TP/TD 10 : ALGORITHME DES K-MOYENNES ET ANALYSE EN COMPOSANTE PRINCIPALES (ACP)

COURS D'APPRENTISSAGE, ECOLE NORMALE SUPÉRIEURE, 27 NOVEMBRE 2015

Jean-Baptiste Alayrac
jean-baptiste.alayrac@inria.fr

RÉSUMÉ. Ce TP a pour but de mettre en pratique deux méthodes couramment utilisées en apprentissage statistique, à savoir l'algorithme K-means et l'ACP. Dans la littérature on trouve plus souvent leur appellation anglosaxonne : Kmeans et PCA. L'algorithme Kmeans est utilisé pour faire du clustering. La PCA peut être utilisée en autre pour faire de la réduction de dimension. Nous illustrerons ces deux points via deux applications simples.

1. K-means

Commençons par quelques rappels sur l'algorithme. Soient $(x_i)_{i=1,\dots,n}$ des vecteurs appartenant à $\mathbb{R}^p$. On notera $X \in \mathbb{R}^{n\times p}$ la matrice de design associée. Le but est de partitionner ces données en K clusters. On introduit alors une variable indicatrice $Z \in \{0,1\}^{n\times K}$ qui indique l'appartenance des points x_i aux différentes clusters k. Plus précisément $Z_{ik} = 1$ si x_i est assignée au cluster k et $Z_{ik} = 0$ sinon. On introduit aussi K "moyennes" $(\mu_k)_{k=1,\dots,K}$ dans $\mathbb{R}^p$ (on verra dans la suite que les μ_k sont exactement les moyennes de chaque cluster). On notera $\mu \in \mathbb{R}^{K\times p}$ la matrice associée. Le critère qu'on utilise pour qualifier un bon partitionnement selon Z et μ est la distorsion $J(\mu, Z)$ définie comme suit :

$$J(\mu, Z) = \sum_{i=1}^{n}\sum_{k=1}^{K} Z_{ik}\|x_i - \mu_k\|^2.$$

Le but est alors de minimiser cette distorsion. L'algorithme K-means consiste en une minimisation alternée entre μ et Z :

- **Etape 0 :** Choix d'une matrice μ de manière aléatoire
- **Etape 1 :** Minimisation de J par rapport à Z (revient à assigner chaque point x_i au cluster le plus proche)
- **Etape 2 :** Minimisation de J par rapport à μ : $\mu_k = \frac{\sum_i Z_{ik}x_i}{\sum_i Z_{ik}}$
- **Etape 3 :** Revenir à l'étape 1 jusqu'à convergence.

1) Montrer que l'étape 2 de l'algorithme correspond à l'annulation du gradient de J par rapport à chacun de μ_k avec Z fixé.

2) Implémentez l'algorithme K-means (la fonction `sqdist` peut être à nouveau utile ici). Celui ci doit prendre en entrée une matrice de design `X` et un nombre de clusters `K` et sortir des moyennes μ ainsi que la matrice de correspondance `Z`. Une bonne chose serait aussi de sortir l'évolution de la distortion au cours des itérations. **Remarque :** Un critère d'arrêt de l'algorithme peut être lorsque la distortion n'évolue plus.

3) Reprenez un des jeu de données du DM (par exemple accessibles à l'adresse `www.di.ens.fr/~slacoste/teaching/apprentissage-fall2014/TP/classificationA.train`). Appliquez alors votre algorithme pour différentes valeurs de K. Tracez l'évolution de la distortion en fonction des itérations.

Affichez ensuite les résultats (sous forme de nuage de points colorés en fonction du cluster).

4) Nous allons maintenant regardez une application plus visuelle. Choisissez une image de votre choix (à défaut on pourra choisir la classique Léna `https://upload.wikimedia.org/wikipedia/en/2/24/Lenna.png`). On va alors utiliser K-means pour encoder l'image sur seulement K couleurs. Pour cela on va considérer que chaque pixel i de l'image est une observation x_i appartenant à $\mathbb{R}^3$. Appliquez alors Kmeans en commençant avec $K = 5$.

5) Affichez la nouvelle image ainsi obtenue (image qui contient pour chaque pixel la couleur associée à son cluster). Essayez différentes valeurs de K.

6) ***Bonus*** (si vous avez le temps) : Comme vous l'avez peut être remarqué l'algorithme K-means dépend beaucoup de son initialisation et il peut s'avérer assez lent. Afin de régler ce problème, Arthur et al. [1] ont proposé une meilleure initialisation (qui possède en plus des garanties théoriques quant au vrai écart à l'optimum). Cette méthode a été baptisé K-means ++ et consiste à remplacer l'étape 0 de l'algorithme K-means par l'algorithme suivant :

- **Etape 0 :** Choisir de manière uniforme un point x_i comme étant le premier centre.
- **Etape 1 :** Pour chaque point x_i calculez la distance entre ce point et le centre le plus proche. On note $D_{\mu_t}(x_i)$ cette distance.
- **Etape 2 :** Choisir un nouveau point x_i comme étant un nouveau centre mais cette fois ci en suivant une loi de probabilité pondéré par $D_{\mu_t}(x_i)^2$.
- **Etape 3 :** Répétez l'étape 1 et 2 jusqu'à avoir K centres.

Implémentez cette méthode et vérifiez qu'elle fonctionne mieux en pratique (meilleur objectif final ainsi que convergence plus rapide).

2. PCA

Nous allons utiliser la PCA afin de réduire la dimension du jeu de données MNIST. Ceci va nous permettre de visualiser l'effet de la PCA. Pour ceux qui n'ont pas le jeu de données vous pouvez le téléchargez à l'adresse suivante : (`http://www.math.ens.fr/cours-apprentissage/mnist_digits.mat`). Formez une matrice de design $X \in \mathbb{R}^{n\times d}$ en sélectionnant un sous ensemble de points ($n = 5000$). Veillez toutefois à gardez un bon nombre de représentants par classe.

7) Appliquez la PCA à la matrice de design X. Pour cela on pensera à tout d'abord centrer les données ($X_c = X - \frac{1}{n}\mathbf{1}_n X^T \mathbf{1}_n$). Ensuite on diagonalisera la matrice de corrélation : $X_c^T X_c$. (command `eig` en Matlab). Dans la suite on considérera que les vecteurs propres ont été ordonnés par valeurs propres décroissantes.

8) Tracez l'évolution du spectre de la matrice de corrélation. Rappelez l'interprétation de ces valeur propres et commentez ce que vous observez.

9) Représentez sous forme d'image le premier et le second vecteur propre. Commentez.

10) Projetez les images initiales sur l'espace engendré par les deux premiers vecteurs propres. Représentez alors le jeu de données MNIST sous forme d'un nuage de points. Affichez différentes couleurs pour les différentes classes. En faisant le lien avec la question précédente, commentez la position des différents chiffres dans cet espace.

Références

[1] D. Arthur and S. Vassilvitskii. *k-means++ : The advantages of Careful seeding*. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007.