Clustering

Benjamin Monmege
benjamin.monmege@lsv.ens-cachan.fr

24 avril 2013

On considère une famille $X = (x_i)_{1\leq i\leq N}$ d'observations avec $x_i \in \mathbf{R}^D$. L'objectif est de regrouper ces données en un nombre K fixé de classes. On cherche donc les *variables de classes* $Y = (y_i)_{1\leq i\leq N}$ avec $y_i \in \{1, 2, \ldots, K\}$ associant à chaque exemple sa classe. Pour ce faire, on fixe une *fonction de dissemblance* d: $\mathbf{R}^D \times \mathbf{R}^D \to [0, +\infty[$ qui donne la distance séparant deux données x et x'. On cherche les variables Y de classes minimisant le critère

$$W(Y) = \frac{1}{2}\sum_{k=1}^{K} \sum_{i\neq j | y_i=k, y_j=k} d(x_i, x_j)$$

c'est-à-dire qu'on souhaite trouver $\widehat{Y} = \arg\min_Y W(Y)$.

Exercice 1. Estimer le nombre total de partitions de l'ensemble d'observations, puis le nombre de partitions à K classes. En déduire qu'une énumération complète de l'espace des solutions n'est pas envisageable.

On fixe dans la suite de ce TD la fonction de dissemblance égale à la distance euclidienne au carré : pour $x, x' \in \mathbf{R}^D$,

$$d(x, x') = \|x - x'\|^2 = \sum_{i=1}^{N} (x_i - x'_i)^2$$

Exercice 2 (K-means)**.** Vérifier qu'on cherche donc les variables Y minimisant

$$W(Y) = \frac{1}{2}\sum_{k=1}^{K} \sum_{i|y_i=k} \sum_{j|y_j=k} \|x_i - x_j\|^2$$

1. Montrer que
$$W(Y) = \sum_{k=1}^{K} N_k \sum_{i|Y_i=k} \|x_i - \mu_k\|^2$$
avec N_k le nombre de données classées dans la classe k, et μ_k la moyenne de ces N_k données.
2. Expliquer pourquoi la paire formée par $\widehat{Y}$ et sa famille $(\widehat{\mu}_k)_{1\leq k\leq K}$ de moyennes est également obtenu grâce à la minimisation
$$(\widehat{Y}, (\widehat{\mu}_k)_{1\leq k\leq K}) = \arg\min_{Y,(\mu_k)_{1\leq k\leq K}} \sum_{k=1}^{K} N_k \sum_{i|y_i=k} \|x_i - \mu_k\|^2$$
où on a, à nouveau, noté N_k le nombre de données de la classe k induit par Y.
3. L'algorithme K-means propose de trouver une solution Y qui est un minimum local de cette équation. L'idée est d'itérer le processus suivant : 1. pour une partition par classe Y donnée, minimiser le critère en fonction des variables $(\mu_k)_{1\leq k\leq K}$, 2. pour un ensemble de variables $(\mu_k)_{1\leq k\leq K}$ donné, minimiser le critère en fonction de la variable Y. Détailler comment réaliser chacune de ces deux étapes. Montrer que cet algorithme converge (vers un minimum local).

4. Trouver un exemple où cet algorithme ne converge pas vers le minimum global.
5. Implémenter cet algorithme et le tester sur des données générées artificiellement, par exemple sur le fichier `http://www.lsv.ens-cachan.fr/~monmege/teach/learning2013/cluster_data.mat` avec $K = 3$. Répéter l'algorithme avec différentes initialisations et d'autres valeurs de K.

Exercice 3 (Compression d'images)**.** On considère une image en couleurs dont chaque pixel est encodé par trois valeurs (RGB) entre 0 et 255. Pour information, en Matlab, on peut charger une image, par exemple enregistrée dans le fichier `image.png`, grâce à la commande

```
A = double(imread('image.png')) / 255;
```

On obtient alors une matrice tri-dimensionnelle dont chacun des coefficients est un niveau d'une des trois couleurs codée entre 0 et 1. Comme il est plus commode d'utiliser une matrice bi-dimensionnelle, on utilisera l'encodage

```
X = reshape(A, size(A,1) * size(A,2), 3);
```

qui fournit une matrice à trois colonnes (une pour chaque couleur) et autant de lignes qu'il y a de pixels dans l'image.

1. Expliquer comment appliquer une méthode de clustering afin de compresser l'image.
2. Appliquer l'algorithme K-means et le tester sur une image de votre choix ou sur l'image que vous pouvez télécharger à l'adresse `http://www.lsv.ens-cachan.fr/~monmege/teach/learning2013/bird_small.png`. Il sera sans doute nécessaire d'arrêter l'algorithme K-means après un nombre fixé d'itérations (car la convergence peut être très lente). Tester avec plusieurs valeurs de K. Pour information, vous pouvez visualiser l'image X' ainsi obtenue (représentée par une matrice bi-dimensionnelle avec trois colonnes) aux côtés de l'image A initiale grâce aux commandes

```
B = reshape(X', size(A,1), size(A,2), 3);
subplot(1,2,1);
imagesc(A);
subplot(1,2,2);
imagesc(B);
```

Exercice 4 (EM)**.** Comme à l'habitude, on essaie d'appliquer un raisonnement probabiliste afin d'étendre et/ou améliorer l'algorithme précédent. On suppose donc que les observations sont des échantillons de variables aléatoires $X_1, \dots, X_N$ i.i.d. suivant une distribution inconnue de densité f qu'on cherche à estimer. On va s'intéresser au cas où la densité est un *mélange de gaussiennes*, c'est-à-dire est de la forme

$$f(x) = \sum_{k=1}^{K} \pi_k \varphi_{\mu_k, \Sigma_k}(x)$$

avec $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$, et

$$\varphi_{\mu,\Sigma}(x) = \frac{1}{\sqrt{(2\pi)^D \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

est la densité de probabilité d'une loi gaussienne multidimensionnelle de moyenne μ et de matrice de covariance Σ. Intuitivement, une telle densité f représente un découpage en K clusters centrés sur μ_k. De plus, les matrices de covariance fournissent une information nouvelle sur le découpage : ainsi, on appelle *centroïde* la paire (μ_k, Σ_k) décrivant le k-ième cluster. Les poids π_k représentent les fréquences relatives des clusters.

1. On fixe K. On cherche alors le mélange de gaussiennes f maximisant la *log-vraisemblance* en fonction des données $X = (x_n)_{1 \leq n \leq N}$. Exprimer en fonction des données et des centroïdes $\theta = (\mu_k, \Sigma_k)_{1 \leq k \leq K}$ la log-vraisemblance $L(\theta) = \log p(X \mid \theta)$, et réécrire le problème comme un problème d'optimisation non convexe.

   On ne connaît pas de solution analytique à ce problème en général. Ainsi, on propose un algorithme itératif permettant de converger vers un maximum local de la log-vraisemblance. Pour ce faire, on utilise des *variables latentes*, dans ce cas les variables de classe Y. L'idée est d'alterner des étapes de deux sortes afin de calculer une suite (θ_t) de centroïdes. À l'étape $t \geq 1$, on exécute :
   - l'étape E : calculer la densité de probabilité des variables de classe $p(Y = y \mid X, \theta_{t-1})$, puis l'espérance de la log-vraisemblance

     $$Q(\theta, \theta_{t-1}) = \sum_y p(Y = y \mid X, \theta_{t-1}) \log p(X, Y = y \mid \theta)$$

   - l'étape M : maximiser l'espérance précédente pour estimer θ_t

     $$\theta_t = \arg\max_\theta Q(\theta, \theta_{t-1})$$

   L'algorithme s'arrête lorsque on a atteint la convergence des centroïdes ou de la log-vraisemblance.

2. Dans le cas de mélange de gaussiennes, estimer les différentes quantités intervenant dans l'algorithme EM présenté précédemment. Montrer que l'étape M revient à estimer de nouvelles quantités π_k, μ_k, Σ_k pour chaque classe $1 \leq k \leq K$.

3. Grâce à la fonction `gmdistribution.fit` de Matlab, appliquer cet algorithme sur l'exemple de l'exercice 2, et tracer le résultat pour observer les lignes de niveau des gaussiennes.