Algorithmes d'Apprentissage Non Supervisé

Module : Sujets Spéciaux

Rédigé par
BENABOU Mohamed El Ghali

Encadrant
ZOGLAT Abdelhak

Mars 2023

Contents

Introduction

Les algorithmes d'apprentissage non supervisé sont une catégorie de techniques de Machine Learning qui permettent à un modèle de découvrir des structures et des patterns cachés dans les données de manière autonome, sans utiliser de données étiquetées ou structurées. Ils sont souvent utilisés lorsque les données sont trop complexes ou volumineuses pour être étiquetées manuellement, ou lorsque l'objectif est de comprendre les caractéristiques cachées des données plutôt que de faire des prédictions précises.

Les algorithmes d'apprentissage non supervisé sont largement utilisés dans de nombreuses applications, notamment la segmentation de clientèle, la classification de documents, la détection de fraudes et la génération de recommandations. Il existe plusieurs types d'algorithmes d'apprentissage non supervisé, tels que les algorithmes de regroupement, de densité de données et de projection de données. Chacun de ces algorithmes a ses propres avantages et inconvénients en fonction de l'application considérée.

Dans ce rapport, nous allons examiner en profondeur les algorithmes d'apprentissage non supervisé et leur utilisation dans différentes applications. Nous verrons comment ces algorithmes sont conçus et comment ils fonctionnent, ainsi que les défis auxquels ils sont confrontés et les méthodes utilisées pour les surmonter. Nous examinerons également les applications courantes de ces algorithmes et leur impact sur l'industrie et la société en général.

1 Types d'algorithmes d'apprentissage non supervisé

1.1 Algorithmes de Regroupement - Clustering

Les algorithmes de regroupement sont utilisés pour regrouper des éléments similaires dans des groupes ou des clusters. Ils fonctionnent en comparant les caractéristiques des différents éléments et en les assignant à des groupes en fonction de leur similitude. Les algorithmes de regroupement peuvent être utilisés lorsque les données ne sont pas étiquetées ou lorsque l'objectif est de découvrir des structures ou des patterns cachés dans les données.

Il existe plusieurs types d'algorithmes de regroupement, tels que l'algorithme de K-means, l'algorithme de regroupement hiérarchique et l'algorithme de regroupement basé sur la densité. Chacun de ces algorithmes a ses propres avantages et inconvénients en fonction de l'application considérée.

La classification, ou analyse de groupes, consiste à regrouper des objets en fonction de leurs similitudes. Elle peut être utilisée dans de nombreux domaines tels que l'apprentissage automatique, la création d'images de synthèse, la reconnaissance de formes, l'analyse d'images, la recherche d'informations, la bio-informatique et la compression de données [1].

Les clusters sont un concept délicat, d'où la présence de nombreux algorithmes de classification différents. Différents modèles de cluster sont utilisés, et pour chacun de ces modèles, il peut y avoir différents algorithmes. Les clusters trouvés par un algorithme de classification seront sûrement différents de ceux trouvés par un autre algorithme.

Dans les systèmes d'apprentissage automatique, nous regroupons souvent des exemples comme première étape pour comprendre le jeu de données. Le regroupement d'un exemple non étiqueté est appelé classification non supervisée. Si les exemples sont étiquetés, il s'agit alors de classification supervisée.

Les algorithmes de clustering sont utilisés pour regrouper des points de données sur la base de certaines similitudes. Il n'y a pas de critère pour un bon clustering, celui-ci dépend principalement de l'utilisateur et du scénario spécifique. Les modèles de cluster typiques comprennent les modèles de connectivité, les modèles de centroïdes, les modèles de distribution, les modèles de densité, les modèles de groupe, les modèles basés sur les graphes et les modèles neuronaux [2].

Il existe également différents types de clustering. Le clustering dur divise les points de données en groupes distincts et exclusifs, tandis que le clustering mou donne une probabilité de présence à chaque point de données dans chaque groupe. Par exemple, dans la segmentation de la clientèle en quatre groupes, chaque client peut appartenir à l'un des quatre groupes selon une probabilité.

1.2 Algorithmes de Réduction de Dimensionalité

Les algorithmes de projection de données sont un type d'algorithmes d'apprentissage non supervisé qui sont utilisés pour réduire la dimensionnalité des données tout en conservant le maximum d'informations possible. Ces algorithmes sont souvent utilisés lorsque les données sont complexes ou volumineuses et qu'il est difficile de les visualiser ou de les comprendre dans leur forme originale.

Les algorithmes de projection de données fonctionnent en transformant les données d'un espace de haute dimension en un espace de dimension inférieure tout en conservant le maximum d'informations possible. Cette transformation est effectuée en utilisant des techniques mathématiques qui permettent de réduire la dimensionnalité des données tout en conservant leur structure. Par exemple, dans le cas de données de consommation de produits, un algorithme de projection de données pourrait réduire la dimensionnalité des données de consommation de produits tout en conservant le maximum d'informations sur les patterns de consommation des clients [3].

Il existe plusieurs types d'algorithmes de projection de données, tels que la réduction de la dimensionnalité par ACP (Analyse en Composantes Principales) et la réduction de la dimensionnalité par TSNE (t-Distributed Stochastic Neighbor Embedding). L'ACP est un algorithme qui utilise une technique de décomposition matricielle pour réduire la dimensionnalité des données tout en conservant le maximum d'informations possible. TSNE est un algorithme qui utilise une technique de réduction de la dimensionnalité non linéaire pour réduire la dimensionnalité des données tout en conservant la structure de proximité des éléments [4].

En utilisant les algorithmes de projection de données, il est possible de réduire la dimensionnalité des données tout en conservant le maximum d'informations possible. Cela peut être utile pour visualiser et comprendre les données de manière plus simple et plus intuitive. De plus, la réduction de la dimensionnalité peut également aider à accélérer les algorithmes de traitement des données en réduisant la complexité des calculs nécessaires. Cela peut être particulièrement important dans des applications telles que l'analyse de données en temps réel ou le traitement de données volumineuses. En résumé, les algorithmes de projection de données sont un outil puissant pour simplifier les données et en faciliter l'analyse, tout en conservant un maximum d'informations importantes.

1.3 Algorithmes de Densité de Données

Les algorithmes de densité de données représentent un type d'algorithme d'apprentissage non supervisé dont l'objectif est de détecter des structures ou des motifs dans les données en identifiant les zones de forte densité de données. Ils sont souvent privilégiés lorsqu'il n'y a pas d'étiquettes pour les données ou lorsqu'on souhaite explorer les structures sous-jacentes des données.

Pour découvrir les zones de forte densité, les algorithmes de densité de données mesurent les distances entre les différents points de données et identifient les régions de l'espace où la densité de points est la plus importante. Ces

zones de haute densité sont considérées comme des structures ou des motifs dans les données. Par exemple, en analysant les données de consommation de produits, un algorithme de densité de données pourrait identifier les zones de forte densité de clients ayant des achats similaires [5].

Il existe différents types d'algorithmes de densité de données, tels que DBSCAN et OPTICS. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) utilise une seule distance de voisinage et un seuil de densité pour identifier les zones de forte densité dans les données. Quant à OPTICS (Ordering Points to Identify the Clustering Structure), il utilise également une distance de voisinage, mais permet d'identifier des zones de densité variable dans les données. Expectation-Maximization (EM) Gaussian Mixture Model (GMM) est une méthode utilisée pour estimer la densité de probabilité de données observées. Elle est particulièrement utile pour les données multidimensionnelles où les données peuvent être générées à partir de plusieurs distributions gaussiennes sous-jacentes. EM GMM peut être utilisé pour trouver le nombre optimal de clusters et pour attribuer chaque observation à son cluster correspondant [1]. Les applications courantes incluent la segmentation d'image, la détection d'anomalies et la classification non supervisée.

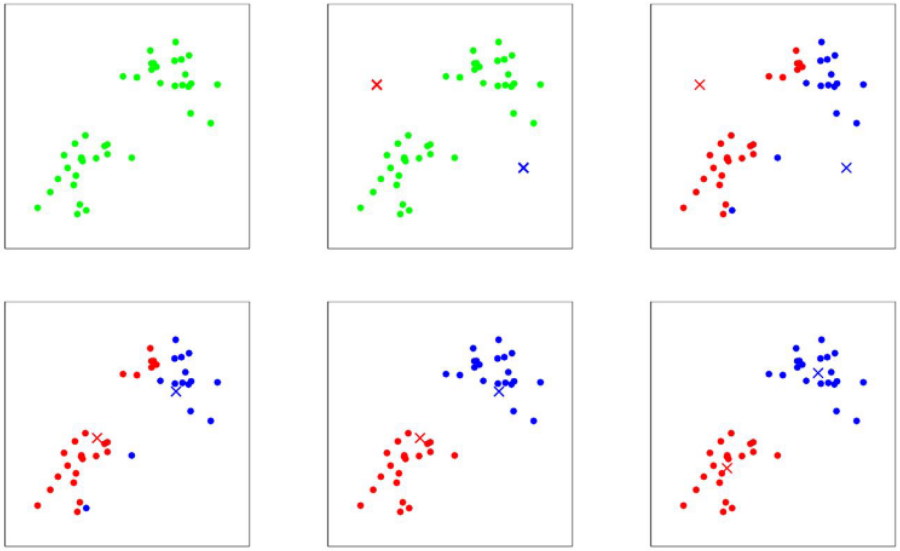
L'utilisation des algorithmes de densité de données permet de découvrir des structures sous-jacentes dans les données et d'obtenir une meilleure compréhension de la distribution des éléments dans l'espace de données. Ces connaissances peuvent être utilisées pour améliorer les stratégies de marketing, optimiser les processus de production ou encore prévenir les fraudes.

2 Exemples d'algorithme d'apprentissage non supervisé

2.1 Algorithme de Regroupement – K-means

L'algorithme K-means est un algorithme de clustering qui, après avoir initialisé K centres des clusters (d'où son nom), fonctionne en itérant à travers deux étapes principales: la première étape consiste à assigner chaque point de données au cluster le plus proche, en utilisant une mesure de distance telle que la distance euclidienne. La seconde étape consiste à recalculer le centre de chaque cluster en utilisant la moyenne des points de données attribués à ce cluster [4].

Ces deux étapes sont répétées jusqu'à ce que la variation de la position des centres de clusters devienne très faible, ou que le nombre d'itérations prédéfini soit atteint. Le résultat final est un ensemble de k clusters, chacun avec son propre centre de gravité, qui représente une partition des données en groupes similaires.



2.1.1 Initialisation des Clusters ou des Centres de Données

L'initialisation des clusters ou des centres de données est une étape cruciale qui consiste à définir les centres de chaque cluster avant de débuter l'apprentissage.

Il existe plusieurs méthodes pour initialiser les centres de données, chacune ayant ses propres avantages et inconvénients. Une méthode couramment utilisée consiste à sélectionner au hasard quelques points de données comme centres initiaux. Cette méthode est simple à mettre en œuvre, mais elle peut donner de mauvais résultats si les points sélectionnés au hasard ne sont pas représentatifs de l'ensemble des données.

Une autre méthode couramment utilisée consiste à utiliser un algorithme de regroupement hiérarchique pour déterminer les centres initiaux. Cette méthode est plus robuste que la sélection au hasard, mais elle est plus coûteuse en temps de calcul.

Il est important de choisir une méthode d'initialisation adéquate car elle peut avoir un impact significatif sur les résultats finaux de l'algorithme de regroupement. Il est recommandé de tester plusieurs méthodes d'initialisation et de comparer les résultats pour trouver celle qui donne les meilleurs résultats pour chaque application particulière.

L'algorithme K-means utilise généralement une initialisation aléatoire pour ses clusters. Cependant, il est possible d'utiliser des hypothèses éclairées afin de limiter le nombre de calculs nécessaires lors des étapes suivantes.

2.1.2 Calcul de la Distance des Données aux Centres

L'étape de calcul de la distance des données aux centres de cluster est fondamentale pour déterminer à quel cluster chaque point de donnée devra être affecté. Elle consiste à mesurer la distance entre chaque point de donnée et chaque centre de cluster pour déterminer le cluster approprié pour chaque point de donnée.

Il existe plusieurs méthodes pour calculer la distance entre les données et les centres, chacune ayant ses propres avantages et inconvénients. La méthode la plus couramment utilisée est la distance Euclidienne, qui calcule la distance entre deux points par exemple $(a_1, a_2, ..., a_n)$ et $(b_1, b_2, ..., b_n)$ en utilisant la formule suivante :

distance $= \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}$

Cette méthode est simple à mettre en œuvre et donne de bons résultats dans de nombreuses situations. Cependant, elle peut donner de mauvais résultats si les dimensions des données sont très dissemblables ou si certaines dimensions ont une importance beaucoup plus grande que d'autres.

D'autres méthodes de calcul de distance, telles que la distance de Manhattan et la distance de Mahalanobis, peuvent être utilisées dans des situations où la distance Euclidienne ne donne pas de bons résultats.

Pour l'algorithme K-means, la distance couramment utilisée est la distance Euclidienne.

2.1.3 Attribution des Données aux Clusters

Cette étape consiste à affecter chaque point de donnée à un cluster en fonction de la distance de ce point aux centres de cluster.

Il existe plusieurs façons d'attribuer les données aux clusters, chacune ayant ses propres avantages et inconvénients. Une méthode couramment utilisée consiste à affecter chaque point de donnée au cluster dont le centre est le plus proche. Cette méthode est simple à mettre en œuvre et donne de bons résultats dans de nombreuses situations.

Une autre méthode consiste à affecter chaque point de donnée au cluster dont la distance est la plus faible. Cette méthode est plus robuste que la méthode précédente, mais elle est plus coûteuse en temps de calcul.

Il est important de choisir une méthode d'attribution adéquate car elle peut avoir un impact significatif sur les résultats finaux de l'algorithme de regroupement.

2.1.4 Recalcul des Centres et Condition d'Arrêt

Cette étape consiste à mettre à jour les centres de chaque cluster en fonction des points de données qui leur sont affectés. Une méthode couramment utilisée dans l'algorithme K-means, consiste à recalculer les centres comme étant la moyenne de tous les points de données affectés au cluster.

Après avoir fait le recalcul Dans l'algorithme K-Means, la condition d'arrêt est atteinte lorsque les centres des clusters ne changent plus ou que la différence

entre les centres des clusters à deux itérations consécutives est inférieure à un seuil prédéfini.

Algorithm 1 Algorithme de clustering K-means

Input: Ensemble d'entraînement $x^{(1)}, ..., x^{(n)}$
Output: Centres des clusters $\mu_1, ..., \mu_k$
Initialisation des centres $\mu_1, ..., \mu_k \in \mathbb{R}^d$ aléatoirement;
repeat
  for $i = 1$ **to** n **do**
    $c^{(i)} := \text{argmin}_j \, ||x(i) - \mu_j||^2$;
  end
  for $j = 1$ **to** k **do**
    $\mu_j := \frac{\sum_{i=1}^{n} 1\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{n} 1\{c^{(i)}=j\}}$;
  end
until *convergence*;

2.1.5 Exemple d'application pratique

a) Présentation des Données

Le jeu de données "iris" auquel on s'intéresse est un ensemble de données contenant des informations sur les caractéristiques de différentes espèces d'iris. Il est inclus dans le langage de programmation R et contient 150 observations de 5 variables : Sepal.Length, Sepal.Width, Petal.Length, Petal.Width et Species. Les observations sont réparties également entre les trois espèces : setosa, versicolor et virginica, avec 50 observations pour chaque espèce.

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Les variables de mesure incluent la longueur et la largeur des sépales et des pétales, qui sont mesurées en centimètres. La variable cible est l'espèce d'iris, qui est utilisée pour classer les observations en fonction de leur caractéristiques morphologiques.

b) Prétraitement des Données

Nous avons choisi de sélectionner les 4 premières colonnes des données "iris" pour réaliser une analyse de cluster non supervisée. Cette sélection permet de

considérer les mesures de longueur et de largeur des sépales et des pétales comme des variables pertinentes pour distinguer les différentes espèces de fleurs "iris". De plus, ces variables étant numériques et indépendantes les unes des autres, elles répondent aux critères requis pour appliquer l'algorithme k-means.

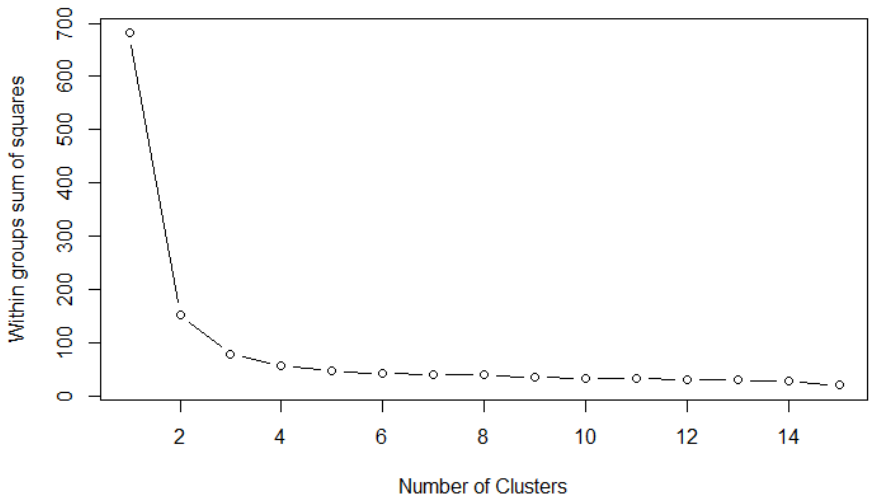
```
> mydata = select(iris, c(1,2,3,4))
> head(mydata)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
```

c) Choix du Nombre de Clusters

Pour déterminer le nombre de clusters à utiliser dans l'algorithme k-means, il y a deux approches possibles. La première consiste à choisir directement le nombre de clusters en fonction de la connaissance préalable des types d'iris, dans notre cas 3. La deuxième approche consiste à utiliser la fonction wssplot pour déterminer le nombre optimal de clusters.

La fonction wssplot calcule la somme des carrés intra-cluster (WSS) pour une plage de nombres de clusters allant de 1 à nc (par défaut 15), en exécutant plusieurs fois l'algorithme k-means avec des nombres de clusters différents sur les données d'entrée data. Le WSS mesure la quantité de variation à l'intérieur de chaque cluster, et est calculé comme la somme des distances au carré entre chaque observation et son centre de cluster.

Ensuite, la fonction trace les valeurs WSS en fonction du nombre de clusters, et permet d'identifier visuellement le "point de coude" dans le graphique. Ce point de coude correspond au nombre de clusters à partir duquel la diminution du WSS commence à se stabiliser. Ainsi, ce point de coude indique le nombre optimal de clusters pour l'ensemble de données donné.



En observant le graphique du WSS, on peut remarquer que le nombre de clusters 2 et 3 sont les points de coude, où la diminution du WSS commence à se stabiliser. Cela suggère que le choix optimal pour le nombre de clusters serait soit 2 ou 3.

d) Entraînement des Modèles

Nous allons entraîner deux modèles, l'un avec 2 clusters et l'autre avec 3 clusters, afin de les comparer.

```
KM1 = kmeans(mydata,2)
KM2 = kmeans(mydata,3)
```

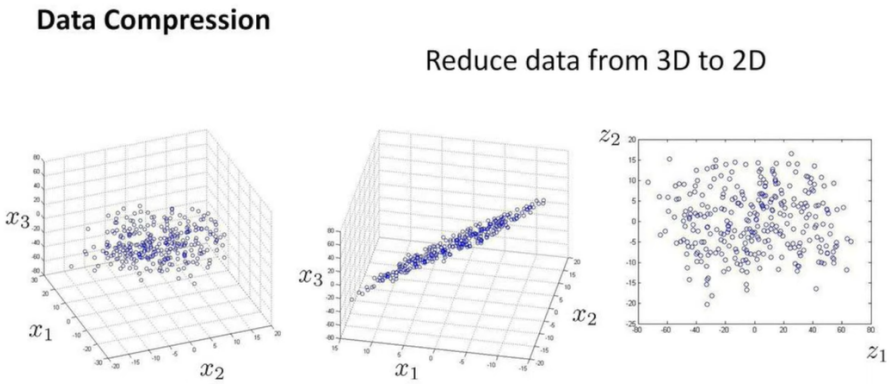
```
autoplot(KM1, mydata, frame = TRUE)
autoplot(KM2, mydata, frame = TRUE)
```

Nous pouvons observer que le modèle avec 2 clusters ne s'est pas bien adapté aux valeurs aberrantes proches du deuxième cluster, tandis que le modèle à 3 clusters s'est mieux adapté aux données.

2.2 Algorithme de Réduction de Dimensionalité – ACP

L'Analyse en Composantes Principales (ACP) est une méthode statistique de réduction de dimensionnalité. Elle est utilisée pour explorer les données multidimensionnelles et réduire leur complexité en identifiant les variables les plus importantes dans un ensemble de données. Cette technique est couramment utilisée en recherche scientifique, en ingénierie et dans d'autres domaines où la compréhension des relations entre les variables est importante [1].

Cet algorithme permet de transformer les données d'origine en un ensemble de variables linéaires indépendantes, appelées composantes principales. Ces composantes sont ordonnées en fonction de leur contribution à la variance totale des données, de sorte que les premières composantes principales capturent la majeure partie de la variance des données. En réduisant la dimensionnalité de l'ensemble de données, l'ACP peut simplifier la visualisation et l'analyse des données, ainsi que faciliter la détection de relations complexes entre les variables (figure ci-dessous). Cette méthode est également efficace d'un point de vue computationnel, car elle ne nécessite qu'un calcul des vecteurs propres.



Cette méthode repose sur deux étapes fondamentales. La première étape consiste à prétraiter et normaliser les données. Cela est réalisé en soustrayant la moyenne de chaque attribut et en divisant par l'écart-type empirique de chaque attribut, permettant ainsi de centrer les données autour de zéro et de les mettre à la même échelle. La deuxième étape consiste à projeter les données sur un sous-espace de dimension inférieure en trouvant les vecteurs propres de la matrice de covariance des données. Ces vecteurs propres définissent les directions dans lesquelles les données varient le plus. Les vecteurs propres associés aux plus grandes valeurs propres représentent les directions principales de variation des données et sont utilisés pour construire le sous-espace optimal.

2.2.1 Prétraitement des Données

Supposons que l'on dispose d'un ensemble de données $\{x^{(i)}; i = 1, ..., n\}$ comprenant des d attributs pour n observations. Soit $x^{(i)} \in \mathbb{R}^d$ pour chaque i

($d \ll n$). Il est préférable que le nombre d'attributs soit beaucoup plus petit que le nombre d'observations pour que l'ACP fonctionne efficacement. Si le nombre d'attributs est trop grand, cela peut entraîner des problèmes de surajustement et rendre difficile la détection des principales composantes qui résumeraient l'essentiel des informations du jeu de données. De plus, plus le nombre d'attributs est grand, plus la matrice de covariance associée à ces attributs sera difficile à calculer et à gérer numériquement. Pour ces raisons, il est conseillé de limiter le nombre d'attributs autant que possible, en sélectionnant uniquement ceux qui sont pertinents pour l'analyse [3].

De plus, l'ACP est une méthode qui nécessite un ensemble de données avec des attributs quantitatifs et non qualitatifs. Si les attributs de l'ensemble de données sont qualitatifs, la méthode utilisée sera l'analyse factorielle des correspondances. Avant d'appliquer l'ACP, il est important de normaliser les données. La normalisation est cruciale car elle permet de transformer les attributs en données de même échelle, c'est-à-dire avec une moyenne de 0 (peut être omis pour les données connues pour avoir une moyenne nulle, par exemple, des séries temporelles correspondant à la parole ou à d'autres signaux acoustiques) et une variance de 1 :

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{\sigma_j}$$

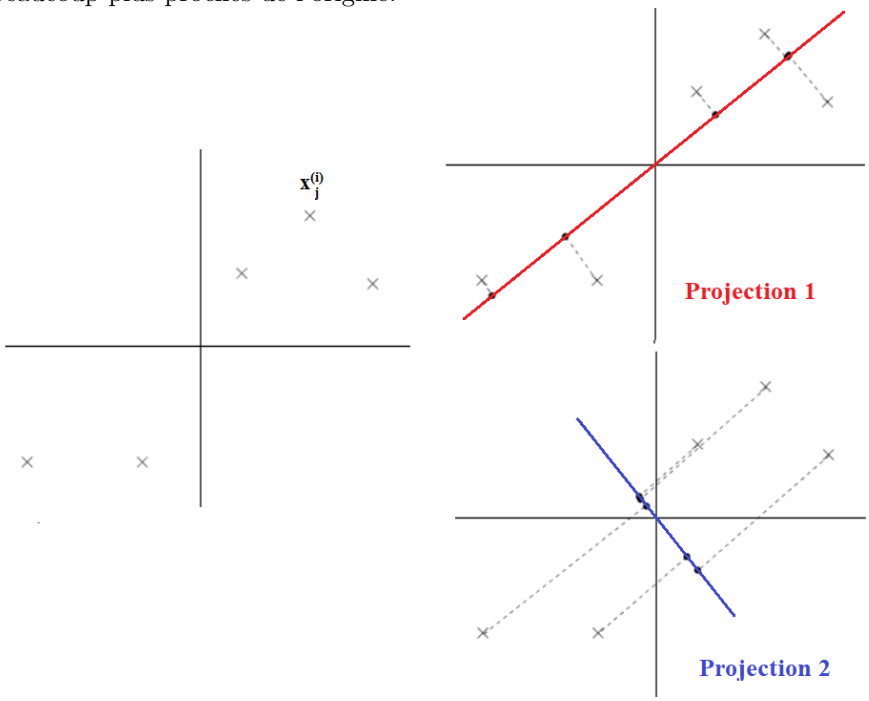
où $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}$ et $\sigma_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_j^{(i)} - \mu_j)^2$ sont la moyenne et la variance de la caractéristique j, respectivement. Cela garantit que les attributs sont tous traités sur la même échelle, ce qui est important pour l'interprétation des résultats. De plus, la normalisation facilite le calcul des composantes principales et permet d'éviter que les attributs ayant des valeurs plus élevées ne dominent les calculs.

2.2.2 Projection des Données

Maintenant que nos données sont normalisées, comment calculons-nous le vecteur principal de variation u, c'est-à-dire la direction approximative des données ? Une façon de formuler ce problème consiste à trouver le vecteur unitaire u tel que lorsque les données sont projetées sur la direction correspondant à u, la variance des données projetées est maximisée. Intuitivement, les données contiennent une certaine quantité de variance/information. Nous voudrions choisir une direction u de manière à ce que si nous approximons les données comme étant situées dans la direction/sous-espace correspondant à u, autant de cette variance que possible soit encore conservée.

Considérons l'ensemble de données suivant, sur lequel nous avons déjà effectué les étapes de normalisation (voir figure ci-dessous). Supposons que nous choisissons u pour correspondre à la direction indiquée dans la figure ci-dessous. Les cercles représentent les projections des données d'origine sur cette ligne rouge (projection 1). Nous constatons que les données projetées ont encore une

variance assez importante, et que les points tendent à être éloignés de zéro. En revanche, si nous avions choisi la direction suivante (projection 2) à la place, les projections auraient une variance considérablement plus petite et seraient beaucoup plus proches de l'origine.



Nous souhaitons sélectionner automatiquement la direction $\boldsymbol{u}$ correspondant à la première des deux figures présentées ci-dessus (projection 1). Pour formaliser cela, notons que la longueur de la projection d'un point $\boldsymbol{x}$ sur un vecteur unitaire $\boldsymbol{u}$ est donnée par $\boldsymbol{x^T u}$. Autrement dit, si $\boldsymbol{x^{(i)}}$ est un point de notre ensemble de données (l'une des croix sur le graphique), alors sa projection sur $\boldsymbol{u}$ est à une distance de $\boldsymbol{x^{(i)T} u}$ de l'origine. Ainsi, pour maximiser la variance des projections, nous souhaitons choisir un vecteur unitaire $\boldsymbol{u}$ de manière à maximiser :

$$\frac{1}{n}\sum_{i=1}^{n}(x^{(i)T}u)^2 = \frac{1}{n}\sum_{i=1}^{n} u^T x^{(i)} x^{(i)T} u = u^T \left(\frac{1}{n}\sum_{i=1}^{n} x^{(i)} x^{(i)T}\right) u.$$

Nous reconnaissons facilement que la maximisation de cette expression sous la contrainte $|\boldsymbol{u}|^2 = 1$ donne le vecteur propre principal de $\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x^{(i)} x^{(i)T}}$, qui est simplement la matrice de covariance empirique des données (en supposant qu'elle ait une moyenne nulle).

En résumé, nous avons trouvé que si nous souhaitons trouver un sous-espace de dimension 1 pour approximer les données, nous devrions choisir $\boldsymbol{u}$ comme étant le vecteur propre principal de $\boldsymbol{\Sigma}$. Plus généralement, si nous souhaitons

projeter nos données dans un sous-espace de dimension k ($k < d$), nous devrions choisir $\boldsymbol{u_1}, \ldots, \boldsymbol{u_k}$ comme étant les k premiers vecteurs propres de $\boldsymbol{\Sigma}$. Les vecteurs $\boldsymbol{u_i}$ forment maintenant une nouvelle base orthogonale pour les données.

Ensuite, pour représenter $x^{(i)}$ dans cette base, nous avons seulement besoin de calculer le vecteur correspondant :

$$y^{(i)} = \begin{pmatrix} u_1^T x^{(i)} \\ u_2^T x^{(i)} \\ \vdots \\ u_k^T x^{(i)} \end{pmatrix} \in \mathbb{R}^k.$$

Ainsi, alors que $x^{(i)} \in \mathbb{R}^d$, le vecteur $\mathbf{y}^{(i)}$ donne maintenant une approximation/représentation de dimension inférieure k pour $x^{(i)}$. L'ACP est donc également appelé algorithme de réduction de dimension. Les vecteurs $\mathbf{u}_1, ..., \mathbf{u}_k$ sont appelés les k premières composantes principales des données.

Afin de pouvoir interpréter les projections des données sur les vecteurs propres, nous avons besoin d'écrire u_i en fonction de $x^{(i)}$. En effet, les projections des données ne sont pas directement interprétables car elles sont exprimées dans un espace différent de l'espace d'origine. En écrivant u_i en fonction de $x^{(i)}$, nous pouvons interpréter la projection des données sur u_i comme la combinaison linéaire de leurs variables originales pondérées par les coefficients correspondants de u_i. Cela nous permet de comprendre quelles variables ont le plus grand impact sur la projection des données sur ce vecteur propre et de tirer des conclusions sur les caractéristiques les plus importantes des données.
Pour exprimer u_i en fonction de $x^{(i)}$, nous pouvons utiliser la formule de projection $x^{(i)} = Tu$, où T est la matrice de transformation formée par les vecteurs propres. En résolvant cette équation pour u, nous avons :

$$u_i = (T^T x^{(i)})_i$$

où $(T^T x^{(i)})_i$ représente la i-ème composante du vecteur $T^T x^{(i)}$. En d'autres termes, nous multiplions la transposée de la matrice de transformation T par le vecteur projeté $x^{(i)}$, puis nous prenons la i-ème composante de ce résultat pour obtenir la valeur de la i-ème composante du vecteur u_i.

Algorithm 2 Algorithme de Réduction de Dimensionalité ACP

Input: Ensemble d'entraînement normalisé $x^{(1)}, ..., x^{(n)}$
Output: Composantes principales des données $u_1, ..., u_d$
et Données transformées $y^{(1)}, ..., y^{(n)}$
1 : Calculer la matrice de covariance empirique
$\boldsymbol{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x^{(i)} x^{(i)T}}$
2 : Calculer les valeurs propres et les vecteurs propres de $\boldsymbol{\Sigma}$
3 : Définir la matrice de transformation $T = [u_1, ..., u_d]$, avec d vecteurs propres associés aux d plus grandes valeurs propres
4 : Projeter les données X dans le sous-espace ACP
$y^{(i)} = Tx^{(i)}$

2.2.3 Exemple d'application pratique

a) Prétraitement des Données

Nous allons de nouveau travailler avec la base de données iris de R. Il est important de noter que l'ACP ne peut être effectuée que sur des variables quantitatives. Par conséquent, nous allons supprimer la dernière colonne de la base de données iris.

```
> mydata = select(iris, c(1,2,3,4))
> head(mydata)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
```

Bien que la normalisation soit nécessaire pour entraîner notre modèle, nous ne la ferons pas à ce stade. Elle sera réalisée lors de l'entraînement du modèle.

b) Entraînement du Modèle

Nous appliquons l'analyse en composantes principales à notre nouvelle table de données (iris en omettant la colonne 5 – Species), avec une mise à l'échelle de chaque variable (scale = TRUE).

```
> model = prcomp(mydata, scale = TRUE)
> model
Standard deviations (1, .., p=4):
[1] 1.7083611 0.9560494 0.3830886 0.1439265

Rotation (n x k) = (4 x 4):
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

Ceci nous fournit les principales composantes du modèle en fonction de nos variables d'origine. Par exemple :

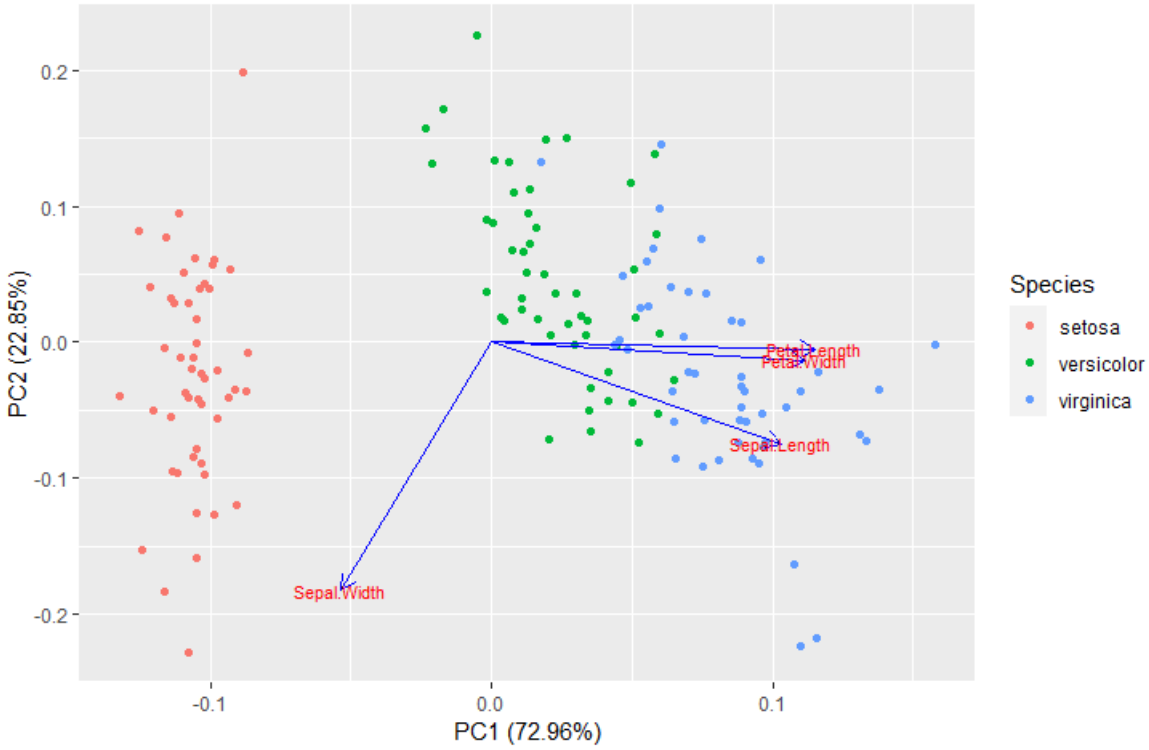
$$PC1 = 0.5210659 \times Sep.Length - 0.2693474 \times Sep.Width + 0.5804131 \times Pet.Length + 0.5648565 \times Pet.Width$$

```
> summary(model)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

La commande "summary" nous permet de voir le pourcentage de variance expliquée par chaque nouvelle composante.

c) Interprétations et Conclusions

Selon les résultats de la fonction summary, les deux premières composantes principales expliquent 95,81% de la variance totale. Par conséquent, nous allons utiliser ces deux composantes pour modéliser nos données et nos variables d'origine, afin de pouvoir interpréter les résultats de manière significative.

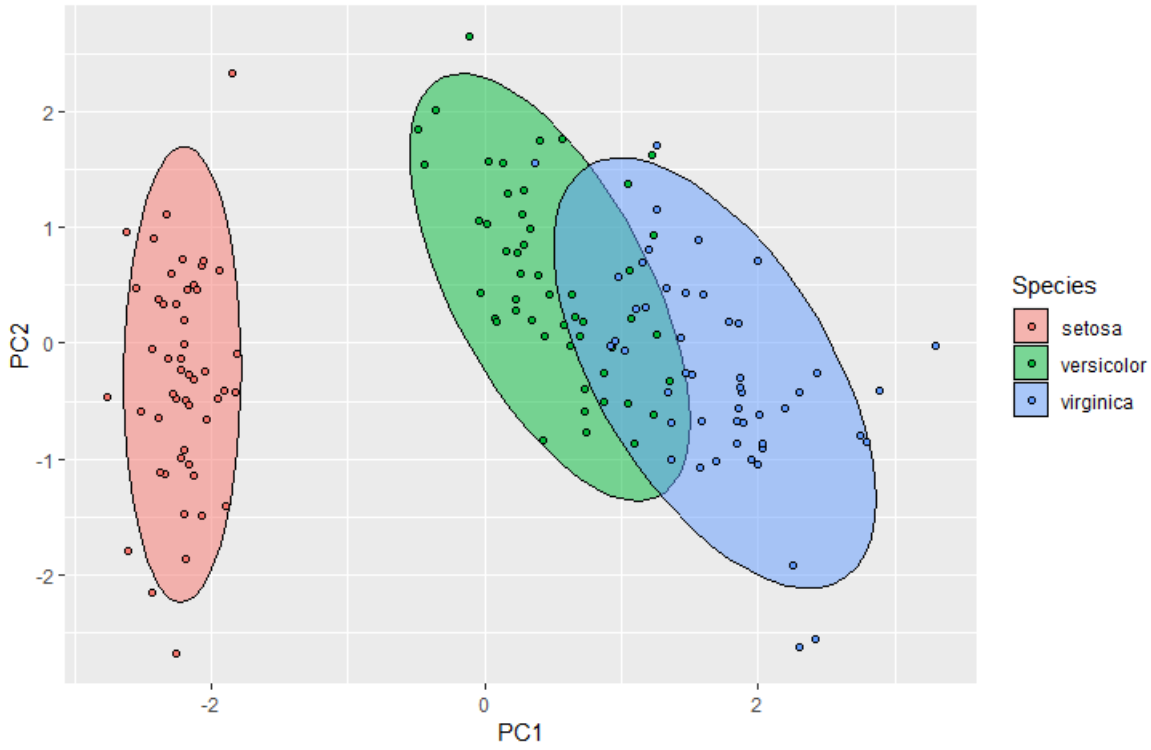


Il est également possible d'observer la corrélation entre les composantes principales retenues et les variables d'origine :

```
                    PC1         PC2
Sepal.Length  0.8901688 -0.36082989
Sepal.Width  -0.4601427 -0.88271627
Petal.Length  0.9915552 -0.02341519
Petal.Width   0.9649790 -0.06399985
```

```
ggplot(dataPR, aes(PC1, PC2, col = Species, fill = Species))+
  stat_ellipse(geom = "polygon", col = "black", alpha =0.5)+
  geom_point(shape = 21, col="black")
```

Le graphique présente les deux premières composantes principales en tant que coordonnées x et y, où la couleur et le remplissage sont définis par l'espèce. De plus, des ellipses sont ajoutées pour représenter les données de chaque espèce dans le graphique.



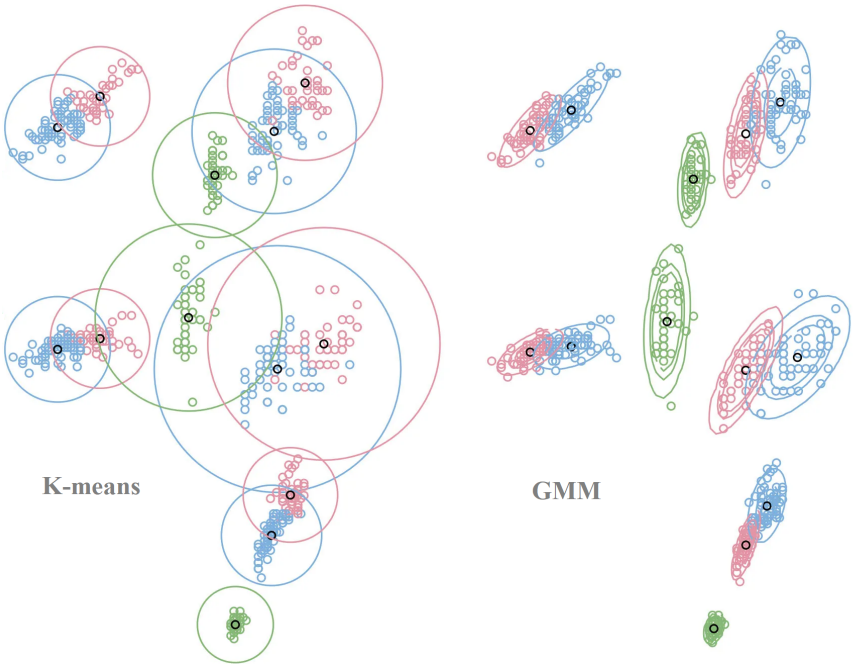
C'est une représentation de la projection des données dans un espace en deux dimensions, où chaque point représente une observation (dans ce cas, une fleur de la base de données iris) et les axes x et y correspondent aux deux premières

composantes principales.

Ce type de graphe est très utile pour visualiser les relations entre les observations dans un espace de dimensions réduit, et peut aider à identifier des groupes ou des clusters d'observations qui se regroupent de manière cohérente. Il permet dans notre cas de visualiser la distribution des différentes espèces de fleurs dans l'espace défini par les deux premières composantes principales, et de voir si elles forment des groupes distincts ou se chevauchent.

2.3 Algorithme de Densité de Données – EM Gaussien Mixture Models

L'algorithmie d'Expectation-Maximisation (EM) est un algorithme clé dans le domaine de l'apprentissage non-supervisé pour modéliser des données de densité inconnue. L'EM, en particulier l'algorithme de mélange gaussien, est utilisé pour décomposer des distributions de probabilité complexes en une combinaison de distributions gaussiennes simples, appelées composants gaussiens. Chaque composant gaussien représente une partie de la distribution globale et est caractérisé par ses propres paramètres de moyenne et de covariance. Le graphique ci-dessous illustre un exemple dans lequel l'utilisation de la GMM est plus appropriée que l'utilisation de K-means pour résoudre un problème de regroupement de données.



L'EM résout ce problème en utilisant un processus itératif en deux étapes: l'étape d'espérance et l'étape de maximisation. L'étape d'espérance est utilisée pour calculer la vraisemblance que chaque point de données appartienne à chaque composant gaussien, tandis que l'étape de maximisation est utilisée pour maximiser la log-vraisemblance des données en ajustant les paramètres des composants gaussiens. L'EM est couramment utilisé dans de nombreux domaines, notamment l'analyse de données, la reconnaissance de formes, la vision par ordinateur, la bioinformatique et l'apprentissage automatique.

Supposons que l'on nous donne un ensemble d'entraînement $\{x^{(1)}, ..., x^{(n)}\}$. Étant donné que nous sommes dans le cadre de l'apprentissage non supervisé, ces points ne sont associés à aucune étiquette. Nous souhaitons modéliser les données en spécifiant une distribution conjointe $p(x^{(i)}, z^{(i)}) = p(x^{(i)}|z^{(i)})p(z^{(i)})$. Ici, $z^{(i)}$ suit une loi multinomiale (Φ) (où Φ_j 0, $\sum_{i=1}^{k} \Phi_j = 1$, et le paramètre Φ_j donne la probabilité que $z^{(i)} = j$), et $x^{(i)}|z^{(i)} = j$ suit une loi normale N (μ_j, Σ_j). Nous désignons par k le nombre de valeurs que peuvent prendre les $z^{(i)}$. En pratique, k représente le nombre choisi de clusters. Ainsi, notre modèle pose que chaque $x^{(i)}$ a été généré en choisissant aléatoirement $z^{(i)}$ dans $\{1, ..., k\}$, puis en tirant $x^{(i)}$ à partir de l'un des k gaussiennes selon $z^{(i)}$. Cela est appelé modèle de mélange de gaussiennes (Gaussien mixture models). Notez également que les $z^{(i)}$ sont des variables aléatoires latentes, ce qui signifie qu'elles sont cachées/non observées. C'est ce qui rend notre problème d'estimation difficile. La distribution de la probabilité de la variable aléatoire X peut être exprimée comme suivant:

$$p(x; \mu, \sigma) = \sum_z p(x, z; \mu, \sigma, \Phi) = \sum_z p(z; \Phi)p(x|z; \mu, \sigma) = \sum_z \pi_z \mathcal{N}(x; \mu_z, \sigma_z)$$

Où $\pi_z = p(z; \Phi)$ représente l'amplitude de la loi normale suivie par cluster z. La fonction de densité de probabilité de la distribution gaussienne multivariée est donnée par :

$$\mathcal{N}(x; \mu_z, \Sigma_z) = \frac{1}{(2\pi)^{d/2}|\Sigma_z|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_z)^T \Sigma_z^{-1}(x - \mu_z)\right)$$

où $x \in \mathbb{R}^d$ est un vecteur de variables aléatoires continues, $\mu_z \in \mathbb{R}^d$ est le vecteur moyen, $\Sigma_z \in \mathbb{R}^{d \times d}$ est la matrice de covariance, $|\Sigma|$ est le déterminant de la matrice de covariance et T est l'opérateur de transposition.

Les modèles de mélange gaussien supposent que chaque classe latente a un ensemble différent de moyennes et de covariances. Cependant, comme chaque classe est inconnue, nous devons commencer par initialiser ces paramètres et les mettre à jour de manière itérative. Les méthodes d'initialisation sont une étape importante dans la modélisation de mélange qui peuvent grandement affecter la cohérence et l'exactitude des résultats. Ainsi, il vaut la peine de mentionner brièvement quelques techniques.

Une approche consiste à échantillonner aléatoirement k nombres de moyennes et de covariances dans la plage de nos données. Cependant, comme l'algorithme EM est un algorithme de "hill climbing" (c'est-à-dire de maximisation), le choix aléatoire des points de départ peut altérer les performances. Le package R

mclust aborde le problème en sélectionnant des moyennes et des covariances initiales par l'application de la classification hiérarchique - une technique de classification non supervisée qui regroupe de manière itérative les points/groupes jusqu'à ce que le nombre souhaité de clusters soit atteint.

2.3.1 Etape d'Espérance

L'étape E, ou étape d'Espérance, est la première étape de l'algorithme EM pour les modèles de mélange gaussien. Au cours de cette étape, les paramètres de chaque distribution de Gauss sont traités comme des constantes et la responsabilité de chaque point de données pour chaque classe cachée est calculée. La responsabilité est une mesure de la probabilité relative que chaque point de données appartienne à une classe donnée. Ces responsabilités sont utilisées pour pondérer l'importance de chaque point de données dans la mise à jour des paramètres de la distribution de Gauss lors de l'étape M suivante. L'objectif de l'étape E est d'estimer les valeurs des responsabilités, qui peuvent être interprétées comme les poids assignés à chaque classe pour chaque point de données. Cette étape est itérative, car la valeur des responsabilités est utilisée pour mettre à jour les paramètres de la distribution de Gauss dans l'étape M, qui est ensuite répétée jusqu'à ce que les paramètres convergent. La valeur de responsabilité r_{iz} qui mesure la probabilité relative que le point de données x_i appartienne au cluster z est calculé comme suit:

$$r_{iz} = \frac{\pi_z \mathcal{N}(x_i{:}\mu_z, \Sigma_z)}{\sum_c \pi_c \mathcal{N}(x_i{:}\mu_c, \Sigma_c)}$$

2.3.2 Etape de Maximisation

La maximisation est la deuxième étape de l'algorithme EM (Expectation-Maximisation) utilisé en apprentissage automatique pour estimer les paramètres d'un modèle probabiliste. Après l'étape d'espérance (E-step) où l'on a calculé les valeurs de responsabilité pour chaque observation, l'étape de maximisation (M-step) vise à trouver les valeurs optimales des paramètres du modèle qui maximisent la vraisemblance des données observées. Dans le cadre des modèles de mélange gaussien, l'étape de maximisation consiste à ajuster les paramètres des gaussiennes pour améliorer la qualité du modèle. Cette étape est itérative et continue jusqu'à ce que les valeurs de paramètres convergent vers une valeur stable ou jusqu'à ce qu'un critère d'arrêt soit atteint. La précision et la stabilité des résultats obtenus à partir de l'algorithme EM dépendent en grande partie de la qualité des initialisations et du nombre d'itérations effectuées.

Nous cherchons à dériver les formules pour les nouveaux paramètres gaussiens tout en cherchant à maximiser la fonction de vraisemblance:

$$\mathcal{L}(\mu, \sigma) = \sum_{i=1}^{n} \log p(\mathbf{x}^{(i)}; \mu, \sigma) = \sum_{i=1}^{n} \log \sum_{\mathbf{z}} Q(\mathbf{z}) \frac{p(\mathbf{x}^{(i)}, \mathbf{z}; \mu, \sigma)}{Q(\mathbf{z})}$$

En utilisant l'inégalité de Jensen pour les fonctions convexes, on obtient le résultat suivant :

$$\mathcal{L}(\mu, \sigma) \geq \sum_{i=1}^{n} \sum_{\mathbf{z}} Q(\mathbf{z}) \log \frac{p(\mathbf{x}^{(i)}, \mathbf{z}; \mu, \sigma)}{Q(\mathbf{z})}$$

Pour rendre la borne serrée pour une valeur particulière de (μ, σ), nous devons nous assurer que l'étape impliquant l'inégalité de Jensen dans notre dérivation ci-dessus soit vraie avec l'égalité. Pour cela, nous savons qu'il suffit que l'espérance soit prise sur une variable aléatoire à valeur "constante". C'est-à-dire que nous avons besoin que $\frac{p(\mathbf{x}^{(i)},\mathbf{z};\mu,\sigma)}{Q(\mathbf{z})} = c$ pour une constante c qui ne dépend pas de z. Cela est facilement réalisé en choisissant $Q(z)\ \alpha\ p(x, z; \mu, \sigma)$. Cela engendre:

$$Q(z) = p(z|x; \mu, \sigma) = r_z$$

Cela signifie que nous avons trouvé la borne inférieure de l'évidence (ELBO) et la maximiser signifie maximiser la fonction de vraisemblance. Si nous remplaçons Q(z) par la nouvelle expression, nous trouvons:

$$ELBO(x; r_z, \mu, \sigma) =$$

$$\sum_{i=1}^{n}\sum_{j=1}^{k} r_{z^{(i)}=j} \log \frac{\frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)}-\mu_j)^T\Sigma_j^{-1}(x^{(i)}-\mu_j)\right)\pi_{z^{(i)}=j}}{r_{z^{(i)}=j}}$$

Ensuite, dans l'étape M, nous devons maximiser, par rapport à nos paramètres, la quantité ELBO. Si nous dérivons par rapport au paramètres, nous obtenons les expressions des nouveaux paramètres.

$$m_j = \sum_{i=1}^{n} r_{ij}$$

$$\pi_{z^{(i)}=j} = \frac{m_j}{n}$$

$$\mu_j = \frac{1}{m_j}\sum_{i=1}^{n} r_{ij}x_i$$

$$\Sigma_j = \frac{1}{m_j}\sum_{i=1}^{n} r_{ij}(x_i - \mu_j)^T(x_i - \mu_j)$$

Dans l'étape de maximisation (M-step), il est important de définir des critères de convergence pour garantir que l'algorithme a convergé vers une solution stable. Les critères de convergence courants incluent la tolérance de la variation des paramètres, qui arrête l'algorithme si les valeurs des paramètres ne changent plus significativement entre deux itérations consécutives. Une autre mesure de convergence est la log-vraisemblance, qui mesure à quel point les paramètres actuels correspondent aux données d'entraînement. Si la log-vraisemblance ne s'améliore plus entre deux itérations consécutives, l'algorithme peut être arrêté. Il est également possible de définir un nombre maximal d'itérations pour garantir que l'algorithme ne fonctionne pas indéfiniment. Les critères de convergence doivent être choisis judicieusement en fonction du problème et des données, afin d'assurer une convergence rapide et précise vers une solution optimale.

Algorithm 3 Algorithme de densité des données EM pour GMM

Input: Données d'entraînement X, nombre de clusters K, critère d'arrêt ϵ
Output: Paramètres optimaux du modèle de mélange gaussien
Initialiser les paramètres $\theta = (\pi_1, \mu_1, \Sigma_1, ..., \pi_K, \mu_K, \Sigma_K)$;
Initialiser la log-vraisemblance à $-\infty$;
repeat
E-étape : Calculer la responsabilité r_{ic} pour chaque observation x_i et chaque cluster c;

$$r_{ic} = \frac{\pi_c N(x_i|\mu_c, \Sigma_c)}{\sum_{k=1}^{K} \pi_k N(x_i|\mu_k, \Sigma_k)}$$

M-étape : Mettre à jour π_c, μ_c, et Σ_c :

$$\pi_c = \frac{1}{n} \sum_{i=1}^{n} r_{ic}$$

$$\mu_c = \frac{\sum_{i=1}^{n} r_{ic} x_i}{\sum_{i=1}^{n} r_{ic}}$$

$$\Sigma_c = \frac{\sum_{i=1}^{n} r_{ic}(x_i - \mu_c)(x_i - \mu_c)^T}{\sum_{i=1}^{n} r_{ic}}$$

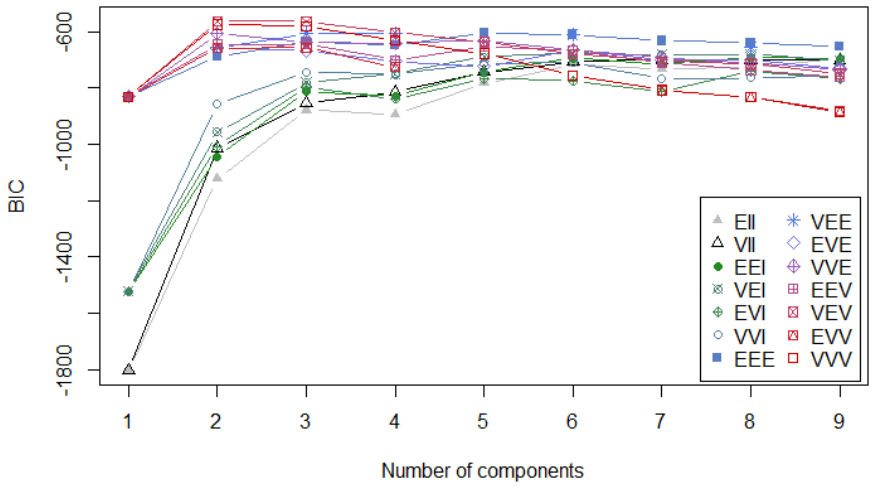
Calculer la log-vraisemblance et comparer à la précédente;
until *convergence*;

2.3.3 Exemple d'application pratique

Nous allons de nouveau travailler avec la base de données iris de R. Pour ce faire, nous allons prétendre une fois de plus que nous ne disposons pas de la variable "Species".

Comme nous devrons de nouveau choisir le nombre de clusters, nous allons utiliser la fonction mclustBIC du package mclust pour déterminer quel type de modèle et quel nombre de clusters est le meilleur en utilisant le critère de BIC.

```
> mydata = iris[-5]
> mBIC = mclustBIC(mydata)
fitting ...
  |=============================================================================================================| 100%
> summary(mBIC)
Best BIC values:
             VEV,2        VEV,3      VVV,2
BIC      -561.7285 -562.5522369 -574.01783
BIC diff    0.0000   -0.8237748  -12.28937
```

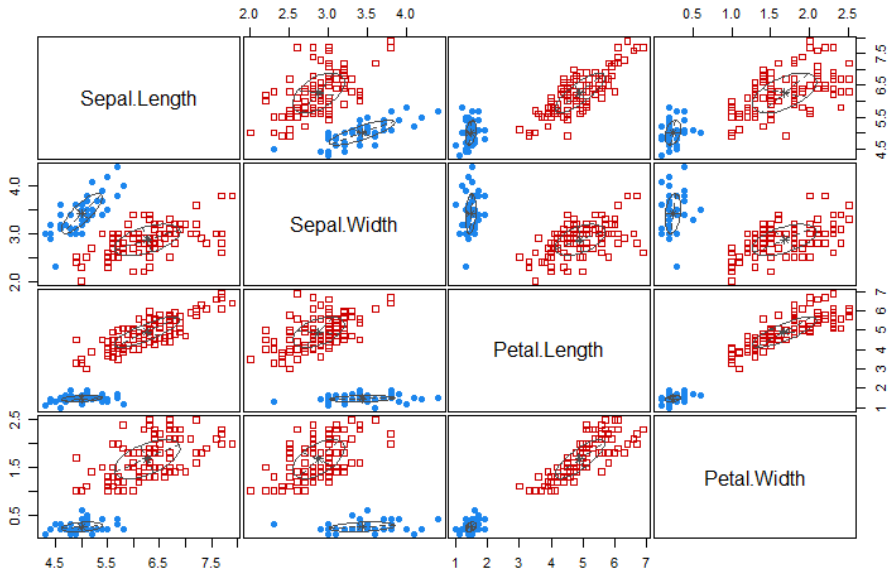


Les meilleurs modèles selon le critère BIC sont les modèles (VEV, 2), (VEV, 3) et (VVV, 2). Dans le modèle VEV, les données dans chaque cluster sont supposées suivre une loi normale multivariée avec une même matrice de covariance pour tous les clusters. La forme de la région de l'espace des variables où les points de données sont plus susceptibles d'être générés est représentée par une forme ellipsoïdale en trois dimensions, contrôlée par la matrice de covariance. "Ellipsoidal, equal shape" signifie que chaque cluster est représenté par une ellipsoïde de même forme, mais peut avoir des centres et des tailles différents. En revanche, le modèle VVV suppose que chaque cluster est modélisé par une ellipse pouvant avoir des volumes, des formes et des orientations différents.

```
> mBIC1 = Mclust(iris[-5], x = mBIC)
> summary(mBIC1, parameters = TRUE)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VEV (ellipsoidal, equal shape) model with 2 components:

 log-likelihood   n df       BIC       ICL
       -215.726 150 26 -561.7285 -561.7289

Clustering table:
  1   2
 50 100

Mixing probabilities:
        1         2
0.3333319 0.6666681

Means:
                  [,1]     [,2]
Sepal.Length 5.0060022 6.261996
Sepal.Width  3.4280049 2.871999
Petal.Length 1.4620007 4.905992
Petal.Width  0.2459998 1.675997

Variances:
[,,1]
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length   0.15065114  0.13080115   0.02084463  0.01309107
Sepal.Width    0.13080115  0.17604529   0.01603245  0.01221458
Petal.Length   0.02084463  0.01603245   0.02808260  0.00601568
Petal.Width    0.01309107  0.01221458   0.00601568  0.01042365
[,,2]
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    0.4000438  0.10865444    0.3994018  0.14368256
Sepal.Width     0.1086544  0.10928077    0.1238904  0.07284384
Petal.Length    0.3994018  0.12389040    0.6109024  0.25738990
Petal.Width     0.1436826  0.07284384    0.2573899  0.16808182
```

On utilise alors la fonction Mclust pour entraîner le modèle avec les hyper-paramètres (VEV,2).

Le modèle (VEV, 2) identifie deux clusters, l'un ayant 50 observations et l'autre ayant 100 observations. Les moyennes et les matrices de covariance pour chaque cluster sont également fournies dans le résultat de la fonction summary. Cependant, étant donné que les données initiales concernent 3 types de fleurs, il est également pertinent de considérer les modèles à 3 clusters. Le modèle (VEV, 3) ayant le meilleur score BIC, nous allons l'utiliser pour créer une nouvelle classification.



Le modèle (VEV, 3) permet d'identifier trois clusters, avec des tailles de 50, 45 et 55 observations respectivement. Les vecteurs de moyenne et les matrices de covariance pour chaque cluster sont également fournis dans le résultat de la fonction summary. Le fait que chaque cluster ait presque une proportion équivalente d'observations est un indicateur positif, étant donné que les données d'origine comportaient 3 types de fleurs avec 50 observations pour chaque type.

```
> mBIC2 = Mclust(iris[-5], G = 3, modelNames = "VEV")
fitting ...
  |============================================================
> summary(mBIC2, parameters = TRUE)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VEV (ellipsoidal, equal shape) model with 3 components:

 log-likelihood   n df       BIC       ICL
       -186.074 150 38 -562.5522 -566.4673

Clustering table:
 1  2  3
50 45 55

Mixing probabilities:
        1         2         3
0.3333333 0.3005423 0.3661243

Means:
              [,1]     [,2]     [,3]
Sepal.Length 5.006 5.915044 6.546807
Sepal.Width  3.428 2.777451 2.949613
Petal.Length 1.462 4.204002 5.482252
Petal.Width  0.246 1.298935 1.985523

Variances:
[,,1]
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length   0.13320850  0.10938369  0.019191764 0.011585649
Sepal.Width    0.10938369  0.15495369  0.012096999 0.010010130
Petal.Length   0.01919176  0.01209700  0.028275400 0.005818274
Petal.Width    0.01158565  0.01001013  0.005818274 0.010695632
[,,2]
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length   0.22572159  0.07613348   0.14689934  0.04335826
Sepal.Width    0.07613348  0.08024338   0.07372331  0.03435893
Petal.Length   0.14689934  0.07372331   0.16613979  0.04953078
Petal.Width    0.04335826  0.03435893   0.04953078  0.03338619
[,,3]
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length   0.42943106  0.10784274   0.33452389  0.06538369
Sepal.Width    0.10784274  0.11596343   0.08905176  0.06134034
Petal.Length   0.33452389  0.08905176   0.36422115  0.08706895
Petal.Width    0.06538369  0.06134034   0.08706895  0.08663823
```

En utilisant le modèle (VEV, 3), nous avons effectué une prédiction pour les six premières observations de notre jeu de données. Nous avons constaté que la probabilité que ces six observations appartiennent au premier cluster était de 1,

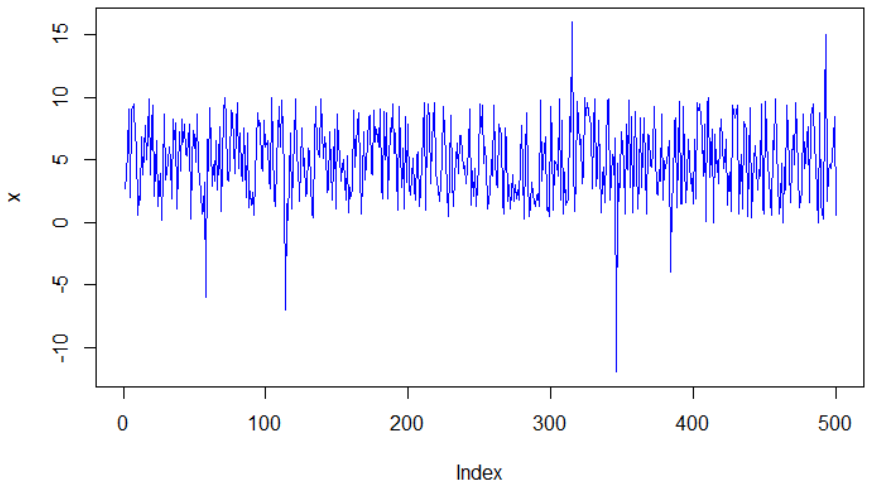
ce qui suggère une forte certitude que ces observations appartiennent effectivement à ce cluster et donc à la même catégorie.

```
> head(mBIC2$z)
     [,1]         [,2]         [,3]
[1,]    1 4.916819e-40 3.345948e-29
[2,]    1 5.846760e-29 1.599452e-23
[3,]    1 2.486168e-33 2.724243e-25
[4,]    1 2.050996e-28 2.180764e-22
[5,]    1 8.283066e-42 8.710458e-30
[6,]    1 2.118466e-41 1.791499e-29
```

2.3.4 Application dans la Détection des Anomalies

Nous allons créer un ensemble de données d'échantillonnage aléatoire pour ce modèle et le visualiser dans un graphique pour le vérifier visuellement.

```
set.seed(1)
n = 500
x = runif(n)*10
x[sample(1:n, 10)] <- sample(-20:20, 10)
plot(x, col="blue", type='l', pch=19)
```



Le but serait donc de pouvoir détecter les anomalies i.e les points aberrants dans le graphe.

```
x = scale(x)[,1]
xfit = Mclust(x, G=3, model="V")
```

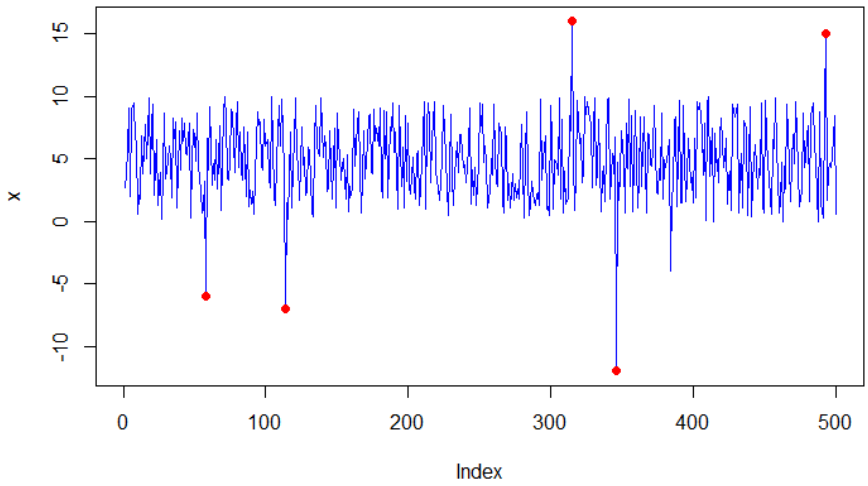
On normalise nos données et on entraîne un modèle de (V,3).

```
> summary(xfit)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust V (univariate, unequal variance) model with 3 components:

 log-likelihood   n df       BIC       ICL
      -661.5075 500  8 -1372.732 -1455.607

Clustering table:
  1   2   3
  6 334 160


pred = predict(xfit)
xpred = pred$z[,1]

thr = quantile(xpred, .99)
print(thr)
```

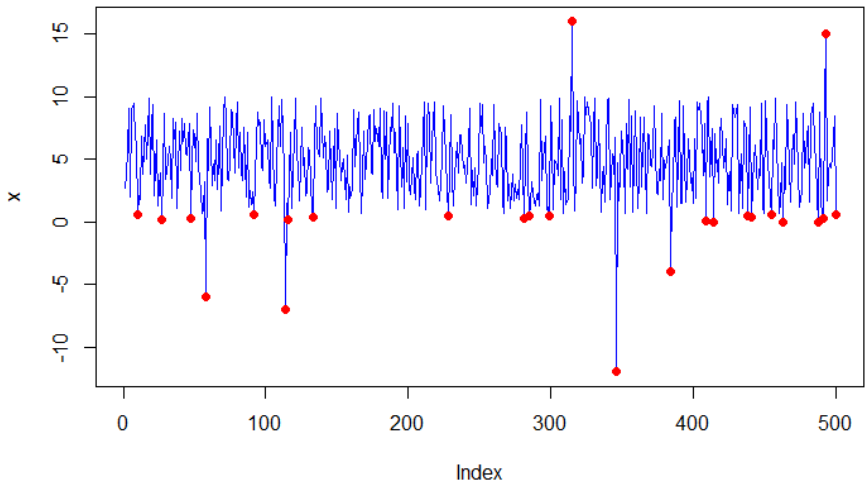
Nous allons utiliser la première colonne de la propriété z. Ensuite, nous allons extraire les valeurs seuils à partir des scores de probabilité en utilisant la fonction quantile(). Ici, 0,99 signifie que nous allons trouver la valeur du quantile de 0,99.

```
outliers = which(xpred >= thr)
index = x[outliers]
plot(x, col="blue", type='l', pch=19)
points(outliers,index, pch=19, col="red")
```

En utilisant le seuil 0.99, nous trouverons les échantillons dont les scores sont égaux ou supérieurs à ce seuil. Ensuite, nous obtiendrons l'indice de ces valeurs. Enfin, nous visualiserons les résultats dans un graphique en mettant en évidence les anomalies avec une couleur.



Il est possible d'expérimenter avec différents seuils, tels que 0,95 ou 0,90, pour détecter d'autres outliers dans les données. Par exemple, voici le résultat pour le seuil 0,95 :



Toutefois, le choix du seuil approprié doit être basé sur la nature des données et le contexte de l'analyse. Si le seuil est trop élevé, des outliers importants

pourraient être manqués, tandis qu'un seuil trop bas peut produire un grand nombre d'outliers qui pourraient fausser l'analyse.

3 Défis de l'utilisation des algorithmes d'apprentissage non supervisé

3.1 Choix du Nombre de Clusters

Le choix du nombre de clusters à utiliser dans un algorithme de clustering est une étape importante de l'analyse de données. Il existe plusieurs méthodes pour déterminer le nombre de clusters approprié pour un jeu de données donné.

L'une des méthodes les plus couramment utilisées est la méthode du coude (ou méthode du genou), qui consiste à tracer un graphique de la valeur de l'indice de performance de l'algorithme en fonction du nombre de clusters. L'indice de performance peut être l'indice de Davies-Bouldin, l'indice de silhouette ou tout autre indice de qualité de clustering. Le nombre de clusters optimal est généralement considéré comme celui où le graphique forme un coude. Cependant, il peut être difficile de déterminer avec précision où se trouve le coude et il peut être nécessaire de faire plusieurs essais avec différentes valeurs de k pour trouver le meilleur nombre de clusters.

Une autre méthode consiste à utiliser des techniques de réduction de dimensionnalité, telles que l'analyse en composantes principales (ACP), pour visualiser les données et essayer de déterminer visuellement le nombre de clusters approprié. Cependant, cette approche peut être subjective et il peut être difficile de déterminer avec précision le nombre de clusters à utiliser.

En fin de compte, le choix du nombre de clusters dépend de l'objectif de l'analyse et de la structure des données. Il peut être nécessaire de faire plusieurs essais avec différentes valeurs de k pour trouver le meilleur nombre de clusters pour un jeu de données donné.

3.2 Traitement des Données Manquantes ou Bruitées

Le traitement des données manquantes ou bruitées est une étape importante dans l'analyse de données. Les données manquantes peuvent être causées par des erreurs de saisie, des problèmes de collecte de données ou tout autre facteur qui empêche une valeur de se trouver dans une cellule de données. Les données bruitées, quant à elles, sont des données qui ont été altérées par des erreurs ou des distorsions, ce qui peut affecter la qualité des résultats de l'analyse.

Il existe plusieurs méthodes pour traiter les données manquantes ou bruitées. L'une des méthodes les plus courantes consiste à utiliser la moyenne, la mediane ou la mode des valeurs environnantes pour remplacer la valeur manquante ou bruitée. Cependant, cette approche peut ne pas être appropriée dans tous les cas et peut affecter la qualité des résultats de l'analyse.

Une autre méthode consiste à utiliser des algorithmes de imputation de données, qui sont conçus pour remplacer les valeurs manquantes ou bruitées

par des valeurs estimées de manière plus précise. Il existe plusieurs algorithmes d'imputation de données, tels que l'imputation par la moyenne, l'imputation par la régression et l'imputation par l'analyse en composantes principales. Ces algorithmes peuvent être utilisés seuls ou en combinaison pour remplacer les valeurs manquantes ou bruitées de manière plus précise et éviter d'influencer négativement les résultats de l'analyse.

Il est également possible de supprimer simplement les observations contenant des valeurs manquantes ou bruitées. Cette approche peut être appropriée si le nombre de valeurs manquantes ou bruitées est faible par rapport à l'ensemble des données. Cependant, il est important de prendre en compte l'impact de la suppression de ces observations sur la qualité et la représentativité des données restantes.

En fin de compte, le choix de la méthode de traitement des données manquantes ou bruitées dépend de la nature des données, de l'objectif de l'analyse et de la quantité de données manquantes ou bruitées. Il est important de choisir une méthode qui minimise l'impact sur la qualité et la représentativité des données, tout en permettant d'obtenir des résultats précis et fiables.

3.3 Traitement des Données de Grande Dimension

Le traitement des données de grande dimension peut être un défi lors de l'analyse de données. Les données de grande dimension ont un grand nombre de variables ou de caractéristiques, ce qui peut rendre difficile leur visualisation et leur traitement par les algorithmes d'apprentissage automatique.

Il existe plusieurs méthodes pour traiter les données de grande dimension. L'une des approches les plus courantes consiste à utiliser des techniques de réduction de dimensionnalité, telles que l'analyse en composantes principales (ACP) ou la réduction par puits de données (DDR), qui permettent de réduire le nombre de dimensions des données tout en conservant le maximum d'informations. Ces techniques peuvent être utiles pour visualiser et analyser les données de grande dimension, mais elles peuvent également conduire à une perte d'informations.

Une autre approche consiste à utiliser des algorithmes de machine learning qui sont spécialement conçus pour traiter les données de grande dimension, tels que les arbres de décision, les réseaux de neurones ou les modèles linéaires généralisés. Ces algorithmes peuvent être plus efficaces pour traiter les données de grande dimension, mais ils peuvent également être plus complexes et prendre plus de temps à entraîner.

En fin de compte, le choix de la méthode de traitement des données de grande dimension dépend de l'objectif de l'analyse et de la nature des données. Il est important de trouver un équilibre entre la précision des résultats et la complexité de l'analyse. Si l'objectif est de visualiser et de comprendre les données de manière générale, des techniques de réduction de dimensionnalité peuvent être utiles. Si l'objectif est de prédire avec précision des valeurs cibles à partir des données, des algorithmes de machine learning peuvent être plus adaptés.

Il est également important de prendre en compte les limites de chaque méthode et de choisir celle qui convient le mieux à la situation. Par exemple, les techniques de réduction de dimensionnalité peuvent être moins précises que les algorithmes de machine learning, mais elles peuvent être plus faciles à comprendre et à interpréter. Les algorithmes de machine learning, quant à eux, peuvent être plus précis, mais ils peuvent être plus difficiles à comprendre et à interpréter.

En fin de compte, le choix de la méthode de traitement des données de grande dimension dépend de l'objectif de l'analyse et de la nature des données. Il est important de trouver un équilibre entre la précision des résultats et la complexité de l'analyse pour obtenir des résultats pertinents et fiables.

4 Impact des algorithmes d'apprentissage non supervisé sur l'industrie et la société

L'apprentissage non supervisé est un domaine de l'IA qui a un impact significatif dans de nombreux domaines de la société, allant des industries manufacturières à la santé, en passant par la finance, le marketing et l'analyse des réseaux sociaux. Cette approche permet d'extraire des informations utiles à partir de données non étiquetées, aidant ainsi les entreprises à prendre des décisions éclairées.

Dans l'industrie manufacturière, l'apprentissage non supervisé peut aider à améliorer la qualité des produits tout en réduisant les coûts de production. Les entreprises peuvent utiliser des techniques de clustering pour identifier des groupes de produits similaires et optimiser les processus de production en conséquence.

Dans le domaine de la santé, l'apprentissage non supervisé peut aider à diagnostiquer les maladies et à identifier les groupes de patients ayant des caractéristiques similaires. Cette approche peut être utilisée pour identifier les groupes de patients présentant des profils de symptômes similaires, ce qui permet de proposer des traitements personnalisés.

Dans le domaine de la finance, l'apprentissage non supervisé peut aider à détecter les fraudes, identifier les modèles de dépenses et de revenus, et prévoir les tendances économiques.

En marketing, l'apprentissage non supervisé peut aider à segmenter les clients en groupes et à personnaliser les campagnes publicitaires en fonction de leurs préférences individuelles.

En analysant les réseaux sociaux, l'apprentissage non supervisé peut identifier des groupes de personnes ayant des intérêts similaires, aidant ainsi les entreprises à cibler des publicités en fonction de ces intérêts.

Cependant, il est important de prendre en compte les risques potentiels associés à l'apprentissage non supervisé. Si les algorithmes sont mal utilisés ou mal compris, ils peuvent amplifier les préjugés et les discriminations présents dans les données d'entrée. Il est donc important de valider les résultats et de s'assurer que les modèles sont justes et équitables.

De plus, les résultats de l'apprentissage non supervisé peuvent être difficiles à interpréter et à expliquer, en particulier pour les algorithmes de clustering. Les clusters peuvent sembler évidents pour l'algorithme, mais peuvent être difficiles à comprendre pour les humains. Il est donc important de valider les résultats en utilisant des méthodes de visualisation et de tests supplémentaires.

Enfin, l'apprentissage non supervisé peut poser des problèmes de confidentialité si les données utilisées contiennent des informations sensibles ou personnelles. Les entreprises doivent donc prendre des mesures pour protéger les données et respecter la vie privée des utilisateurs.

En somme, l'apprentissage non supervisé offre de nombreuses opportunités pour extraire des informations utiles à partir de données non étiquetées. Toutefois, il est important de prendre en compte les limites et les précautions associées à son utilisation afin de garantir des résultats de qualité et de respecter la vie privée des utilisateurs.

Il convient également de noter que l'apprentissage non supervisé peut être plus complexe que l'apprentissage supervisé, car il nécessite souvent une exploration plus approfondie des données et des techniques de modélisation plus avancées. Cela peut rendre l'apprentissage non supervisé moins accessible aux utilisateurs moins expérimentés dans le domaine.

Malgré ces défis, l'apprentissage non supervisé reste un domaine de recherche actif et en constante évolution, avec de nouvelles techniques et applications émergentes. Il est donc important de continuer à explorer les avantages et les limites de cette technique pour maximiser son potentiel et minimiser les risques associés.

En fin de compte, le choix entre l'apprentissage supervisé et non supervisé dépendra des besoins spécifiques de la tâche à accomplir et des données disponibles. Les deux approches ont leurs avantages et leurs limites, et il est important de bien comprendre ces différences pour faire le choix le plus éclairé et le plus approprié.

Conclusion

En conclusion, les algorithmes d'apprentissage non supervisé sont des méthodes puissantes pour explorer et comprendre les données sans la nécessité de l'intervention humaine pour fournir des étiquettes ou des réponses préalables. Les trois types d'algorithmes d'apprentissage non supervisé les plus couramment utilisés sont les algorithmes de regroupement, de réduction de dimensionalité et de densité de données. Chacun de ces types peut être appliqué à différentes situations et peut être utilisé pour résoudre des problèmes de la vie réelle.

Les exemples d'algorithmes que nous avons examinés, tels que K-means, ACP et EM Gaussien Mixture Models, ont démontré leur efficacité dans des applications telles que la reconnaissance des formes, la détection des anomalies, la réduction de dimensionalité,... etc.

Cependant, l'utilisation de ces algorithmes présente des défis tels que le choix du nombre de clusters, le traitement des données manquantes ou bruitées, le traitement des données de grande dimension, etc. Par conséquent, il est essentiel de bien comprendre les caractéristiques des données à traiter et de choisir l'algorithme approprié en conséquence.

En fin de compte, les algorithmes d'apprentissage non supervisé ont un impact significatif sur l'industrie et la société en permettant une meilleure compréhension des données, une amélioration de la prise de décision et une automatisation accrue de diverses tâches.

References

[1] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[2] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[3] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[4] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[5] Geoffrey J McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.