Computational Intelligence SEW SS18
Homework 5
Expectation Maximization and k-means Algorithm

Christian Knoll

Tutors:	Peter Lorenz, peter.lorenz@student.tugraz.at Markus Huber, markus.huber@student.tugraz.at
Points to achieve:	33 pts
Extra points:	5* pts
Info hour:	19.06.2018, 14:00-15:00, HS i12
Deadline:	27.06.2018 23:55
Hand-in mode:	Use the **cover sheet** from the website. Submit (i) all **python files (as.zip)** and (ii) a colored version of **your report (as.pdf)** at the teachcenter `https://tc.tugraz.at`. (Please name your files *hw5-Familyname1Familyname2Familyname3.pdf* and *hw5-Familyname1Familyname2Familyname3.zip*)
Course information:	`https://www.spsc.tugraz.at/courses/computational-intelligence-sew`
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 the EM algorithm, the K-means algorithm, and PCA. For the evaluation use the (modified) iris flower data set. The data set is provided and loaded in the skeleton file. The iris flower data set (and additional information) can also be found in the UCI Machine Learning Repository (`https://archive.ics.uci.edu/ml/datasets/iris`)

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

1 Background

This section explains the algorithms that have to be implemented. The actual tasks are provided in the subsequent sections.

1.1 Expectation Maximization Algorithm

We want to fit a maximum-likelihood Gaussian Mixture Model (GMM) to the provided data. Using K Gaussian components, this model is given as

$$p(\mathbf{x}_n|\mathbf{\Theta}) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\mathbf{x}_n|\underline{\mu}_k, \mathbf{\Sigma}_k) \tag{1}$$

with the parameter vector $\mathbf{\Theta}$ containing the component weights α_k, the means $\underline{\mu}_k$, and the covariance matrices $\mathbf{\Sigma}_k$. 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 $\mathbf{\Theta}_0$. Therefore, write a function `alpha_0, mean_0, cov_0 = init_EM(dimension, nr_components, scenario)` that provides the initial values `alpha_0`, `mean_0`, and `cov_0`.

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

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

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

3. **Maximization Step:** The effective number of samples for the k-th component is given by $N_k = \sum_{n=1}^{N} r_k^n$. With this, the entries of $\mathbf{\Theta}$ can be updated according to:

$$\underline{\mu}_{k,i+1} = \frac{1}{N_k}\sum_{n=1}^{N} r_k^n \mathbf{x}_n \tag{3}$$

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

$$\alpha_{k,i+1} = \frac{N_k}{N} \tag{5}$$

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

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

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

1. Write a function that implements the EM algorithm:
   `alpha, mean, cov, LL, labels = EM(X, K, alpha_0, mu_0, sigma_0, max_iter, tol).`
   Here, `X` is the data set and consists of $N = 150$ samples. `K` is the number of Gaussian components. The initial parameter vector is `alpha_0`, `mean_0`, and `cov_0`. The maximum number of iterations and the tolerance are given by `max_iter` and `tol`. After convergence the function returns the final parameter vector given by `alpha`, `mu`, and `cov`, as well as the log-likelihood over the iterations in `LL`. The function further returns the labels that are obtained by performing soft classification of each sample according to (2)
   You may use the provided function `P = likelihood_multivariate_normal(X,mu,cov,log)` that estimates the likelihood, or the log-likelihood, of `X` given the specified multivariate Gaussian distribution.

1.2 K-means algorithm

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 `centers_0 = init_k_means(dimension, nr_clusters, scenario)` that provides the initial cluster centers `centers_0`.

2. Further write a function that implements the K-means algorithm:
   `centers, D, labels = k_means(X, K, centers_0, max_iter, tol)`
   Here, `X` is the data, `K` is the number of clusters, `centers_0` are the initial cluster centers and `max_iter` and `tol` are the maximum number of iterations and the tolerance. The function returns the optimized cluster centers `centers` and the cumulative distance over the iterations in `D`. All `labels` are obtained by performing hard classification for each sample.

1.3 Principal Component Analysis (PCA)

You should also implement PCA as discussed in the lecture and apply it to reduce the dimension of the data from $D = 4$ to $M = 2$.

1. Write a function `Y,V = PCA(data, M, whitening)` that computes the first `M` principal components and transforms the data, so that $\mathbf{y}_n = \mathbf{U}^T\mathbf{x}_n$. The function should further return `V` that explains the amount of variance that is explained by the transformation.

2. (bonus-task) If different features are on different scales it can be an important pre-processing step to transform the data such that `Y` has zero mean and a covariance matrix equal to the identity matrix. Implement this and perform the according transformation if `whitening = True`.

2 Classification/ Clustering

We use three different scenarios (Sec. 2.1- 2.3) in order to evaluate the EM- and the K-means algorithm. Each scenario uses a different pre-processing step before applying the algorithms. Test your EM-algorithm for all three scenarios:

1. Compare the result with the labeled data set (i.e., consider `labels` as well). 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,nr_points,title)` for plotting each of the Gaussian components. The factors α_k are neglected for the visualization. Make sure to choose an appropriate range for plotting.

2. For your tests, select the correct number of components ($K = 3$), 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?

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

4. 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_k^n, and plot the samples in different colors for each of the three classes. You may use the provided functions `plot_iris_data(data,labels)` and `reassign_class_labels(labels)` in order to produce comparable plots.

Further apply and evaluate your K-means algorithm for all three scenarios:

5. 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.

6. 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?

2.1 2 dimensional feature [14 + 2* Points]

In this scenario we consider only two of the available features (sepal length and petal length), i.e., `x_2dim`.

1. Perform all of the above-mentioned tasks for the EM algorithm. [7 Points]

2. Perform all of the above-mentioned tasks for the K-means algorithm. [7 Points]

3. (bonus-task) You may additionally choose any other pair of features; how would this change the classification accuracy? [2* Points]

2.2 4 dimensional feature [7 Points]

In this scenario we apply our algorithms to the full data set with four features, i.e., `x_4dim`. The classification has to be performed for $D = 4$; it is sufficient, however to visualize your results by plotting the same two features as in scenario 2.1. Again perform all of the above-mentioned tasks for both algorithms.

1. How do the convergence properties and the accuracy of you classification change in comparison to scenario 2.1? [4 Points]

2. Within your EM-function confine the structure of the covariance matrices to diagonal matrices! What is the influence on the result. [3 Points]

2.3 Processing the data with PCA [10 + 3* Points]

Here we perform PCA first to reduce the dimension to $M = 2$ while preserving most of the variance and then apply our algorithms to the transformed data set, i.e., `x_2dim_pca`.

1. How much of the variance in the data is explained this way? [3 Points]

2. How does the performance of your algorithms compare to scenario 2.1 and scenario 2.2? [5 Points]

3. For scenario 2.2 we visualized the result by neglecting two features. We could also apply PCA as post-processing step for visualization. PCA takes the covariance of the whole data set into account; how are the covariance matrices of the individual components in scenario 2.2 alternated if you apply PCA this way. Provide an analytical expression! [2 Points]

4. (bonus-task) Apply PCA with whitening, so that the transformed data has zero mean and a unit covariance matrix. How does this influence the choice of your initialization? [3* Points]

3 Samples from a Gaussian Mixture Model [4 Points]

1. Write a function `Y = sample_GMM(alpha, mu, cov, N)`, that draws `N` samples from a two-dimensional Gaussian Mixture distribution given by the parameters `alpha`, `mu` and `cov`!
   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 ($K > 3$), demonstrate the correctness of your function!