

Big Data Mining

Apprentissage Supervisé

Master 2 BIBD et Master 2 SISE

Guillaume Metzler

guillaume.metzler@univ-lyon2.fr



**INSTITUT
de la
communication**



Université de Lyon, Lyon 2, ERIC EA3083, Lyon, France

Automne 2020


Introduction

Plan du cours

Deux séances de CM

- Une introduction au (big) data mining
- Classification non supervisée
- Classification supervisée
- Réseaux de neurones

Cinq séances de TD

- Mise en pratique des différentes méthodes sous le logiciel 
- Apprentissage déséquilibré et exemple d'application industrielle
- Projet à effectuer sur des données réelles

Plan du cours

Ambitions

- Vous présentez un large champ d'algorithmes : Arbres, Forêts aléatoires, k -NN, Boosting, Analyse Discriminante, Réseaux de neurones en tout genre, k-means, one class SVM - local outlier factor
- Vous faire un projet sur des données réelles dans un contexte particulier.
- Vous présentez, pour ceux qui le veulent, comment établit des résultats (garanties théoriques) sur des algorithmes simple en machine learning.

Volonté

- Vous faire participer à la conception du cours.
- Répondre à vos attentes si jamais vous en avez.

(Big) Data Mining

Qu'est-ce que le data mining ?

Le data mining, ou fouille de données, a pour objectif d'extraire de la connaissance à partir d'un ensemble de données (structurées) ou non.

Cette extraction a pour but d'acquérir des connaissances sur des sujets particuliers définis par l'utilisateur. Elle peut permettre la compréhension d'un phénomène ou encore la réalisation de certaines tâches (comme de la classification - régression).

L'extraction de connaissances se fait au moyen d'outils statistiques qui sont à la base d'extraction de connaissances et qui pourront servir d'outils d'aide à la décision.

Evolution du data mining

Historiquement, l'analyse de données se faisait sur une quantité de données très faibles mais aussi présentant un faible nombres de variables (features). Les outils informatiques actuels n'étant pas à disposition des scientifiques de l'époque.

Avec le développement constant de la capacité de stockage des données de la part des entreprises, on commence à se rendre compte de l'importance (marketing, économique) que peuvent représenter les données. Les sociétés s'y intéressent d'avantage et n'hésite pas à accroître des moyens de **stockage** et d'**analyse** de ces données, on peut alors parler d'émergence du **data mining**.

Evolution du data mining

Les capacités de stockage ne cessent d'augmenter (loi de Moore) et on commence à étudier une variété plus importante de données, comme les données génomiques dont le coût d'acquisition est très important mais qui ont la particularité de présenter un très grand nombres de variables → **statistiques en grande**.

Ce n'est que depuis peu de temps que nous sommes maintenant capables d'analyser des données en quantités importantes mais aussi avec un grand nombre de variables : **graphes** avec l'émergence des réseaux sociaux. Ces avancées poussent au développement à de nouveaux outils d'analyses et de traitements de plus en plus sophistiqués.

Les étapes du Data Mining

Le data mining se compose de 4 étapes principales

- Collecte et intégration des données
- Pré-traitement des données
- Analyse des données : régression, classification, segmentation
- Validation du protocole expérimentale et test

Collecte et intégration des données

- Il s'agit avant tout de définir l'objet d'étude ainsi que l'objectif de l'étude afin de définir quelles sont les informations à récolter auprès de la population.
- Définir un protocole pour récolter les données
- Intégrer l'ensemble des données récoltées afin de les mettre dans des bases de données. Il faudra aussi définir la forme de la base de données et voir comment stocker ses données (mais ce n'est pas l'objet de ce cours → cf. cours de bases de données).

Pré-traitement des données

Il s'agit d'effectuer un premier traitement des données afin d'en faciliter l'analyse, soit par une phase exploratoire ou par des premières transformations

- Visualisation des données
- Analyse de corrélations
- Etudes des tendances de chaque variable : moyenne, variance
- Voir quelles sont les tendances dans le jeu de données avec des premières analyses (clustering, ACP)

Transformation - Gestion des données manquantes/aberrantes

- Supprimer les données atypiques qui peuvent fausser une analyse
- Estimer ou supprimer les valeurs manquantes
- Normalisation des données (peut jouer un rôle majeur dans la suite !)
- Transformation des variables

Analyse des données

C'est là l'objet de ce cours ... donc on va préserver un peu de suspense pour la suite.

Validation des modèles

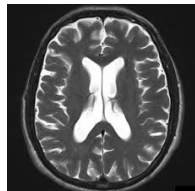
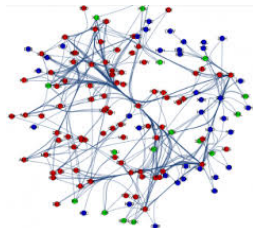
C'est également une étape très importante. Elle va permettre, à l'aide de données non utilisées pour l'analyse, de valider les modèles appris et ainsi de vérifier que nos analyses et résultats sont fondés.

En tant que data analyst et pour les applications concrètes, il est toujours intéressant d'interpréter ces résultats.

Applications



Nature des données



105	12,7	1,2	0,00	10,15	1,2
104	12,7	1,2	0,00	11,89	0,5
103	10,4	11,8	0,1	1,15	15,78
102	12,6	10,3	0,3	0,00	16,31
101	16,6	11,8	1,1	-0,06	10,56
100	13,2	1,9	-0,03	11,89	1,8
99	15	16,9	0,9	0,00	12,81
98	18,7	0,4	0,12	10,92	0
97	10,1	1,7	0,04	11,83	0



Pourquoi Big Data ?

La notion de big data comprend essentiellement trois caractéristiques :

Volume : les données sont présentes en quantité massives, mais le seul nombre ne suffit pas à les caractériser. Ces données présentent également un nombre important de variables/descripteurs/features : les données génomiques

Pluralité : les données peuvent prendre plusieurs formes, il peut s'agir de textes, de tableaux de chiffres, d'images voir un mélange de toutes ces données

Vitesse d'acquisition : outre le volume important des données, il est important de voir que ces quantités massives arrivent de façon continue et extrêmement rapide dans le temps. Cela nécessite de mettre en place une gestion fine des bases de données et mise à jour perpétuelle des algorithmes d'analyse.

Classification non supervisée

Programme

On va aborder cela en deux temps

Clustering

- k-means et k-médoïdes
- clustering hiérarchique

Classification - détection d'anomalies

- One Class SVM

Notations

Dans toute cette partie, nous allons considérer des données numériques de taille $m \times p$ (individus \times features), *i.e.*, elles se présenteront sous la forme

$$\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,p-1} & x_{1,p} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,p-1} & x_{2,p} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_{m,1} & x_{m,2} & x_{m,3} & \cdots & x_{m,p-1} & x_{m,p} \end{pmatrix}$$

On notera également y_i la variable "réponse" associée à la donnée \mathbf{x}_i , il peut s'agit d'un groupe dans le cas du clustering, d'une étiquette ou d'un ensemble d'étiquettes quand on parlera de classification. Finalement il peut aussi s'agir d'un nombre réel dans le cas de la régression.

Clustering

Objectif du clustering : déterminer un partitionnement de nos données, *i.e.* assigner à chaque individu x_i un groupe d'appartenance y_i . Ces groupes pourront ainsi être interprétés mais aussi représentés par un seul individu (moyenne de chaque groupe)

On pourra, avec ce type d'algorithme, apprendre une représentation simplifiée de nos données (pratique si on souhaite réduire la quantité de données à analyser !)

Cela permet également de mettre en exergue différents profils présents dans nos données, ce qui peut se révéler très important en marketing pour orienter les publicités.

Clustering

Fonctionnement : il repose sur une seule et unique chose fondamentale : la notion de **distance**.

D'ailleurs ... pouvez-vous me rappeler comment est définie une distance ?

On peut utiliser n'importe quelle distance pour cet algorithme. Bien évidemment, la distance utilisée va conditionner le clustering :

$$d_p(\mathbf{x}, \mathbf{x}')^p = \sum_{j=1}^p |x_j - x'_j|^p.$$

On prendra classiquement la distance euclidienne, *i.e.* $p = 2$.

Clustering

Supposons maintenant que l'on dispose maintenant deux partitions Y_1 et Y_2 de nos données. Comment savoir si ces deux partitions sont différentes ou non. Pour cela on calcule le **Rand Index** RI ,

$$RI = \frac{a + d}{a + b + c + d},$$

où

	groupées dans Y_2	séparées dans Y_2
groupés dans Y_1	a	b
séparés dans Y_1	c	d

Question : quelle est la valeur de $a + b + c + d$?

Le **Rand Index** est une mesure de similarité entre deux partitionnements.

Clustering

Prenons un exemple pour voir ce que cela donne. Supposons qu'un algorithme de clustering que l'on aurait itéré trois fois nous donne les partitions suivantes :

- $Y_1 = \{1, 1, 2, 2, 1, 2\}$
- $Y_2 = \{2, 2, 1, 2, 2, 1\}$
- $Y_3 = \{1, 1, 1, 1, 2, 2\}$

Quel partitionnement est le plus proche de Y_1 ?

Clustering

Prenons un exemple pour voir ce que cela donne. Supposons qu'un algorithme de clustering que l'on aurait itéré trois fois nous donne les partitions suivantes :

- $Y_1 = \{1, 1, 2, 2, 1, 2\}$
- $Y_2 = \{2, 2, 1, 2, 2, 1\}$
- $Y_3 = \{1, 1, 1, 1, 2, 2\}$

Quel partitionnement est le plus proche de Y_1 ?

On calcule $RI(Y_1, Y_2)$ et $RI(Y_1, Y_3)$ sachant que $a + b + c + d = 15$.

- $RI(Y_1, Y_2) = 10/15 = 2/3$
- $RI(Y_1, Y_3) = 6/15 = 2/5$

Clustering : K-means

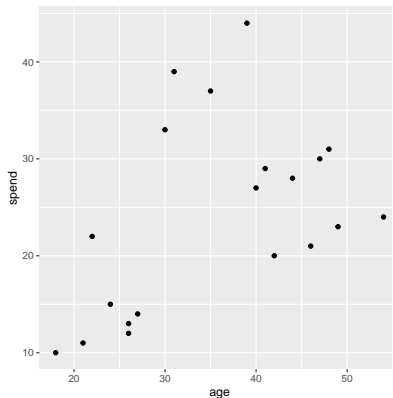
Il s'agit donc d'un algorithme de partitionnement des données qui fonctionnent de la façon suivante :


- On commence par fixer le nombre K de cluster que l'on souhaite créer, *i.e.* fixer le nombre de groupes dans lesquels sont répartis les individus.
- On tire aléatoirement un vecteur moyenne μ_k pour chaque groupe.
- Tant que les groupes ne sont pas stables :
 - 1) assigner chaque exemple \mathbf{x}_i à son centre de groupe le plus proche μ_k
 - 2) mettre à jour μ_k :

$$\mu_k = \frac{1}{m_k} \sum_{i \in I_k} \mathbf{x}_i$$

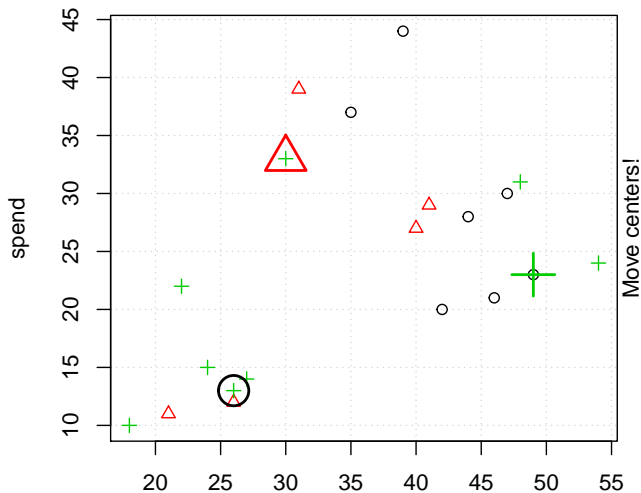
Clustering : K-means

Regardons, ce que cela donne sur un exemple pour mettre en avant la construction.

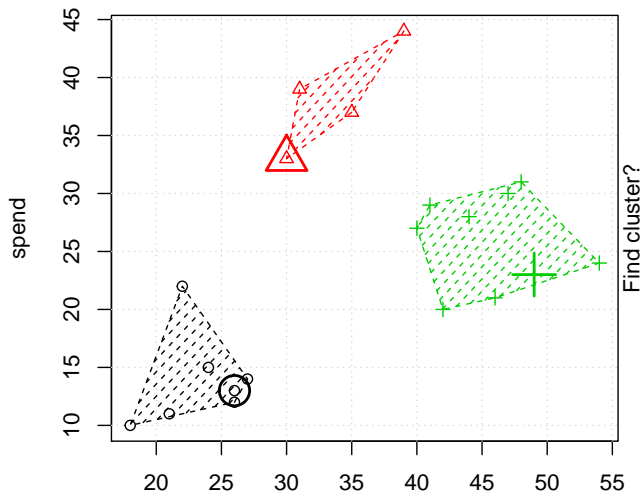


Pourriez-vous identifier rapidement les clusters présents dans ce jeu de données ? (on utilisera *kmeans* et *animation* de )

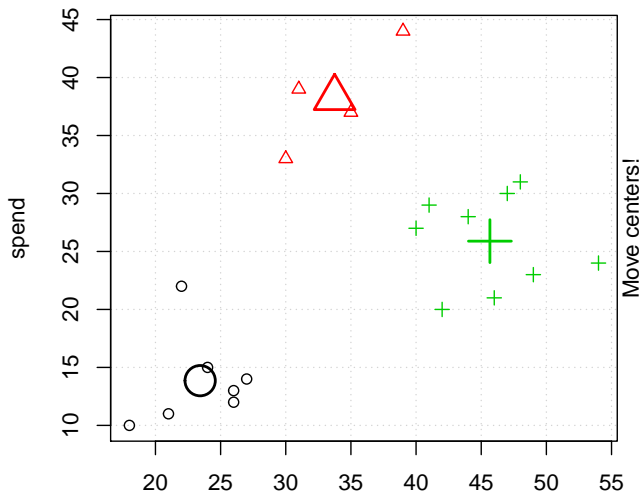
Clustering : K-means



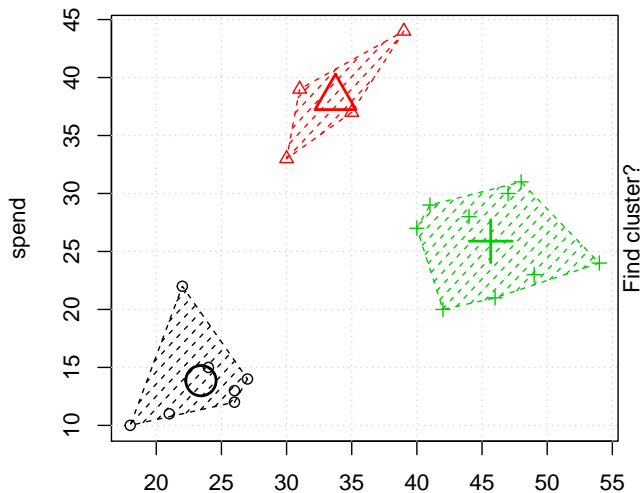
Clustering : K-means



Clustering : K-means



Clustering : K-means



Clustering : K-means

Objectif : l'algorithme k-means cherche à résoudre le problème d'optimisation suivant :

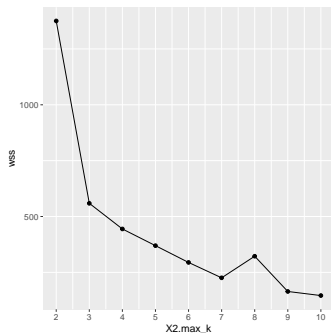
$$\arg \min_{\mathcal{P}} \sum_{k=1}^K \sum_{i \in \mathcal{P}_k} \|\mathbf{x}_i - \mu_k\|,$$

i.e. il cherche à minimiser l'**inertie (ou la variance)** intra-classe, c'est à dire la variance au sein des différents groupes.

Convergence : l'algorithme converge mais uniquement vers un optimum local, il n'y a pas de garantis que la solution soit optimale !

Clustering : K-means

Choix du nombre de classes : il est évident que plus le valeur de K est grande, plus la valeur de notre problème d'optimisation sera faible. Alors en pratique, pour choisir la valeur de K , on va rechercher un coude dans la graphe suivant :





Clustering : K-means

Exemple

Appliquer l'algorithme k-means, pour $k=2$, sur les données suivantes :

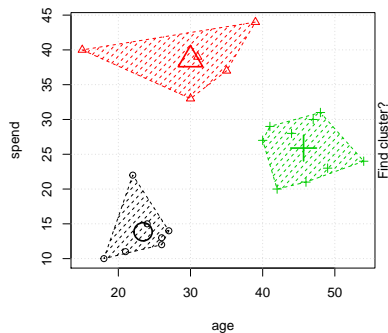
$x_1 = 0, x_2 = 3, x_3 = 7, x_4 = 10, x_5 = 1$ et $x_6 = 8$.


Vérifiez ensuite votre résultat à l'aide du logiciel .

Faites de même avec le jeu de données iris sur . Comparer le clustering obtenu à l'espèce de fleurs de chaque exemple.

Clustering : variante

Un inconvénient de l'algorithme k -means et sa sensibilité aux outliers, *i.e.* aux données atypiques, qui ont un comportement qui ne ressemblent pas à celui des autres données.



Variante moins sensible k -médoides également disponible sous .

Clustering hiérarchique

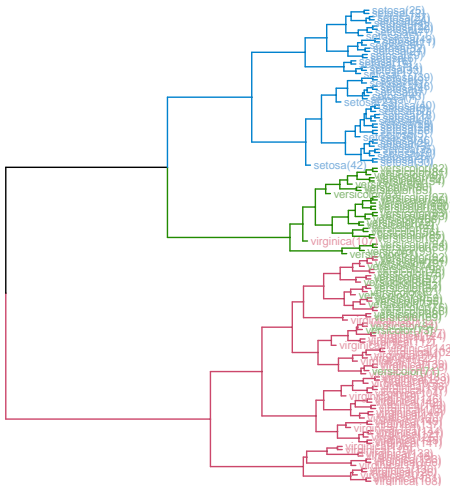
Nous avons vu comment regrouper un ensemble de points en K groupes distincts, lorsque K est fixé par l'utilisateur. Il existe un autre algorithme de clustering appelé **clustering hiérarchique (ascendant)** qui consiste à regrouper **successivement** les observations les plus proches.

Algorithme :

- On considère notre jeu de données avec m observations et on définit une distance D
- Tant que tous les points ne sont pas reliés :
 - 1) calculer les distances deux à deux entre chaque classe à l'aide de d
 - 2) regrouper les deux classes les plus proches au sens de d

Clustering hiérarchique

Clustered Iris data set
(the labels give the true flower species)



Clustering hiérarchique

Quel(s) critère(s) employer pour regrouper nos données ?

Il faudra à nouveau faire le choix d'une distance d qui va permettre de calculer la dissimilarité entre nos groupes. On aura ensuite plusieurs critères possibles, considérons deux groupes G_1 et G_2

- Complete linkage :

$$\max\{d(\mathbf{x}_1, \mathbf{x}_2) \mid \mathbf{x}_1 \in G_1, \mathbf{x}_2 \in G_2\}$$

- Single linkage :

$$\min\{d(\mathbf{x}_1, \mathbf{x}_2) \mid \mathbf{x}_1 \in G_1, \mathbf{x}_2 \in G_2\}$$

- Weighted linkage :

$$\frac{1}{|G_1||G_2|} \sum_{\mathbf{x} \in G_1} \sum_{\mathbf{x}' \in G_2} d(\mathbf{x}_1, \mathbf{x}_2)$$

Clustering hiérarchique

Quel(s) critère(s) employer pour regrouper nos données ?

Il existe encore bien d'autres critères pour faire la liaison entre deux groupes

- Ward Criterion
- Minimum Energy Clustering
- Sum of Intra-Variance

Un petit exemple à tester

```
1 d = dist(USArrests)
2 hc <- hclust(d, method = "complete")
3 plot(hc, hang = -1)
4 cluster <- cutree(hc, k = 3)
```

Comparer les résultats de k -means et d'un clustering hiérarchique sur le jeu de données iris.

Clustering hiérarchique


Exemple

Réaliser un clustering hiérarchique sur les données suivantes :

$x_1 = 0, x_2 = 3, x_3 = 7, x_4 = 10, x_5 = 1$ et $x_6 = 8$.

1) avec le critère de *complete linkage*

2) avec le critère de *single linkage*

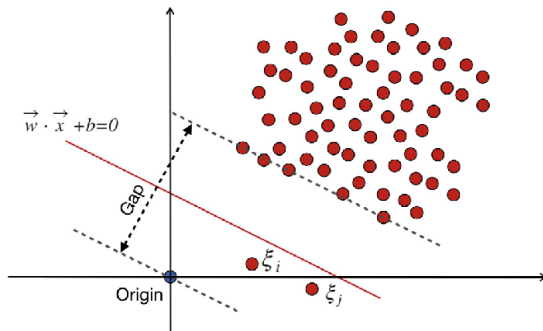
Vérifiez ensuite votre résultat à l'aide du logiciel 

Représenter le dendrogramme associé à chacun des résultats et comparer les résultats avec un k -means, pour $k = 2$.

One Class SVM

On a vu deux méthodes de clustering, *i.e.* de regroupement des données. Regardons maintenant une méthode que l'on appelle les One Class SVM, qui permettent de faire de la détection d'anomalies.

On peut voir cette tâche comme un problème de classification binaire mais non supervisé !



One Class SVM

Vous reconnaissez visuellement l'algorithme bien connu des *Support Vector Machine* ou *Séparateurs à Vastes Marges*.

L'objectif est très différent de ce que nous avons vu jusqu'à présent. Il s'agit ici de séparer les individus en deux classes avec d'un côté, les exemples "normaux" et d'un autre côté les exemples "aberrants" ou "anormaux"

Cela peut notamment servir, quand on ne peut le voir visuellement, à supprimer des données qui ont une distribution qui ne coïncide pas avec la distribution de la majorité des données.

Mais regardons déjà un problème plus ancien ... vraiment ancien ... le problème du cercle minimum (Sylvester, 1857)

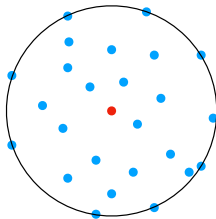
One Class SVM

Formulation du problème

Etant donné un ensemble de m points $S = \{\mathbf{x}_i\}_{i=1}^m$ non étiquetés, trouvez le centre \mathbf{c} et le rayon R de la plus petite sphère contenant l'ensemble des points de S .

On peut résoudre cette tâche en résolvant le problème d'optimisation suivant :

$$\begin{aligned} \min_{\mathbf{c}, R} \quad & R^2, \\ \text{s.t.} \quad & \|\mathbf{x}_i - \mathbf{c}\|^2 \leq R^2, \forall i = 1, \dots, m, \end{aligned}$$



One Class SVM

Formulation équivalente

Les deux formulations suivantes sont équivalentes (Elzinga and Hearn, 1972) :

$$\begin{aligned} \min_{\mathbf{c}, R} \quad & R^2, \\ \text{s.t.} \quad & \|\mathbf{x}_i - \mathbf{c}\|^2 \leq R^2, \quad \forall i = 1, \dots, m, \end{aligned}$$

et

$$\begin{aligned} \min_{\rho, \mathbf{w}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 - \rho, \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{x}_i \geq \rho + \frac{1}{2} \|\mathbf{x}_i\|^2, \quad \forall i = 1, \dots, m, \end{aligned}$$

où $\rho = \frac{1}{2} (\|\mathbf{c}\|^2 - R^2)$ et $\mathbf{c} = \mathbf{w}$

Exercice : montrer l'équivalence entre les deux formulations

One Class SVM

Cette dernière formulation est plus connue sous le nom de **SVDD** : **Support Vector Data Description** (Tax and Duin, 1999, 2004)

$$\begin{aligned} \min_{\rho, \mathbf{w}} \quad & \frac{1}{2} \|\mathbf{w}\|^2 - \rho, \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{x}_i \geq \rho + \frac{1}{2} \|\mathbf{x}_i\|^2, \quad \forall i = 1, \dots, m, \end{aligned}$$

Dans le cas où l'on suppose toutes nos données \mathbf{x}_i de norme constante, *i.e.* $\|\mathbf{x}_i\|^2 = \beta$, on retrouve alors la formulation des **One Class SVM** (Scholkopf and Smola, 2001)

$$\begin{aligned} \min_{\mathbf{w}, \rho'} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 - \rho', \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{x}_i \geq \rho', \quad \forall i = 1, \dots, m, \end{aligned}$$

$$\text{où } \rho' = \rho + \frac{1}{2} \|\mathbf{x}_i\|^2.$$

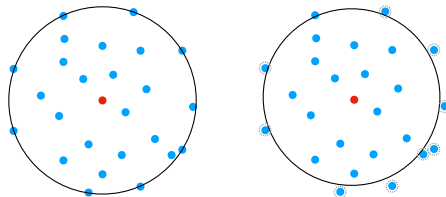
One Class SVM

Vers un problème de détection d'anomalies

$$\begin{aligned} \min_{\mathbf{c}, R} \quad & R^2 + \frac{C}{m} \sum_{i=1}^m \xi_i, \\ \text{s.t.} \quad & \|\mathbf{x}_i - \mathbf{c}\|^2 \leq R^2 + \xi_i, \quad \forall i = 1, \dots, m, \\ & \xi_i \geq 0, \quad \forall i = 1, \dots, m. \end{aligned}$$

Ajout de variables dites "**slacks**" qui prennent en compte les erreurs de l'algorithme, *i.e.* on autorise certains points à se trouver en dehors du cercle.

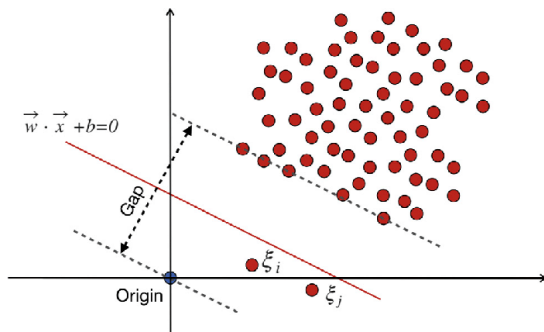
Les données qui se trouvent en dehors du cercle sont considérés comme des anomalies



One Class SVM


Problème d'optimisation du One Class SVM associé

$$\begin{aligned} \min_{\mathbf{w}, \rho'} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{1}{m\mu} \sum_{i=1}^m \xi_i \rho', \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{x}_i \geq \rho' - \xi_i, \quad \forall i = 1, \dots, m, \\ & \xi_i \geq 0, \quad \forall i = 1, \dots, m. \end{aligned} \tag{1}$$



Map-reduce

Retour au calcul parallèle

Un petit exemple rapide pour vous rappeler/montrer comment faire du calcul parallèle sous .

```
1
2 # Les librairies a telecharger
3 library(doParallel)
4 library(foreach)
5 library(iterators)
6
7 # Initialisation du calcul parallele
8 no_cores <- detectCores()
9 cl <- makeCluster(no_cores)
10 registerDoParallel(cl)
11
12 # Une boucle tres simple
13 res <- foreach(i = 1:10000, .combine = 'c', .multicombine=TRUE) %dopar% {
14   sqrt(i)
15 }
```


Retour au calcul parallèle

Paralléliser ses calculs est une bonne chose si l'on souhaite effectuer plusieurs tâches en même temps, mais cela est à éviter pour des tâches simples.

Regarder ce que cela donne en effectuant les deux calculs suivants :

```
1 # Une boucle "for" simple
2 system.time(
3   for (i in seq(1:10000)) {
4     sqrt(i)
5   }
6 )
7
8 no_cores <- detectCores()
9 cl <- makeCluster(no_cores)
10 registerDoParallel(cl)
11
12 # Une boucle foreach avec dopar
13 system.time(
14   foreach(i = 1:10000, .combine = 'c') %dopar% {
15     sqrt(i)
16   }
17 )
18 stopCluster(cl)
```

Le regroupement des résultats (ici concaténation) est une opération qui nécessite du temps !

Retour au calcul parallèle

Prenons maintenant un exemple un peu plus complexe pour voir l'intérêt de la parallélisation.

```
1 # On commence par definir une fonction qui tire un echantillon et determine l'ecart-type
  de ce dernier
2 myfun <- function(r, mean = 0, sd = 1) {
3   x <- rnorm(r, mean = mean, sd = sd)
4   s <- sd(x)
5   return(s)
6 }
7
8 # Une application simple avec la fonction sapply
9 system.time(
10   resultats$res_non_par <- sapply(r_values, FUN = myfun,
11                                   mean = 5, sd = 10)
12 )
13
14 # Une application en utilisant clusterApply
15 cl <- makeCluster(detectCores()-1)
16 registerDoParallel(cl)
17 system.time(
18   res_par <- clusterApply(cl, r_values, fun = myfun,
19                             mean = 5, sd = 10)
20 )
21 stopCluster(cl)
```

Retour au calcul parallèle


L'inconvénient de ce type de méthode est la nécessité de transférer l'ensemble des données à l'ensemble des processeurs de calculs.

Les données sont donc dupliquées autant de fois qu'il y a de processeurs de calcul, ce qui peut rapidement se révéler problématique lorsque nous avons une quantité importante de données à traiter.

Une solution intéressante serait de distribuer les calculs ... mais aussi les données ! Cela éviterait tout problème de saturation de la mémoire.

Map Reduce

Objectif

- utiliser le package *rmr2* pour simuler des gestions de fichiers HDFS sans Hadoop
- écrire des fonctions Map et Reduce dans 

Installation

- *rmr2* n'est pas disponible sur CRAN, il faudra passer par GitHub pour le télécharger
- l'installation peut également nécessiter d'autres packages comme devtools qu'il faudra installer

Chargement

```
1 library(rmr2)
2 # Pour utiliser tout cela en local uniquement
3 rmr.options(backend = "local")
```

Map Reduce : quelques exemples

```
1 library(rmr2)
2 rmr.options(backend = "local")
3
4 # assignation (clef, valeurs)
5 keyval(1, 1:10)
6
7 #creation d'un objet big data
8 objetBG = to.dfs(keyval(1,1:10))
9
10 # transformation d'un objet big data, en un objet R
11 objetR = from.dfs
12
13 # Visualisation de la valeur des objets uniquement
14 values(objetR)
```

Map Reduce : quelques exemples

Regardons ensemble quelques exemples. On commence par deux exemples simples qui consistent à calculer le carré d'un entier, ou la somme des carrés d'entiers

```
1 # Input
2 small.ints = to.dfs(1:10)
3
4 # A. Carres d'entiers
5 calsquare = mapreduce(
6   input = small.ints,
7   map = function(k, v) cbind(v,v^2),
8   # notez que par default, la fonction reduce est la fonction NULL
9 )
10 results = from.dfs(calsquare)
11 results
12
13 # B. Somme des carres d'entiers
14 calsumsquare = mapreduce(
15   input = small.ints,
16   map = function(k, v) keyval (1,v^2) # on associe la meme clef a tout le monde
17   reduce = function(k,vv) sum(vv) # on fait la somme des elements
18 )
19 results = from.dfs(calsumsquare)
20 values(results)
```

Map Reduce : quelques exemples

Regardons un exemple un peu différent qui consiste à compter le nombre de fois où une valeur apparaît dans un échantillon

```
1 # Input : on tire aleatoirement des nombres selon une loi binomiale de parametre "prob"
2 input = rbinom(32, n =50, p = 0.4)
3
4 # Il ne reste qu'a ecrire les fonctions pour compter le nombre d'occurrences
5 groupe = to.dfs(input)
6 comptage = mapreduce(
7   input = input,
8   # a chaque entier dans "groupe" on associe une paire (entier ,1)
9   map = function (., v) keyval (v, 1))
10  # par default, les clefs de meme valeur sont associes dans une liste ou un vecteur
11  # pour une valeur de clef donnee, reduce compte le nombre de 1
12  reduce = function (k, vv) keyval (k, length(vv))
13 )
```

N'hésitez pas à commenter la fonction *reduce* de ce qui précède afin de regarder ce que donne la fonction *map*.

Map Raduce : K-means

On commence par écrire nos fonctions *map* et *reduce* qui seront appliquées à chaque itération.

```
1 # On commence par ecrire notre fonction principale
2 kmeans.mr = function(data, num.cluster, num.iter){
3   # fonction a definir pour calculer la distance euclidienne entre les centres et les
      donnees
4   dist.fun = function(C, data){
5     apply(C, 1, function(x) colSums( (t(data) - x)^2))
6   }
7   # fonction map
8   km.map = function(., P){
9     nearest = {
10       # on commence par l'initialisation des centres
11       if (is.null(C))
12         sample(1:num.clusters, nrow(data), replace = TRUE)
13       # sinon on recherche les centre les plus proches
14       else{
15         D = dist.fun(C, data)
16         nearest = max.col(-D)
17       }
18     }
19     keyval(nearest, data)
20   }
21   km.reduce = function(., G) t(as.matrix(apply(G, 2, mean)))
22   # a suivre ...
```


Map Raduce : K-means

Il ne reste plus qu'à écrire la boucle à effectuer

```
1 # ... suite
2 C = NULL
3 for(i in 1:num.iter){
4   res = from.dfs( mapreduce(data, map = km.map, reduce = km.reduce))
5   C = values(res)
6   # si des centres disparaissent
7   if (nrow(C) < num.clusters) {
8     C = rbind(C, matrix(rnorm((num.clusters - nrow(C))*nrow(C)),
9                           ncol = nrow(C)) %% C)
10  }
11 }
12 }
```


Il ne reste plus qu'à tester tout cela sur un exemple.

Map Raduce : K-means

Pour cela, on va simuler des données de façon aléatoire

```
1 # Simulation de 5 centre + bruit gaussien
2
3 set.seed(1)
4 data = do.call(rbind, rep(list(matrix(rnorm(10,sd=10), ncol=2)), 20)) + matrix(rnorm(200)
5     , ncol = 2)
6 # Il ne reste plus qu'a tester !
7
8 out = kmeans.mr(to dfs(data),
9     num.clusters = 5,
10    num.iter = 8
11 )
12 plot(P)
13 points(out, col = "blue", pch =16)
```

Map Reduce

Pour un petits tutoriels sur l'utilisation de *Map Reduce* sous  (je me suis servi de ce tutoriel pour cette partie)

<https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-tutor5-R-mapreduce.pdf>

<https://github.com/RevolutionAnalytics/rmr2/blob/master/docs/tutorial.md>
ou encore le lien suivant pour le calcul matriciel

<https://lendap.wordpress.com/2015/02/16/matrix-multiplication-with-mapreduce/>

References I

- Elzinga, D. J. and Hearn, D. W. (1972). The minimum covering sphere problem. *Management Science*, 19(1):96–104.
- Scholkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.
- Sylvester, J. (1857). A question in the geometry of situation. *Quarterly Journal of Pure and Applied Mathematics*.
- Tax, D. M. J. and Duin, R. P. W. (1999). Data domain description using support vectors. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 251–256.
- Tax, D. M. J. and Duin, R. P. W. (2004). Support vector data description. *Machine Learning Journal*, 54(1):45–66.