IFT3395/6390, Assignment 3

Date posted: Nov. 19, 2014,
Due: Nov. 27, 2014, at the *start* of class

---

1. (2/10) Use PCA to project the training images from assignment 1 into a two-dimensional space for visualisation. Recall that the input data files can be found here:

   `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`

   Perform PCA on the whole set of training images (data from all classes together). Then make ten scatter plots, each one showing the projection of the *test-data* of one of the ten classes into two dimensions.

2. (2/10) Show that minimizing the average reconstruction error in PCA is equivalent to maximizing the average norm of the latent representation (in other words, the empirical total variance of the latent representation).

3. (6/10) In this question, we will implement a Gaussian mixture model (GMM) on some $D = 2$-dimensional data, and we will train it with the EM algorithm. The 2-dimensional GMM can be defined as follows:

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

   where $\pi_k = p(z_k = 1)$ is a prior probability over states ("mixing proportion"), and where $p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a Gaussian observation probability.

   What you need to do:

   - The model is given by the following parameters: a K-vector $\boldsymbol{\pi}$ representing the prior probabilities $p(z_k = 1)$; K D-vectors $\boldsymbol{\mu}_k$, each representing the mean for one state; and K $D \times D$-matrices $\boldsymbol{\Sigma}_k$, each representing the covariance matrix for one state.
   - Implement a function that, given the parameters and a matrix of N D-dimensional data-points, returns L, the log-probability of the data:

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

- Implement an E-step which, given the model parameters and the data, computes the $(N \cdot K)$ "responsibilities":

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

- Implement an M-step which, given the responsibilities, the model parameters and the data, updates the model parameters. The three update equations are given the lecture slides.
- Use your implementation to train the model on the 'old-faithful' data-set, available here:

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

  The data-set consists of $N = 272$ observations in $D = 2$ dimensions. Train the model with $K = 2, 3, 5$ components, using multiple (sensible) random initializations over parameters each time, by iterating the E-steps and M-steps and evaluating the log-likelihood after each M-step.
- For each model ($K = 2, 3, 5$), make a scatter-plot of the data, and plot all Gaussians as one-standard-deviation ellipses in the same plot. You may use the following code for plotting the Gaussian ellipses:

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

Hints:

- You should use the logsumexp-trick to do various computations that might be unstable otherwise.
- Your log-likelihood should never decrease during learning. If it does, something is wrong.
- For debugging purposes, it may be helpful to leave out the covariance updates at first. In other words, initialize the covariance matrices to some sensible values and keep them fixed initially to see if everything else behaves as expected.
- It can happen, once in a while, that one Gaussian becomes responsible for a single data-point, which will cause numerical problems, because the variances of this state will then approach zero. One solution is to re-start learning whenever this happens. Another solution is to threshold the eigenvalues of each $\mathbf{\Sigma}_k$ after every M-step, using code like the following:

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

What to hand in:

- The value of the log-likelihood of the data for a model with $K = 2, 3, 5$ states, where each mixing proportion is set to $\frac{1}{K}$, each mean to $\mathbf{0}$, and each covariance matrix to the 2×2-identity matrix.

- Three scatter-plots (for $K = 2, 3, 5$) including the Gaussian ellipses for the *best* trained model, ie. the model with the highest log-likelihood among your multiple random initializations. Also report the log-probability of the data for each of the three.