Machine Learning Fall 2015 Homework 4

Make sure to explain you reasoning or show your derivations. Except for answers that are especially straightforward, you will lose points for unjustified answers, even if they are correct.

General Instructions

Submit your homework electronically on Canvas. We recommend using LaTeX, especially for the written problems. But you are welcome to use anything as long as it is neat and readable.

Include a README file that describes all other included files. Indicate which files you modified. You are welcome to create additional functions, files, or scripts, and you are also welcome to modify the included interfaces for existing functions if you prefer a different organization.

Since we will work on some of the homework in class, clearly indicate which parts of your submitted homework are work done in class and which are your own work.

Relatedly, cite all outside sources of information and ideas. **List any students you discussed the homework with.**

Written Problems

1. We will derive the expectation-maximization (EM) algorithm using *variational* analysis. Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ be a set of data vectors. Let $Z = \{z_1, \ldots, z_n\}$ be set of multinomial variables corresponding to which Gaussian data is generated from. Let the Gaussian parameters be means $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ and covariance matrices $\{\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K\}$. Let $\Theta = \{\theta_1, \ldots, \theta_K\}$ be multinomial prior probabilities over which Gaussian data is sampled from.

   Each data point is generated by first sampling a Gaussian index from $p(z|\Theta)$, then sampling from the Gaussian $\mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$. The log likelihood of any observations given these mixture model parameters is

$$L(X, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \Theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \theta_k \, \mathcal{N}(x_i | \boldsymbol{\mu}_k, \Sigma_k) \right) \tag{1}$$

   Since we will never observe any z variables, these are considered *hidden* or *latent* variables. When computing the likelihood of observed variables, we sum over all possible states of the latent variables, weighted by the probability of those states.

   (a) (2 points) We start by doing something a little weird. We create an independent distribution q for the latent variables such that

$$q(z_1, \ldots, z_n) := q(z_1) q(z_2) \ldots q(z_n). \tag{2}$$

   Rewrite the log likelihood from Equation (1) so that each data point's likelihood is multiplied by the q distribution divided by itself. In other words, multiply the terms in the innermost summation by $\frac{q(z_i=k)}{q(z_i=k)}$.

   (b) (6 points) Jensen's inequality guarantees that for any convex function φ and any distribution over random variable X

$$\varphi\left(\mathbb{E}\left[X\right]\right) \leq \mathbb{E}\left[\varphi(X)\right] \tag{3}$$

   Use Jensen's inequality and the fact that $\log$ is a **concave** function (i.e., $-\log$ is a convex function) to form a lower bound on the log likelihood. You should get an expression that looks like a sum of "cross-entropies" and entropies.

(c) (6 points) You now have a lower bound objective function that depends on the Gaussian mixture model parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, and Θ and the variational distribution q. If we hold q fixed, maximizing the Gaussian mixture model parameters gets us the Gaussian EM updates (the so-called "m-step"). You will prove this for one of the mixture parameters (you are welcome to try the other parameters too for fun). Show that maximizing the lower bound with respect to the cluster mixture probabilities Θ is exactly the EM update for these variables. You will need to use a Lagrange multiplier to enforce that $1 = \sum_{k=1}^{K} \theta_k$.

(d) (6 points) Show how to find the q parameters for each data point that maximize the lower bound. Hints: You'll need to use Lagrange multipliers to enforce that $1 = \sum_k q(z_i = k)$, and you should be able to consider each data point's q distribution independently.

2. (10 points) Project proposal. See instructions on project homepage.

Programming Assignment

For this programming assignment, we have provided a lot of starter code. Your tasks will be to complete the code in a few specific places, which will require you to read and understand most of the provided code, but will only require you to write a small amount of code yourself.

The script `makeSyntheticData.m` generates synthetic data using the following procedure:

- First, it generates a 2D random Gaussian mixture model with five Gaussians.
- It then generates n (2000) 2D data points from the mixture model.
- It projects the 2D data into 64-dimensional data using a random projection matrix.
- Finally, it adds random noise in 64-dimensional space to the projected data.

The resulting data comes from a 2D mixture model, but it is presented using 64-dimensions. You will use PCA and expectation maximization to attempt to fit a mixture model to the data.

The main experiment script is `runSyntheticExperiment.m`. It first uses PCA to transform the data into the two most descriptive dimensions. Then it runs expectation maximization using different numbers of Gaussians and records the log-likelihood of held-out validation data for each cluster count parameter.

Since MATLAB includes the functionality to do PCA and Gaussian mixture fitting, you will not be allowed to submit code that uses these built-in functions. You are however allowed to used the built-in numerical linear algebra functions `det`, `eig`, `svd`, `eigs`, `svds`, or `chol` (you don't need all of these, but you may want to use one or two).

1. (10 points) Complete `myPca.m`. This function should take a data matrix as input and return a transformed data matrix and the variance of the transformed data. The dimensions of the transformed data should be ordered such that the earlier indices have the highest variance. Using this ordering, one should be able to truncate to a lower-dimensional space while preserving the highest reconstruction accuracy by using the first few dimensions of the returned data.

2. (12 points) Complete `gmm.m`. This function should take a data matrix and a number of clusters as input and fit the Gaussians using expectation maximization. The main framework for the algorithm is set up for you already. You need to complete the code for updating the parameters (the Gaussian means and covariances and the cluster priors) and the latent variable probabilities (the cluster assignments).

   One important note for fitting Gaussians is that sometimes we can get degenerate data that creates numerically unstable covariances. To avoid this problem, one fix is to add a small constant to the diagonal of the estimated covariance matrix. The matrix `reg` in `gmm.m` is set up for you to do this. You can argue that this trick is regularization, or if you're more honest, it's just a reasonable hack to make things work.

3. (8 points) Complete `gmmLL.m`. This function should take a data matrix and Gaussian mixture model parameters as input and output a log likelihood. The formula for the log likelihood is

$$\begin{aligned}
\mathrm{ll}(\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\theta}) &= \log \prod_{i=1}^{n} p(\boldsymbol{x}_i | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\theta}) \\
&= \log \prod_{i=1}^{n} \sum_{k=1}^{K} \theta_k \mathcal{N}(x_i | \boldsymbol{\mu}_k, \Sigma_k) \\
&= \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \theta_k \mathcal{N}(x_i | \boldsymbol{\mu}_k, \Sigma_k) \right) \\
&= \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \frac{\theta_k}{\sqrt{(2\pi)^k |\Sigma_k|}} \exp\left(-\frac{1}{2} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k) \right) \right).
\end{aligned} \tag{4}$$

With some manipulation, you should be able to use the log-sum-exp trick to compute the terms in the summation while avoiding numerical underflow.

Table 1: Included files and brief descriptions. Unless indicated here in bold, the source files are just placeholders for code you should complete. You should not need to implement or modify bolded files.

File Path	Description
synthData.mat	Synthetic data of 64-dimensional points
trueData.mat	Synthetic 2D data that was used to generate the 64 dimensional points. Included for reference if you're interested in comparing.
src/gaussianLL.m	Script that computes the log likelihood of points under a Gaussian distribution
src/gmm.m	Function **you will complete** that fits a Gaussian mixture model to data using expectation maximization
src/gmmLL.m	Function **you will complete** that computes the log likelihood of data under a Gaussian mixture distribution
src/makeSyntheticData.m	Script that generates the synthetic data. You do not need to run this script since `synthData.mat` saves the relevant output.
src/myPca.m	Function **you will complete** that finds an orthogonal basis for the input data such that early dimensions capture the most variance in the data.
src/plotGMM.m	Function that plots ellipses representing the Gaussians in a mixture model.
src/runSyntheticExperiment.m	Script that runs the main experiment. It runs PCA and EM using different numbers of clusters and reports the log likelihoods.
src/sampleGaussian.m	Script that samples data from a multivariate Gaussian distribution. You won't need this, but it's used to generate the synthetic data.
src/sampleMultinomial.m	Script that samples data from a multinomial distribution. You won't need this, but it's used to generate the synthetic data.