University of Caen-Normandy
Lab of Mathematics Nicolas Oresme

Statistics & Data Science
UMR CNRS 6139

Algorithme des centres moblies (K-means)

par Faïcel Chamroukhi

Table des matières

1 Objectifs

L'objectif de ce travail est d'implémenter l'algorithme des centres mobiles (connu sous le nom de K-means) pour la classification automatique (en anglais *clustering*) et de l'appliquer, par exemple pour la classification de données multidimensionnelles.

2 Contexte

2.1 Qu'est ce que la classification automatique ?

En quelques mots, la classification automatique est la tâche qui consiste à regrouper, de façon non supervisée (ç-à-d sans l'aide préalable d'un expert), un ensemble d'objets ou plus largement de données, de telle manière que les objets d'un même groupe (appelé cluster) sont plus proches (au sens d'un critère de (dis)similarité choisi) les unes au autres que celles des autres groupes (clusters). Il s'agit d'une tâche principale dans la fouille exploratoire de données, et une technique d'analyse statistique des données très utilisée dans de nombreux domaines, y compris l'apprentissage automatique, la reconnaissance de formes, le traitement de signal et d'images, la recherche d'information, etc.
L'idée est donc de découvrir des groupes au sein des données, de façon automatique.

2.2 Quelles données ?

Les données traitées en classification automatique peuvent être des images, signaux, textes, autres types de mesures, etc. Dans le cadre de ce travail les données seront des données multidimensionnelles, par exemple une image (couleur). Chaque donnée est donc composée de plusieurs variables (descripteurs). Pour le cas de données multidimensionnelles standard, chaque donnée étant de dimension d (donc un point dans l'espace $\mathbb{R}^d$) et éventuellement étiquetée (dans le cas où l'on connaitrait sa classe d'appartenance (le "label")) et peut donc être modélisée, par exemple, par un e structure contenant au moins les coordonnées du point et le champ "label". Les n données peuvent donc être modélisées comme étant un tableau de n éléments, chaque élément du tableau étant une structure "donnée" comme décrite précédemment. Dans le cas d'une image (couleur), l'image contenant n lignes et m colonnes et donc $n \times m$ pixels couleurs, chaque pixel est composé de trois composantes RVB, peut être modélisée par un tableaux de $n \times m$ structures. Chaque structure "pixel" est composée au moins des champs, couleur et "label".
Pour cette partie `ImageSeg-Kmeans` du travail, on commencera par des données simulées, ensuite on traitera les données Iris et en fin une image couleur.

2.3 Qu'est ce qu'une classe ?

En gros, une classe (ou groupe) est un ensemble de données formée par des données homogènes (qui "se ressemblent" au sens d'un critère de similarité (distance, densité de probabilité, etc)). Par exemple, si on a une base d'image de chiffres manuscrits, on aura la classes des zéros, la classe des uns, etc. De même pour le cas d'images de lettres manuscrites. Une classe peut être aussi une région dans une image couleur, un évènement particulier dans un signal sonore, la classe spam et classes non spam dans le cas détection de spams dans un mail, etc.

2.4 Combien de classes ?

Le nombre de groupes (qu'on notera K), pour commencer, peut être supposé fixe (donné par l'utilisateur). C'est le cas par exemple si l'on s'intéresse à classer des images de chiffres manuscrits (nombre de classes = 10 : $0, ..., 9$) ou de lettres manuscrites (nombre de classes = nombres de caractères de l'alphabet), etc.

Néanmoins, il existe des critères, dits de *choix de modèle*, qui permettent de choisir le nombre optimal de classes dans un jeu de données. on verra par la suite comment on pourrait calculer le nombre de groupe optimal par ce type de critères.

2.5 Exemple Introductif

Considérons par exemple une image couleur, chaque image contient n pixels $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, chaque pixel $\mathbf{x}_i$ contient $d = 3$ valeurs (RGB). On peut donc représenter donc le $i$ème pixel $(i = 1, \ldots, n)$ par un vecteur $\mathbf{x}_i$ de dimension $d = 3 : \mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3})^T \in \{0, 1, ..., 255\}^3$. Si l'on connaît les classes de certains pixels, on pourra prédire les classes des autres pixels on choisissant une mesure de (dis)similarité, par exemple une simple distance, ou une mesure de probabilité, etc. Chaque pixel à classer aura donc la classe de celui qui lui est le plus proche au sens de la mesure de (dis)similarité choisie. Ceci peut être utilisé par exemple en segmentation d'image. De manière générale, on peut représenter les données comme un ensemble de vecteurs $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, chaque $\mathbf{x}_i$ est composée de d composantes réelles : $\mathbf{x}_i = (x_{i1}, ..., x_{ij}, ..., x_{id})^T \in \mathbb{R}^d$.

3 Travail à réaliser

Il s'agit d'implémenter (sous `Matlab`, `R`, `Pyhton` ou `Octave`) l'algorithme des centres mobiles (K-means) pour la classification automatique d'un ensemble de données $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ K-means minimise le critère d'erreur (distorsion) suivant par rapport aux centres des classes $\mathbf{\Psi} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$ et les classes $\mathbf{z} = (z_1, \ldots, z_n) : (\mathbf{z}, \mathbf{\Psi})$:

$$\mathcal{J}(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \mathbf{z}) \quad = \quad \sum_{k=1}^{K} \sum_{i=1}^{n} z_{ik} \parallel \mathbf{x}_i - \boldsymbol{\mu}_k \parallel^2 \tag{1}$$

qui correspond à la distance euclidienne totale entre chaque données $\mathbf{x}_i$ et le centre $\boldsymbol{\mu}_{z_i}$ dont elle est la plus proche au sens de la distance Euclidienne :

$$\parallel \mathbf{x}_i - \boldsymbol{\mu}_k \parallel = d(\mathbf{x}_i, \boldsymbol{\mu}_k) = \sqrt{\sum_{j=1}^{d} (x_{ij} - \mu_{kj})^2} \tag{2}$$

Dans l'expression du critère $\mathcal{J}$, z_{ik} est une variable binaire qui vaut 1 si la classe du $i$ème exemple $\mathbf{x}_i$ est k et 0 sinon.

3.1 Algorithme K-means

L'algorithme K-means est composée des trois étapes suivantes :

— **Initialisation :** On initialise les centres des classes $(\boldsymbol{\mu}_1^{(0)}, \ldots, \boldsymbol{\mu}_K^{(0)})$ (à votre choix) pour donner le pas de départ de l'algorithme (par exemple on choisissant aléatoirement des centres "virtuels", ou K données parmi les données à traiter). Il s'agit donc de démarrer à l'itération $t = 0$ avec des valeurs initiales pour les paramètres du modèle $(\boldsymbol{\mu}_1^{(0)}, \ldots, \boldsymbol{\mu}_K^{(0)})$.

1. **Etape d'affectation (classification)** : Chaque donnée est assignée à la classe du centre dont elle est la plus proche : $\forall i = 1, \ldots, n$

$$z_{ik}^{(t)} = \begin{cases} 1 \text{ si } k = \arg \min\limits_{z \in \{1,,\ldots,K\}} \parallel \mathbf{x}_i - \boldsymbol{\mu}_z \parallel^2 \\ 0 \text{ sinon.} \end{cases} \tag{3}$$

2. **Etape de recalage des centres** : le centre $\boldsymbol{\mu}$ de chaque classe k est recalculé comme étant la moyenne arithmétique de toutes les données appartement à cette classe (suite à l'étape d'affectation précédente) : $\forall k = 1, \ldots, K$

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{n} z_{ik}^{(t)} \mathbf{x}_i}{\sum_{i=1}^{n} z_{ik}^{(t)}}, \tag{4}$$

t étant l'itération courante.
La convergence peut être considérée comme atteinte si la valeur relative au niveau de la distorsion J (1) devient inférieure à un seuil petit préfixé ou si un nombre maximum d'itérations préfixé a été atteint.

3.2 Classification des données d'apprentissage

Une fois l'algorithme a convergé, on a donc une estimation des classes $(z_1, \ldots, z_n)$ et des centres $(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$. On connait donc l'appartenance z_i (la classe) de chaque donnée $\mathbf{x}_i$ à partir de z_{ik}.
L'algorithme K-means est donné par le pseudo-code 1 ci-après.

Algorithm 1: Algorithme K-means

Input: Données $(\mathbf{x}_1, ..., \mathbf{x}_n)$, nombre de clusters K

Algorithme K-means :

Itération : $t \leftarrow 0$

`/* Initialisation : */`

Initialiser les centres des classes : $\mathbf{\Psi}^{(0)} = (\boldsymbol{\mu}_1^{(0)}, ..., \boldsymbol{\mu}_K^{(0)})$

`/* */`

Nombre d'itérations max : maxIter $\leftarrow 1000$

Fixer un seuil petit pour le test de convergence : $\epsilon > 0$; e.g., $\epsilon = 10^{-6}$

Distorsion : $\mathcal{J}^{(t)} \leftarrow +\infty$

`/* K-means */`

while $((converge \neq 0)$ **and** $(t \leq maxIter))$ **do**

`/* Étape affectation */`

for $i \leftarrow 1$ ***to*** n **do**

for $k \leftarrow 1$ ***to*** K **do**

Calculer $z_{ik}^{(t)}$ en utilisant l'équation (3)

end

end

`/* Étape recalage (recalcul des centres) */`

for $k \leftarrow 1$ ***to*** K **do**

Calculer $\boldsymbol{\mu}_k^{(t+1)}$ en utilisant l'équation (4) :

for $j \leftarrow 1$ ***to*** d **do**

Calculer $\mu_{kj}^{(t+1)}$

end

end

$t \leftarrow t + 1$ (itération suivante)

`/*Test de convergence*/`

Calculer la distorsion $\mathcal{J}^{(q+1)}$ en utilisant l'équation (1)

if $\left(\left(\frac{|\mathcal{J}^{(t+1)} - \mathcal{J}^{(t)}|}{|\mathcal{J}^{(t)}|} \leq \epsilon\right) || \, (t \geq maxIter)\right)$ **then**

$converge = 1$

end

end

Result: les classes $(z_1, \ldots, z_n)$ et les centres des classes $(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$

4 Données et présentation des résultats

Pour les données, vous pouvez par exemple utiliser celles-ci : Xtrain. Vous pouvez également tester votre programme sur les données iris téléchargeable ici (Iris, un jeu de données très utilisé en classification automatique).
Pour la présentation des résultats, pour commencer, les résultats trouvés peuvent être affichés directement à l'écran ou écrites dans un fichier.
Pour aller plus loin, la présentation des résultats pourra se faire par des graphiques en affichant par exemple les données dans l'espace, chaque ensemble de données appartenant à une même classe est colorée par une couleur différente, etc. Vous pouvez vous inspirez de l'exemple suivant de la figure 2. Les données pour cet exemple sont : Xtrain,

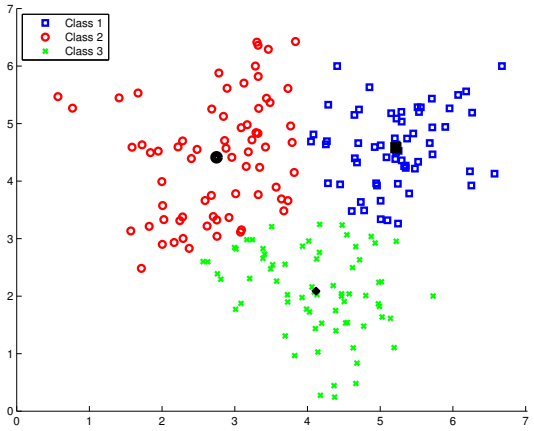

FIGURE 1 – Un exemple où l'on a trois classes. Les données sont colorées selon les classes d'appartenances fournies par l'algorithme des K-means. Les centres estimés des classes sont représentés en noir.

Si vous manipulez des données pour lesquelles vos connaissez les classes des données de test (le cas des iris par exemple), vous donnerez également le taux d'erreur de classification en comparant les classes fournies par l'algorithme avec les vraies classes (par exemple en créant une fonction `classif_error`). Pour le cas de segmentation d'images, essayer de regarder si les classes trouvées correspondent à des régions homogènes au niveau de l'image.

FIGURE 2 – Exemple de segmentationd d'image couleurs en deux classes par l'algorithme des K-means