ridge de $\mathbf{a}$ n'est donc pas sans biais et de ce point de vue il est moins bon que l'estimateur des MCO.
- Variance de $\hat{\mathbf{a}}_{ridge}$:

$$\mathrm{V}_{Y|X}(\hat{\mathbf{a}}_{ridge}|\mathbf{X}) = \sigma^2\left(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_p\right)^{-1}\mathbf{X}^\top\mathbf{X}\left(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_p\right)^{-1}$$

- $\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_p$ et $\mathbf{X}^\top\mathbf{X}$ ont vecteurs propres mais les valeurs propres de $\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_p$ sont plus grandes que celles de $\mathbf{X}^\top\mathbf{X}$.
- On en déduit que la variance de l'estimateur de $\mathbf{a}_{ridge}$ est plus petite que celle de $\mathbf{a}_{mco}$. De ce point de vue, on peut attendre de la régression ridge qu'elle donne des prédictions meilleures que celles de la régression linéaire classique sur des données non observées.

Régression ridge (suite)

- Comment choisir la valeur de λ, le coefficient de pénalité ?
- On prend une séquence de nombres $\mathbb{S}$ allant de 0 jusqu'à un nombre positif maximal, on remplace λ par chacune de ses valeurs et on estime itérativement plusieurs modèles et on sélectionne le meilleur λ selon un critère.
- Il existe plusieurs méthodes. On propose ici une approche basée sur la validation croisée LOOCV :

$$\hat{\lambda} = \arg\min_{\lambda\in\mathbb{S}} \sum_{i=1}^{n}\left(y_i - \hat{f}_{\lambda,\setminus i}(\mathbf{x}_i)\right)^2$$

où $\hat{f}_{\lambda,\setminus i}$ est le modèle estimé avec λ mais sans l'objet $\mathbf{x}_i$ (LOOCV).
- La procédure LOOCV est coûteuse dès lors que la taille de l'échantillon est grand et/ou $|\mathbb{S}|$ est grand.

Régression ridge (suite)

- Si la procédure LOOCV est trop coûteuse, on pourra choisir λ qui minimise le critère de validation croisée généralisée :

$$gcv(\lambda) = \sum_{i=1}^{n}\left(\frac{y_i - \hat{f}_\lambda(\mathbf{x}_i)}{1 - tr(\mathbf{X}(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_p)^{-1}\mathbf{X}^\top)/n}\right)^2$$

- Ici pas de LOOCV, on estime un seul modèle pour chaque λ : $\hat{f}_\lambda$
- *gcv* tient compte de la quantité suivante appelée également **nombre de degrés de liberté effectifs** :

$$\begin{aligned} ddl(\lambda) &= tr(\mathbf{X}(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I}_p)^{-1}\mathbf{X}^\top) \\ &= \sum_{j=1}^{p}\frac{d_j^2}{d_j^2+\lambda} \end{aligned}$$

- Quand $\lambda = 0$ (pas de pénalité), $ddl(0) = p$ le nombre de variables libres mais quand $\lambda \to +\infty$ on a $ddl(\lambda)\to 0$.

Régression lasso

- Dans l'idée, la régression lasso est très proche de la régression ridge. L'estimateur lasso noté $\hat{\mathbf{a}}_{lasso}$ est défini de la manière suivante :

$$\hat{\mathbf{a}}_{lasso} = \arg \min_{\mathbf{a} \in \mathbb{R}^{p+1}} \left\{ \sum_{i=1}^{n} \left(y_i - (a_0 + \sum_{j=1}^{p} a_j x_{ij}) \right)^2 + \lambda \|\mathbf{a}_{\backslash 0}\|_{\ell_1} \right\}$$

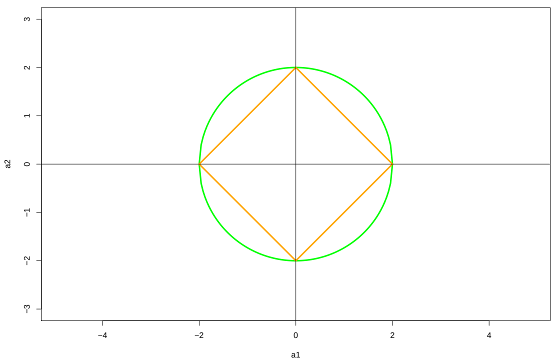
- $R(\mathbf{a}_{\backslash 0}) = \|\mathbf{a}_{\backslash 0}\|_{\ell_1} = \sum_{j=1}^{p} |a_j|$ est une **fonction de pénalité** basée sur la norme ℓ_1 plutôt que la norme ℓ_2 comme c'est le cas pour la régression ridge.
- Le problème précédent est aussi équivalent au problème d'optimisation contraint :

$$\begin{array}{ll} & \min_{\mathbf{a} \in \mathbb{R}^{p+1}} \sum_{i=1}^{n} \left(Y_i - (a_0 + \sum_{j=1}^{p} a_j X^j) \right)^2 \\ slc & \sum_{j=1}^{p} |a_j| \leq \tau \end{array}$$

Régression lasso (suite)

- La différence de norme dans la fonction de pénalité a en fait un impact important. Il existe ainsi des différences fortes entre régression lasso et régression ridge :
- Contrairement à la régression ridge, il n'y a pas de solution analytique car la valeur absolue rend le problème non différentiable.
- On a donc recours à des méthodes d'optimisation numérique ou à des algorithmes spécifiques ("Least Angle Regression - Stagewise" [Efron et al., 2004]).
- Quand τ est relativement petit, la solution obtenue par **la régression lasso est parcimonieuse** càd que certains coefficients estimés seront nuls. La régression lasso peut ainsi être vue comme une **méthode de sélection de variables**. Il s'agit d'un modèle davantage parcimonieux que les modèles précédents. Lasso : "Least Absolute Shrinkage and Selection Operator".
- Quand $\tau \geq \|\hat{\mathbf{a}}_{mco}\|_{\ell_1}$ alors $\hat{\mathbf{a}}_{lasso} = \hat{\mathbf{a}}_{mco}$.

Comparaison ridge et lasso - domaines de recherche



- En vert la frontière ridge : $a_1^2 + a_2^2 = 2$.
- En orange la frontière lasso : $|a_1| + |a_2| = 2$.

Régression lasso (suite)

- En pratique, comme pour la régression ridge, on centre-réduit $\mathbf{X}$ et on centre $\mathbf{y}$. $\mathbf{X}$ étant centrée, on peut à nouveau de séparer en deux l'inférence. On retrouve notamment dans ce cas $\hat{a}_{lasso,0} = \bar{\mathbf{y}}$.
- En prenant $\mathbf{a} = (a_1, \ldots, a_p)$, l'estimateur lasso est donné par :

$$\hat{\mathbf{a}}_{lasso} = \arg \min_{\mathbf{a} \in \mathbb{R}^{p}} \left\{ \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} a_j x_{ij} \right)^2 + \lambda \|\mathbf{a}\|_{\ell_1} \right\}$$

- Les prédictions sont calculées de la façon suivante :

$$\hat{\mathbf{y}} = \underbrace{\bar{\mathbf{y}}}_{\hat{a}_{lasso,0}} \mathbf{1}_n + \mathbf{X}\hat{\mathbf{a}}_{lasso} \quad \text{et} \quad \hat{f}(\mathbf{x}) = \underbrace{\bar{\mathbf{y}}}_{\hat{a}_{lasso,0}} + \mathbf{x}^\top \hat{\mathbf{a}}_{lasso}$$

où $\mathbf{x}$ est un objet quelconque et $\mathbf{x}$ est un vecteur de taille $(p \times 1)$ et de terme général : $x_j = \frac{x_j - \bar{\mathbf{x}}^j}{\hat{\sigma}_{\mathbf{x}^j}}$.

Régression lasso (suite)

- Comment choisir le coefficient de pénalité ?
- On considère le problème équivalent suivant :

$$\min_{\mathbf{a}\in\mathbb{R}^p} \sum_{i=1}^n \left(Y_i - \sum_{j=1}^p a_j X^j\right)^2$$
$$slc \quad \frac{\sum_{j=1}^p |a_j|}{\|\hat{\mathbf{a}}_{mco}\|_{\ell_1}} \leq \tau$$

  où $\|\hat{\mathbf{a}}_{mco}\|_{\ell_1}$ est une constante précalculée.
- Une méthode consiste alors à faire varier τ de 0 à 1. On voit que lorsque τ vaut 1 on a $\hat{\mathbf{a}}_{lasso} = \hat{\mathbf{a}}_{mco}$.
- On peut alors procéder par validation croisée LOOCV comme précédemment.
- La décision peut être basée sur plusieurs types de critères. On prendra typiquement *mse*.

Procédures classiques de sélection de modèle

- Le lasso permet de faire de la **sélection de modèles** mais rappelons au préalable qu'il existe des techniques simples dans ce cas :
  - **Recherche exhaustive** du meilleur sous-ensemble de variables. Si p est le nombre d'attributs, il y a 2^p possibilités (impraticable si p est grand).
  - **Recherche pas à pas** (approche gloutone, localement optimale) :
    - **ascendante** : on ajoute itérativement la variable qui permet d'améliorer le plus un critère de sélection de modèles,
    - **descendante** : on part du modèle avec p variables et on enlève itérativement la variable qui permet d'améliorer le plus un critère de sélection de modèles.
- Les **critères de selection de modéles** sont de plusieurs sortes :
  - R^2 ajusté,
  - C_p de Mallows,
  - le critère AIC (Akaïke Information Criterion),
  - le critère BIC (Bayesian Information Criterion) ...

Least Angle Regression - Stagewise (lars)

- Nous avons cité précédemment l'algorithme lars. Il s'agit d'un algorithme efficace permettant de déterminer le **chemin de régularisation** ("regularization path" ou "solution path") du lasso càd l'ensemble des solutions $\hat{\mathbf{a}}_{lasso}(\lambda)$ pour $\lambda \in [0, \infty]$ [Efron et al., 2004].
- $\hat{\mathbf{a}}_{lasso}(\lambda)$ est linéaire par morceau (cf slide plus loin).
- L'algorithme est proche d'une méthode de recherche pas à pas ascendante dite "**forward stagewise regression**" où on cherche itérativement la variable la plus corrélée avec le vecteur des résidus.
- lars commence avec $\lambda = \infty$ et dans ce cas $\hat{\mathbf{a}}_{lasso} = \mathbf{0}$. Puis il détermine itérativement la valeur λ (qui décroît) permettant de faire entrer une variable dans l'ensemble actif (càd tel que le coefficient est non nul). Lorsque $\lambda = 0$ on obtient la solution des MCO.
- L'algorithme a une complexité cubique en p. Plus récemment des méthodes d'optimisation numérique ("cyclic coordinate descent") ont montré de meilleures performances que lars [Friedman et al., 2010].

Limites de la régression lasso

- Quand $p > n$ (données de grande dimension), la méthode lasso ne sélectionne que n variables (en raison de la nature même du modèle d'optimisation sous-jacent).
- Si plusieurs variables sont corrélées entre elles, la méthode lasso ne sélectionnera qu'une seule d'entre elles et ignorera les autres.
- Dans de nombreux cas classiques avec $n > p$, s'il y a de fortes corrélations entre les variables explicatives, on trouve empiriquement que la méhode ridge donne de meilleures performances que la méthode lasso.

Mélange de ridge et de lasso

- On se place dans le même contexte que pour ridge et lasso où $\mathbf{X}$ est centrée-réduite et $\mathbf{y}$ est centrée.
- Pour surmonter les problèmes précédents, l'idée est de prendre un mélange de ridge et lasso. On obtient alors le problème suivant :

$$\min_{\mathbf{a}\in\mathbb{R}^p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij}\right)^2 + \lambda_1 \|\mathbf{a}\|_{\ell_1} + \lambda_2 \|\mathbf{a}\|_{\ell_2}^2$$

où λ_1 et λ_2 sont des paramètres de positifs ou nuls.
- En posant $\alpha = \lambda_1/(\lambda_1 + \lambda_2)$ on peut montrer que le problème est équivalent à :

$$\min_{\mathbf{a}\in\mathbb{R}^p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij}\right)^2 + \lambda \left(\alpha \|\mathbf{a}\|_{\ell_1} + (1-\alpha)\|\mathbf{a}\|_{\ell_2}^2\right)$$

La régression elasticnet

- L'estimateur elasticnet naïf (cf plus loin) est défini par [Zou and Hastie, 2005] :

$$\begin{aligned}\hat{\mathbf{a}}_{nen} &= \arg\min_{\mathbf{a}\in\mathbb{R}^p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij}\right)^2 \\ &\quad + \lambda \left(\alpha \|\mathbf{a}\|_{\ell_1} + (1-\alpha)\|\mathbf{a}\|_{\ell_2}^2\right)\end{aligned}$$

où $R(\mathbf{a}) = \alpha\|\mathbf{a}\|_{\ell_1} + (1-\alpha)\|\mathbf{a}\|_{\ell_2}^2$ est la **fonction de pénalité** de la régression elasticnet.
- Ce problème est équivalent au problème contraint suivant :

$$\begin{aligned} &\min_{\mathbf{a}\in\mathbb{R}^p} \textstyle\sum_{i=1}^n \left(Y_i - \sum_{j=1}^p a_j X^j\right)^2 \\ slc \quad &\alpha \textstyle\sum_{j=1}^p |a_j| + (1-\alpha)\sum_{j=1}^p |a_j|^2 \leq \tau \end{aligned}$$

- Si $\alpha = 0$ on a l'estimateur ridge et si $\alpha = 1$ on a l'estimateur lasso. Les cas nous intéressant sont donc ceux pour lesquels $0 < \alpha < 1$.

La régression elasticnet (suite)

Lemme.

Soit $\mathbf{X}$ la matrice des variables explicatives de taille $(n \times p)$, et $\mathbf{y}$ le vecteur de la variable cible réelle de taille n. Soit $(\lambda_1, \lambda_2) \in \mathbb{R}^+ \times \mathbb{R}^+$. Soient les données augmentées $\mathbf{X}^*$ et $\mathbf{y}^*$ de tailles respectives $((n+p)\times p)$ et $n+p$:

$$\mathbf{X}^* = \frac{1}{\sqrt{1+\lambda_2}} \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\mathbf{I}_p \end{pmatrix} \quad et \ \mathbf{y}^* = \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix}$$

Soit $\gamma = \lambda_1/\sqrt{1+\lambda_2}$. Alors la fonction objectif de la régression eslaticnet peut s'écrire de façon équivalente :

$$\|\mathbf{y}^* - \mathbf{X}^*\mathbf{a}^*\|_{\ell_2}^2 + \gamma\|\mathbf{a}^*\|_{\ell_1}$$

Soit $\hat{\mathbf{a}}^*$ le minimiseur de cette fonction on a alors :

$$\hat{\mathbf{a}}_{nen} = \frac{1}{\sqrt{1+\lambda_2}} \hat{\mathbf{a}}^*$$

La régression elasticnet

- Ce lemme de [Zou and Hastie, 2005] montre que la solution de la régression elasticnet peut être obtenue par la solution de la régression lasso avec des données augmentées !
- Comme $\mathbf{X}^*$ est de rang p, la solution elasticnet peut donc sélectionner potentiellement p variables contrairement à la régression lasso.
- Ce lemme permet également de montrer que la méthode elasticnet permet de faire de la **sélection de variables** comme la méthode lasso et contrairement à la méthode ridge.
- Dans le cas de grande dimension $n << p$, on observe souvent un **effet de groupes** entre variables qui ont tendance à être linéairement dépendantes. La régression elasticnet permet de tenir compte de cet effet : les variables fortement corrélées ont tendance à avoir la même valeur de coefficient dans $\hat{\mathbf{a}}_{nen}$.

Effet de groupe de la régression elasticnet

Théorème.

Soient $\mathbf{X}$ *et* $\mathbf{y}$ *les données du problème de régression où les variables explicatives sont supposées centées-réduites et la variable à expliquer centrée. Soit* (λ_1, λ_2) *des paramètres non négatifs. Soit* $\hat{\mathbf{a}}_{nen}(\lambda_1, \lambda_2)$ *la solution elasticnet naïve. Supposons que* $\hat{a}_{nen,i}(\lambda_1, \lambda_2)\hat{a}_{nen,j}(\lambda_1, \lambda_2) > 0$ *alors :*

$$\frac{1}{\|\mathbf{y}\|_{\ell_1}}|\hat{a}_{nen,i}(\lambda_1, \lambda_2) - \hat{a}_{nen,j}(\lambda_1, \lambda_2)| \leq \frac{1}{\lambda_2}\sqrt{2(1-\rho_{ij})}$$

où $\rho_{ij} = \langle \mathbf{x}^i, \mathbf{x}^j \rangle$ *est le coefficient de corrélation entre les* X^i *et* X^j*.*

- Ce théorème de [Zou and Hastie, 2005] permet de caractériser l'effet de groupe d'elasticnet : l'écart entre les coefficients de deux variables est borné supérieurement par une grandeur qui dépend de la corrélation linéaire entre celles-ci.

Ré-échelonnement de l'estimateur elasticnet naïf

- L'estimateur elasticnet vu jusqu'à présent est dit "**naïf**".
- En théorie, il permet de tenir compte des limites du lasso identifiées précédemment.
- En pratique, il ne donne satisfaction que lorsqu'il est proche de l'estimateur ridge ou de l'estimateur lasso.
- Ce comportement est en fait dû à un **double effet de rétrécissement** qui porte atteinte au modèle (on a une faible diminution de la variance pour une forte augmentation du biais).
- L'**estimateur elasticnet** $\hat{\mathbf{a}}_{en}$ retenu est alors un **ré-échelonnement** de la solution précédente :

$$\hat{\mathbf{a}}_{en} = (1+\lambda_2)\hat{\mathbf{a}}_{nen} = \sqrt{1+\lambda_2}\hat{\mathbf{a}}^*$$

- En pratique, l'estimateur ré-échelonné $\hat{\mathbf{a}}_{en}$ donne de meilleurs résultats pour $0 < \alpha < 1$ et peut ainsi surpasser le lasso.
- Pourquoi ce facteur $(1+\lambda_2)$?

Elasticnet vue comme stabilisation du lasso

- On peut écrire la fonction objectif de l'estimateur lasso comme suit :

$$\hat{\mathbf{a}}_{lasso} = \arg\min_{\mathbf{a}\in\mathbb{R}^p} \mathbf{a}^\top(\mathbf{X}^\top\mathbf{X})\mathbf{a} - 2\mathbf{y}^\top\mathbf{X}\mathbf{a} + \lambda_1\|\mathbf{a}\|_{\ell_1}$$

- On montre que celle de l'estimateur elasticnet peut s'écrire :

$$\hat{\mathbf{a}}_{en} = \arg\min_{\mathbf{a}\in\mathbb{R}^p} \mathbf{a}^\top\left(\frac{\mathbf{X}^\top\mathbf{X}+\lambda_2\mathbf{I}_p}{1+\lambda_2}\right)\mathbf{a} - 2\mathbf{y}^\top\mathbf{X}\mathbf{a} + \lambda_1\|\mathbf{a}\|_{\ell_1}$$

- Les variables étant centrées-réduites, $\mathbf{X}^\top\mathbf{X} = \hat{\mathbf{\Sigma}}$ la matrice variance-covariance empirique. On a par ailleurs :

$$\frac{\mathbf{X}^\top\mathbf{X}+\lambda_2\mathbf{I}_p}{1+\lambda_2} = (1-\gamma)\hat{\mathbf{\Sigma}} + \gamma\mathbf{I}_p$$

  en posant $\gamma = 1/(1+\lambda_2)$.
- Elasticnet peut donc être vu comme un lasso où la matrice de variance-covariance est rétrécie càd proche de la matrice identité.

Régression logistique polytomique

- Nous avons traité la régression par le modèle linéaire pénalisé.
- Dans le cadre du problème de catégorisation, le modèle linéaire (généralisé) adéquat est la **régression logistique polytomique**.
- Nous rappelons ce modèle et présentons sa version pénalisée.
- La régression logistique vise à modéliser la probabilité conditionnelle de chaque classe C_l étant donné le vecteur aléatoire X.
- Ce sont en fait le logarithme des odds-ratio qui sont modélisés par des fonctions linéaires. On a $q-1$ fonctions linéaires suivantes :

$$\begin{aligned}
\log\frac{P(Y=C_1|X=\mathbf{x})}{P(Y=C_q|X=\mathbf{x})} &= a_{10} + \mathbf{a}_1^\top\mathbf{x} \\
&\vdots \\
\log\frac{P(Y=C_{q-1}|X=\mathbf{x})}{P(Y=C_q|X=\mathbf{x})} &= a_{q-10} + \mathbf{a}_{q-1}^\top\mathbf{x}
\end{aligned}$$

  C_q au dénominateur est une classe de référence qui est arbitraire.

Régression logistique polytomique (suite)

- Les membres de gauche sont des **fonctions logit** :

$$\text{logit}(p) = \log(\frac{p}{1-p}), \text{ avec } p \in]0,1[$$

- Chaque fonction logit compare une classe $C_1, \ldots, C_{q-1}$ à une classe de référence C_q et est modélisée par une fonction affine.
- Le modèle spécifié précédemment conduit aux propriétés suivantes $\forall k = 1, \ldots, q-1$:

$$P(Y = C_k|X = \mathbf{x}) = \frac{\exp(a_{k0} + \mathbf{a}_k^\top \mathbf{x})}{1 + \sum_{l=1}^{q-1} \exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x})}$$

$$P(Y = C_q|X = \mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{q-1} \exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x})}$$

- Nous pouvons clairement vérifier que $\sum_{l=1}^{q} P(Y = C_l|X = \mathbf{x}) = 1$.
- Ici, l'ensemble des paramètre est $\mathbb{P} = \{(a_{l0}, \mathbf{a}_l)\}_{l=1}^{q-1}$.

Régression logistique polytomique (suite)

- Dans l'équation précédente, on voit que la classe C_q, prise comme référence, est traitée de manière particulière. Afin de rendre les classes uniformes nous poserons plus particulièrement :

$$\forall k = 1, \ldots, q \; : P(Y = C_k|X = \mathbf{x}) = \frac{\exp(a_{k0} + \mathbf{a}_k^\top \mathbf{x})}{\sum_{l=1}^{q} \exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x})}$$

- La fonction $\frac{\exp(a_{k0} + \mathbf{a}_k^\top \mathbf{x})}{\sum_{l=1}^{q} \exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x})}$ est appelée fonction **softmax** et nous voyons que dans ce cas $\sum_{l=1}^{q} P(Y = C_l|X = \mathbf{x}) = 1$.
- L'appellation softmax vient du fait que s'il existe une classe C_k telle que $a_{k0} + \mathbf{a}_k^\top \mathbf{x}$ est largement supérieure aux autres classes $C_l \neq C_k$ alors, la fonction softmax retourne une valeur proche de 1. Ainsi la fonction agit comme la "fonction max" excepté qu'**elle est différentiable**.

Régression logistique polytomique (suite)

- Dans le cas multiclasse, nous représentons l'appartenance des objets aux différentes classes par une matrice binaire $\mathbf{Y}$ de taille $(n \times q)$ et de terme général :

$$y_{il} = \begin{cases} 1 & \text{si } \mathbf{x}_i \in C_l \\ 0 & \text{sinon} \end{cases}$$

- On modélise la probabilité par une **distribution multinomiale**. Sous l'hypothèse i.i.d., la vraisemblance s'écrit alors comme suit :

$$vr(\mathbb{P}) = \prod_{l=1}^{q} \prod_{i=1}^{n} P(Y = C_l|\mathbf{x}_i; \mathbf{a}_l)^{y_{il}}$$

- La log-vraisemblance vaut alors :

$$lvr(\mathbb{P}) = \sum_{l=1}^{q} \sum_{i=1}^{n} y_{il} \log(P(Y = C_l|\mathbf{x}_i; \mathbf{a}_l))$$

Régression logistique polytomique (suite)

- En remplaçant $P(Y = C_l|\mathbf{x}_i; \mathbf{a}_l)$ par la forme paramétrique introduite précédemment, on a :

$$lvr(\mathbb{P}) = \sum_{l=1}^{q} \sum_{i=1}^{n} y_{il} \log\left(\frac{\exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x}_i)}{\sum_{k=1}^{q} \exp(a_{k0} + \mathbf{a}_k^\top \mathbf{x}_i)}\right)$$

- Nous pouvons utiliser à nouveau l'**algorithme de Newton-Raphson** pour déterminer une solution approchée de l'estimateur du MV. Pour cela, il faut déterminer le gradient de la *lvr* par rapport à $\mathbf{a}_l$ ainsi que la matrice hessienne...

Pseudo-code de l'algorithme de Newton-Raphson

Input : $f \in \mathcal{C}^2, \mathbf{x}^0$
$k \leftarrow 0$
Tant que condition d'arrêt non satisfaite **faire**
  $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \nabla^2 f(\mathbf{x}^{(k)})^{-1} \nabla f(\mathbf{x}^{(k)})$
  $k \leftarrow k+1$
Fin Tant que
Output : $\mathbf{x}^{(k)}$

où la matrice $D^2 f(\mathbf{x})$ est appelée **matrice hessienne** de f en $\mathbf{x}$ avec :

$$\nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

Régression logistique pénalisée

- Le **principe de régularisation** pour obtenir un modèle de faible variance et/ou parcimonieux a été également appliqué à d'autres fonctions objectif que les MCO telle que la log-vraisemblance.
- En fait, le principe de pénalisation a été introduit en modélisation mathématique en 1963 par Tikhonov et en statistique en 1971 par Good et Gaskins.
- Dans le cas plus générale de la régression logistique polytomique, notons l'ensemble des paramètres $\mathbb{P} = \{(a_{l0}, \mathbf{a}_l) \in \mathbb{R}^{p+1}\}_{l=1}^q$ nous obtenons le modèle pénalisé suivant :

$$\max_{\{(a_0, \mathbf{a}_l)\}_l \in \mathbb{R}^{(p+1)q}} lvr(\mathbb{P}) - \lambda \sum_{l=1}^{q} \left((1-\alpha)\|\mathbf{a}_l\|_{\ell_1} + \alpha \|\mathbf{a}_l\|_{\ell_2}^2\right)$$

- Si $\alpha = 1$ on retrouve une pénalisation ℓ_2 et le problème peut-être estimé par l'algorithme de Newton-Raphson (le fonction objectif étant différentiable 2 fois).

Rappel du Sommaire

Introduction

- Ce sont des modèles qui s'inspirent du fonctionnement du cerveau et de travaux provenant des sciences cognitives et des neurosciences.
- Le cerveau peut être vu comme un système de traitement d'informations qui possède des capacités extraordinaires qui dépassent bien évidemment celles d'un ordinateur.
- Le cerveau effectue par exemple des tâches de reconnaissance de formes visuelles ou sonores, ou est capable d'apprendre à partir d'exemples ou d'un enseignant.
- Ces quelques tâches que nous venons de citer sont les problèmes que traite l'intelligence artificielle. Puisque le cerveau est capable de résoudre ces tâches alors il est tentant de modéliser le fonctionnement du cerveau, de le programmer et de demander à une machine de reproduire les capacités du cerveau.

Introduction (suite)

- Le cerveau est à bien des égards différent d’un ordinateur ! Mais si l’on devait appréhender le cerveau comme un système de traitement d’informations, on voit qu’un ordinateur à un (ou quelques) processeur alors que le cerveau est composé d’un très large nombre de “processeurs” que sont les **neurones** (le cerveau humain en compte environ 10^{11}).
- Toutefois, le neurone en tant qu’unité de traitement est plus “simple” qu’un processeur. Ce qui fait la particularité du cerveau est sa capacité à traiter de manière parallèle l’information. Ceci est possible par la très grande connectivité entre neurones reliés entre eux par les **synapses**.
- Les synapses sont les éléments du cerveux qui permettent la transmission (en parallèle) d’information entre neurones. On compte en moyenne 10000 synapses par neurone.

Schéma d’un neurone biologique

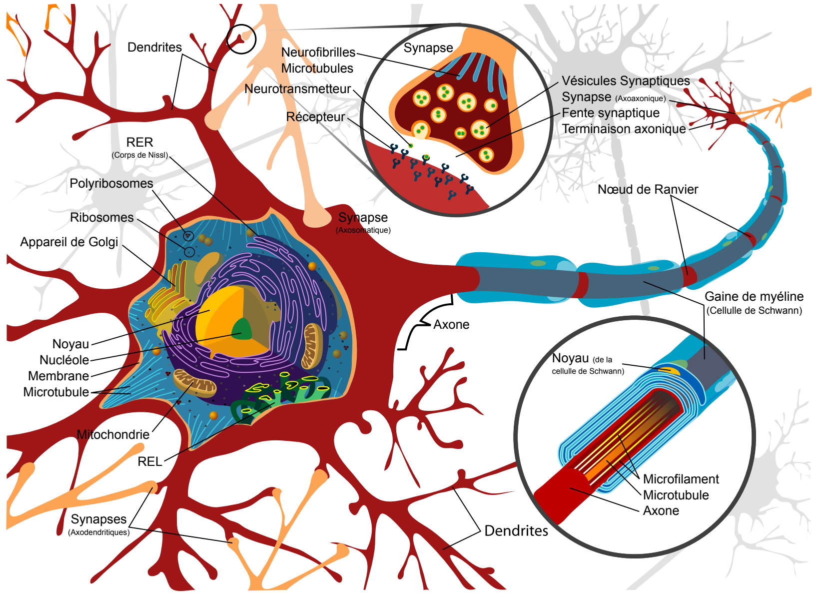
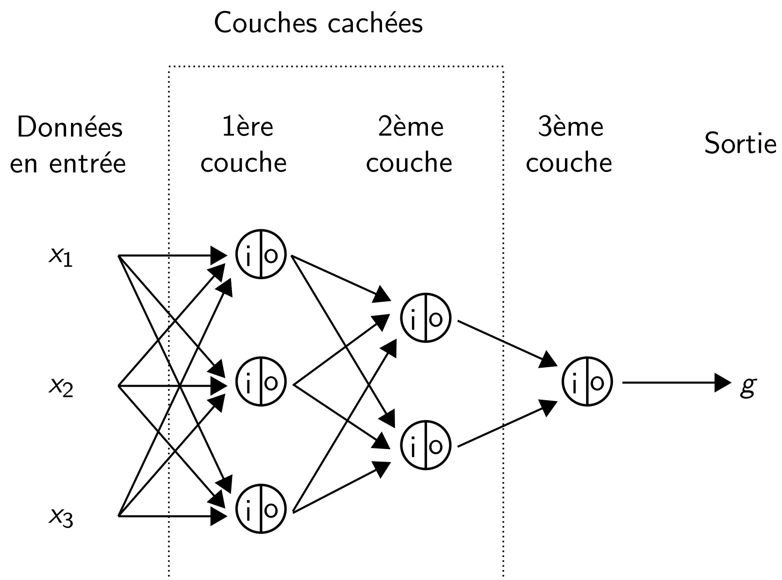


Schéma d’un réseau de neurones artificiel (RN)

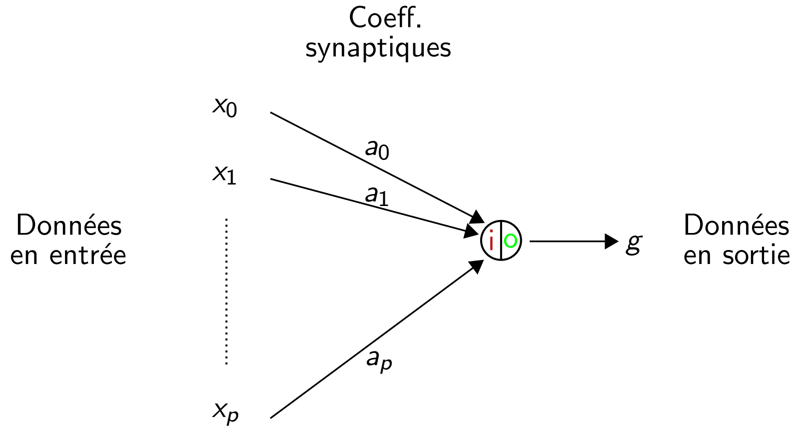


Le perceptron

- Le **perceptron** est l’élément de base d’un RN : il reçoit des données en entrée provenant de signaux internes (d’autres perceptrons) ou externes (données du problème) et donne en sortie un signal qui peut être communiqué en interne (à d’autres perceptrons) ou en externe (prédiction du perceptron).
- Soit un vecteur (signal) $\mathbf{x} = (x_1, \dots, x_p)$ appartenant à $\mathbb{X} \subseteq \mathbb{R}^p$ qui est le signal d’entrée transmis à un percpetron.
- A chaque x_{ij} on lui associe un poids a_j. On a ainsi un vecteur $\mathbf{a} = (a_1, \dots, a_p)$ que l’on appelle les **coefficients synaptiques**.
- On a également une constante a_0 que l’on appele le **biais**.
- Le signal **post-synaptique** est défini de la façon suivante :

$$s(\mathbf{x}) = a_0 + \sum_{i=1}^{p} a_j x_j$$

Le perceptron (suite)



- Dans la suite on notera $\mathbf{x} = (1, x_1, \ldots, x_p)$ et $\mathbf{a} = (a_0, a_1, \ldots, a_p)$ si bien que le signal post-synaptique s'écrit :

$$s(\mathbf{x}) = \mathrm{i} = \mathbf{a}^\top \mathbf{x}$$

Le perceptron (suite)

- Le signal post-synaptique est ainsi une combinaison liénaire des données en entrée : $s(\mathbf{x}) = \mathbf{a}^\top \mathbf{x}$.
- Le neurone qui reçoit ce signal le transforme par le biais d'une **fonction d'activation**, $h(s(\mathbf{x})) = \mathrm{o}$ et qui peut prendre plusieurs formes :
  - La fonction identité :

$$h(s(\mathbf{x})) = h(\mathrm{i}) = \mathrm{o} = \mathbf{a}^\top \mathbf{x}$$

  - Une fonction à seuil comme la fonction d'Heaviside :

$$h(s(\mathbf{x})) = \mathrm{o} = \begin{cases} 1 & \text{si } \mathbf{a}^\top \mathbf{x} > 0 \\ 0 & \text{sinon} \end{cases}$$

  - La fonction sigmoïd (fonction inverse de logit) :

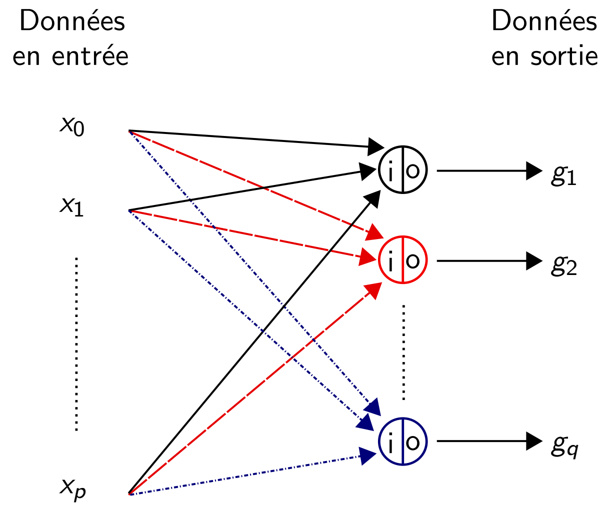
$$h(s(\mathbf{x})) = \mathrm{o} = \frac{1}{1 + \exp(-\mathbf{a}^\top \mathbf{x})}$$

Le perceptron (suite)

- Nous voyons que le perceptron est une forme d'allégorie permettant de retrouver les modèles linéaires vus précédemment :
  - Le perceptron associé à la fonction d'activation identité revient à un modèle linéaire pour le problème de régression.
  - Le perceptron associé à des fonctions à seuil détermine des frontières de décision linéaires (hyperplans) dans l'espace $\mathbb{X}$ ce qui est équivalent à des modèles linéaires en catégorisation telle que l'AFD.
  - Le perceptron associé à la fonction sigmoïd permet d'obtenir en sortie des valeurs normées entre 0 et 1 ce qui permet une interprétation en termes de probabilité pour le problème de catégorisation binaire. On remarquera dans ce cas que si $P(C_1|X) = \frac{1}{1+\exp(-\mathbf{a}^\top \mathbf{x})}$ alors $P(C_2|X) = 1 - P(C_1|X) = \frac{\exp(-\mathbf{a}^\top \mathbf{x})}{1+\exp(-\mathbf{a}^\top \mathbf{x})}$ (rég. log. binomiale).
- Un perceptron permet donc de modéliser le problème de régression et de catégorisation binaire. Dans ce dernier cas, quand est-il du problème multiclasse ?

$\Rightarrow$ On utilise en parallèle q perceptrons.

Problème multiclasse : q perceptrons en parallèle



Problème multiclasse : q perceptrons en parallèle

- Dans ce cas, nous avons q systèmes de coefficients synaptiques ce qui nous conduit à définir la matrice $\mathbf{A}$ de taille $((p+1)\times q)$ dont les colonnes $\mathbf{a}_l$ sont les coefficients synaptiques du $l$ème perceptron.
- Chaque perceptron reçoit un message post-synaptique défini par $s_l(\mathbf{x}) = \mathbf{a}_l^\top \mathbf{x}$ et Chaque perceptron est indépendant des autres.
- Globalement, on obtient une fonction vectorielle $\mathbf{s} : \mathbb{X} \to \mathbb{R}^q$ qui, étant donné un signal en entrée $\mathbf{x}$, calcule les signaux post-synaptiques de chaque perceptron l de la façon suivante :

$$\mathbf{s}(\mathbf{x}) = \mathbf{A}^\top \mathbf{x} \text{ avec } s_l(\mathbf{x}) = \mathbf{a}_l^\top \mathbf{x}$$

- Les neurones appliquent ensuite la fonction d'activation et on note :

$$\mathbf{g}(\mathbf{x}) = h(\mathbf{A}^\top \mathbf{x}) \text{ avec (par abus de notations) : } g_l(\mathbf{x}) = h(\mathbf{a}_l^\top \mathbf{x})$$

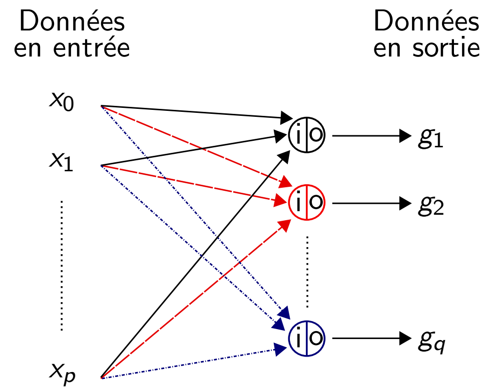
- La **règle de décision** est alors la suivante :

$$f(\mathbf{x}) = C_l \Leftrightarrow \forall l' \neq l \ : g_l(\mathbf{x}) \geq g_{l'}(\mathbf{x})$$

q perceptrons en parallèle et régression logistique

- La régression logistique polytomique est équivalente à un RN avec q perceptrons en parallèle et des fonctions d'activation softmax :

$$g_k(\mathbf{x}) = h(\mathbf{a}_k^\top \mathbf{x}) = \frac{\exp(\mathbf{a}_k^\top \mathbf{x})}{\sum_{l=1}^q \exp(\mathbf{a}_l^\top \mathbf{x})}$$



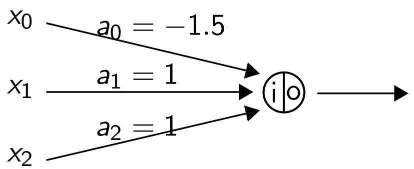
Cas de la fonction booléenne “AND”

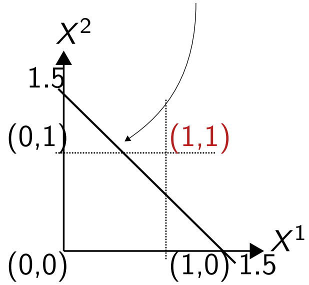
Signal post-synaptique et fonction d'Heaviside :

X^1	X^2	AND
1	1	1
1	0	0
0	1	0
0	0	0

$$a_0 + a_1 X^1 + a_2 X^2 \lesseqgtr 0$$
$$\begin{cases} a_0 + a_1 + a_2 > 0 \\ a_0 + a_1 \leq 0 \\ a_0 + a_2 \leq 0 \\ a_0 \leq 0 \end{cases}$$

$$-1.5 + X^1 + X^2 = 0$$





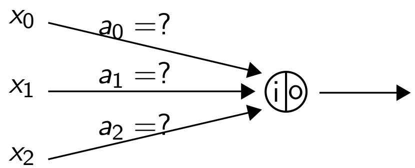
Cas de la fonction booléenne “XOR”

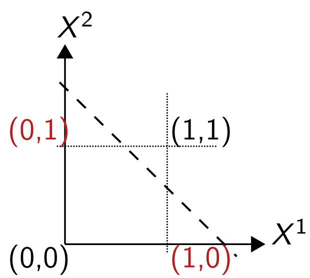
Signal post-synaptique et fonction d'Heaviside :

X^1	X^2	XOR
1	1	0
1	0	1
0	1	1
0	0	0

$$a_0 + a_1 X^1 + a_2 X^2 \lesseqgtr 0$$
$$\begin{cases} a_0 + a_1 + a_2 \leq 0 \\ a_0 + a_1 > 0 \\ a_0 + a_2 > 0 \\ a_0 \leq 0 \end{cases}$$

Ensemble de contraintes **incompatibles**.





Le perceptron multicouche

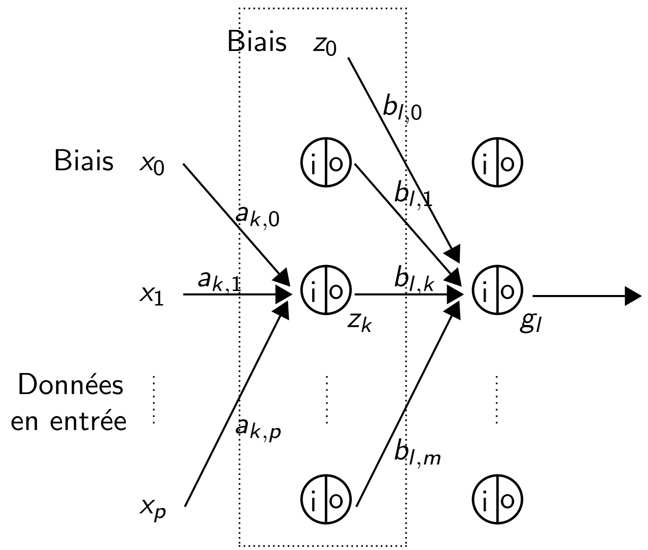
- Les modèles précédents définissent des modèles linéaires avec certaines limites.
- Le perceptron multicouche ("multilayer perceptron") est une généralisation de ces modèles :
  - En régression il permet de traiter les cas **non linéaires** de régression.
  - En classification, il permet de déterminer des fonctions de décision **non linéaire** permettant de résoudre le problème "XOR" précédent par exemple.
- Il consiste en l'ajout de **couches de neurones dites cachées** entre les données en entrée et les données en sortie.
- Dans la suite nous supposerons qu'il y a une seule couche cachée mais d'autres configurations sont possibles.

Le perceptron multicouche (suite)

- Nous supposerons :
  - p données en entrée plus un biais x_0.
  - Une 1ère couche de m perceptrons (la couche cachée).
  - Une 2ème couche de q perceptrons ($q = 1$ dans le cas de la régression).
  - La 1ère couche contient $(p+1) \times m$ coefficients synaptiques.
  - La 2ème couche contient $(m+1) \times q$ coefficients synaptiques.
- Les données en entrée $\mathbf{x}$ sont envoyées à chaque perceptron k de la 1ère couche qui combine linéairement celles-ci en un signal post-synaptique : $s_k(\mathbf{x}) = \mathbf{a}_k^\top \mathbf{x}$.
- Le perceptron k de la 1ère couche applique ensuite la fonction d'activation h_1 : $z_k = h_1(s_k(\mathbf{x}))$.
- Les données $\mathbf{z} = (z_1, \ldots, z_m)$ sont envoyées à chaque perceptron l de la 2ème couche qui combine linéairement celles-ci : $s_l(\mathbf{z}) = \mathbf{b}_l^\top \mathbf{z}$.
- Le perceptron l de la 2ème couche applique enfin la fonction d'activation h_2 pour obtenir les données en sortie : $g_l = h_2(s_l(\mathbf{z}))$.
- Remarque : les fonctions d'activation entre la 1ère et la 2ème couche peuvent être différentes

Le perceptron multicouche (suite)

- Perceptron avec une couche cachée et coefficients synaptiques :



Le perceptron multicouche (suite)

- Fonction d'activation de la couche de sortie (deuxième couche) :
  - Pour la régression on utilise un seul perceptron de sortie et la fonction d'identité :

$$h_2(s_l(\mathbf{z})) = s_l(\mathbf{z}) = \mathbf{b}_l^\top \mathbf{z}$$

  - Pour la catégorisation en $q = 2$ classes on utilise un seul perceptron de sortie et la fonction sigmoïd :

$$h_2(s_l(\mathbf{z})) = \frac{1}{1 + \exp(-\mathbf{b}_l^\top \mathbf{z})}$$

  - Pour la catégorisation en $q > 2$ classes on utilise q perceptrons de sortie et la fonction softmax :

$$h_2(s_l(\mathbf{z})) = \frac{\exp(\mathbf{b}_l^\top \mathbf{z})}{\sum_{l=1}^{q} \exp(\mathbf{b}_l^\top \mathbf{z})}$$

- Pour des problèmes de catégorisation, la variable en sortie g_l peut être interprétée telle la probabilité d'appartenir à C_l.

Le perceptron multicouche (suite)

- Le perceptron à deux couches peut être vu comme un modèle comportant une phase de projection **non linéaire**. En effet, la 1ère couche produit une projection des données de $\mathbb{X}$ de dimension $p+1$ vers un nouvel espace de dimension $m+1$. Ensuite la 2ème couche peut être vue comme q perceptrons en parallèle mais prenons en entrée non pas les données initiales mais les données projetées dans une espace intermédiaire (et caché).
- En statistique, des modèles similaires appelés “**projection pursuit**” avaient été proposés avant les réseaux de neuronnes [Hastie et al., 2011].
- Le perceptron multicouche est un **approximateur universel**. Toute proposition logique peut être représentée par une disjonction de conjonctions. Le perceptron à 2 couches peut approximer toute proposition logique : chaque conjonction est représentée par un perceptron de 1ère couche cachée et la disjonction est représentée par la 2ème couche.

Résolution du cas de la fonction booléenne “XOR”

- Définition logique de l'opération binaire “XOR”

X^1	X^2	XOR
1	1	0
1	0	1
0	1	1
0	0	0

- Contraintes linéaires incompatibles (dans l'espace initial) :

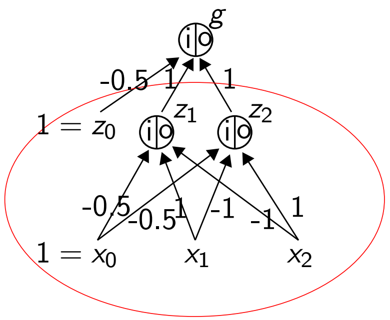
$$\begin{cases} a_0 \leq 0 \\ a_0 + a_1 + a_2 \leq 0 \\ a_0 + a_1 > 0 \\ a_0 + a_2 > 0 \end{cases}$$

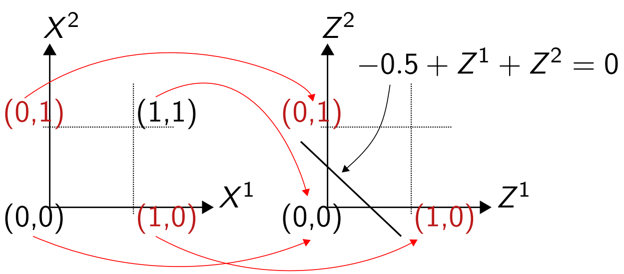
- Ecriture sous la forme normale disjonctive :

$$X^1 \text{ XOR } X^2 \Leftrightarrow (X^1 \text{ AND } \neg X^2) \text{ OR } (\neg X^1 \text{ AND } X^2)$$

- Pour résoudre le problème avec un perceptron à 2 couches on prend $m = 2$ et h la fonction d'Heaviside.

Résolution du cas de la fonction booléenne “XOR” (suite)





Le perceptron multicouche (suite)

- On aborde maintenant le problème de l'inférence des paramètres du modèle. L'ensemble des paramètres est constitué de coefficients synaptiques de chaque perceptron des deux couches $\mathbb{P} = \{\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathbf{b}_1, \ldots, \mathbf{b}_q\}$ où :
  - $\forall k = 1, \ldots, m : \mathbf{a}_k$ est de taille $(p+1) \times 1$
  - $\forall l = 1, \ldots, q : \mathbf{b}_l$ est de taille $(m+1) \times 1$
- Le vecteur $\mathbf{z}$ obtenu à l'issue de la 1ère couche est de terme général :

$$z_k = h_1(\mathbf{a}_k^\top \mathbf{x}) = h_1(\langle \mathbf{a}_k, \mathbf{x} \rangle)$$

- Le vecteur $\mathbf{g}$ obtenu à l'issue de la 2ème couche est de terme général :

$$g_l = h_2(\mathbf{b}_l^\top \mathbf{z}) = h_2(\langle \mathbf{b}_l, \mathbf{z} \rangle)$$

- Ainsi la fonction de décision vectorielle $\mathbf{g}(\mathbf{x})$ donnée en sortie du perceptron à 2 couches est de terme général :

$$g_l(\mathbf{x}) = h_2(\underbrace{\mathbf{b}_l^\top \mathbf{z}}_{s_l(\mathbf{z})}) = h_2(\langle \underbrace{(b_{l,0}, \ldots, b_{l,m})}_{\mathbf{b}_l}, \underbrace{(z_0, h_1(\langle \mathbf{a}_1, \mathbf{x} \rangle), \ldots, h_1(\langle \mathbf{a}_m, \mathbf{x} \rangle))}_{\mathbf{z}} \rangle)$$

Le perceptron multicouche (suite)

- La fonction objectif est de deux types selon le problème traité :
  - Pour la **régression**, on utilise la somme des carrés des résidus :

$$err(g) = scr(g) = \sum_{i=1}^{n} \underbrace{(y_i - g(\mathbf{x}_i))^2}_{err_i} \text{ où } \forall i : y_i \in \mathbb{R}$$

  - Pour la **catégorisation**, on utilise la cross-entropie définie par :

$$err(\mathbf{g}) = ce(\mathbf{g}) = \sum_{i=1}^{n} \underbrace{\sum_{l=1}^{q} -y_{il} \log(g_l(\mathbf{x}_i))}_{err_i}$$

  où $\forall i, l : y_{il} = 1$ si $y_i = C_l$ et $y_{il} = 0$ sinon.
- Il s'agit de problèmes de minimisation non contraints mais non convexe. Chercher un minimiseur pose parfois des problèmes de sur-apprentissage. Dans ce cas, on ajoute un terme de pénalité ou on arrête la recherche du minimiseur prématurément.

Le perceptron multicouche (suite)

- Pour la recherche du minimiseur on utilise une descente de gradient. En raison de la structure multi-niveaux du perceptron multicouche, on a une méthode particulière dite de **rétro-propagation de l'erreur**.
- Considérons err_i l'erreur partielle relative ou locale à l'élément $\mathbf{x}_i$.
- L'erreur totale notée err vaut donc :

$$err(g) = \sum_{i=1}^{n} err_i$$

- Rappel : la **descente de gradient** consiste à chercher un minimiseur de façon itérative en suivant à chaque étape la direction opposée du gradient de sorte à déterminer un point critique (CNPO).
- Dans ce qui suit nous étudions l'algorithme de rétro-propagation de l'erreur pour le perceptron multicouche avec une couche cachée tel que décrit schématiquement au slide 123.

Le perceptron multicouche (suite)

- On définit les gradients relatifs à l'objet $\mathbf{x}_i$ suivants :

$$\nabla_{\mathbf{b}_l} err_i(\mathbb{P}) = \begin{pmatrix} \frac{\partial}{\partial b_{l,0}} err_i \\ \vdots \\ \frac{\partial}{\partial b_{l,m}} err_i \end{pmatrix} \quad ; \quad \nabla_{\mathbf{a}_k} err_i(\mathbb{P}) = \begin{pmatrix} \frac{\partial}{\partial a_{k,0}} err_i \\ \vdots \\ \frac{\partial}{\partial a_{k,p}} err_i \end{pmatrix}$$

- Les formules itératives de **descente de gradient** sont données par :

$$\mathbf{b}_l^{(r+1)} = \mathbf{b}_l^{(r)} - \alpha_r \sum_{i=1}^{n} \nabla_{\mathbf{b}_l} err_i$$

$$\mathbf{a}_k^{(r+1)} = \mathbf{a}_k^{(r)} - \alpha_r \sum_{i=1}^{n} \nabla_{\mathbf{a}_k} err_i$$

  où l'exposant (r) indique l'itération r de l'algorithme.
- La structure multi-niveaux du perceptron multicouche permet un **calcul efficace et local des gradients**.

Le perceptron multicouche (suite)

- La rétro-propagation repose sur les règles de dérivation suivantes dites "chaining rule" :

$$\frac{\partial err_i}{\partial b_{l,k}} = \frac{\partial err_i}{\partial g_l}\frac{\partial g_l}{\partial b_{l,k}} \text{ et } \frac{\partial err_i}{\partial a_{k,j}} = \sum_{l=1}^{q} \frac{\partial err_i}{\partial g_l}\frac{\partial g_l}{\partial z_k}\frac{\partial z_k}{\partial a_{k,j}}$$

- Intuitivement, pour connaître comment varie err_i quand $a_{k,j}$ varie, on regarde d'abord comment varie z_k quand $a_{k,j}$ varie ; puis comment varie g_l quand z_k varie et enfin on peut voir comment varie err_i quand g_l varie.
- Inversement, en lisant la formule de la gauche vers la droite, elle traduit d'une certaine façon, comment err_i se propage dans le réseau de la 2ème couche vers la 1ère d'où le terme de rétro-propagation.
- Comme les couches se succèdent, ces calculs de dérivées peuvent se faire de la même façon d'une couche à une autre.

Le perceptron multicouche (suite)

- Les étapes de l'algorithme de retro-propagation sont les suivantes et sont typiquement basées sur deux étapes dites "forward" et "backward" qui s'alternent de façon itérative :

1. On initialise aléatoirement des coefficients synaptiques $\mathbf{a}_k$ et $\mathbf{b}_l$.
2. A l'itération r on utilise les valeurs courantes $\mathbf{a}_k^{(r)}$ et $\mathbf{b}_l^{(r)}$ et on calcule les prédictions $\mathbf{g}^{(r)}(\mathbf{x})$ (étape "forward").
3. On calcule les erreurs au niveau de la sortie et celles-ci sont rétro-propagées vers les perceptrons de la 2ème puis de la 1ère couche et on ajuste ainsi les coefficients $\mathbf{b}_l^{(r+1)}$ et $\mathbf{a}_k^{(r+1)}$ en utilisant la descente de gradient (étape "backward").
4. On itère 2 et 3 jusqu'à convergence ou jusqu'à la vérification d'un critère d'arrêt.

Le perceptron multicouche : cas de la régression non linéaire

- Cas de la régression non linéaire :
  - Un seul neurone dans la 2ème couche ie $q = 1$
  - Fonction objectif :

$$err(g) = \sum_{i=1}^{n} \underbrace{(y_i - g(\mathbf{x}_i))^2}_{err_i}$$

  - Fonction d'activation de la 1ère couche, la fonction sigmoïd :

$$h_1(s_k(\mathbf{x}_i)) = 1/(1 + \exp(-\mathbf{a}_k^\top \mathbf{x}_i)) = z_k$$

  - Remarque : on a la propriété suivante,

$$\frac{\partial \text{sigmoïd}(x)}{\partial x} = \text{sigmoïd}(x)(1 - \text{sigmoïd}(x))$$

  - Fonction d'activation de la 2ème couche, la fonction identité :

$$h_2(s(\mathbf{z})) = \mathbf{b}^\top \mathbf{z} = g$$

Le perceptron multicouche : cas de la régression non linéaire

- Erreur associée à l'objet X_i :

$$err_i = (y_i - \mathbf{b}^\top \mathbf{z})^2$$

- Dérivées partielles par rapport à $\mathbf{b}$ (2ème couche) :

$$\frac{\partial err_i}{\partial b_k} = \underbrace{-2(y_i - \mathbf{b}^\top \mathbf{z})}_{\partial err_i/\partial g} \underbrace{z_k}_{\partial g/\partial b_k}$$

- Dérivées partielles par rapport à $\mathbf{a}_k$ (1ère couche) :

$$\frac{\partial err_i}{\partial a_{k,j}} = \underbrace{-2(y_i - \mathbf{b}^\top \mathbf{z})}_{\partial err_i/\partial g} \underbrace{b_k}_{\partial g/\partial z_k} \underbrace{z_k(1 - z_k)}_{\partial z_k/\partial s_k} \underbrace{x_j}_{\partial s_k/\partial a_{k,j}}$$

Pseudo-code (batch) de rétro-propagation

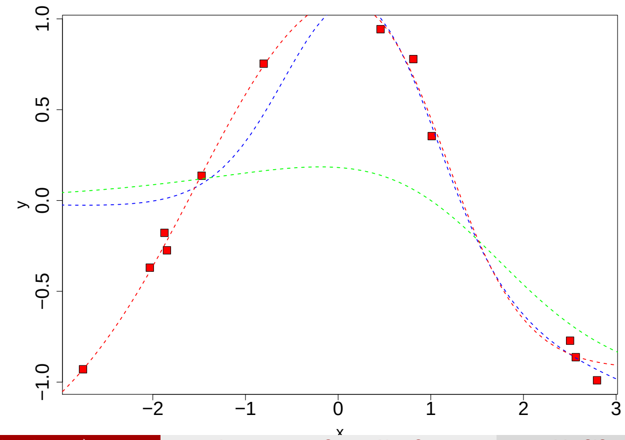
Input : $\mathbb{E}, \alpha_r$
Initialiser $a_{k,j}$ et b_k aléatoirement par des valeurs dans $[-0.01, 0.01]$
Tant que condition d'arrêt non satisfaite **faire**
    $r \leftarrow 1$
    Pour tout $i = 1, \ldots, n$ **faire**
        Pour tout $k = 1, \ldots, m$ **faire**
            $z_k \leftarrow 1/(1 + \exp(-\mathbf{a}_k^\top \mathbf{x}_i))$
        Fin Pour
        $\hat{y}_i \leftarrow \mathbf{b}^\top \mathbf{z}$
    Fin Pour
    $\mathbf{b}^{(r+1)} \leftarrow \mathbf{b}^{(r)} - \alpha_r \sum_{i=1}^n \nabla_{\mathbf{b}} err_i$
    Pour tout $k = 1, \ldots, m$ **faire**
        $\mathbf{a}_k^{(r+1)} = \mathbf{a}_k^{(r)} - \alpha_r \sum_{i=1}^n \nabla_{\mathbf{a}_k} err_i$
    Fin Pour
    $r \leftarrow r + 1$
Fin Tant que
Output : $\mathbf{b}^{(r)}, \{\mathbf{a}_k^{(r)}\}_{k=1}^m$

Pseudo-code (on-line) de rétro-propagation

Input : $\mathbb{E}, \alpha_r$
Initialiser $a_{k,j}$ et b_k aléatoirement par des valeurs dans $[-0.01, 0.01]$
Tant que condition d'arrêt non satisfaite **faire**
  $L \leftarrow sample(n)$ (permutation aléatoire de $1, 2, \ldots, n$)
  $r \leftarrow 1$
  Pour tout $i \in L$ **faire**
    Pour tout $k = 1, \ldots, m$ **faire**
      $z_k \leftarrow 1/(1 + \exp(-\mathbf{a}_k^\top \mathbf{x}_i))$
    Fin Pour
    $\hat{y}_i \leftarrow \mathbf{b}^\top \mathbf{z}$
    $\mathbf{b}^{(r+1)} \leftarrow \mathbf{b}^{(r)} - \alpha_r \nabla_{\mathbf{b}} err_i$
    Pour tout $k = 1, \ldots, m$ **faire**
      $\mathbf{a}_k^{(r+1)} = \mathbf{a}_k^{(r)} - \alpha_r \nabla_{\mathbf{a}_k} err_i$
    Fin Pour
    $r \leftarrow r + 1$
  Fin Pour
Fin Tant que
Output : $\mathbf{b}^{(r)}, \{\mathbf{a}_k^{(r)}\}_{k=1}^m$

Perceptron multicouche : cas de la régression non linéaire (suite)

- Résultats (version on-line) avec $m = 3$ après 100, 200 et 400 itérations



Le perceptron multicouche : cas de la catégorisation

- Cas de la catégorisation avec $q = 3$:
  - Trois neurones dans la 2ème couche ie $q = 3$.
  - Fonction objectif :

$$err(\mathbf{g}) = \sum_{i=1}^{n} \sum_{l=1}^{q} -y_{il} \log(g_l(\mathbf{x}_i))$$

  - Fonction d'activation de la 1ère couche, la fonction sigmoïd :

$$h_1(s_k(\mathbf{x})) = \frac{1}{1 + \exp(-\mathbf{a}_k^\top \mathbf{x})} = z_k$$

  - Fonction d'activation de la 2ème couche, la fonction softmax :

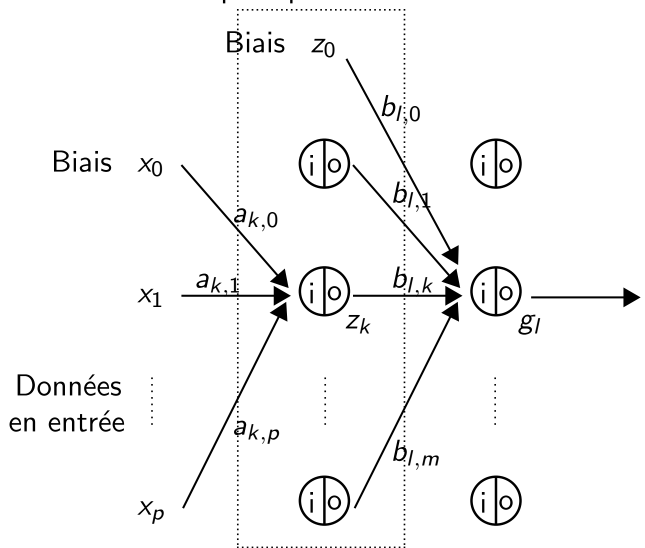
$$h_2(s_l(\mathbf{z})) = \frac{\exp(\mathbf{b}_l^\top \mathbf{z})}{\sum_{l=1}^{q} \exp(\mathbf{b}_l^\top \mathbf{z})} = g_l$$

  - Remarque : on a aussi la propriété suivante,

$$\frac{\partial \text{softmax}(x)}{\partial x} = \text{softmax}(x)(1 - \text{softmax}(x))$$

Le perceptron multicouche : cas de la catégorisation (suite)

- Rappelons le schéma d'un perceptron multicouche :



Le perceptron multicouche : cas de la catégorisation (suite)

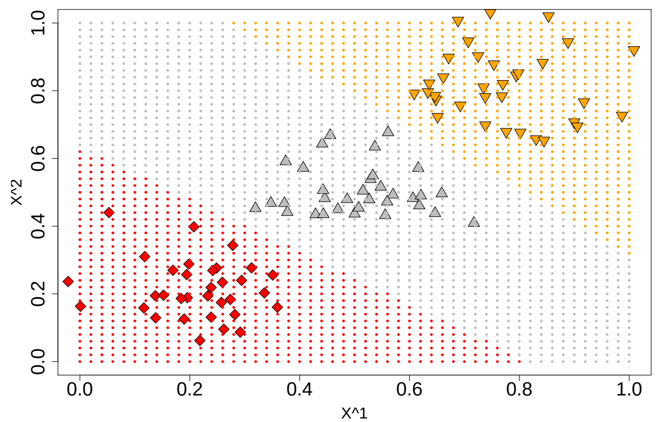
- Erreur associée à l'objet X_i :
$$err_i = \sum_{l=1}^{q} -y_{il} \log\left(\frac{\exp(\mathbf{b}_l^\top \mathbf{z})}{(\sum_{l=1}^{q} \exp(\mathbf{b}_l^\top \mathbf{z}))}\right)$$
- Dérivées partielles par rapport à $\mathbf{b}$ (2ème couche) :
$$\frac{\partial err_i}{\partial b_{l,k}} = \sum_{l'=1}^{q} \frac{\partial err_i}{\partial g_{l'}} \frac{\partial g_{l'}}{\partial b_{l,k}} = \sum_{l'=1}^{q} \frac{\partial err_i}{\partial g_{l'}} \frac{\partial g_{l'}}{\partial s_l} \frac{\partial s_l}{\partial b_{l,k}}$$
- En explicitant chaque calcul on obtient les résultats partiels suivants :
  - $\partial err_i/\partial g_{l'} = -y_{il'}/g_{l'}$
  - $\partial g_{l'}/\partial s_l = g_{l'} ind(l = l') - (g_{l'})^2$ (ind étant la fonction indicatrice)
  - $\partial s_l/\partial b_{l,k} = z_k$
- Puis en regroupant le tout, on obtient finalement :
$$\frac{\partial err_i}{\partial b_{l,k}} = \underbrace{-(y_{il} - g_l)}_{\sum_{l'}(\partial err_i/\partial g_{l'})(\partial g_{l'}/s_l)} \underbrace{z_k}_{\partial s_l/\partial b_{l,k}}$$

Le perceptron multicouche : cas de la catégorisation (suite)

- Rappelons que : $z_k = h_1(s_k(\mathbf{x})) = \text{sigmoïd}(s_k(\mathbf{x})) = \frac{1}{1+\exp(-\mathbf{a}_k^\top \mathbf{x})}$.
- Dérivées partielles par rapport à $\mathbf{a}_k$ (1ère couche) :
$$\frac{\partial err_i}{\partial a_{k,j}} = \sum_{l'=1}^{q} \frac{\partial err_i}{\partial g_{l'}} \sum_{l''=1}^{q} \frac{\partial g_{l'}}{\partial s_{l''}} \frac{\partial s_{l''}}{\partial z_k} \frac{\partial z_k}{\partial s_k} \frac{\partial s_k}{\partial a_{k,i}}$$
- On obtient le résultat suivant :
$$\begin{aligned}\frac{\partial err_i}{\partial a_{k,j}} &= \sum_{l''=1}^{q} \underbrace{\left(\sum_{l'=1}^{q} \frac{\partial err_i}{\partial g_{l'}} \frac{\partial g_{l'}}{\partial s_{l''}}\right)}_{-(y_{il''}-g_{l''})} \frac{\partial s_{l''}}{\partial z_k} \frac{\partial z_k}{\partial s_k} \frac{\partial s_k}{\partial a_{k,i}} \\ &= \sum_{l''=1}^{q} \underbrace{-(y_{il''} - g_{l''})}_{\sum_{l'}(\partial err_i/\partial g_{l'})(\partial g_{l'}/\partial s_{l''})} \underbrace{b_{l'',k}}_{\partial s_{l''}/\partial z_k} \underbrace{z_k(1-z_k)}_{\partial z_k/\partial s_k} \underbrace{x_j}_{\partial s_k/\partial a_{k,j}}\end{aligned}$$

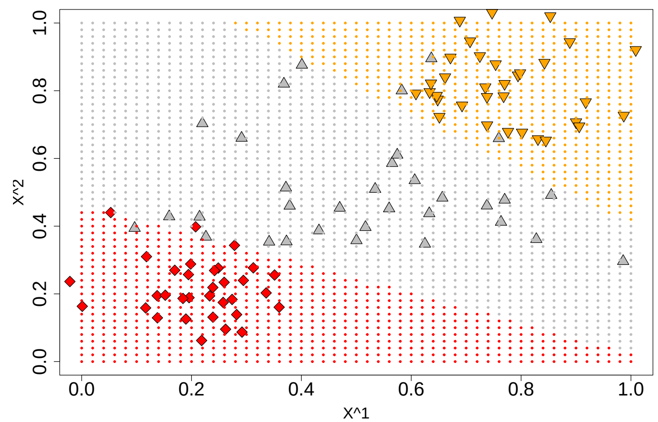
Le perceptron multicouche : cas de la catégorisation (suite)

- Résultats (version on-line) avec $m = 5$ et $q = 3$ après 100 itérations.



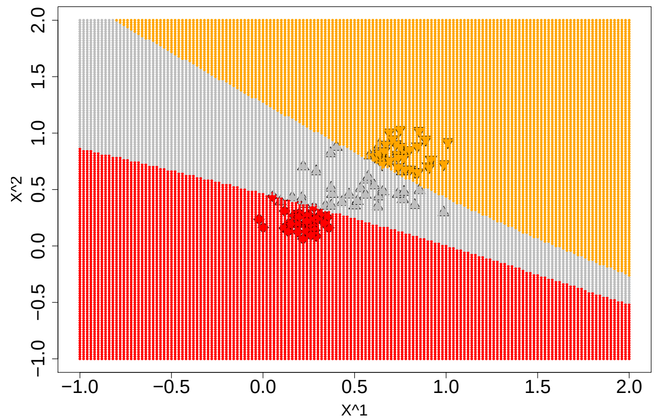
Le perceptron multicouche : cas de la catégorisation (suite)

- Résultats (version on-line) avec $m = 5$ et $q = 3$ après 100 itérations.



Le perceptron multicouche : cas de la catégorisation (suite)

- Résultats (version on-line) avec $m = 5$ et $q = 3$ après 100 itérations.



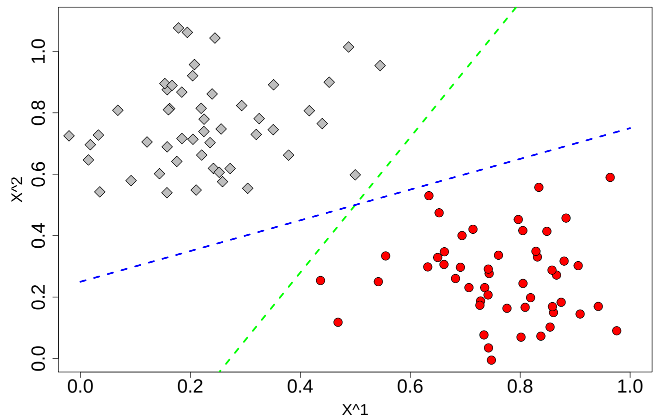
Rappel du Sommaire

Introduction

- C'est une famille de méthodes “récentes” proposée initialement dans [Vapnik, 1995] dans les années 90.
- Nous étudierons dans un premier temps l'application de cette méthode pour le problème de catégorisation puis nous verrons comment elle permet également de traiter les problèmes de régression.
- C'est une **méthode discriminante** mais qui estime directement la frontière de décision entre deux catégories (ce qui est distinct des fonctions discriminantes et de la modélisation probabiliste $P(Y|X)$).
- Cette **frontière peut-être définie par des objets de** $\mathbb{E}$ et non nécessairement par les variables $\mathbb{A}$.
- La méthode repose sur la matrice de Gram càd la matrice des produits scalaires entre objets de $\mathbb{E}$ (et non nécessairement sur la représentation vectorielle).
- La méthode cherche à résoudre un problème d'**optimisation convexe** et il existe donc une **solution unique**.

Hyperplans de séparation entre deux classes

- On suppose un problème avec deux catégories C_1 et C_2.
- Il existe une infinité d'hyperplans permettant de séparer deux nuages de points linéairement séparable.



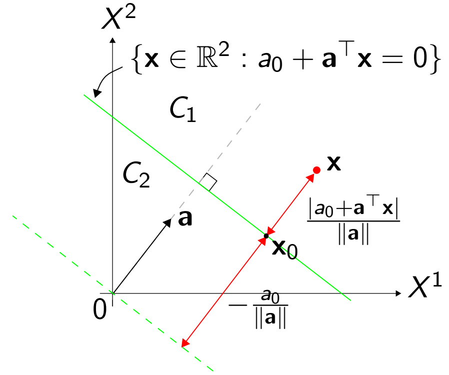
Hyperplans de séparation optimale entre deux classes

- Dans le cas des svm, on cherche la frontière linéaire représentée par $a_0 \in \mathbb{R}$ et $\mathbf{a} \in \mathbb{R}^p$ telle que :

$$\begin{cases} a_0 + \mathbf{a}^\top \mathbf{x} \geq \delta & \text{pour tout } \mathbf{x} \in C_1 \\ a_0 + \mathbf{a}^\top \mathbf{x} \leq -\delta & \text{pour tout } \mathbf{x} \in C_2 \end{cases}$$

  avec $\delta \geq 0$.
- Contrairement aux fonctions discriminantes où on regardait uniquement le signe par rapport à la frontière ($g(\mathbf{x}) \lessgtr 0$), on veut aussi une distance δ par rapport à la frontière.
- On appelle la **marge**, la distance entre la frontière et les objets $\mathbf{x}$ les plus proches de celle-ci.
- L'apprentissage consiste alors à déterminer l'hyperplan permettant de **maximiser la marge** (on traduit parfois svm par "Séparateur à Vaste Marge") afin d'obtenir une meilleure généralisation.

Optimisation de la marge



- Dans $\mathbb{R}^p$, le vecteur normal de la frontière est $\mathbf{a}$.
- La distance (signée) entre la frontière et l'origine est $-a_0/\|\mathbf{a}\|$.
- Soit $\mathbf{x}_0$ un point de la frontière, la distance entre $\mathbf{x}$ et la frontière est :

$$\frac{|\mathbf{a}^\top(\mathbf{x} - \mathbf{x}_0)|}{\|\mathbf{a}\|} = \frac{|\mathbf{a}^\top \mathbf{x} + a_0|}{\|\mathbf{a}\|}$$

Optimisation de la marge (suite)

- On a alors le problème suivant :

$$\max_{a_0, \mathbf{a} \in \mathbb{R}^p} \delta$$
$$slc \quad \forall i, y_i \frac{(\mathbf{a}^\top \mathbf{x}_i + a_0)}{\|\mathbf{a}\|} \geq \delta$$

  où $y_i = 1$ si $\mathbf{x}_i \in C_1$ et $y_i = -1$ si $\mathbf{x}_i \in C_2$.
- Dans les contraintes, on peut écrire de façon équivalente :

$$y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq \delta \|\mathbf{a}\|$$

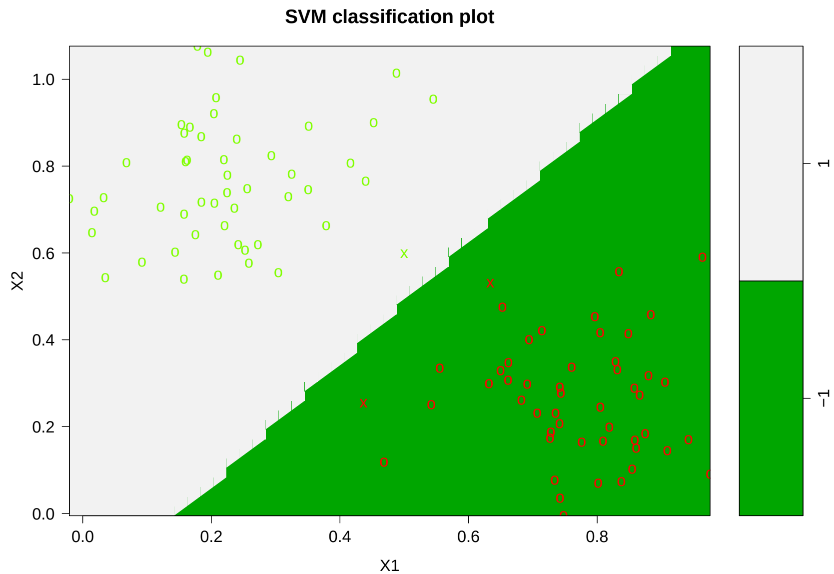
- Puis sans perte de généralité on peut poser :

$$\delta = \frac{1}{\|\mathbf{a}\|}$$

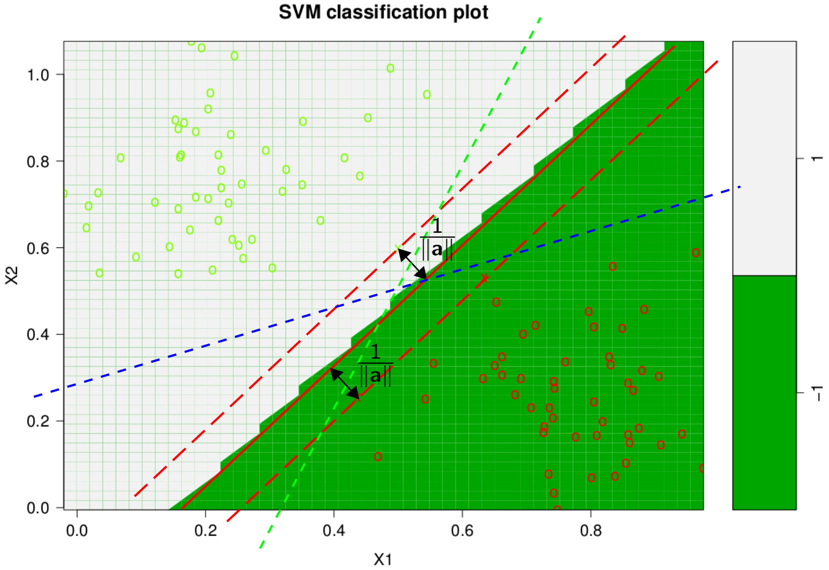
- Le problème devient alors :

$$\min_{a_0, \mathbf{a} \in \mathbb{R}^p} \frac{1}{2}\|a\|^2$$
$$slc \quad \forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1$$

Exemple



Exemple (suite)



Problème quadratique contraint

- La marge $\delta = 1/\|\mathbf{a}\|$ donc $2/\|\mathbf{a}\|$ est l'épaisseur de la bande (ou tube).
- Il n'y a uniquement que quelques points (ceux marqués d'une croix dans l'exemple précédent) qui participent à la définition de la frontière (cf plus loin).

- Pour **maximiser la marge** cela revient donc à **minimiser la norme** euclidienne au carré du vecteur normal **a** de la frontière. Il s'agit d'un **problème quadratique avec des contraintes d'inégalités linéaires** (de type $\geq$). Il s'agit donc d'un **problème convexe** que l'on peut résoudre en utilisant des solvers où en appliquant des méthodes d'optimisation numériques dédiées à ce problème.

- Toutefois, on peut reformuler de façon équivalente ce problème en écrivant le **Lagrangien associé** et en formant ainsi le **dual**.

Lagrangien et problème dual

- Reprenons le problème primal suivant :

$$\begin{array}{ll} & \min_{a_0,\mathbf{a}\in\mathbb{R}^p} \frac{1}{2}\|a\|^2 \\ slc & \forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1 \end{array}$$

- Le **Lagrangien (primal)** est noté $lag_p(a_0, \mathbf{a}, \boldsymbol{\alpha})$ où $\boldsymbol{\alpha}$ est le vecteur de taille $(n \times 1)$ des **multiplicateurs de Lagrange**. Il est défini par :

$$lag_p(a_0, \mathbf{a}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{a}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) - 1 \right)$$

- Le Lagrangien doit être minimisé selon a_0 et **a** et maximiser par rapport à $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$.
- Par conséquent les CNPO impliquent que la solution se trouve en un point selle.

Lagrangien et problème dual (suite)

- Le problème étant convexe, il est équivalent de résoudre le dual qui consiste à maximiser le Lagrangien lag_d par rapport à $\boldsymbol{\alpha}$ sous les contraintes que les gradients de lag_p par rapport à a_0 et **a** soient nuls :

$$\left\{ \begin{array}{lll} \frac{\partial lag_p}{\partial a_0} & = & 0 \\ \frac{\partial lag_p}{\partial \mathbf{a}} & = & \mathbf{0} \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lll} \sum_{i=1}^n \alpha_i y_i & = & 0 \\ \mathbf{a} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i & = & \mathbf{0} \end{array} \right.$$

- On obtient les relations suivantes $\sum_{i=1}^n \alpha_i y_i = 0$ et $\mathbf{a} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$
- En intégrant ces relations au sein du Lagrangien lag_p on obtient :

$$\begin{aligned} lag_p(a_0, \mathbf{a}, \boldsymbol{\alpha}) &= \frac{1}{2}\|\mathbf{a}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) - 1 \right) \\ &= \ldots \\ &= \underbrace{\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j}_{lag_d(\boldsymbol{\alpha})} \end{aligned}$$

Lagrangien et problème dual (suite)

- Le **problème dual** est alors le suivant :

$$\begin{aligned} &\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ slc \quad &\forall i, \alpha_i \geq 0 \\ &\sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

- En plus de la contrainte sur les multiplicateurs de Lagrange, la solution optimale du dual doit également satisfaire les **conditions de Karush-Kuhn-Tucker (KKT)** suivantes :

$$\forall i, \alpha_i \left(y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) - 1 \right) = 0$$

- Ces conditions s'interprètent de la façon suivante :
  - Si $\alpha_i > 0$ alors la contrainte est saturée càd $y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) = 1$ et $\mathbf{x}_i$ se situe sur une frontière de la bande.
  - Si $y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) > 1$ alors $\alpha_i = 0$ et dans ce cas $\mathbf{x}_i$ se situe hors de la bande.

Interprétation du svm

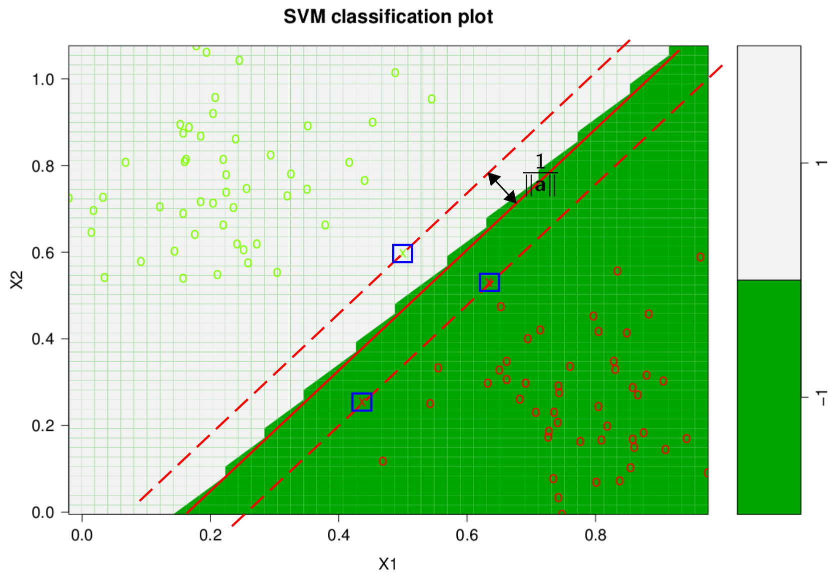
- Rappelons que nous avons $\hat{\mathbf{a}}_{svm} = \sum_{i=1}^n \hat{\alpha}_i y_i \mathbf{x}_i$.
- De plus, seuls les $\mathbf{x}_i$ sur les frontières de la bande sont tels que $\hat{\alpha}_i > 0$. On les appelle les **vecteurs supports**.
- En d'autres termes, $\hat{\mathbf{a}}_{svm}$ est défini comme une combinaison linéaire des vecteurs supports.
- Les objets $\mathbf{x}_i$ tel que $\hat{\alpha}_i = 0$ sont des points hors de la bande et ne sont pas intéressants pour définir la frontière entre les deux classes (ils sont relativement loins de la frontière).
- On obtient $\hat{a}_{svm,0}$ à l'ade de l'équation suivante pour n'importe quel vecteur support (càd tel que $\alpha_i > 0$) :

$$\hat{a}_{svm,0} = y_i - \hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i$$

- La fonction de décision $\hat{f}(\mathbf{x})$ dépend de $\hat{g}(\mathbf{x}) = \hat{\mathbf{a}}_{svm}^\top \mathbf{x} + \hat{a}_{svm,0}$:

$$\hat{f}(\mathbf{x}) \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) > 0 \\ C_2 & \text{sinon} \end{cases}$$

Interprétation du svm (suite)



Le cas non linéairement séparable

- Nous avons traité précédemment le cas linéairement séparable.
- Si dans l'espace de description initial $\mathbb{X}$, les **classes se recouvrent** alors elles ne sont pas linéairement séparables et **le problème d'optimisation n'a pas de solution**.
- En effet, il est alors impossible de satisfaire toutes les contraintes :

$$\forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1$$

- On cherche alors un hyperplan qui continue à maximiser la marge mais tout en faisant le moins d'erreur possible.
- Pour ce faire, on intègre des **variables d'écart** $\xi_i \geq 0$ qui permettent des erreurs :

$$\forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1 - \xi_i$$

- On parle alors de "**soft margin**" ou de **méthodes discriminantes flexibles**.

Le cas non linéairement séparable (suite)

$$\forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1 - \xi_i$$

- Nous remarquerons les cas particuliers suivants :
  - Si $\xi_i = 0$ alors il n'y a pas de problème de catégorisation avec $\mathbf{x}_i$.
  - Si $0 < \xi_i < 1$ alors $\mathbf{x}_i$ est du bon côté de la frontière mais se situe dans la bande.
  - Si $\xi_i \geq 1$ alors $\mathbf{x}_i$ est catégorisée de façon incorrecte.
- $|\{\mathbf{x}_i \in \mathbb{E} : \xi_i > 1\}|$ est le nb de vecteurs incorrectement classifiés.
- $|\{\mathbf{x}_i \in \mathbb{E} : \xi_i > 0\}|$ est le nb de vecteurs non linéairement séparables.
- On définit alors le "**soft error**" :

$$\sum_i \xi_i$$

- Celui-ci est ajouté dans la fonction objectif comme terme de **pénalité**.

Hyperplan flexible de séparation optimale

- Nous avons le problème suivant :

$$\begin{array}{rl} & \min_{a_0, \mathbf{a} \in \mathbb{R}^p, \boldsymbol{\xi} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{a}\|^2 + c\sum_{i=1}^n \xi_i \\ slc & \forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1 - \xi_i \\ & \forall i, \xi_i \geq 0 \end{array}$$

  où c est une constante positive tel un coefficient de pénalité, permettant de contrôler l'équilibre entre la maximisation de la marge et les erreurs. Nous remarquerons que pour un cas linéairement séparable les ξ_i sont nuls et donc "$c = \infty$".
- Le Lagrangien (primal) est alors donné par :

$$lag_p(a_0, \mathbf{a}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu})$$
$$=$$
$$\frac{1}{2}\|\mathbf{a}\|^2 + c\sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i \left(y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) - (1 - \xi_i)\right) - \sum_{i=1}^n \mu_i \xi_i$$

  où $\boldsymbol{\alpha} \in \mathbb{R}^{+n}$ et $\boldsymbol{\mu} \in \mathbb{R}^{+n}$ sont les multiplicateurs de Lagrange.

Lagrangien et problème dual

- On doit minimiser le Lagrangien par rapport à a_0, $\mathbf{a}$, $\boldsymbol{\xi}$ et le maximiser par rapport à $\boldsymbol{\alpha}$ et $\boldsymbol{\mu}$.
- Comme précédemment, on peut de façon équivalente maximiser le Lagrangien par rapport à $\boldsymbol{\alpha}$ et $\boldsymbol{\mu}$ sous les contraintes que les gradients de lag_p par rapport aux variables primales soient nuls :

$$\left\{\begin{array}{lcl} \frac{\partial lag_p}{\partial a_0} & = & 0 \\ \frac{\partial lag_p}{\partial \mathbf{a}} & = & \mathbf{0} \\ \frac{\partial lag_p}{\partial \boldsymbol{\xi}} & = & \mathbf{0} \end{array}\right. \Leftrightarrow \left\{\begin{array}{lcl} \sum_{i=1}^n \alpha_i y_i & = & 0 \\ \mathbf{a} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i & = & \mathbf{0} \\ c\mathbf{1} - \boldsymbol{\alpha} - \boldsymbol{\mu} & = & \mathbf{0} \end{array}\right.$$

  où $\mathbf{1}$ est le vecteur de taille $(n \times 1)$ rempli de 1.
- On obtient les relations suivantes $\sum_{i=1}^n \alpha_i y_i = 0$, $\mathbf{a} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ et $\forall i, \alpha_i = c - \mu_i$.
- Comme $\forall i, \mu_i \geq 0$, la dernière condition implique que $\forall i, 0 \leq \alpha_i \leq c$.

Lagrangien et problème dual (suite)

- Après simplification, on obtient le problème dual suivant :

$$\begin{array}{rl} & \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ slc & \forall i, 0 \leq \alpha_i \leq c \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{array}$$

- Nous obtenons la solution $\hat{\boldsymbol{\alpha}}$ et le vecteur normal de la frontière de décision $\hat{\mathbf{a}}_{svm} \in \mathbb{X}$, est tel que :

$$\hat{\mathbf{a}}_{svm} = \sum_{i=1}^n \hat{\alpha}_i y_i \mathbf{x}_i$$

- Les conditions KKT suivantes permettent par ailleurs de caractériser également la solution optimale obtenue vis à vis du primal :

$$\left\{\begin{array}{l} \forall i, \alpha_i \left(y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) - (1 - \xi_i)\right) = 0 \\ \forall i, \mu_i \xi_i = 0 \end{array}\right.$$

Lagrangien et problème dual (suite)

- Le vecteur normal de la frontière étant :

$$\hat{\mathbf{a}}_{svm} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i$$

- Nous avons les interprétations suivantes :
  1. Si $\hat{\alpha}_i > 0$ alors $\mathbf{x}_i$ participe à la définition de $\hat{\mathbf{a}}_{svm}$.
  2. Comme $\hat{\mu}_i \hat{\xi}_i = 0$ (KKT), alors $\hat{\xi}_i = 0 \Rightarrow \hat{\mu}_i > 0$ et dans ce cas, $0 \leq \hat{\alpha}_i < c$.
     Par ailleurs, si $\hat{\xi}_i = 0$ alors $\hat{\alpha}_i \left(y_i(\hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i + \hat{a}_{svm,0}) - 1\right) = 0$ (KKT).
     Ainsi, si de plus $0 < \hat{\alpha}_i$, cela implique que $\mathbf{x}_i$ est sur une frontière de la bande puisque $y_i(\hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i + \hat{a}_{svm,0}) = 1$.
  3. On en déduit également que $\hat{\xi}_i > 0 \Rightarrow \hat{\mu}_i = 0$ et dans ce cas $\hat{\alpha}_i = c$.
     Alors, $\mathbf{x}_i$ est dans l'intérieur de la bande puisque $y_i(\hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i + \hat{a}_{svm,0}) = 1 - \hat{\xi}_i$. En fonction du signe du membre de droite, il peut être bien ou mal catégorisé.

Lagrangien et problème dual (suite)

- On obtient $\hat{a}_{svm,0}$ à l'aide de l'équation suivante pour n'importe quel vecteur support (càd tel que $0 < \hat{\alpha}_i < c$) :

$$\hat{a}_{svm,0} = y_i - \hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i$$

- La **fonction de décision** $\hat{f}(\mathbf{x})$ dépend alors de la fonction $\hat{g}(\mathbf{x}) = \hat{\mathbf{a}}_{svm}^\top \mathbf{x} + \hat{a}_{svm,0}$:

$$\hat{f}(\mathbf{x}) = \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) > 0 \\ C_2 & \text{sinon} \end{cases}$$

- Le problème dual est plus simple à résoudre que le problème primal.
- Le problème **dual** permet de faire dépendre la **complexité** du problème **en fonction de** n plutôt qu'en fonction de p !
- Les svm peuvent ainsi traiter les **problèmes de grande dimension** ($n << p$) plus efficacement que les modèles linéaires précédents !

Choix du paramètre c

- On remarquera la similitude entre la fonction objectif du svm et les modèles pénalisés précédents :

$$\min_{a_0, \mathbf{a} \in \mathbb{R}^p, \boldsymbol{\xi} \in \mathbb{R}^n} \frac{1}{2} \underbrace{\|\mathbf{a}\|^2}_{\text{pénalité}} + c \underbrace{\sum_{i=1}^{n} \xi_i}_{\text{perte}}$$

- Le svm nécessite également le "tuning" du paramètre c qui arbitre entre la fonction de perte et la fonction de pénalité.
- c peut être sélectionné par validation croisée comme indiquer en slide 83 (mais en utilisant le taux d'erreur comme critère).
- Il existe aussi des travaux pour déterminer le **chemin de régularisation** d'un svm, càd le calcul de $\hat{\mathbf{a}}_{svm}(c)$ pour $c \in [0, \infty]$.
- Dans [Hastie et al., 2004], les auteurs montrent que $\hat{\mathbf{a}}_{svm}(c)$ est linéaire par morceaux (comme le lasso). Leur algorithme est inspiré de lars.

Expansions de base et noyaux

- Si le problème n'est pas linéairement séparable, nous pouvons appliquer une **expansion de base** de $\mathbb{X}$ dans un espace étendu $\mathbb{F}$:

$$\phi : \mathbb{X} \rightarrow \mathbb{F}$$

- Dans ce cas un modèle linéaire dans $\mathbb{F}$ correspond à un modèle non linéaire dans $\mathbb{X}$. Donc au lieu de manipuler les vecteurs $\mathbf{x} \in \mathbb{X}$, on manipule des vecteurs $\phi(\mathbf{x}) \in \mathbb{F}$.
- Les développements précédents sont les mêmes pour obtenir le problème dual suivant :

$$\begin{array}{ll} & \max_{\boldsymbol{\alpha} \in \mathbb{F}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \\ slc & \forall i, 0 \leq \alpha_i \leq c \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{array}$$

Expansions de base et noyaux

- La solution du dual est donnée par :

$$\hat{\mathbf{a}}_{svm} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \phi(\mathbf{x}_i)$$

où $\hat{\mathbf{a}}_{svm} \in \mathbb{F}$.
- Par ailleurs :

$$\hat{a}_{svm,0} = y_i - \hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i$$

où $\mathbf{x}_i$ est un vecteur support càd tel que $0 < \hat{\alpha}_i$.
- La fonction de score ou de discrimination du svm est donnée par :

$$\hat{g}(\mathbf{x}) = \hat{\mathbf{a}}_{svm}^\top \phi(\mathbf{x}) + \hat{a}_{svm,0}$$

- La fonction de décision reste :

$$\hat{f}(\mathbf{x}) = \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) > 0 \\ C_2 & \text{sinon} \end{cases}$$

Expansions de base et noyaux

- Regardons de plus prés la fonction de score du svm :

$$\begin{aligned} \hat{g}(\mathbf{x}) &= \hat{\mathbf{a}}_{svm}^\top \phi(\mathbf{x}) + \hat{a}_{svm,0} \\ &= \underbrace{\sum_{i=1}^{n} \hat{\alpha}_i y_i \phi(\mathbf{x}_i)^\top}_{\hat{\mathbf{a}}_{svm}^\top} \phi(\mathbf{x}) + \hat{a}_{svm,0} \end{aligned}$$

- En utilisant le dual, les éléments importants dans le cadre du svm peuvent s'exprimer en termes de **produit scalaires** dans l'espace étendu $\mathbb{F}$: $\phi(\mathbf{x}_i)^\top \phi(\mathbf{x})$.
- Posons alors $K : \mathbb{F} \times \mathbb{F} \to \mathbb{R}$ tel que :

$$K(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x})$$

Expansions de base et noyaux

- Le problème d'optimisation dual s'écrit donc :

$$\begin{aligned} & \max_{\boldsymbol{\alpha} \in \mathbb{F}} \textstyle\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ slc \quad & \forall i, 0 \leq \alpha_i \leq c \\ & \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

- La fonction de score obtenue également :

$$\hat{g}(\mathbf{x}) = \sum_{i=1}^{n} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + \hat{a}_{svm,0}$$

- La fonction $K(.,.)$ est appelée **noyau** ("kernel") et les méthodes qui remplacent le produit scalaire dans $\mathbb{X}$ par un produit scalaire dans un espace issu d'une expansion de base $\mathbb{F}$ sont dites **méthodes à noyaux** ("kernel methods" ou "kernel machines").
- L'intérêt de ces fonctions est qu'elles ne nécessitent pas de représenter explicitement $\mathbf{x}$ dans $\mathbb{F}$ (càd on ne calcule jamais $\phi(\mathbf{x})$ - "**kernel trick**").

Les noyaux

- $K(\mathbf{x}, \mathbf{y})$ représente un produit scalaire et doit satisfaire plusieurs types de contraintes.
- Notons $\mathbf{K}$ la matrice carrée de taille $(n \times n)$ de produits scalaires dont le terme général est tel que :

$$\begin{aligned} k_{ij} &= K(\mathbf{x}_i, \mathbf{x}_j) \\ &= \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \\ &= \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \end{aligned}$$

- On appelle une matrice de produits scalaires une **matrice de Gram**.
- La matrice $\mathbf{K}$ doit alors satisfaire les propriétés suivantes :
  - Symétrie : $\forall i, j, k_{ij} = k_{ji}$.
  - Semi-définie positivité : $\forall \mathbf{z} \in \mathbb{R}^n, \mathbf{z}^\top \mathbf{K} \mathbf{z} \geq 0$.

Les noyaux (suite)

- Soit dans $\mathbb{X} = \mathbb{R}^2$ deux vecteurs $\mathbf{x} = (x_1, x_2)$ et $\mathbf{y} = (y_1, y_2)$.
- Exemple classique de noyau :

$$\begin{aligned} K(\mathbf{x},\mathbf{y}) &= (\langle \mathbf{x},\mathbf{y}\rangle + 1)^2 \\ &= (x_1y_1 + x_2y_2 + 1)^2 \\ &= (x_1y_1)^2 + (x_2y_2)^2 + 1 + 2x_1y_1x_2y_2 + 2x_1y_1 + 2x_2y_2 \end{aligned}$$

- Ce noyau correspond à l'expansion de base ϕ suivante :

$$\phi(\mathbf{x}) = (x_1^2, x_2^2, 1, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2)$$

- On vérifie bien en effet que : $K(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$.
- En utilisant K, la complexité de calcul reste en $O(dim(\mathbb{X}))$ plutôt que $O(dim(\mathbb{F}))$!

Les noyaux (suite)

Il existe plusieurs familles de noyaux :

- Les **noyaux polynomiaux** de degré d ("Polynomial kernels") :

$$K(\mathbf{x},\mathbf{y}) = (\langle \mathbf{x},\mathbf{y}\rangle + 1)^d$$

Ces noyaux sont relatifs à une expansion de base reposant sur des polynômes de degré d des composantes initiales. Le cas $d = 1$ est appelé noyau linéaire (produit scalaire dans l'espace initial $\mathbb{X}$).

- Les **fonctions à bases radiales** ("Radial basis functions" (RBF)) :

$$K(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$$

Ces noyaux reposent sur la notion de voisinage (hypersphère de centre $\mathbf{x}$ et de rayon σ^2).

Les noyaux (suite)

- Expansion de base associée au noyau RBF :

$$\begin{aligned} K(\mathbf{x},\mathbf{y}) &= \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right) \\ &= \frac{\exp\left(\frac{\langle \mathbf{x},\mathbf{y}\rangle}{\sigma^2}\right)}{\sqrt{\exp\left(\frac{\langle \mathbf{x},\mathbf{x}\rangle}{\sigma^2}\right)\exp\left(\frac{\langle \mathbf{y},\mathbf{y}\rangle}{\sigma^2}\right)}} \end{aligned}$$

- Rappelons le développement en séries de Taylor de l'exponentielle :

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

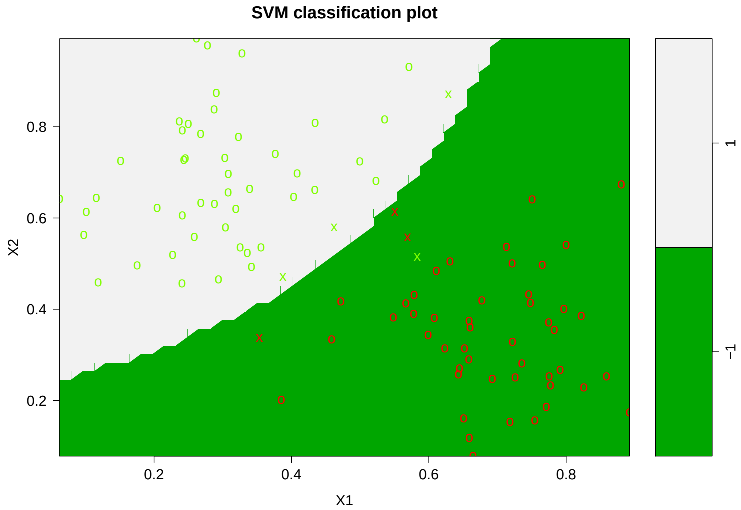
- Le noyau RBF correspond donc à une mesure **cosinus** dans un espace de **dimension infinie**.

Les noyaux (suite)

- Les noyaux permettent donc de travailler implicitement dans un espace $\mathbb{F}$ qui peut être de très grande dimension.
- En projetant les données dans $\mathbb{F}$, on espère pouvoir rendre le problème davantage linéairement séparable que dans $\mathbb{X}$. Ceci permettrait d'utiliser le concept d'optimisation de la marge dans un espace plus adéquat afin d'avoir de meilleures performances.
- Dans l'espace $\mathbb{F}$ on obtient donc une frontière linéaire qui s'exprime à l'aide de vecteurs supports : $\hat{g}(\mathbf{x}) = \sum_{i=1}^{n} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + \hat{a}_{svm,0}$.
- En revanche, dans l'espace initial $\mathbb{X}$ on obtient une frontière de décision **non linéaire**.
- Pour un noyau polynomial, plus le paramètre d est petit, plus la frontière dans $\mathbb{X}$ que l'on obtient est lisse ("smooth").
- Pour un noyau RBF, plus le paramètre σ^2 est grand, plus la frontière dans $\mathbb{X}$ que l'on obtient est lisse.
- Les paramètres des noyaux peuvent être estimés par validation croisée.

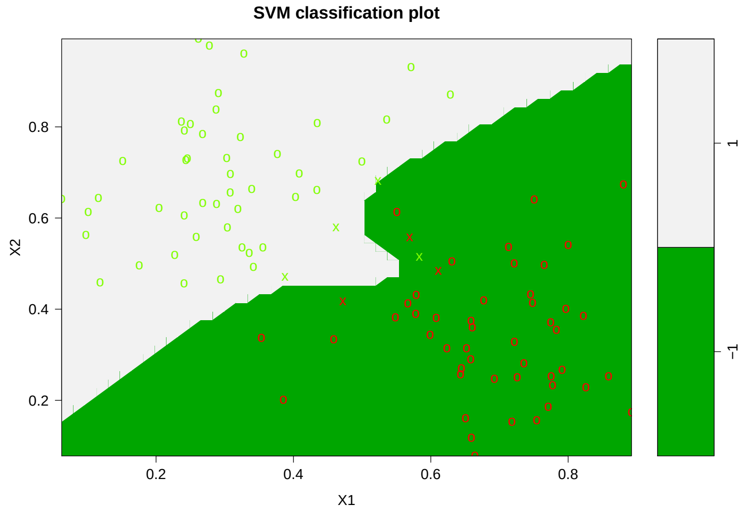
Exemple (suite)

- Avec un noyau polynomial $K(\mathbf{x},\mathbf{y}) = (\langle \mathbf{x},\mathbf{y}\rangle + 1)^2$ $(d = 2)$.



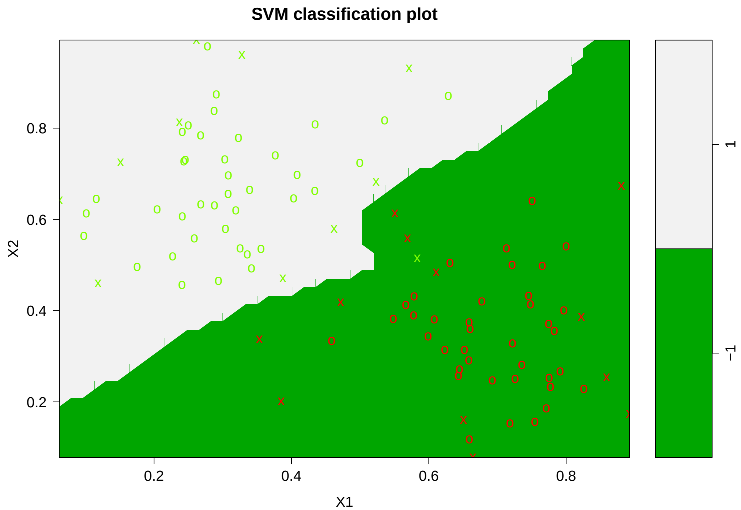
Exemple (suite)

- Avec un noyau polynomial $K(\mathbf{x},\mathbf{y}) = (\langle \mathbf{x},\mathbf{y}\rangle + 1)^{10}$ $(d = 10)$



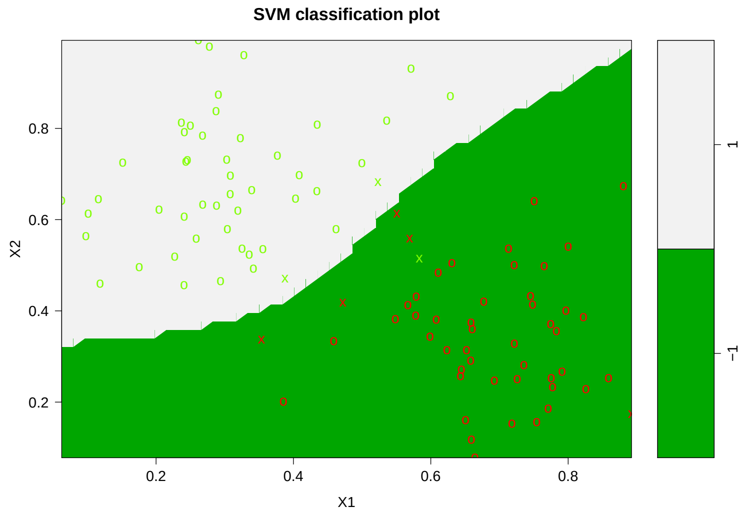
Exemple (suite)

- Avec un noyau RBF $K(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{0.5}\right)$ $(\sigma^2 = 0.25)$



Exemple (suite)

- Avec un noyau RBF $K(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2}\right)$ $(\sigma^2 = 1)$



Les svm appliqués au problème de régression

- Nous supposons maintenant que $\mathbb{Y} = \mathbb{R}$.
- Les idées de marge, de variables d'écarts, de noyaux... peuvent être généralisées pour le problème de régression.
- Supposons d'abord un noyau linéaire. Nous avons alors la famille d'hypothèses $\mathbb{H}$ qui est l'ensemble des fonctions de type :

$$f(\mathbf{x}) = a_0 + \mathbf{a}^\top \mathbf{x}$$

- Rappelons que la régression linéaire et le problème des MCO sont :

$$\min_{a_0,\mathbf{a}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - (a_0 + \mathbf{a}^\top \mathbf{x}))^2$$

- Par ailleurs, la régression ridge ajoute au *scr* une fonction de pénalité :

$$\min_{a_0,\mathbf{a}\in\mathbb{R}^p} \sum_{i=1}^{n} \left(y_i - (a_0 + \mathbf{a}^\top \mathbf{x})\right)^2 + \lambda\|\mathbf{a}\|^2$$

Fonction de perte

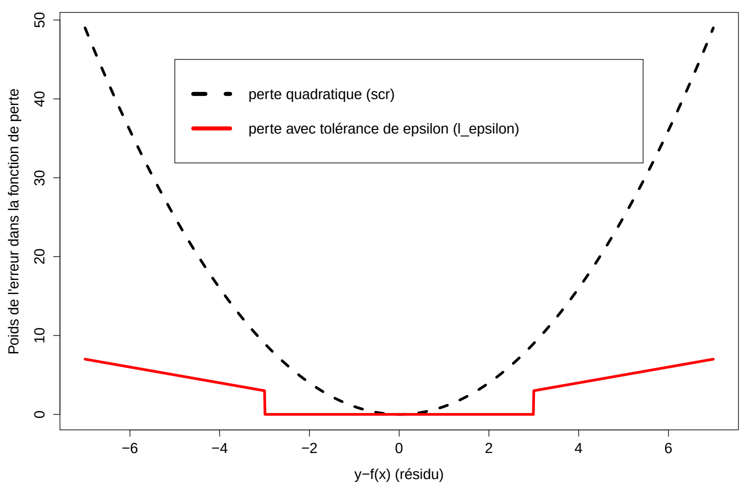
- En comparaison des méthodes précécentes, les svm cherchent à minimiser la fonction de perte suivante :

$$l_\epsilon(f(\mathbf{x}), y) = \begin{cases} 0 & \text{si } |y - f(\mathbf{x})| < \epsilon \\ |y - f(\mathbf{x})| - \epsilon & \text{sinon} \end{cases}$$

où $\epsilon > 0$ est un paramètre relatif à une **marge** d'erreur.
- On peut interpréter l_ϵ de la façon suivante :
  - On tolère des erreurs d'ajustement jusqu'à une quantité ϵ.
  - Au delà de ϵ le poids d'une erreur est linéaire et non quadratique.
  - l_ϵ est plus robuste vis à vis du bruit.
- Les svm pour la régression combinent $l_\epsilon(f(\mathbf{x}), y)$ et la fonction de pénalité quadratique :

$$\min_{a_0,\mathbf{a}\in\mathbb{R}^p} c\sum_{i=1}^{n} l_\epsilon(f(\mathbf{x}_i), y_i) + \|\mathbf{a}\|^2$$

Fonction de perte (suite)



Fonction de perte (suite)

- Sortir de l'intervalle de tolérance de taille $\epsilon > 0$ se produit quand :
  - $a_0 + \mathbf{a}^\top \mathbf{x}_i < y_i - \epsilon$: le modèle prédit une valeur trop faible.
  - $a_0 + \mathbf{a}^\top \mathbf{x}_i > y_i + \epsilon$: le modèle prédit une valeur trop forte.
- On introduit des variables d'écarts pour formaliser ces "sorties" (de la bande ou du tube). Soient $\forall i, \xi_i^+ \geq 0$ et $\xi_i^- \geq 0$ telles que :

$$\begin{cases} y_i - (a_0 + \mathbf{a}^\top \mathbf{x}_i) & \leq & \epsilon + \xi_i^+ \\ (a_0 + \mathbf{a}^\top \mathbf{x}_i) - y_i & \leq & \epsilon + \xi_i^- \end{cases}$$

- On voit que $|y_i - (a_0 + \mathbf{a}^\top \mathbf{x}_i)| \leq \epsilon \Leftrightarrow \xi_i^+ = \xi_i^- = 0$
- Minimiser les variables d'écart est équivalent à minimiser l_ϵ.
- Le problème peut donc se reformuler de façon équivalente comme :

$$\begin{array}{ll} & \min_{a_0,\mathbf{a}\in\mathbb{R}^p,\boldsymbol{\xi}^+,\boldsymbol{\xi}^-\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{a}\|^2 + c\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) \\ slc & \forall i, y_i - (a_0 + \mathbf{a}^\top \mathbf{x}_i) \leq \epsilon + \xi_i^+ \\ & \forall i, (a_0 + \mathbf{a}^\top \mathbf{x}_i) - y_i \leq \epsilon + \xi_i^- \\ & \forall i, \xi_i^+, \xi_i^- \geq 0 \end{array}$$

Lagrangien et problème dual

- Le Lagrangien (primal) dépend des variables primales $a_0, \mathbf{a}, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-$ et des multiplicateurs de Lagrange $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\mu}^+, \boldsymbol{\mu}^-$ qui sont des vecteurs de $\mathbb{R}^n$. Il est donné par :

$$\begin{array}{rcl} lag_p & = & \frac{1}{2}\|\mathbf{a}\|^2 + c\sum_{i=1}^n(\xi_i^+ + \xi_i^-) \\ & & -\sum_i \alpha_i^+\left(\epsilon + \xi_i^+ - y_i + (a_0 + \mathbf{a}^\top \mathbf{x}_i)\right) \\ & & -\sum_i \alpha_i^-\left(\epsilon + \xi_i^+ + y_i - (a_0 + \mathbf{a}^\top \mathbf{x}_i)\right) \\ & & -\sum_i \left(\mu_i^+\xi_i^+ + \mu_i^-\xi_i^-\right) \end{array}$$

- A l'optimum, les gradients de lag_p par rapport aux variables primales sont nuls :

$$\left\{\begin{array}{rcl} \frac{\partial lag_p}{\partial a_0} & = & 0 \\ \frac{\partial lag_p}{\partial \mathbf{a}} & = & \mathbf{0} \\ \frac{\partial lag_p}{\partial \boldsymbol{\xi}^+} & = & \mathbf{0} \\ \frac{\partial lag_p}{\partial \boldsymbol{\xi}^-} & = & \mathbf{0} \end{array}\right. \Leftrightarrow \left\{\begin{array}{rcl} \sum_{i=1}^n(\alpha_i^+ - \alpha_i^-)y_i & = & 0 \\ \mathbf{a} - \sum_{i=1}^n(\alpha_i^+ - \alpha_i^-)\mathbf{x}_i & = & \mathbf{0} \\ c\mathbf{1} - \boldsymbol{\alpha}^+ - \boldsymbol{\mu}^+ & = & \mathbf{0} \\ c\mathbf{1} - \boldsymbol{\alpha}^- - \boldsymbol{\mu}^- & = & \mathbf{0} \end{array}\right.$$

où $\mathbf{1}$ est le vecteur de taille $(n \times 1)$ rempli de 1

Lagrangien et problème dual (suite)

- En injectant les relations précédentes dans la fonction objectif, on obtient le problème dual suivant :

$$\begin{array}{rl} \max\limits_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^- \in \mathbb{R}^n} & -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)\mathbf{x}_i^\top\mathbf{x}_j \\ & -\epsilon\sum_{i=1}^n(\alpha_i^+ + \alpha_i^-) - \sum_{i=1}^n y_i(\alpha_i^+ - \alpha_i^-) \\ slc & \forall i, 0 \leq \alpha_i^+ \leq c \\ & \forall i, 0 \leq \alpha_i^- \leq c \\ & \sum_{i=1}^n(\alpha_i^+ - \alpha_i^-) = 0 \end{array}$$

- Une fois résolu ce problème quadratique contraint, on obtient la fonction de prédiction suivante qui dépend donc de vecteurs supports :

$$\begin{array}{rcl} \hat{f}(\mathbf{x}) & = & \hat{a}_0 + \hat{\mathbf{a}}^\top\mathbf{x} \\ & = & \hat{a}_{svm,0} + \sum\limits_{i=1}^n(\hat{\alpha}_i^+ - \hat{\alpha}_i^-)\mathbf{x}_i^\top\mathbf{x} \end{array}$$

Lagrangien et problème dual (suite)

- Les conditions KKT permettent d'avoir les interprétations suivantes :

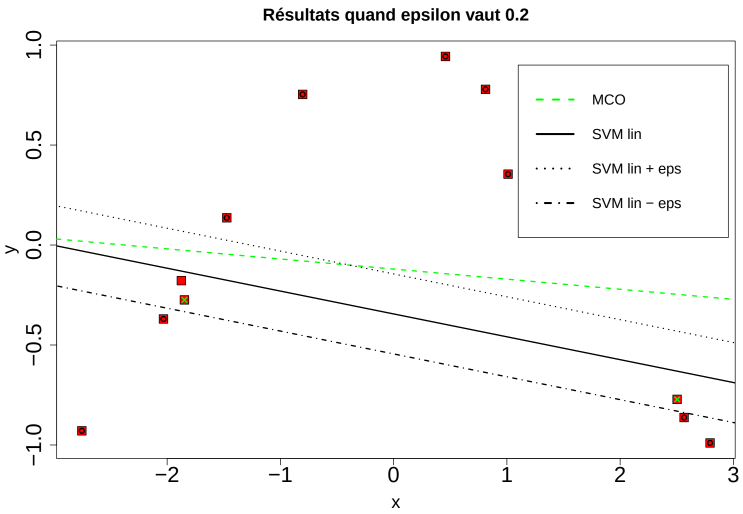
1. Si $\hat{\alpha}_i^+ = \hat{\alpha}_i^- = 0$ alors $\mathbf{x}_i$ est dans le tube de précision ϵ.
2. Si $\hat{\alpha}_i^+ > 0$ ou $\hat{\alpha}_i^- > 0$ alors on a deux sous-cas :
   a. Si $0 < \hat{\alpha}_i^+ < c$ ou $0 < \hat{\alpha}_i^- < c$ alors $\mathbf{x}_i$ est sur une frontière du tube (et donc $\xi_i^+ = \xi_i^- = 0$).
   b. Si $\hat{\alpha}_i^+ = c$ ou $\hat{\alpha}_i^- = c$ alors $\mathbf{x}_i$ est à l'extérieur du tube.

- Dans le cas de la régression, les points à l'intérieur du tube ne sont pas des vecteurs supports.
- Les points $\mathbf{x}_i$ sur la frontière qui sont tels que $0 < \hat{\alpha}_i^+ < c$ ou $0 < \hat{\alpha}_i^- < c$ permettent de calculer $\hat{a}_{svm,0}$ puisque dans ce cas, nous avons (KKT) :

$$\epsilon - y_i + (\hat{a}_{svm,0} + \hat{\mathbf{a}}_{svm}^\top\mathbf{x}_i) = 0 \text{ ou } \epsilon + y_i - (\hat{a}_{svm,0} + \hat{\mathbf{a}}_{svm}^\top\mathbf{x}_i) = 0$$

Exemple

- Avec un noyau linéaire



Les noyaux

- Comme pour la catégorisation, le problème dual et la fonction de discrimination s'expriment par le biais de produits scalaires.
- Nous pouvons donc étendre l'approche à des noyaux conduisant alors à des modèles non linéaires dans $\mathbb{X}$.
- Formellement, les svm appliquées au problème de régression consistent à résoudre le problème suivant :

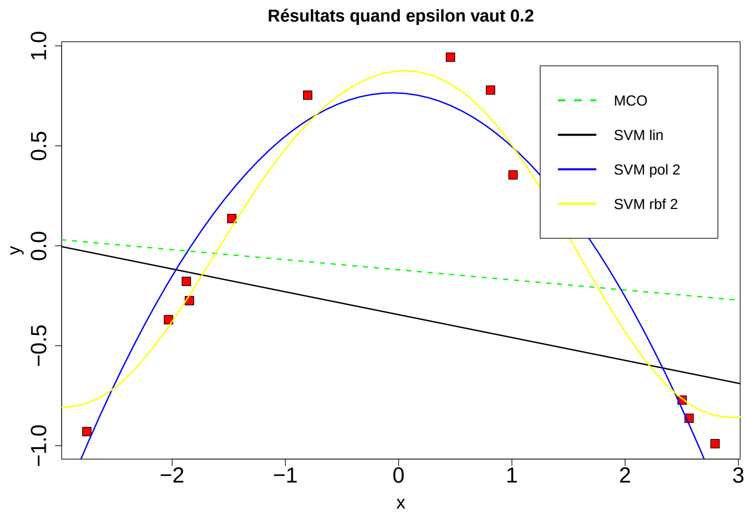
$$\begin{aligned}
&\max_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-\in\mathbb{R}^n} -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n(\alpha_i^+-\alpha_i^-)(\alpha_j^+-\alpha_j^-)K(\mathbf{x}_i,\mathbf{x}_j)\\
&\qquad\qquad -\epsilon\sum_{i=1}^n(\alpha_i^++\alpha_i^-)-\sum_{i=1}^n y_i(\alpha_i^+-\alpha_i^-)\\
slc\quad &\forall i, 0\leq\alpha_i^+\leq c\\
&\forall i, 0\leq\alpha_i^-\leq c\\
&\sum_{i=1}^n(\alpha_i^+-\alpha_i^-)=0
\end{aligned}$$

- La fonction de prédiction est alors :

$$\hat{f}(\mathbf{x})=\hat{a}_{svm,0}+\sum_{i=1}^n(\hat{\alpha}_i^+-\hat{\alpha}_i^-)K(\mathbf{x}_i,\mathbf{x})$$

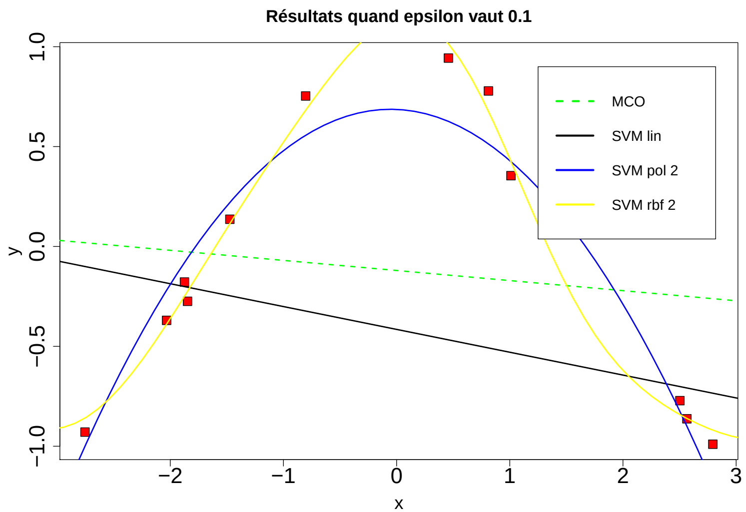
Exemple (suite)

- Avec différents types de noyaux et $\epsilon = 0.2$.



Exemple (suite)

- Avec différents types de noyaux et $\epsilon = 0.1$ (moins de tolérance).



Rappel du Sommaire

Rappel du Sommaire

Rappel

- Apprentissage automatique :
  - **Supervisé** : on dispose d'un ensemble d'objets et pour chaque objet une valeur cible associée ; il faut apprendre un modèle capable de prédire la bonne valeur cible d'un objet nouveau.
  - ▷ **Non-supervisé** : on dispose d'un ensemble d'objets sans aucune valeur cible associée ; il faut apprendre un modèle capable d'extraire les **régularités** présentes au sein des objets pour mieux visualiser ou appréhender la structure de l'ensemble des données.
- Une façon "probabiliste" de présenter la différence entre supervisé et non supervisé est la suivante :
  - En supervisé on est intéressé par modéliser $P(Y|X)$ (discriminatif) ou $P(X,Y)$ (génératif).
  - En non-supervisé on est plutôt intéressé par modéliser $P(X)$.

Deux problèmes classiques

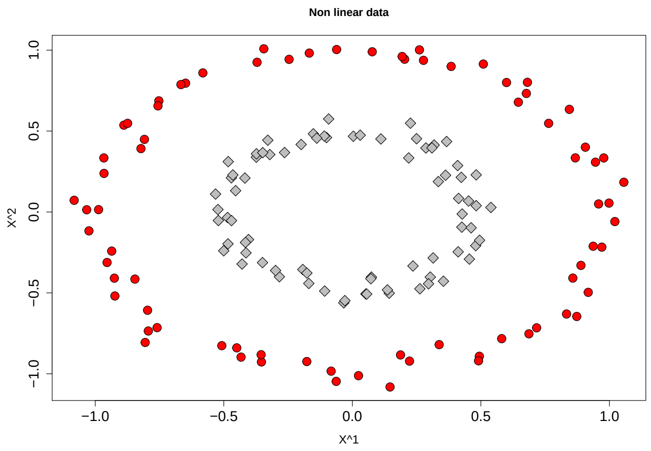
- Nous allons considérer deux problèmes classiques en apprentissage non-supervisé :
  - **Recherche d'espaces latents** : Y'a t-il dans $\mathbb{X}$ (l'espace de description), des sous-espaces où la densité d'objets est plus importante que d'autres ? Comment déterminer et caractériser ces régions ?
  - **Classification automatique (clustering)** : Peut-on déterminer des groupes homogènes d'objets tels qu'ils soient plus similaires entre eux qu'avec les autres. Comment caractériser et déterminer ces groupes ?
- L'un ou l'autre des problèmes permet d'appréhender $P(X)$.
- Les notions de similarités/distances entre objets et de corrélations/association entre variables sont fondamentales pour modéliser le concept de régularité.

Deux solutions classiques

- Nous allons considérer deux solutions classiques à ces deux problèmes :
  - **Recherche d'espaces latents** : l'Analyse en Composantes Principales.
  - **Classification automatique** : la méthode des k-means.
- Les distances/métriques utilisées sont euclidiennes.
- Hypothèses implicites :
  - ACP : les données appatiennent à des espaces linéaires et non courbés.
  - k-means : les groupes sont représentés dans l'espace par des ellipsoïdes.

▷ Ces méthodes présentent donc des limites si les données appatiennent à des **espaces courbés**. C'est le cas des données rencontrées en multimedia, en bio-informatique...

Illustration



- Que donnerait l'ACP ici ?
- Comment se positionneraient les barycentres des deux classes ?

Rappel du Sommaire

Introduction

- Les noyaux permettent de projetter implicitement les objects dans des espaces de grande dimension.
- Cette projection peut être vue comme une expansion de base reposant sur des **transformations non-linéaires** des variables initiales.
- Nous avons vu que les noyaux permettaient de traiter plus efficacement certains problèmes (les cas non linéairement séparables) en apprentissage supervisé.
- Mais de façon plus générale, ils permettent d'étendre plusieurs méthodes classiques et notamment en apprentissage non-supervisé.
- Nous voyons ci-après l'application de ces concepts dans le cadre :
  - des méthodes de réduction de dimension : l'**ACP à noyaux** [Schölkopf et al., 1998],
  - des méthodes de clustering : les k-**means à noyaux** et de façon plus générale le **spectral clustering**.

Rappel des objectifs de l'ACP

- L'ACP est une méthode de **réduction de dimension** :
  - Représentation d'un nuage de points décrit initialement dans un espace euclidien de grande dimension, dans un sous-espace de faible dimension.
  - Ce sous-espace de faible dimension doit préserver au maximum l'information contenue dans l'espace initial.
  - Le concept d'information en ACP est formalisé par la variance ou inertie du nuage de points.
  - La projection sur le sous-espace de faible dimension doit "déformer" le moins possible le nuage.
  - La résolution du problème d'optimisation sous-jacent conduit à une solution explicite qui est la décomposition spectrale de la matrice des corrélations (ACP normée).
- Dans la suite, nous introduisons brièvement les notations et rappelons les résultats de l'ACP normé.

Données traitées par l'ACP et notations

- L'ACP traite des tables de données dont les **variables sont quantitatives continues**.
- Ces données se retrouvent dans la matrice $\mathbf{T}$ de taille $(n \times p)$:

$$\mathbf{T} = \begin{array}{c} \\ \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_i \\ \vdots \\ \mathbf{t}_n \end{array} \begin{array}{c} \begin{array}{ccccc} \mathbf{t}^1 & \ldots & \mathbf{t}^k & \ldots & \mathbf{t}^p \end{array} \\ \left(\begin{array}{ccccc} & & \vdots & & \\ & & \vdots & & \\ \ldots & \ldots & t_{ik} & \ldots & \ldots \\ & & \vdots & & \\ & & \vdots & & \end{array}\right) \end{array}$$

- On suppose n objets $\{\mathbf{t}_1, \ldots, \mathbf{t}_i, \ldots, \mathbf{t}_n\}$ où $\mathbf{t}_i \in \mathbb{R}^p$.
- On suppose p variables $\{\mathbf{t}^1, \ldots, \mathbf{t}^k, \ldots, \mathbf{t}^p\}$ où $\mathbf{t}^k \in \mathbb{R}^n$.
  où $\forall i = 1, \ldots, n; \forall k = 1, \ldots, p : t_{ik} \in \mathbb{R}$.

Proximité et nuage des variables

- L'ensemble des vecteurs $\{\mathbf{t}_1, \ldots, \mathbf{t}_n\}$ de $\mathbb{R}^p$ forme le nuage $\mathbb{NO}$.
- L'ensemble des vecteurs $\{\mathbf{t}^1, \ldots, \mathbf{t}^p\}$ de $\mathbb{R}^n$ forme le nuage $\mathbb{NA}$.
- Pour mesurer la **liaison** entre deux variables k et l on utilise le **coefficient de corrélation** noté $r(\mathbf{t}^k, \mathbf{t}^l)$.
- Etant donné une variable $\mathbf{t}^k$, on introduit au préalable :
  - sa **moyenne empirique** notée m_k (ie $\bar{\mathbf{t}}^k$) :

$$m_k = \frac{1}{n}\sum_{i=1}^{n} t_{ik}$$

  - sa **variance empirique** notée s_k^2 :

$$s_k^2 = \frac{1}{n}\sum_{i=1}^{n}(t_{ik} - m_k)^2$$

- Le vecteur $\mathbf{m} = (m_1, \ldots, m_p)$ est appelé barycentre de $\mathbb{NO}$.

Proximité et nuage des variables (suite)

- On **centre et on réduit** la variable $\mathbf{t}^k$ puis on **divise par** $\sqrt{n}$ et on obtient la variable $\mathbf{x}^k$ dont le terme général est donné par :

$$x_{ik} = \frac{t_{ik} - m_k}{s_k\sqrt{n}}$$

- On s'intéresse aux **relations de "proximité" entre variables** et on utilise ici le **coefficient de corrélation** (empirique) donné par :

$$r(\mathbf{t}^k, \mathbf{t}^l) = \sum_{i=1}^{n} \underbrace{\left(\frac{t_{ik} - m_k}{s_k\sqrt{n}}\right)}_{x_{ik}} \underbrace{\left(\frac{t_{il} - m_l}{s_l\sqrt{n}}\right)}_{x_{il}}$$

- Remarques sur le coefficient de corrélation :
  - $r(\mathbf{t}^k, \mathbf{t}^l)$ est le produit scalaire entre les vecteurs $\mathbf{x}^k$ et $\mathbf{x}^l$:
    $r(\mathbf{t}^k, \mathbf{t}^l) = \langle \mathbf{x}^k, \mathbf{x}^l \rangle = \sum_{i=1}^{n} x_{ik}x_{il}$.
  - $\forall k : r(\mathbf{t}^k, \mathbf{t}^k) = \langle \mathbf{x}^k, \mathbf{x}^k \rangle = \|\mathbf{x}^k\|^2 = 1$, donc tous les vecteurs $\mathbf{x}^k$ sont de norme unitaire et ils appartiennent tous à une hypersphère.

Transformation des données et ACP normée

- Nous supposons désormais que la matrice des données est celle des variables centrées, réduites et divisée par $\sqrt{n}$.
- Nous noterons $\mathbf{X}$ cette matrice :

$$\mathbf{X} = \begin{array}{c} \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_i \\ \vdots \\ \mathbf{x}_n \end{array} \begin{array}{c} \begin{array}{ccccc} \mathbf{x}^1 & \ldots & \mathbf{x}^k & \ldots & \mathbf{x}^p \end{array} \\ \left(\begin{array}{ccccc} & & \vdots & & \\ & & \vdots & & \\ \ldots & \ldots & x_{ik} & \ldots & \ldots \\ & & \vdots & & \\ & & \vdots & & \end{array}\right) \end{array}$$

Transformation des données et ACP normée (suite)

- Dans la suite, les vecteurs représentants les individus seront donc notés $\mathbf{x}_i$ et les vecteurs représentants les variables seront notés $\mathbf{x}^k$.
- Les propriétés de $\mathbf{X}$ sont alors les suivantes, $\forall k = 1, \ldots, p$:

$$\sum_{i=1}^{n} x_{ik} = 0 \quad \text{et} \quad \sum_{i=1}^{n} (x_{ik})^2 = 1$$

- Remarques :
  - **m** le barycentre de $\mathbb{NO}$ calculé dans le repère affine initial devient l'origine du nouveau repère. L'opération de centrage agit telle une **translation** de $\mathbb{NO}$ de l'origine initiale au barycentre.
  - Il est intéressant de noter que le centrage ne change pas les distances euclidiennes entre objets.
  - Après réduction les variables appartiennent à une hypersphère.
  - En pratique, la réduction permet aux variables de s'affranchir de leurs unités de mesure ce qui rend l'analyse plus robuste face aux biais associés aux différences d'échelles. On parle d'**ACP normée**.

Variance ou inertie du nuage des objets

- La notion de variance de nuage de points est centrale puisque c'est la quantité que nous cherchons à préserver.
- Dans le repère initial $\mathbb{A}$, $\mathbb{NO}$ a une **inertie totale** définie par :

$$int_{\mathbb{A}}(\mathbb{NO}) = \frac{1}{2n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} d^2(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{n} \sum_{i=1}^{n} d^2(\mathbf{x}_i, \bar{\mathbf{x}})$$

où $d^2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2 = \sum_{j=1}^{p} (x_j - y_j)^2$ est la dist. eucl. au carré.
- L'information contenue dans $\mathbb{NO}$ est mesurée par l'inertie qui indique, en moyenne, de combien s'éloignent les points du barycentre.
- Si l'inertie est faible alors les points sont en moyenne très proche du barycentre ce qui est peu informatif. Si l'inertie est grande, c'est donc qu'il y a bcp de disparités entre les objets. Le but alors est d'appréhender cette information de façon synthétique.

Variance ou inertie du nuage des individus

- Du fait du centrage, le barycentre devient l'origine du nouveau repère. On montre que dans ce nouveau repère affine :

$$int_{\mathbb{A}}(\mathbb{NO}) = \frac{1}{n} \sum_{i=1}^{n} d^2(\mathbf{x}_i, \mathbf{0}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i\|^2$$

- Donc dans l'espace affine de $\mathbb{R}^p$ dont l'origine est le barycentre, l'inertie est la moyenne pondérée des normes des vecteurs des individus.

▷ L'objectif est de déterminer des **sous-espaces** de sorte à ce que la projection des $\mathbf{x}_i$ dans ceux-ci **conservent au mieux l'inertie**. Autrement dit, on souhaite que l'image de $\mathbb{NO}$ dans ce sous-espace soit **le moins déformé possible**.

Ajustement du nuage des individus

- Pour déterminer en pratique ces sous-espaces, on cherche une **suite de s directions privilégiées** ($s < p$) dans $\mathbb{R}^p$ notées $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_s$ qui permettent de **maximiser l'inertie du nuage projeté**.
- Ce sont en fait s vecteurs de $\mathbb{R}^p$ appelés **axes factoriels (ou principaux)** qui ont les propriétés suivantes :
  - $\mathbf{u}_1$ est le sous-espace de dimension 1 qui maximise l'inertie du nuage projeté.
  - $\mathbf{u}_2$ est orthogonal à $\mathbf{u}_1$ et le plan engendré par $\{\mathbf{u}_1, \mathbf{u}_2\}$ est l'espace de dimension 2 qui maximise l'inertie du nuage projeté.
  - $\mathbf{u}_3$ est orthogonal à $\mathbf{u}_1$ et $\mathbf{u}_2$ et le sous-espace engendré par $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ est l'espace de dimension 3 qui maximise l'inertie du nuage projeté.
  - ...
- Les vecteurs $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_s$ sont donc orthogonaux entre eux et permettent de maximiser l'inertie des points images lorsque l'on projette le nuage $\mathbb{NO}$.

Détermination des axes factoriels (ou principaux)

- On montre que les vecteurs $\mathbf{u}_1, \ldots, \mathbf{u}_s$ peuvent être obtenus en **diagonalisant la matrice des coefficients de corrélation** que l'on notera par $\mathbf{C}$ et qui est de taille $(p \times p)$.
- En utilisant la matrice de données centrées-réduites $\mathbf{X}$, on a :

$$\mathbf{C} = \mathbf{X}^\top \mathbf{X}$$

où $\mathbf{X}^\top$ est la transposée de $\mathbf{X}$

- De façon explicite, nous avons le terme général de $\mathbf{C}$, $\forall k, l$:

$$\mathbf{C}_{kl} = \sum_{i=1}^{n} x_{ik} x_{il} \quad \text{(cf slide 203)}$$

▷ Pour tout $m = 1, \ldots, s$, $\mathbf{u}_m$ **est le vecteur propre associé à** λ_m **la** m**-ème plus grande valeur propre de** $\mathbf{C}$.

Extension à des noyaux

- Supposons maintenant qu'au lieu de représenter les objets dans $\mathbb{X}$, l'espace euclidien initial engendré par les variables $\mathbb{A}$, nous les représentions dans un espace de Hilbert de plus grande dimension $\mathbb{F}$ que l'on peut atteindre par l'application $\phi : \mathbb{X} \rightarrow \mathbb{F}$.
- Pour ne pas alourdir les notations notons $\mathbf{X}$ la matrice des composantes des vecteurs $\phi(\mathbf{x}_i)$ dans $\mathbb{F}$:

$$\mathbf{X} = \begin{array}{c} \\ \phi(\mathbf{x}_1) \\ \vdots \\ \phi(\mathbf{x}_i) \\ \vdots \\ \phi(\mathbf{x}_n) \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & \ldots & k & \ldots & \ldots \end{array} \\ \begin{pmatrix} & & \vdots & & \\ & & \vdots & & \\ \ldots & \ldots & [\phi(\mathbf{x}_i)]_k & \ldots & \ldots \\ & & \vdots & & \\ & & \vdots & & \end{pmatrix} \end{array}$$

Extension à des noyaux (suite)

- Supposons également que les $\phi(\mathbf{x}_i)$ sont des vecteurs dont les variables ont été centrées et réduites et divisée par $\sqrt{n}$ pour toute dimension k de $\mathbb{F}$ (comme dans le slide 204) :

$$\sum_{i=1}^{n} [\phi(\mathbf{x}_i)]_k = 0 \quad \text{et} \quad \sum_{i=1}^{n} [\phi(\mathbf{x}_i)]_k^2 = 1$$

- On peut exprimer $\mathbf{X}$ comme une collection de n vecteurs lignes et dans ce cas, $\mathbf{C}$, la matrice des coefficients de corrélations (dans $\mathbb{F}$), peut être reformulée comme suit :

$$\mathbf{C} = \mathbf{X}^\top \mathbf{X} = \begin{pmatrix} \phi(\mathbf{x}_1) & \ldots & \phi(\mathbf{x}_n) \end{pmatrix} \begin{pmatrix} \phi(\mathbf{x}_1)^\top \\ \vdots \\ \phi(\mathbf{x}_n)^\top \end{pmatrix} = \sum_{i=1}^{n} \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^\top$$

Extension à des noyaux (suite)

- Les axes factoriels $\mathbf{u}_1, \ldots, \mathbf{u}_s$ ($s < dim(\mathbb{F})$) sont des vecteurs propres de $\mathbf{C}$ et donc, $\forall m = 1, \ldots, s$:

$$\mathbf{C}\mathbf{u}_m = \lambda_m \mathbf{u}_m$$

- Par ailleurs, $\mathbf{u}_1, \ldots, \mathbf{u}_s \in \mathbb{F}$ et sont des **combinaisons linéaires** des $\phi(\mathbf{x}_i)$. Il existe donc $\forall m, \boldsymbol{\alpha}^m = (\alpha_1^m, \ldots, \alpha_n^m)$ tel que :

$$\mathbf{u}_m = \sum_{i=1}^{n} \alpha_i^m \phi(\mathbf{x}_i)$$

Autrement dit : $\mathbf{u}_m \in Vec\{\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)\}$.

- Le problème $\mathbf{C}\mathbf{u}_m = \lambda_m \mathbf{u}_m$ est alors équivalent à l'ensemble des problèmes suivants :

$$\phi(\mathbf{x}_i)^\top \mathbf{C}\mathbf{u}_m = \lambda_m \phi(\mathbf{x}_i)^\top \mathbf{u}_m, \quad \forall i = 1, \ldots, n$$

Extension à des noyaux (suite)

- Ensuite, en utilisant $\mathbf{C} = \sum_{k=1}^{n} \phi(\mathbf{x}_k)\phi(\mathbf{x}_k)^\top$, il vient :
  $\phi(\mathbf{x}_i)^\top \mathbf{C}\mathbf{u}_m = \lambda_m \phi(\mathbf{x}_i)^\top \mathbf{u}_m$
  $\Leftrightarrow \phi(\mathbf{x}_i)^\top \sum_k \phi(\mathbf{x}_k)\phi(\mathbf{x}_k)^\top \sum_j \alpha_j^m \phi(\mathbf{x}_j) = \lambda_m \phi(\mathbf{x}_i)^\top \sum_j \alpha_j^m \phi(\mathbf{x}_j)$
  $\Leftrightarrow \sum_k \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_k) \sum_j \alpha_j^m \phi(\mathbf{x}_k)^\top \phi(\mathbf{x}_j) = \lambda_m \sum_j \alpha_j^m \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$
  $\Leftrightarrow \sum_j \alpha_j^m \sum_k \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_k)\phi(\mathbf{x}_k)^\top \phi(\mathbf{x}_j) = \lambda_m \sum_j \alpha_j^m \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$
- Soit $\mathbf{K}$ la matrice de Gram avec $k_{ij} = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, alors :
  $\phi(\mathbf{x}_i)^\top \mathbf{C}\mathbf{u}_m = \lambda_m \phi(\mathbf{x}_i)^\top \mathbf{u}_m$
  $\Leftrightarrow \sum_j \alpha_j^m \sum_k k_{ik} k_{kj} = \lambda_m \sum_j \alpha_j^m k_{ij}$
- Si on considère l'équation précédente pour l'ensemble des $i = 1, \ldots, n$ alors on obtient le résulat suivant :

$$\mathbf{C}\mathbf{u}_m = \lambda_m \mathbf{u}_m \Leftrightarrow \mathbf{K}^2 \boldsymbol{\alpha}^m = \lambda_m \mathbf{K} \boldsymbol{\alpha}^m$$

Extension à des noyaux (suite)

▷ On montre ensuite dans [Schölkopf et al., 1998] que pour résoudre $\mathbf{K}^2 \boldsymbol{\alpha}^m = \lambda_m \mathbf{K} \boldsymbol{\alpha}^m$ il est suffisant de résoudre le problème spectral suivant pour les valeurs propres $\lambda_m \neq 0$:

$$\mathbf{K} \boldsymbol{\alpha}^m = \lambda_m \boldsymbol{\alpha}^m$$

- On cherche donc les coefficients de la combinaison linaire permettant de définir $\mathbf{u}_m$ en fonction des objets. En effet, une fois déterminé $\boldsymbol{\alpha}^m$, rappelons que le m-ème axe principal est donné par :

$$\mathbf{u}_m = \sum_{i=1}^{n} \alpha_i^m \phi(\mathbf{x}_i)$$

- Comme pour les svm, le problème initial $\mathbf{C}\mathbf{u}_m = \lambda_m \mathbf{u}_m$ est dans $\mathbb{F}$ alors que le "problème dual", $\mathbf{K}\boldsymbol{\alpha}^m = \lambda_m \boldsymbol{\alpha}^m$, est dans $\mathbb{R}^n$ (peut importe $\mathbb{F}$). On voit donc que cette approche permet aussi de traiter les données en grande dimension.

Composantes principales

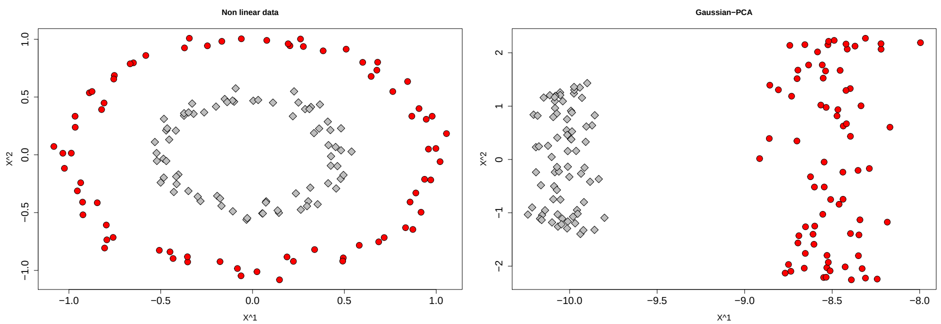
- Une fois déterminée les axes $\mathbf{u}_m, m = 1, \ldots, s$, il faut projeter les objets sur ceux-ci pour avoir la représentation de faible dimension.
- Ces coordonnes sont appelées **composantes principales**.
- Soit $\mathbf{f}^m \in \mathbb{R}^n$ les composantes principales des objets sur l'axe $\mathbf{u}_m$.
- Ces coordonnées peuvent être déterminées à l'aide de la matrice $\mathbf{K}$! On a en effet, $\forall i = 1, \ldots, n$:

$$\begin{aligned} f_i^m &= \langle \phi(\mathbf{x}_i), \mathbf{u}_m \rangle \\ &= \langle \phi(\mathbf{x}_i), \sum_{j=1}^{n} \alpha_j^m \phi(\mathbf{x}_j) \rangle \\ &= \sum_{j=1}^{n} \alpha_j^m \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \\ &= \sum_{j=1}^{n} \alpha_j^m k_{ij} \end{aligned}$$

Pseudo-code de l'ACP à noyaux

```
Input : X (données initiales), K (fonction noyau), s (nb d'axes)
Calculer la matrice K de terme général k_ij = K(x_i, x_j)
Résoudre Kα = λα (décomposition spectrale).
Pour tout m = 1,...,s faire
    Normaliser α^m de sorte que λ_m<α^m, α^m> = 1
    Pour tout i = 1,...,n faire (composantes principales)
        Calculer f_i^m = Σ_{j=1}^n α_j^m k_ij
    Fin Pour
Fin Pour
Ouput : {f^1,...f^s}
```

Illustration



- Résultats de l'ACP avec un noyau RBF avec $\sigma^2 = 2$.

Rappel des objectifs des k-means

- Les k-means est une méthode de **partitionnement** qui sépare $\mathbb{O}$ en k classes disjointes.
- Notons $\mathbb{P}(\mathbb{O}) = \{\mathbb{C}_1, \ldots, \mathbb{C}_k\}$ une partition de $\mathbb{O}$ en k classes.
- On suppose que les objets sont représentés dans un espace euclidien.
- Les k-means minimisent la somme des carrés des résidus [1] suivantes :

$$scr(\mathbb{P}(\mathbb{O})) = \sum_{l=1}^{k} \sum_{\mathbf{x}_i \in \mathbb{C}_l} \|\mathbf{x}_i - \mathbf{m}_l\|^2$$

où $\mathbf{m}_l = \frac{1}{|\mathbb{C}_l|} \sum_{\mathbf{x}_i \in \mathbb{C}_l} \mathbf{x}_i$ est le barycentre de $\mathbb{C}_l$.

- Remarques :
  - $\mathbf{m}_l$ est le représentant ou prototype de la classe $\mathbf{C}_l$.
  - $scr(\mathbb{P}(\mathbb{O}))$ peut être interprétée comme suit : mesure de la perte d'information si on devait représenter chaque objet par son prototype.

1. Equivalent ici à l'inertie inter-classe.

Rappel de l'algorithme des k-means

- Le problème est combinatoire (NP-hard).
- On utilise une heuristique qui détermine un optimum local.
- La complexité de l'heuristique est en $O(knp)$.

Input : $\mathbf{X}$ and k
Initialize $\mathbb{P}(\mathbb{O})$ with k different clusters
While a stopping criterion is not reached **do**
  For all $\mathbf{x}_i \in \mathbb{O}$ **do**
    For all $\mathbb{C}_l \in \mathbb{P}(\mathbb{O})$ **do**
      Compute $\|\mathbf{x}_i - \mathbf{m}_l\|^2$
    End For
    Find $\mathbb{C}_{l^*} = \arg\min_{\mathbb{C}_l \in \mathbb{P}(\mathbb{O})} \|\mathbf{x}_i - \mathbf{m}_l\|^2$
    Move $\mathbf{x}_i$ from its current cluster to $\mathbb{C}_{l^*}$
    Update the mean vectors accordingly
  End For
End While
Ouput : $\mathbb{P}(\mathbb{O})$

Extension à des noyaux

- Supposons maintenant qu'au lieu de représenter les objets dans $\mathbb{X}$, nous les représentions dans un espace de Hilbert de plus grande dimension $\mathbb{F}$ que l'on peut atteindre par l'application $\phi : \mathbb{X} \to \mathbb{F}$.
- Dans $\mathbb{F}$, les k-means minimisent :

$$scr(\mathbb{P}(\mathbb{O})) = \sum_{l=1}^{k} \sum_{\phi(\mathbf{x}_i) \in \mathbb{C}_l} \|\phi(\mathbf{x}_i) - \mathbf{m}_l\|^2$$

où $\mathbf{m}_l = \frac{1}{|\mathbb{C}_l|} \sum_{\phi(\mathbf{x}_i) \in \mathbb{C}_l} \phi(\mathbf{x}_i)$ est le barycentre de $\mathbb{C}_l$ dans $\mathbb{F}$.

- Si on développe la distance euclidienne, il vient :

$$\begin{aligned}\|\phi(\mathbf{x}_i) - \mathbf{m}_l\|^2 &= \langle \phi(\mathbf{x}_i) - \mathbf{m}_l, \phi(\mathbf{x}_i) - \mathbf{m}_l \rangle \\ &= \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle + \langle \mathbf{m}_l, \mathbf{m}_l \rangle - 2\langle \phi(\mathbf{x}_i), \mathbf{m}_l \rangle\end{aligned}$$

Extension à des noyaux (suite)

- Ensuite en considérant $\mathbf{m}_l = \frac{1}{|\mathbb{C}_l|}\sum_{\phi(\mathbf{x}_i)\in\mathbb{C}_l}\phi(\mathbf{x}_i)$ et en utilisant les propriétés de linéarités du produit scalaire, on a :

$$\begin{aligned}\|\phi(\mathbf{x}_i)-\mathbf{m}_l\|^2 =& \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_i)\rangle + \frac{1}{|\mathbb{C}_l|^2}\sum_{\mathbf{x}_i\in\mathbb{C}_l}\sum_{\mathbf{x}_j\in\mathbb{C}_l}\langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle\\ &-2\frac{1}{|\mathbb{C}_l|}\sum_{\mathbf{x}_j\in\mathbb{C}_l}\langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle\end{aligned}$$

▷ Soit $\mathbf{K}$ la matrice de Gram avec $k_{ij} = \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle$, alors :

$$\|\phi(\mathbf{x}_i)-\mathbf{m}_l\|^2 = k_{ii} + \frac{1}{|\mathbb{C}_l|^2}\sum_{i:\mathbf{x}_i\in\mathbb{C}_l}\sum_{j:\mathbf{x}_j\in\mathbb{C}_l}k_{ij} - 2\frac{1}{|\mathbb{C}_l|}\sum_{j:\mathbf{x}_j\in\mathbb{C}_l}k_{ij}$$

- On peut donc calculer les distances euclidiennes en utilisant uniquement la matrice de Gram $\mathbf{K}$ (complexité en n) et non pas la matrice de données $\mathbf{X}$ (complexité en $dim(\mathbb{F})$).

k-means pondérés à noyaux

- La version classique des k-means suppose un poids uniforme pour tous les objets. L'extension au cas non-uniforme peut s'avérer utile.
- Soit $\mathbf{w} = (w_1,\ldots,w_n)\in\mathbb{R}^{+n}$ où w_i est le poids de $\mathbf{x}_i$.
- Les k-means à noyaux peuvent être étendus aux **k-means pondérés à noyaux**. Dans ce cas, le barycentre de $\mathbb{C}_l$ est défini comme suit :

$$\mathbf{m}_l = \frac{1}{\sum_{i:\phi(\mathbf{x}_i)\in\mathbb{C}_l} w_i}\sum_{i:\phi(\mathbf{x}_i)\in\mathbb{C}_l} w_i\phi(\mathbf{x}_i)$$

- Les distances euclidiennes sont alors calculées comme suit :

$$\begin{aligned}\|\phi(\mathbf{x}_i)-\mathbf{m}_l\|^2 =& k_{ii} + \frac{1}{(\sum_{i:\phi(\mathbf{x}_i)\in\mathbb{C}_l} w_i)^2}\sum_{i:\phi(\mathbf{x}_i)\in\mathbb{C}_l}\sum_{j:\phi(\mathbf{x}_j)\in\mathbb{C}_l} w_iw_jk_{ij}\\ &-2\frac{1}{\sum_{i:\phi(\mathbf{x}_i)\in\mathbb{C}_l} w_i}\sum_{j:\phi(\mathbf{x}_j)\in\mathbb{C}_l} w_jk_{ij}\end{aligned}$$

- Notons $d^2(\phi(\mathbf{x}_i),\mathbf{m}_l)$ ce calcul $\forall\phi(\mathbf{x}_i)\in\mathbb{O}, \forall\mathbf{m}_l : \mathbb{C}_l\in\mathbb{P}(\mathbb{O})$.

Pseudo-code des k-means pondérés à noyaux

Input : $\mathbf{K}$ and k
Initialize $\mathbb{P}(\mathbb{O})$ with k different clusters
While a stopping criterion is not reached **do**
    For all $\mathbf{x}_i\in\mathbb{O}$ **do**
        For all $\mathbb{C}_l\in\mathbb{P}(\mathbb{O})$ **do**
            Compute $d^2(\phi(\mathbf{x}_i),\mathbf{m}_l)$
        End For
        Find $\mathbb{C}_{l^*} = \arg\min_{\mathbb{C}_l\in\mathbb{P}(\mathbb{O})} d^2(\phi(\mathbf{x}_i),\mathbf{m}_l)$
        Move $\mathbf{x}_i$ from its current cluster to $\mathbb{C}_{l^*}$
    End For
End While
Ouput : $\mathbb{P}(\mathbb{O})$

Introduction

- Ce sont des méthodes de clustering développées depuis les années 2000.
- Ces approches permettent de tenir compte de la géométrie intrinsèque des données (espaces courbées).
- Les notions de **graphes de similarités et de voisinage** et de **matrice laplacienne** somt importantes.
- Du point de vue méthodologique, le spectral clustering fait usage de la **décomposition spectrale** afin de représenter les données dans un espace euclidien. Il utilise ensuite les k-**means** afin d'obtenir une partition des données.
- Le spectral clustering englobe d'une certaine façon les méthodes non-supervisées vues précédemment.
- On montre qu'il possède de nombreux liens avec d'autres méthodes de partitionnement de graphe, ce qui explique également les bons résultats obtenus par cette famille de méthode.

Graphe de similarités

- La modélisation employée est celle de **graphe non-orienté pondéré**.

Définition. (Graphe non-orienté pondéré)

Un **graphe non-orienté** *G est défini par la donnée de deux ensembles :*

- $\mathbb{V} = \{v_1, \dots, v_n\}$ *dont les éléments sont appelés* **sommets** *(ou* **noeuds***).* $|\mathbb{V}| = n$ *est le nb de sommets. On dit alors que G est d'ordre n.*
- $\mathbb{E} = \{e_1, \dots, e_m\}$ *dont les éléments sont des paires non-orientées de sommets que l'on appelle* **arêtes***.* $|\mathbb{E}| = m$ *désigne le nombre d'arêtes.*

$G = (\mathbb{V}, \mathbb{E})$ *est* **pondéré** *s'il existe une fonction $f : \mathbb{E} \to \mathbb{R}$ donnant une valuation à toute arête de $\mathbb{E}$.*

Représentation matricielle

Définition. (Matrice d'adjacence)

Soit $G = (\mathbb{V}, \mathbb{E})$ un graphe non-orienté d'ordre n. La **matrice d'adjacence** *de G est une matrice carrée binaire d'ordre n notée* **A***. Son terme général est défini comme suit :*

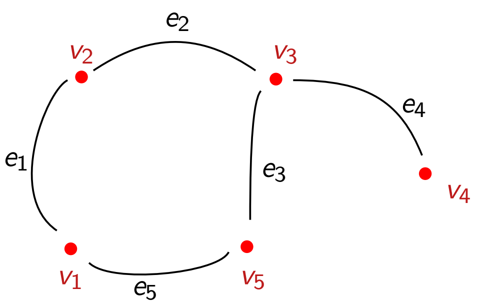
$$a_{ij} = \begin{cases} 1 & si\ (v_i, v_j) \in \mathbb{E} \\ 0 & sinon \end{cases}$$

Si G est pondéré nous noterons plutôt par **W** *sa matrice d'adjacence et dans ce cas, nous avons :*

$$w_{ij} = f(v_i, v_j)$$

- G étant non-orienté, sa matrice d'adjacence est symétrique.

Exemple de matrice d'adjacence



$$\mathbf{A} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

- Si le graphe est pondéré selon les valuations suivantes :

	e_1	e_2	e_3	e_4	e_5
$f(e_j)$	2	3	1	5	6

- On obtient alors la matrice d'adjacence pondérée suivante :

$$\mathbf{W} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & 0 & 0 & 6 \\ 2 & 0 & 3 & 0 & 0 \\ 0 & 3 & 0 & 5 & 1 \\ 0 & 0 & 5 & 0 & 0 \\ 6 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Degré des sommets et matrice des degrés

- Dans ce qui suit, nous supposons que G est non-orienté et pondéré.
- Nous définissant le degré du sommet v_i par :

$$d_i = \sum_{j=1}^{n} w_{ij}$$

- Remarque : comme **W** est symétrique on a aussi $d_i = \sum_{j=1}^{n} w_{ji}$.
- La matrice des degrés est une matrice carrée diagonale d'ordre n notée **D** de terme général :

$$\mathbf{D}_{ij} = \begin{cases} d_i & \text{si } i = j \\ 0 & \text{sinon} \end{cases}$$

Graphe de similarités et de voisinage

- En spectral clustering nous avons les correspondances suivantes :
  - $\mathbb{V} = \mathbb{O} =$ ensemble des objets $\{X_1, \ldots, X_j\}$.
  - $\mathbb{E} =$ ensemble des paires d'objets (X_i, X_j) considérés comme voisins.
  - $w_{ij} =$ mesure de la relation de voisinage pour (X_i, X_j).
- Suppons une fonction s donnant pour chaque paire (X_i, X_j) une mesure de similarité dans $\mathbb{R}$ (une fonction noyau par exemple).
- Soit $\mathbb{NN}_k(X_i)$= ensemble des k objets ayant les plus fortes valeurs de similarité avec X_i.
- **Plusieurs façons de construire W** à partir de s :
  - **Graphe écrêté selon un seuil** $\theta \geq 0$:
  $$w_{ij} = s(X_i, X_j) \text{ si } s(X_i, X_j) \geq \theta$$
  - **Graphe des** k**-plus proches voisins** :
  $$w_{ij} = s(X_i, X_j) \text{ si } X_i \subset \mathbb{NN}(X_j) \vee X_j \subset \mathbb{NN}(X_i)$$
  - **Graphe "connexe"** :
  $$w_{ij} = s(X_i, X_j) \text{ si } s(X_i, X_j) \geq 0$$

Matrice laplacienne d'un graphe

Définition. (Matrice laplacienne)

Soit $G = (\mathbb{V}, \mathbb{E})$ un graphe non-orienté pondéré de matrice d'adjacence **W**. *La matrice laplacienne non normalisée de G, notée* **L**, *est une matrice carré de même ordre que* **W** *qui est définie comme suit :*

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

où **D** *est la matrice des degrés.*

- Exemple :

$$\mathbf{W} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & 0 & 0 & 6 \\ 2 & 0 & 3 & 0 & 0 \\ 0 & 3 & 0 & 5 & 1 \\ 0 & 0 & 5 & 0 & 0 \\ 6 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \rightarrow \mathbf{L} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 8 & -2 & 0 & 0 & -6 \\ -2 & 5 & -3 & 0 & 0 \\ 0 & -3 & 9 & -5 & -1 \\ 0 & 0 & -5 & 5 & 0 \\ -6 & 0 & -1 & 0 & 7 \end{pmatrix} \end{array}$$

Propriétés de la matrice laplacienne

Propriété.

Soit **L** *la matrice laplacienne d'un graphe G non-orienté pondéré d'ordre n alors :*

- *Pour tout vecteur* $\mathbf{f} \in \mathbb{R}^n$ *:*
$$\mathbf{f}^\top \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$
- **L** *est symétrique et semi-définie positive.*
- *La plus petite valeur propre de* **L** *est 0 et son vecteur propre associé est* **1** *(vecteur rempli de 1).*
- **L** *a n valeurs propres réelles, non-négatives :*
$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$$

Propriétés de la matrice laplacienne (suite)

Propriété.

Soit G un graphe non-orienté pondéré d'ordre n dont les valuations sont non-négatives. Alors, l'ordre de multiplicité k de la valeur propre 0 de la matrice laplacienne **L** *est le nombre de composantes connexes du graphe que l'on notera $C_1, \ldots, C_k$. De plus, le sous-espace propre associé à la valeur propre 0 est engendré par les vecteurs indicateurs $\mathbf{1}_{C_1}, \ldots, \mathbf{1}_{C_k}$ de ces composantes.*

- Rappel : une composante connexe est un sous-graphe tels que ses sommets sont connexes.
- Le vecteur indicateur $\mathbf{1}_{C_l}$ appartient à $\{0,1\}^n$ et $[\mathbf{1}_{C_l}]_i = 1$ si $X_i \in C_l$.
- Il est intéressante de noter les liens entre ce résultat issu de la théorie des graphes et la problématique de clustering et notamment le partitionnement en k classes à partir d'une matrice de similarités.

Matrice laplacienne **normalisée** d'un graphe

- En spectral clustering, certaines méthodes utilisent une version normalisée de la matrice laplacienne.

Définition. (Matrice laplacienne normalisée)

Soit $G = (\mathbb{V}, \mathbb{E})$ un graphe non-orienté pondéré de matrice d'adjacence $\mathbf{W}$ et de matrice laplacienne (non-normalisée) $\mathbf{L}$. La matrice laplacienne normalisée de G, notée $\mathbf{L}_{sym}$, est une matrice carré de même ordre que $\mathbf{W}$ qui est définie comme suit :

$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$$

où $\mathbf{D}$ est la matrice des degrés.

- Il existe un autre type de normalisation : $\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L}$ [Shi and Malik, 2000] mais nous ne l'étudierons pas.

Propriétés de la matrice laplacienne **normalisée**

Propriété.

*Soit $\mathbf{L}_{sym}$ la matrice laplacienne **normalisée** d'un graphe G non-orienté pondéré d'ordre n alors :*

- *Pour tout vecteur $\mathbf{f} \in \mathbb{R}^n$:*

$$\mathbf{f}^\top \mathbf{L}_{sym}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$$

- *$\mathbf{L}_{sym}$ est symétrique et sdp et possède n valeurs propres réelles, non-négatives :*

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$$

- *L'ordre de multiplicité k de la valeur propre 0 est le nombre de composantes connexes de G que l'on notera $C_1, \ldots, C_k$. Le sous-espace propre associé à la valeur propre 0 est engendré par les vecteurs $\mathbf{D}^{-1/2}\mathbf{1}_{C_1}, \ldots, \mathbf{D}^{-1/2}\mathbf{1}_{C_k}$.*

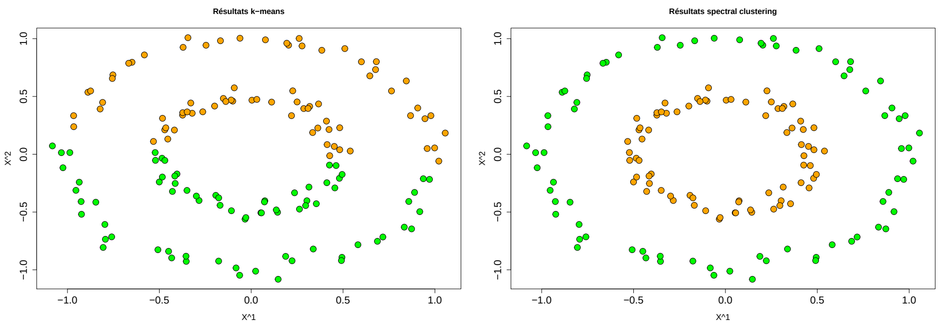
Pseudo-code des méthodes de spectral clustering

1 **Input :** s, méthode pour $\mathbf{W}$, normalisation ou pas, k (nb de classes)
2 Construire le graphe de voisinage $\mathbf{W}$ selon la méthode choisie
3 Construire la matrice laplacienne non-normalisée $\mathbf{L}$ ou normalisée $\mathbf{L}_{sym}$
4 Calculer $\mathbf{f}^1, \ldots, \mathbf{f}^k$, les k premiers vecteurs propres de $\mathbf{L}$ ou $\mathbf{L}_{sym}$
5 Construire $\mathbf{F} = \begin{pmatrix}\mathbf{f}^1 & \ldots & \mathbf{f}^k\end{pmatrix} \in \mathbb{R}^{n\times k}$ la matrice des k vecteurs propres mis en colonne
6 Si $\mathbf{L}_{sym}$ est utilisée, normer les vecteurs lignes de $\mathbf{F}$
7 Utiliser les k-means pour partitionner les n lignes de $\mathbf{F}$
8 **Ouput :** $\mathbb{P}(\mathbb{O})$

Lien entre les différentes méthodes

- Le spectral clustering réduit implicitement les dimensions puis utilise les k-means dans l'espace réduit pour partitionner les objets.
- Si la fonction de similarité est un noyau et si $\mathbf{W}$ est le graphe de voisinage "connexe" alors la méthode spectrale utilisée est clairement liée aux k-means à noyaux.
- Si le graphe de voisinage est limité au voisinage proche alors le spectral clustering va plus loin que les k-means à noyaux puisque $\mathbf{W}$ permet alors d'encoder implicitement la géométrie intrinsèque des données. Dans ce cas, le spectral clustering ne fait donc pas d'hypothèse sur la forme des classes.
- En revanche, il y a plusieurs paramètres à fixer pour la construction de $\mathbf{W}$ et les résultats peuvent être sensibles à ce paramétrage.
- Le spectral clustering est également une approche relaxée du problème de coupe normalisée de graphe dont la version non-normalisée est le dual du problème de flot maximal (Ford-Fulkerson).

Illustration



- Résultats des k-means et du spectral clustering avec un noyau gaussien.

Quelques références I

Abiteboul, S., Bancilhon, F., Bourdoncle, F., Clemencon, S., De La Higuera, C., Saporta, G., and Fogelman Soulié, F. (2014).
L'émergence d'une nouvelle filière de formation : " data scientists ".
Interne, INRIA Saclay.

Alpaydin, E. (2010).
Introduction to Machine Learning.
MIT Press.

Anderberg, M. (1973).
Clustering analysis for applications.
London, Academic Press.

Asuncion, A. and Newman, D. (2007).
UCI machine learning repository.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999).
When is Nearest Neighbor Meaningful ?
In ICDT.

Bishop, C. M. (2006a).
Pattern Recognition and Machine Learning.
Springer.

Bishop, C. M. (2006b).
Pattern recognition and machine learning.
Springer, 1st ed. 2006. corr. 2nd printing edition.

Quelques références II

Breiman, L. (1996).
Bagging predictors.
Machine learning, 24(2) :123–140.

Breiman, L. (2001).
Random forests.
Machine learning, 45(1) :5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984).
Classification and Regression Trees.
Chapman & Hall, New York, NY.

Clark, A., Fox, C., and Lappin, S. (2010).
The Handbook of Computational Linguistics and Natural Language Processing.
Blackwell Handbooks in Linguistics. Wiley.

Cleveland, W. S. (2001).
Data science : An action plan for expanding the technical areas of the field of statistics.

Cornillon, P.-A. and Matzner-Lober, E. (2010).
Régression avec R.

Cornuéjols, A. and Miclet, L. (2003).
Apprentissage artificiel.
Eyrolles.

Quelques références III

Ding, C. and He, X. (2004).
K-means clustering via principal component analysis.
In ICML '04 : Proceedings of the twenty-first international conference on Machine learning, page 29, New York, NY, USA. ACM.

Dreyfus, G. (2008).
Apprentissage Statistique. Réseaux de neurones, Cartes topologiques, Machines à vecteurs supports.
Eyrolles.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004).
Least angle regression.
The Annals of statistics, 32(2) :407–499.

Filippone, M., Camastra, F., Masulli, F., and Rovetta, S. (2008).
A survey of kernel and spectral methods for clustering.
Pattern Recogn., 41(1) :176–190.

Freund, Y., Schapire, R. E., et al. (1996).
Experiments with a new boosting algorithm.
In Icml, volume 96, pages 148–156.

Friedman, J., Hastie, T., and Tibshirani, R. (2010).
Regularization paths for generalized linear models via coordinate descent.
Journal of statistical software, 33(1) :1.

Gan, G., Ma, C., and Wu, J. (2007).
Data Clustering : Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability).
SIAM, Society for Industrial and Applied Mathematics, illustrated edition edition.

Quelques références IV

Han, J. and Kamber, M. (2006).
Data Mining Concepts and Techniques.
Morgan Kaufmann.

Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. (2004).
The entire regularization path for the support vector machine.
Journal of Machine Learning Research, 5(Oct) :1391–1415.

Hastie, T., Tibshirani, R., and Friedman, J. (2011).
The Elements of Statistical Learning.
Springer.

Horn, R. and Johnson, C. (1985).
Matrix analysis.
Cambridge University Press.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999).
Data clustering : a review.
ACM Comput. Surv., 31(3) :264–323.

Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014).
Mining of massive datasets.
Cambridge University Press.

Manning, C. D. and Schütze, H. (1999).
Foundations of Statistical Natural Language Processing.
MIT Press.

Quelques références V

Mirkin, B. (1996).
Mathematical classification and clustering.
Kluwer academic press, Dordrecht.

Mitchell, T. (1997).
Machine Learning.
McGraw Hill.

Ng, A. Y., Jordan, M. I., Weiss, Y., et al. (2001).
On spectral clustering : Analysis and an algorithm.
In NIPS, volume 14, pages 849–856.

Pelleg, D. and Moore, A. (2000).
X-means : Extending k-means with efficient estimation of the number of clusters.
In In Proceedings of the 17th International Conf. on Machine Learning, pages 727–734. Morgan Kaufmann.

Quinlan, J. (1986).
Induction of decision trees.
Machine Learning, 1(1) :81–106.

Quinlan, J. R. (1993).
C4.5 : programs for machine learning.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rokach, L. (2010).
Ensemble-based classifiers.
Artificial Intelligence Review, 33(1-2) :1–39.

Quelques références VI

Saporta, G. (2006).
Probabilités, analyses des données et statistiques.
Editions Technip.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998).
Nonlinear component analysis as a kernel eigenvalue problem.
Neural computation, 10(5) :1299–1319.

Scholkopf, B. and Smola, A. J. (2001).
Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond.
MIT Press, Cambridge, MA, USA.

Shi, J. and Malik, J. (2000).
Normalized cuts and image segmentation.
IEEE Transactions on pattern analysis and machine intelligence, 22(8) :888–905.

Valiant, L. G. (1984).
A theory of the learnable.
Communications of the ACM, 27(11) :1134–1142.

Vapnik, V. N. (1995).
The nature of statistical learning theory.
Springer-Verlag New York, Inc., New York, NY, USA.

Vert, J., Tsuda, K., and Schölkopf, B. (2004).
A primer on kernel methods.
In Schölkopf, B., K. T. and Vert, J., editors, Kernel Methods in Computational Biology, pages 35–70, Cambridge, MA, USA. MIT Press.

Quelques références VII

Von Luxburg, U. (2007).
A tutorial on spectral clustering.
Statistics and computing, 17(4) :395–416.

Xu, R. and Wunsch, D. I. I. (2005).
Survey of clustering algorithms.
IEEE Transactions on Neural Networks, 16(3) :645–678.

Zelnik-Manor, L. and Perona, P. (2005).
Self-tuning spectral clustering.
In Advances in Neural Information Processing Systems 17.

Zou, H. and Hastie, T. (2005).
Regularization and variable selection via the elastic net.
Journal of the Royal Statistical Society : Series B (Statistical Methodology), 67(2) :301–320.