Une introduction à l'Apprentissage Statistique

7 janvier 2015

Première partie

Vocabulaire et notions préliminaires

1 Qu'est-ce que l'Apprentissage Statistique ?

Les problèmes que nous serons amenés à traiter en Machine Learning concernent l'interprétation de données : on nous donne des caractéristiques $X_1, \ldots, X_p$ – que l'on va appeler prédicteurs ou *features*– et on nous demande de prédire ou d'inférer une fonction $Y = f(X)$ à partir de ces données. En pratique la sortie Y qui nous intéresse est bruitée, c'est-à-dire que $Y = f(X) + \varepsilon$, donc nous ferons toujours une petite erreur dans notre prédiction mais nous devrons tout de même travailler à réduire l'erreur entre notre estimation $\hat{f}(X)$ et la vraie valeur de la fonction inconnue $f(X)$.

1.1 Exemple 1 : la recommandation de contenu

On peut voir le problème de la recommandation de contenu comme celui de la prédiction de la note que va donner l'individu i au contenu c. On décrit alors l'individu i par un ensemble de *features* $X_1^i, \ldots, X_p^i$ (typiquement son âge, son adresse, sa profession, etc.) et on caractérise le contenu c par d'autres *features* $X_1^c, \ldots, X_q^c$ (typiquement la catégorie du produit, sa marque, son prix, etc.) et on cherche à prédire

$$Y = f(X_1^i, \ldots, X_p^i, X_1^c, \ldots, X_q^c)$$

où Y peut par exemple représenter la note ou un autre indice d'utilité (achat ou non) auquel on pourrait avoir accès. A partir des données du vendeurs, qui sont des valeurs de cette fonction en certains points de l'espace, on souhaite être capable d'avoir la meilleure estimation possible de f.

1.2 Exemple 2 : Analyse d'un phénomène physique

Cette fois, on suppose que l'on observe un phénomène physique que l'on peut quantifier par Y et que les différentes variables que l'on contrôle sont $X_1, \ldots, X_p$. On souhaite connaître l'influence de chaque variable X_i sur la sortie Y. On peut envisager que certaines variables aient des interactions fortes ou des corrélations que l'on n'observe pas directement mais que l'on souhaite détecter. On appelle ce problème *l'inférence*.

2 Apprentissage supervisé et non-supervisé

Deux cas de figures très différents peuvent se présenter en apprentissage :

— D'une part on peut vous présenter deux ensembles de données : un dont on connaît la sortie Y – données étiquetées – et un autre dont on n'a que les *features* $X_1, \ldots, X_p$. On vous demande alors d'apprendre votre modèle $\hat{f}$ sur les données étiquetées et de le tester sur les données dont l'étiquette est inconnue. On parler alors d'apprentissage supervisé.

— D'autre part on peut vous donner un ensemble de N points $(X_1^i, \ldots, X_p^i)_{i=1}^N$ dont on ne connaît rien a priori. On ne cherche pas nécessairement une sortie particulière Y mais plutôt une structure sur les points : certains sont-ils proches les uns des autres ? Y a-t-il des *features* inutiles ? Peut-on regrouper certaines *features* pour en créer de nouvelles qui mettraient les données plus en valeur ? On parle alors d'apprentissage non-supervisé.

Dans les exemples précédents, celui de la recommandation de contenu peut tout à fait être vu comme un problème d'apprentissage supervisé. Nous verrons d'autres problèmes de ce type dans la suite du cours.

Un exemple classique d'apprentissage non-supervisé est le problème du clustering. Imaginons que l'on vous donne un grand nombre d'individus décrits par des *features* $X_1, \ldots, X_p$. On souhaite en extraire un certain nombre de populations, c'est-à-dire faire émerger des groupes d'individus qui se ressemblent entre eux mais qui ne ressemblent pas aux autres groupes. Ce type de problème est très courant et vient fréquemment se poser en amont des problèmes classiques de Machine Learning. On peut le voir comme un pré-étiquetage automatique des données.

3 Régression ou Classification ?

Dans les problèmes posés, on doit distinguer encore deux types :

— Ceux pour lesquels on nous demande de prédire l'appartenance d'un individu à une catégorie : "si l'individu a les caractéristiques $X_1, \ldots, X_p$ alors j'affirme qu'il va acheter/ne pas acheter le produit". On prédit un 1 ou un 0, ou une catégorie parmi K, pour un ensemble de features. C'est le problème de la classification.

— Ceux pour lesquels on nous demande de prédire une grandeur sur une échelle de valeurs quantitatives : "si l'individu a les caractéristiques $X_1, \ldots, X_p$

alors j'estime que son score vaut 17". Il s'agit alors d'un problème de régression.

On peut ensuite faire émerger une classification à partir d'une régression : imaginez que l'on prédise des scores pour chaque individu et que l'on en déduise s'il vont acheter ou non le produit qu'on propose. C'est le cas par exemple lorsque l'on fait du *scoring*.

Deuxième partie

Quelques algorithmes importants

Avant de démarrer cette section, il est important de noter que nous ne présentons pas ici une liste exhaustive des méthodes existantes en matière de machine learning. En outre, nous souhaitons rester concis en ce qui concerne les résultats mathématiques qui garantissent la convergence et la consistance de ces mtéhodes. Pour plus de détails, vous pouvez toujours vous référer aux deux ouvrages suivants :[1] pour une introduction claire pouvant même inclure des exemple de code en R, et [2] pour une approche plus complète des modèles mathématiques que soutendent les algorithmes présentés.

4 Arbres

Les méthodes fondées sur des arbres possèdent de nombreux avantages :

- Très faciles à comprendre et à visualiser ;
- Demandent peu de préparation des données ;
- Permettent de traiter des données numériques et catégoriques ;
- Permettent de traiter des poblèmes à sorties multiples ;
- Permettent une interprétation rapide et claire des résultats : pour chaque donnée, on sait pourquoi telle ou telle catégorie a été décidée en remontant les variables booléennes de l'arbre.

En outre, les méthodes fondées sur des arbres ont été largement étudiées dans la littérature et on trouve de nombreux résultats garantissant leur robustesse Enfin, des algorithmes élaborés comme les *Random Forests* ou le *Bagging* sont directement dérivés de ces méthodes et donnent d'excellents résultats en régression ou en classification.

4.1 Arbres de décision

Le principe est le suivant : on commence par choisir une des variables X_i et on trouve la valeur seuil s_i qui sépare les données en deux catégories le mieux possible. On obtient ainsi deux parties de l'espace dans lesquelles on réitère indépendamment la première opération. On obtient ainsi quatre zones. On continues ainsi jusqu'à avoir séparé tous les points appartenant à des catégories différentes. La dernière étape consiste alors à élaguer l'arbre pour ne pas

FIGURE 1 – Exemple de surfaces de décisions pour la construction d'un arbre.

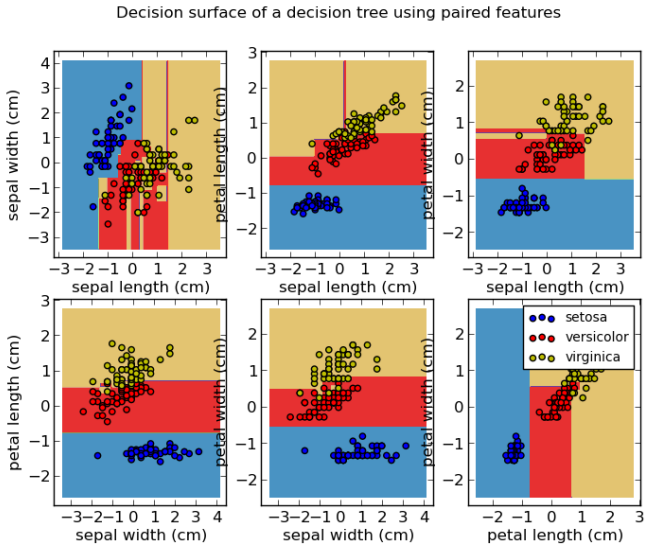

figure : manuel Scikit-learn

sur-apprendre notre classifieur : on supprime certaines des séparations qu'on avait dessinées suivant un critère de complexité à choisir. Par exemple, on peut pénaliser le nombre de feuilles de l'arbre à l'aide d'un paramètre α et chercher l'arbre qui minimise l'erreur pénalisée

$$\left(\sum_{m \in feuilles} \sum_{x \in surface\ de\ m} (y_x - \hat{y}_m)^2 \right) + \alpha |N_{feuilles}|$$

On obtient alors des zones de décision, comme présentées figure 1, qui nous permettent de construire l'arbre présenté figure 2.

4.2 Random Forests

L'idée des Random Forests est de booster les performances d'un arbre en en construisant quelques centaines à partir de divers échantillons des données. On obtient ainsi de nombreux classifieurs que l'on agrège pour obtenir la décision finale. Cet algorithme est encore jeune (Breiman, 2001) et de mieux en mieux compris même si on ne sait pas encore très bien à quoi est due la robustesse des prédictions obtenues. Une très bonne implémentation de cette méthode est disponible dans la bibliothèque *scikit-learn.*

FIGURE 2 – Exemple d'arbre de décision.

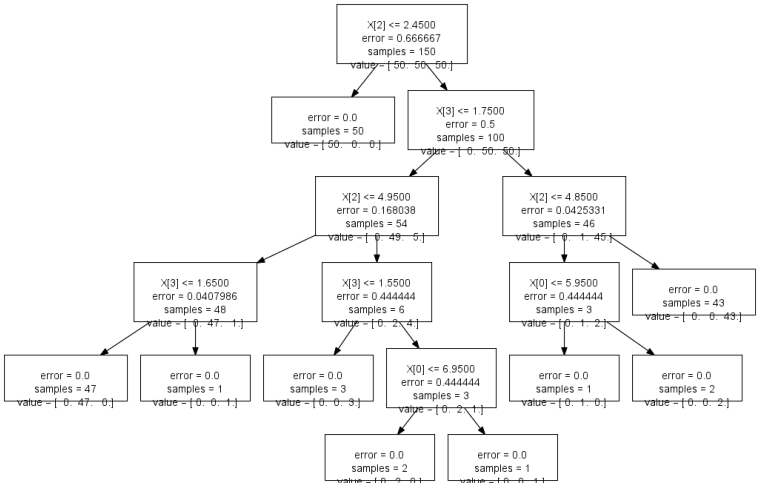

figure : manuel Scikit-learn

5 Trois algorithmes de base pour la classification

Alors que les arbres de la partie précédente pouvaient tout aussi bien classifier que régresser, on se concentre ici sur le problème de la classification : à partir d'un ensemble de features, on cherche à quelle catégorie appartient un point donné.

5.1 La régression logistique

Supposons que nos données X peuvent appartenir à l'une des deux catégories 0 ou 1. La régression logistique cherche à modéliser la probabilité que $Y(X)$ appartienne à une certaine catégorie. On cherche donc à estimer $Pr(Y = 1|X) := p(X)$ et pour cela on choisit de modéliser cette quantité par

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

Afin de pouvoir utiliser notre modèle pour prédire Y, on a besoin de trouver les coefficients β_0 et β_1 qui correspondent le mieux aux données que nous avons à analyser. Pour cela, on utilise une méthode très classique appelée *maximum de vraissemblance*. On écrit la vraissemblance des paramètres β_0 et β_1 étant données les observations :

$$\begin{aligned} L(\beta_0, \beta_1|X_1, \dots X_n) &= \prod_{i=1}^{n} Pr(Y_i|X_i; \beta_0, \beta_1) \\ &= \prod_{i:Y_i=1} p(X_i; \beta_0, \beta_1) \prod_{j:Y_j=0} (1 - p(X_j; \beta_0, \beta_1)) \end{aligned}$$

FIGURE 3 – Exemple de régression logistique.

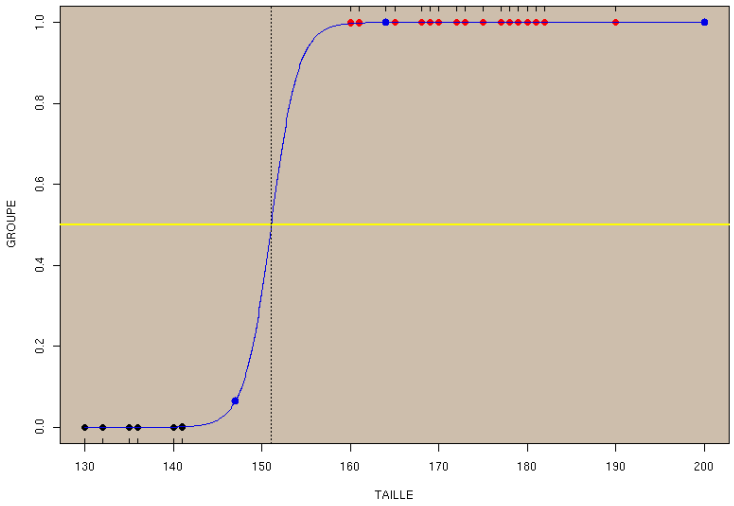

figure : Gilles Hunault [3]

Comme toutes les données observées sont supposées indépendantes, la vraisemblance s'écrit comme le produit des probabilités d'observer ce qu'on observe sachant le modèle que l'on a imposé. La fonction obtenue est convexe en β_0 et β_1. Il ne nous reste plus qu'à la minimiser en utilisant un algorithme comme celui de Newton-Raphson. En pratique, la plupart des langages possèdent une bibliothèque qui vous permmettra de calculer ces coefficients très rapidement.

On obtient alors une courbe comme celle présentée figure 3 : on y présente la classification de personnes en fonction de leur taille. On remarque que la taille 151 cm est la valeur seuil autour de laquelle l'appartenance à une classe particulière est incertaine. Plus la pente de la courbe est grande au niveau de la valeur seuil, plus la classification est sûre.

Ref[3]

5.2 Les K plus proches voisins

Toujours dans l'espoir de classifier des points qui n'ont pas encore de label, on peut essayer de s'appuyer sur leurs voisins. La méthode des K plus proches voisins, dite *K-nearest neighbors (KNN)* en anglais, a l'avantage d'être non paramétrique : à l'opposé de la régression logistique étudiée ci-dessus, on ne suppose aucun modèle a priori sur les données mais on les utilise directement pour classifier les données inconnues.

En pratique, on considère un point non classifié U et on cherche ses K plus proches voisins $X_{1,U}, \dots X_{K,U}$ appartenant à la base d'apprentissage, c'est-à-

FIGURE 4 – Exemple classification par K-NN.

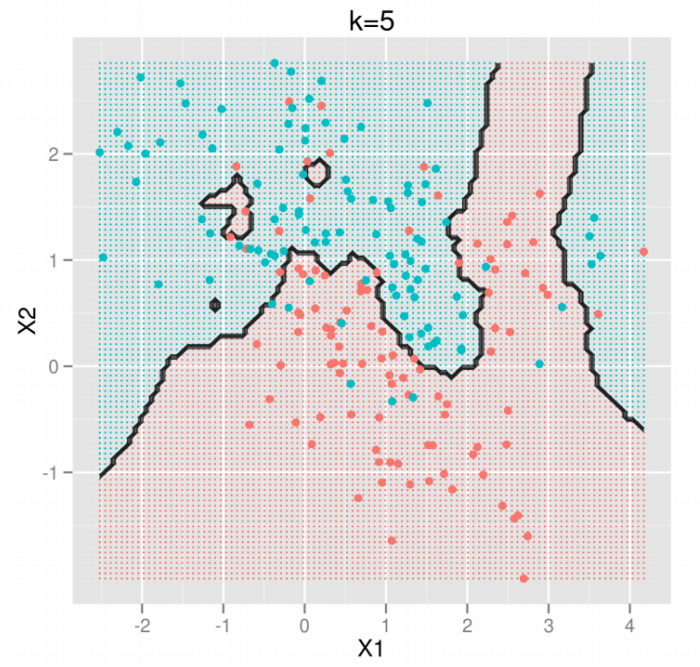

figure : cours MDI343 - Télécom Paristech

dire dont on connaît les labels $Y_{1,U}, \dots Y_{K,U}$. On réalise enfin un vote à la majorité : on décide que U appartient à la classe majoritaire de ses K plus proches voisins. On obtient une carte du plan comme présentée figure 4. Pour réaliser cette carte, on a pris tous les points sur une grille assez serrée et on a évalué quel serait le label d'une donnée située à cet endroit en prenant les 5 plus proches voisins.

5.3 Les machines à vecteur de support (SVM)

Le problème que l'on cherche à résoudre est celui de la classification à deux classes que l'on suppose linéairement séparables : cela signifie qu'il est possible de tracer une droite dans le plan qui sépare les points labellés 1 des points labellés -1.

L'idée clé à retenir est que l'on va chercher à *maximiser la marge* entre la droite tracée et le point le plus proche de cette droite. Dans un espace de dimension d, un hyperplan peut être entièrement décrit sous forme cartésienne par $d+1$ paramètres $p_0, \dots, p_d$:

$$p_0 + p_1 x_1 + \dots + p_d x_d = 0$$

Ainsi, pour un point X^+ situé au-dessus de l'hyperplan, donc ayant un label

$Y^+ = +1$, on aura

$$p_0 + p_1 X_1^+ + \ldots + p_d X_d^+ > 0$$

et réciproquement, pour un point X^- situé en-dessous de l'hyperplan, ayant un label $Y^- = -1$, on aura

$$p_0 + p_1 X_1^- + \ldots + p_d X_d^- < 0$$

En résumé, l'hyperplan séparateur a la propriété

$$Y_{point}(p_0 + p_1 X_1^{point} + \ldots + p_d X_d^{point}) > 0$$

Rappelons la formule de distance d'un point à un hyperplan s'écrit

$$d(x, \mathcal{H}) = \frac{|\mathbf{p}.x|}{|\mathbf{p}|}$$

où $\mathbf{p}$ est le vecteur des $d+1$ paramètres. Si on impose que ces paramètres soient choisis de telle sorte que $|\mathbf{p}| = 1$, on obtient que pour tout point du plan x ayant pour label y, sa distance à l'hyperplan vaut exactement

$$d(x, \mathcal{H}) = y\,(p_0 + p_1 x_1 + \ldots + p_d x_d)$$

Le problème d'optimisation à résoudre consiste donc à maximiser la marge entre l'hyperplan paramétré par $\mathbf{p}$ et les points de la base d'entrainement :

$$\begin{array}{rc} \text{maximiser} & M \\ \text{sous contrainte} & \sum_{i=1}^{d} p_i^2 = 1 \\ & y_j \left(p_0 + p_1 x_1^j + \ldots + p_d x_d^j\right) \quad \forall j \in \{1, \ldots, N\} \end{array}$$

Sans entrer dans les détails des calculs, on peut simplement évoquer le fait que le théorème de Karush Kuhn et Tucker va faire apparaître des conditions d'optimalité de la solution qui mettront en évidence les points situés exactement sur la marge. Ces points limites seront alors nommés *vecteurs supports* et ce sont eux qui définiront les paramètres $\mathbf{p}$ nécessaires pour classifier les données de la base de test dont on ne connaît pas les labels.

Remarque Que faire lorsque le problème n'est pas séparable ? Dans ce cas, il faut relâcher un peu le problème d'optimisation, c'est-à-dire autoriser un buget C de dépassement de la marge. Pour chaque point, on calculera un ε_i appelée *variable ressort*. On imposera alors que la somme des variables ressort $\sum_{i=1}^{N} \varepsilon_i$ soit strictement inférieure au buget C aloué au dépassement. Cela rajoute donc N variables et une contrainte au problème d'optimisation considéré ci-dessus.

FIGURE 5 – Exemple de classification binaire par SVM.

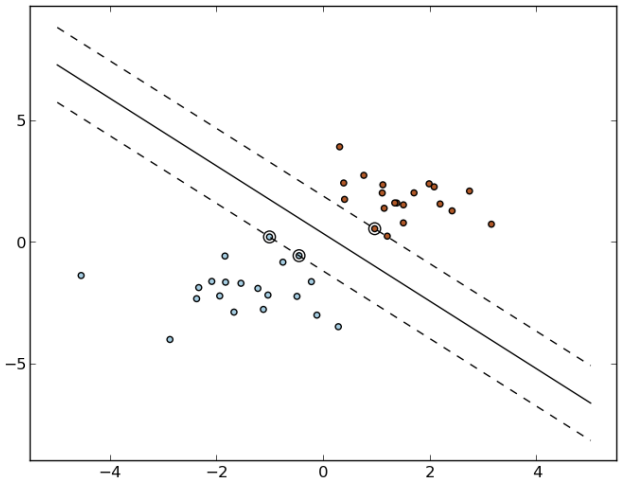

figure : manuel Scikit-learn

6 Deux algorithmes d'apprentissage non supervisé

On ne parle plus ici de labels, ni de classification ou de régression. Les différents objectifs de l'apprentissage non supervisés visent à mieux connaître la structure des données et des variables qui les décrivent. On présente ici deux algorithmes qui résolvent deux problèmes distincts :

— Tout d'abord l'analyse en composantes principales qui vise à réduire la dimension de l'espace des features en projetant les données sur un espace de dimension inférieure tout en perdant le moins d'information possible ;
— Ensuite le clustering K-Means qui cherche à trouver K groupes de points dans le nuage des données.

6.1 L'analyse en composantes principales (PCA)

Dans cette partie, on suppose que les données sont décrites par trois variables X_1, X_2, X_3 . Supposons par exemple que l'on veuille représenter en 2D les points de notre base. Une première solution serait de représenter les $\begin{pmatrix} 3 \\ 2 \end{pmatrix} = 3$ projections sur les axes. Cependant, on se rend compte que lorsque la dimension augmente, $\begin{pmatrix} d \\ 2 \end{pmatrix}$ augmente très vite et la méthode proposée se révèle caduque. En outre, selon la répartition des données, il est probable qu'aucune de ces trois

FIGURE 6 – Exemple de réduction de dimension par PCA.

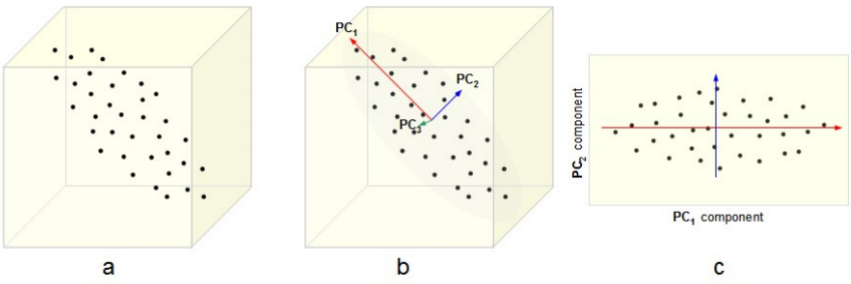

figure : cours MDI343 - Télécom Paristech

représentations ne mette en évidence la forme du nuage. On cherche donc les directions de l'espace le long desquelles les données varient le plus. Appelons Z_1 cette direction. Elle peut s'écrire comme une combinaison linéaire des vecteurs de la base canonique

$$Z_1 = \psi_1^1 X_1 + \psi_2^1 X_2 + \psi_3^1 X_3$$

Chaque point peut alors être caractérisé par un score pour la première composante principale Z_1

$$z_1^i = \psi_1^1 x_1^i + \psi_2^1 x_2^i + \psi_3^1 x_3^i$$

et on cherche à maximiser le score total obtenu pour toutes les données

$$\begin{aligned} &\text{maximiser} && \tfrac{1}{n} \sum_{i=1}^{N} \left(\sum_{j=1}^{3} \psi_j^1 x_j^i \right)^2 \\ &\text{sous contrainte} && \sum_{j=1}^{3} \left(\psi_j^1 \right)^2 = 1 \end{aligned}$$

Une fois la première composante trouvée, il faut chercher la seconde et ainsi de suite. On peut prouver que la résolution de ce problème d'optimisation revient à trouver les vecteurs propres de la matrice de covariance des données, symétrique donc diagonalisable en base orthonormée. Pour notre exemple, il suffit donc de projeter les données sur le plan formé par les deux premiers vecteurs propres que l'on a trouvés afin d'obtenir une visualisation des données comme celle que l'on a figure 6.

6.2 Le clustering K-Means

On se propose de regrouper les points du nuages en $K = 3$ groupes ou clusters. Pour cela, on va faire appel à un algorithme itératif appelé $K-Means$, qui se repose, comme son nom l'indique, sur un point moyen pour chaque cluster que l'on appelera les centroïdes. La sortie de l'algorithme est un ensemble $\mathbf{S} =$

FIGURE 7 – Exemple de clustering avec K-Means.

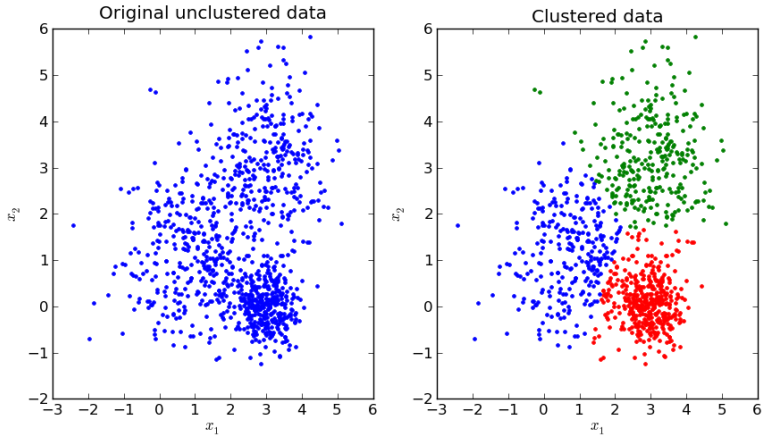

figure : documentation librairie PyPR

$\{S^1, \ldots, S^K\}$ de K ensembles d'assignations : chaque S^i contient les points du cluster i.

On commence donc par initialiser les centroïdes aléatoirement dans l'espace. Même si elle n'en a pas l'air, cette phase d'initialisation est critique et peut complètement influencer la forme de la solution finale. On part donc avec $c^1(0), \ldots, c^K(0)$.

Ensuite, on itère jusqu'à ce qu'on n'observe plus de changement :

— Assigner chaque donnée au cluster le plus proche – mise à jour des ensembles d'assignations– :

$$S^i(t) = \{x \mid \|x - c^i(t)\| \leq \|x - c^j(t)\|, \forall j = 1, \ldots, K\}$$

— Mettre à jour les centroïdes pour les recentrer :

$$c^i(t+1) = \frac{1}{|S^i(t)|} \sum_{\mathbf{x}_j \in S^i(t)} \mathbf{x}_j$$

On obtient ainsi K clusters (ensembles de points assignés) et K centroïdes qui peuvent servir à expliquer ces regroupements par exemple. On présente un exemple figure 7.

Références

[1] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (pp. 303-320). New York: Springer.

[2] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., & Tibshirani, R. (2009). The elements of statistical learning (Vol. 2, No. 1). New York: Springer.

[3] Site de Gilles Hunault. Page dédiée à la régression logistique. http://www.info.univ-angers.fr/~gh/wstat/reglogi.php