Computational Intelligence, SS11

http://www.spsc.tugraz.at/courses/computational-intelligence

Paul Meissner [1]

6 Homework: Expectation Maximization and K-means Algorithm

[Points: 23+8* , Issued: 2011/06/22 , Deadline: 2011/07/01 , Tutor: Georg Kapeller [2]; Info-hour: TBA , Room and Date will be posted in the newsgroup , ;

General: In this homework, you have to implement both the EM and K-means algorithm. For the evaluation, use the dataset provided on the course homepage (`vowels.mat`). The variable `X` contains the unlabeled training data, which is two-dimensional. You already know this dataset from the problem classes.

6.1 Expectation Maximization Algorithm [13 Points]

We want to find a maximum-likelihood Gaussian Mixture Model (GMM) for the training data in X. Using M Gaussian components, this model is given as

$$p(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_{m=1}^{M} \alpha_m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{1}$$

with the parameter vector $\boldsymbol{\Theta}$ containing the component weights α_m, the means $\boldsymbol{\mu}_m$ and the covariance matrices $\boldsymbol{\Sigma}_m$, respectively. The EM algorithm for GMMs tries to iteratively find this ML estimator, i.e., at each iteration i it computes an updated parameter vector $\boldsymbol{\Theta}_i$.

Implement the EM algorithm for GMMs using the following steps:

1. Init: At $i = 0$, select an initial guess of the parameter vector $\boldsymbol{\Theta}_0$.

2. Expectation Step: For each sample $\boldsymbol{x}_n$, calculate the probability $p(m|\boldsymbol{x}_n, \boldsymbol{\Theta}_i)$ that this sample was caused by the m-th component of the GMM. In the lecture, this value was called r_m^n and is written as

$$r_m^n = \frac{\alpha_{m,i}\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_{m,i}, \boldsymbol{\Sigma}_{m,i})}{\sum_{m'=1}^{M} \alpha_{m',i}\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_{m',i}, \boldsymbol{\Sigma}_{m',i})} \tag{2}$$

Remember, this corresponds to a Bayesian "soft classification" of each sample.

3. Maximization Step: Using the r_m^n, the effective number of samples for the m-th component is given by $N_m = \sum_{n=1}^{N} r_m^n$. With this, the entries of $\boldsymbol{\Theta}$ can be updated:

$$\boldsymbol{\mu}_{m,i+1} = \frac{1}{N_m}\sum_{n=1}^{N} r_m^n \boldsymbol{x}_n \tag{3}$$

$$\boldsymbol{\Sigma}_{m,i+1} = \frac{1}{N_m}\sum_{n=1}^{N} r_m^n (\boldsymbol{x}_n - \boldsymbol{\mu}_{m,i+1})(\boldsymbol{x}_n - \boldsymbol{\mu}_{m,i+1})^T \tag{4}$$

$$\alpha_{m,i+1} = \frac{N_m}{N} \tag{5}$$

[1] mailto:paul.meissner@tugraz.at

[2] mailto:georg.kapeller@student.tugraz.at

4. Likelihood calculation: Compute the current value of the log-likelihood function of the samples in X, given the current model $\mathbf{\Theta}_{i+1}$:

$$\log p(X|\mathbf{\Theta}_{i+1}) = \sum_{n=1}^{N} \log \sum_{m=1}^{M} \alpha_{m,i+1} \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_{m,i+1}, \mathbf{\Sigma}_{m,i+1}) \tag{6}$$

Check wether the likelihood has already converged or not. If yes, terminate the algorithm, if no, go back to step 2.

Your tasks are as follows:

1. Write a Matlab function `[alpha, mu, Sigma, L] = EM(X, M, alpha_0, mu_0, Sigma_0, max_iter)` that implements the EM algorithm! `X` is the training data, `M` is the number of Gaussian components, `alpha_0`, `mu_0` and `Sigma_0` are the initial guesses of the parameter vector and `max_iter` is the maximum number of iterations. The function returns the final parameter vector given by `alpha`, `mu` and `Sigma` as well as the log-likelihood over the iterations in `L`. You may use the Matlab function `mvnpdf`.

2. Test your implementation using the dataset given in `X`. You can compare the result with the labeled data set given in the variables `a`, `e`,`i`,`o` and `y`. Make a scatter plot of the data and plot the Gaussian mixture model over this plot. You can use the provided function `plotGaussianContour()` for plotting each of the Gaussian components (the factors α_m are then neglected of course).

3. For your tests, select the correct number of components ($M = 5$), but also check the result when you use more or less components. How do you choose your initialization $\mathbf{\Theta}_0$? Does this choice have an influence on the result?

4. Also plot the log-likelihood function over the iterations! What is the behavior of this function over the iterations?

5. Within your EM-function, confine the structure of the covariance matrices to diagonal matrices! What is the influence on the result?

6. Make a scatter plot of the data that shows the result of the soft-classification that is done in the E-step. This means, classify each sample using the r_m^n, and plot the samples in different colors for each of the M classes.

6.2 K-means algorithm [10 Points]

In this task, you should implement the K-means algorithm as discussed in the lecture and compare the results with the ones obtained with the EM-algorithm. Remember that you can interpret K-means as a version of the EM-algorithm with several simplifications: First, the classes are just represented by their means, second, there are no class weights and third, a hard classification is performed for all samples. The latter also means that for the parameter update, only the points classified to this component play a role.

1. Write a Matlab function `[mu, D] = k_means(X, M, mu_0, max_iter)` that implements the K-means algorithm. Here, `X` is the data, `M` is the number of clusters, `mu_0` are the

initial means (cluster centers) and `max_iter` is the maximum number of iterations. The function returns the optimized cluster centers `mu` and the cumulative distance over the iterations in `D`.

2. Perform the same tasks as for the EM-algorithm to evaluate the performance! Of course, the way to plot the classes/components is different now: In the scatter plot, plot the mean value for each class and plot the points that were classified to this class in a certain color.

3. What is the nature of the boundaries between the classes? Compare with the results of the soft-classification in the EM-algorithm! Also compare with the labeled data, can K-means find the class structure well?

6.3 Samples from a Gaussian Mixture Model [8* Points]

1. Write a Matlab function `Y = sampleGMM(alpha, mu, Sigma, N)`, that draws `N` samples from a two-dimensional Gaussian Mixture distribution given by the parameters `alpha`, `mu` and `Sigma`! Hint: Can you split the problem into two parts (First, sample from a specific distribution, then, conditioned on the results from the first step, sample from another one)? You may use the provided function `Y = sampleDiscretePMF(X, PM, N)` that draws `N` samples from a discrete probability distribution `PM`, defined over the values `X`.

2. Using a GMM of your choice ($M > 3$), demonstrate the correctness of your function!