Computational Intelligence SS17
Homework 5
Expectation Maximization and k-means Algorithm

Christian Knoll

Tutor:	Philipp Gabler, pgabler@student.tugraz.at
Points to achieve:	18 pts
Extra points:	4* pts
Info hour:	13.06.2016, 14:00-15:00, HS i12
Deadline:	19.06.2016 23:59
Hand-in mode:	Submit your **python files and a colored version of your report (as.pdf)** at `https://courses-igi.tugraz.at/`. (Please name your archive *hw5-Familyname1Familyname2Familyname3.zip*)
Hand-in instructions:	`https://www.spsc.tugraz.at/courses/computational-intelligence-cs`
Newsgroup:	tu-graz.lv.ew

General remarks

Your report must be self-contained and must therefore include all relevant plots, results, and discussions. Your submission will be graded based on:

- The correctness of your results (Is your code doing what it should be doing? Are your plots consistent with what algorithm XY should produce for the given task? Is your derivation of formula XY correct?)
- The depth and correctness of your interpretations (Keep your interpretations as short as possible, but as long as necessary to convey your ideas)
- The quality of your plots (Is everything important clearly visible in the print-out, are axes labeled, . . . ?)

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 (`HW5_data.zip`). The file `X.data` contains the unlabeled training data, which is two-dimensional. You already know this dataset from the problem classes. The labeled data is contained in the following files (`a.data, e.data, i.data, o.data, y.data`) where every file corresponds to one class.

Hint: The skeleton file also contains a sanity check where the usage of the provided functions is demonstrated.

1 Expectation Maximization Algorithm [10 Points]

We want to fit a maximum-likelihood Gaussian Mixture Model (GMM) to the training data in $\mathbf{X}$. Using M Gaussian components, this model is given as

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

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

The EM algorithm for GMMs goes through the following steps:

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

2. **Expectation Step:** For each sample $\mathbf{x}_n$, calculate the probability $p(m|\mathbf{x}_n, \mathbf{\Theta}_i)$ 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

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

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

3. **Maximization Step:** 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 $\mathbf{\Theta}$ can be updated according to:

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

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

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

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

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

   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 function that implements the EM algorithm:
   `alpha, mu, Sigma, L = EM(X, M, alpha_0, mu_0, sigma_0, max_iter)`. Here, `X` is the training data and consists of $N = 20000$ samples. `M` is the number of Gaussian components. The initial guesses of the parameter vector are `alpha_0`, `mu_0`, and `Sigma_0`. The maximum number of iterations is given by `max_iter`. After convergence 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 provided function `P = likelihood_bivariate_normal(X,mu,cov)` that estimates the likelihood of `X` given the specified bivariate Gaussian distribution.

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 `plot_gauss_contour(mu,cov,xmin,xmax,ymin,ymax,title)` for plotting each of the Gaussian components. The factors α_m are neglected for the visualization. Make sure to choose an appropriate range for plotting.

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. Therefore classify each sample using r_m^n, and plot the samples in different colors for each of the M classes.

2 K-means algorithm [8 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 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 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! 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 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?

3 Samples from a Gaussian Mixture Model [4* Points]

1. Write a function `Y = sample_GMM(alpha, mu, Sigma, N)`, that draws `N` samples from a two-dimensional Gaussian Mixture distribution given by the parameters `alpha`, `mu` and `Sigma`!
   Hint: You can split the problem into two parts (first, sample from a specific distribution, then, conditioned on these results , sample from another distribution)? You may use the provided function `Y = sample_discrete_pmf(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!