IFT3395/6390, Devoir 3

Date affiché: 19 nov, 2014,
À remettre: 27 nov 2014, au *début* de la classe

---

1. (2/10) Utilisez PCA pour projeter les images d'entraînement du devoir 1 dans un espace de deux dimensions pour la visualisation. Rappelez-vous que les fichiers de données peuvent être trouvés ici:

   `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/train_images.txt`
   `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/test_images.txt`
   `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/train_labels.txt`
   `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/test_labels.txt`

   Effectuez PCA sur l'ensemble des images d'entraînement (les données de toutes les classes simultanément). Ensuite, faites dix graphiques de type nuages de points (scatter plot), chacun montrant la projection des *données de test* de l'une des dix classes en deux dimensions.

2. (2/10) Montrez que minimiser l'erreur moyenne de reconstruction pour PCA est équivalent à maximiser la norme moyenne de la représentation latente (en d'autres termes, la variance totale empirique de la représentation latente).

3. (6/10) On vous demande d'implémenter le modèle de mélange gaussien (MMG) sur certaines données de dimension $D = 2$, et de faire l'entraînement en utilisant l'algorithme EM. Le MMG en deux dimensions peut être défini comme suit:

$$\begin{aligned} p(\mathbf{x}) &= \sum_{k=1}^{K} p(z_k = 1) p(\mathbf{x}|z_k = 1) \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad \Big(= \sum_{k=1}^{K} \pi_k |2\pi\Sigma_k|^{-\frac{1}{2}} \exp\big(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\big)\Big) \end{aligned}$$

   où $\pi_k = p(z_k = 1)$ est une probabilité a priori sur les états, et $p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ est une distribution gaussienne sur l'observation sachant l'état.

   À faire:

   - Le modèle est donné par les paramètres suivants: un K-vecteur $\boldsymbol{\pi}$ représentant les probabilités a priori $p(z_k = 1)$; K D-vecteurs $\boldsymbol{\mu}_k$, chacun représentant la moyenne d'un état; ainsi que K matrices de covariance $\boldsymbol{\Sigma}_k$ de taille $D \times D$ (une par état).

- Implémentez une fonction qui, étant donné les paramètres ainsi qu'une matrice de N points de données de dimension K, retourne L, la log-probabilité des données:

$$L = \sum_{n=1}^{N} \log p(\mathbf{x}) = \sum_{n=1}^{N} \log \sum_{k} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Implémentez l'*E-step* qui, étant donné les paramètres du modèle et des données, retourne les $(N \cdot K)$ *responsabilités*:

$$q_{nk} = p(z = k|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|z_k = 1)\pi_k}{\sum_\ell p(\mathbf{x}_n|z_\ell = 1)\pi_\ell}$$

- Implémentez le *M-step* qui, étant donné les responsabilités, les paramètres du modèle et les données, met à jour les paramètres. Les trois formules de mise à jour sont données dans les diapositives du cours.
- Utilisez votre implémentation pour entraîner le modèle sur l'ensemble de données *old-faithful*, disponible ici:

  `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/faithful.txt`

  L'ensemble de données est constitué de de N = 272 observations en $D = 2$ dimensions. Entraînez le modèle avec $K = 2, 3, 5$ composants, en utilisant plusieurs initialisations aléatoires (raisonnables) sur les paramètres à chaque fois. Pour l'entraînement, itérez les E-step, M-step et réévaluations de la log-vraisemblance.
- Pour chaque modèle ($K = 2, 3, 5$) créez un graphique montrant un nuage de points des données d'entraînement ainsi que (dans le même graphique) chacune des K gaussiennes sous forme d'ellipse en utilisant le code suivant:

  `www.iro.umontreal.ca/~memisevr/teaching/ift3395_2014/devoirs/plotGaussian.py`

Indices:

- Vous devez utiliser l'astuce logsumexp pour différents calculs qui autrement pourraient être instables.
- Votre log-vraisemblance ne doit jamais diminuer au cours de l'apprentissage. Si elle le fait, alors quelque chose n'est pas correct.
- Pour le débogage, il peut d'abord être utile de mettre de côté les mises à jour des matrices de covariance. En d'autres termes, au départ, initialisez les matrices de covariance à certaines valeurs raisonnables et gardez-les fixes pour voir si tout se comporte comme prévu.
- Il peut arriver, de temps en temps, qu'une gaussienne devienne responsable d'un seul point de données, ce qui peut causer des problèmes numériques parce que les écarts (variances) de cette gaussienne iront alors vers zéro. Une solution consiste à redémarrer l'apprentissage chaque fois que cela se produit. Une autre solution est de limiter avec un seuil les valeurs propres de chaque $\boldsymbol{\Sigma}_k$ après chaque M-step, en utilisant le code comme suit:

```
SMALL = 0.001
D, V = numpy.linalg.eigh(Sigma)
Sigma[:, :] = numpy.dot(numpy.dot(V,numpy.diag(D+SMALL)),V.T)
```

À remettre:

- La valeur de la log-vraisemblance des données pour un modèle avec $K = 2, 3, 5$ états, où chaque valeur dans la distribution a priori, $\boldsymbol{\pi}$, est définie comme $\frac{1}{K}$, chaque moyenne comme $\mathbf{0}$, et chaque matrice de covariance comme la matrice d'identité 2×2.
- Trois graphiques nuages de points ($K = 2, 3, 5$), y compris les ellipses gaussiennes, pour le meilleur modèle que vous avez pu entraîner, cad. le modèle avec la log-vraisemblance la plus haute entre plusieurs initialisations aléatoires. Mentionnez également la log-vraisemblance des données pour chacun des trois.