Skolkovo Institute of Science and Technology

Homework #4. EM algorithm

Bayesian Methods – Advanced Machine Learning, Spring 2016, Term 3

Start Date: Saturday, February 20.
Submission Deadline: **Monday, February 29, 23:59.**
Maximum points: **5.0** (+ bonus points)
Programming Languages: Python + NumPy.

Questions regarding this assignment should be sent to *bayesml@gmail.com*. Please use the following prefix for the subject: [BMML Skoltech 2016]

1 Gaussian mixture model

Consider a problem of clustering, that is given a dataset $X = \{x_1, \ldots, x_N\}$ of N points we want to decompose it into K disjoint subsets according to some criteria. K is a hyperparameter (predefined number).

One assumption which is often made in practice is that the points x_i are i.i.d. samples from a mixture of Gaussian distributions with unknown parameters:

$$p(x_i) = \sum_{k=1}^{K} \underbrace{w_k}_{p(z_i=k)} \underbrace{\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \Sigma_k)}_{p(x_i \mid z_i=k)}.$$

Here z_i is a latent variable that describes which class does x_i belongs to. The unknown parameters are $\Theta = \{\{w_k\}_{k=1}^K, \{\mu_k\}_{k=1}^K, \{\sigma_k\}_{k=1}^K\}$.

After introducing the model we now can estimate the parameters and the clustering z_i for each point x_i from the data using the EM-algorithm or its modifications.

2 EM-algorithm

Lets consider the general EM-algorithm. We have the observed variables X, the latent variables Z and the parameters Θ and we want to solve the following problem:

$$\max_{\Theta}.\ \ln p(X \mid \Theta)$$

Note that the following equalities holds for any distribution $q(Z)$:

$$\ln p(X \mid \Theta) = \int q(Z) \ln p(X \mid \Theta) dZ = \int q(Z) \ln \frac{p(X, Z \mid \Theta)}{p(Z \mid X, \Theta)} dZ = \int q(Z) \ln \left[\frac{p(X, Z \mid \Theta)}{q(Z)} \frac{q(Z)}{p(Z \mid X, \Theta)} \right] dZ =$$

$$= \underbrace{\int q(Z) \ln p(X, Z \mid \Theta) dZ - \int q(Z) \ln q(Z) dZ}_{\mathcal{L}(q, \Theta)} + \underbrace{\int q(Z) \ln \frac{q(Z)}{p(Z \mid X, \Theta)} dZ}_{KL(q(Z) \mid\mid p(Z \mid X, \Theta))}$$

where $KL(q(Z) \mid\mid p(Z \mid X, \Theta))$ is the Kullback Leibler divergence between the distribution $q(Z)$ and the posterior distribution $p(Z \mid X, \Theta)$.
Since KL-divergence is always nonnegative, we obtained a lower bound on the log-likelihood:

$$\ln p(X \mid \Theta) \geq \mathcal{L}(q, \Theta) = \int q(Z) \ln p(X, Z \mid \Theta) dZ - \int q(Z) \ln q(Z) dZ.$$

On the E-step we are trying to make our lower bound as precise as possible in the current point:

$$\max_{q}.\ \mathcal{L}(q, \Theta)$$

Since $\ln p(X \mid \Theta) = \mathcal{L}(q, \Theta) + KL(q(Z) \mid\mid p(Z \mid X, \Theta))$ and the left hand side doesn't depend on q, maximizing $\mathcal{L}(q, \Theta)$ is equivalent to minimizing $KL(q(z) \mid\mid p(Z \mid X, \Theta))$:

$$\arg\max_{q}.\ \mathcal{L}(q, \Theta) = \arg\min_{q}.\ KL(q(Z) \mid\mid p(Z \mid X, \Theta)) = p(Z \mid X, \Theta) \tag{1}$$

So the E-step consist in computing the posterior distribution $p(Z \mid X, \Theta)$ in the current point Θ.
Now, on the M-step we fix the obtained $q = p(Z \mid X, \Theta)$ and maximize the lower bound w.r.t. the parameters:

$$\Theta = \arg\max_{\Theta}.\ \mathcal{L}(q, \Theta) = \arg\max_{\Theta}.\ \int q(Z) \ln p(X, Z \mid \Theta) dZ \tag{2}$$

3 Simplified EM

Consider the simplified version of the EM-algorithm, where on the E-step instead of searching for the best distribution $q(Z)$ among all possible distributions, we search among the delta-functions (compare to eq. (1)):

$$\widehat{q}(Z) = \underset{q(Z):\ \text{delta-function}}{\arg\min} KL(q(Z) \mid\mid p(Z \mid X, \Theta)) = \begin{cases} 1 & \text{if } Z = Z_0 \\ 0 & \text{otherwise} \end{cases}$$

This way on the M-step you will have the mathematical expectation with respect to the delta function, which is just the value of the function at the point Z_0:

$$\Theta = \underset{\Theta}{\arg\max.}\ \mathcal{L}(\widehat{q}, \Theta) = \underset{\Theta}{\arg\max.} \int \widehat{q}(Z) \ln p(X, Z \mid \Theta) dZ = \underset{\Theta}{\arg\max.}\ \ln p(X, Z_0 \mid \Theta)$$

4 Assignment

In this assignment you will implement and compare three methods:

1. The full EM-algorithm (as discussed on the seminar);
2. Soft K-Means, that is the EM-algorithm with fixing sigma to be identity matrix.
3. K-Means algorithm;

The assignment consists of the following tasks:

1. Derive formulas for the full EM-algorithm.
2. Apply the following changes to the EM-algorithm and derive formulas for the E and M-steps of the obtained algorithm
   (a) Use uniform prior distribution $p(z_i = k) = \frac{1}{K}$;
   (b) Use homogeneous Gaussians (fix sigma to be identity matrix);
   (c) Minimize over delta functions on the E-step (see Sec. 3).
3. Check that the formulas obtained in task 2 are equivalent to the formulas of the K-Means algorithm.
4. Implement the three methods in separate python classes (see example implementation of the random clusterer in 'clustering.py'). You are provided with some utility functions and with an iPython file which shows how to use clustering algorithms and how to monitor the convergence.
5. Discuss in the report what are the pros and cons of using each of these three methods.
6. Generate two datasets on which
   (a) All the methods work equally good;
   (b) Full EM-algorithm works much better than the others.
7. Include scatter plots and the obtained log-likelihoods on these two datasets in the report.

Notes

1. Make sure that the likelihood of the EM-algorithm is never decreasing (it actually can decrease due to numerical issues, but just a little bit).
2. Work with probabilities in the log domain to avoid numerical issues (see utils.log_likelihood for the example of this).
3. A reasonable way to initialize cluster centers is to place them in K random objects.
4. Some parts of the iPython notebook require 'pyqt' to be installed.

5 Submission Guidelines

The assignment is to be submitted via **Canvas**. The submission must contain:

- Report in PDF format or as IPython Notebook with description of the experiments.
- Source code.

The source code for all classes should be in 'clustering.py'. A clusterer class init method should take the following parameters:

1. *n_clusters* – the number of clusters K.
2. *max_iter* – the maximal number of iterations. Default is 100.
3. *n_init* – the number of runs of the algorithm with different random inits. You should return the results of the best run according to log-likelihood. Default is 10.
4. *min_covar* – when you estimate covariance matrix from the data, it is a good idea to regularize it. Just add the number specified by this parameter to the diagonal of your covariance matrices on each iteration. Default is 0.001.
5. *tol* – if log-likelihood changed less than *tol* on the current iteration, stop the optimization process. Default is 0.001.
6. *logging* – if logging is True, add a self.logs dictionary attribute which consist of the following fields:
   (a) *log_likelihood* – a list of log-likelihoods on each iteration.
   (b) *labels* – a list, for each iteration it contains the labeling of the data points (cluster index for each data point).
   (c) *w* – a list, for each iteration it contains the vector of prior probabilities w.
   (d) *mu* – a list, for each iteration it contains $K \times d$ matrix of cluster centers, where K is the number of clusters and d is the number of features.
   (e) *sigma* (only for the full EM) – a list, for each iteration it contains a $K \times d \times d$ tensor of cluster covariance matrices.

After calling the 'fit(X)' method, the object should contain the following attributes:

1. *labels_* – the labeling of the data points (cluster index for each data point).
2. *w_* – the vector of prior probabilities w.
3. *cluster_centers_* – $K \times d$ matrix of cluster centers.
4. *covars_* (only for the full EM) – $K \times d \times d$ tensor of cluster covariance matrices.

6 Late submission policy

The assignment may be submitted after due date, but with a late submission penalty. The late submission penalty for this assignment is 0.1 points per day, capped at 4 points.

7 Collaboration

The assignment have to be done individually in the sense that no sharing of code or solutions is allowed. Discussion of the assignment is allowed and encouraged.