Computational Intelligence SS23
Homework 4
Expectation Maximization, k-Means and Principal Component Analysis

Harald Leisenberger

Tutor:	Maximilian Graber, maximilian.graber@student.tugraz.at
Points to achieve:	25 pts
Extra points:	4* pts
Info hour:	26.06.2023, 09:00-10:00, in: https://meet157.webex.com/meet/pr25504191823
Deadline:	03.07.2023, 23:59
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 *hw1-Familyname1Familyname2Familyname3.pdf* and *hw1-Familyname1Familyname2Familyname3.zip*.)

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, are axes labeled, . . . ?). Each plot requires a verbal description as well (often, a few lines will be sufficient).
- Your submission should run with Python 3.2+.

1 Introduction

This assignment consists of three parts: Expectation Maximization (Sec.2), k-Means (Sec.3), and Principal Component Analysis (Sec.4). A few tasks of Sec.3 / Sec.4 can only be fulfilled, after certain tasks in Sec.2 / Sec.3 have been performed.

Download `HW4.zip` from the course website and unzip. The file `HW4_skeleton.py` contains a skeleton of the script that you should implement. Specifically, you should complete the #TODO snippets in the file. For the evaluation, we 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`).

Some remarks:

- You are in general **not allowed** to use any functions from `scipy` and `sklearn`; feel free to use any `numpy` function though.
- The skeleton file also contains a sanity check where the usage of the provided functions is demonstrated.
- All over this assignment, we will consider **3 different scenarios** that correspond to different pre-processing steps applied to the data.:
  - `scenario` 1 occurs in Sec.2.1 and Sec.3.1 and involves working on a 2-dimensional slice of the original 4-dimensional iris data set.
  - `scenario` 2 occurs in Sec.2.2 and Sec.3.2 and involves working on the original 4-dimensional iris data set.
  - `scenario` 3 occurs in Sec. 4.1 and involves working on a 2-dimensional data set that is obtained by reducing the dimension of the original iris data set with PCA.

2 Expectation Maximization [11 Points + 4 Bonus Points]

This section is about Gaussian mixture models (GMMs) and the Expectation Maximization (EM) algorithm. We want to fit a GMM to the provided data $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. Using K Gaussian components, this model is given by

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

where the parameter vector $\boldsymbol{\Theta}$ consists of: the weights of the individual components α_k, the means $\boldsymbol{\mu}_k$, and the covariance matrices $\boldsymbol{\Sigma}_k$. The EM algorithm for GMMs iteratively maximizes the log-likelihood by updating the parameters $\boldsymbol{\Theta}^{(t)}$.

The EM algorithm for GMMs goes through the following steps:

1. **Init:** At $t = 0$, select an initial guess of the parameter vector $\boldsymbol{\Theta}^{(0)}$
2. **Expectation Step:** For each sample $\mathbf{x}_n$, calculate the posterior probability $p(k|\mathbf{x}_n, \boldsymbol{\Theta}^{(t)})$ that this sample was generated by the k-th component of the GMM. In the lecture, this number was defined by r_k^n and is computed as

$$r_k^n = \frac{\alpha_k^{(t)} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_{k'}^{(t)}, \boldsymbol{\Sigma}_{k'}^{(t)})}{\sum_{k'=1}^{K} \alpha_{k'}^{(t)} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_{k'}^{(t)}, \boldsymbol{\Sigma}_{k'}^{(t)})} \tag{2}$$

   Recall that 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 $N_k = \sum_{n=1}^{N} r_k^n$. Then, the entries of $\mathbf{\Theta}$ can be updated according to

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{1}{N_k} \sum_{n=1}^{N} r_k^n \mathbf{x}_n \tag{3}$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{N_k} \sum_{n=1}^{N} r_k^n (\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_n - \boldsymbol{\mu}_k^{(t+1)})^T \tag{4}$$

$$\alpha_k^{(t+1)} = \frac{N_k}{N} \tag{5}$$

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

$$\log p(\mathcal{X}|\mathbf{\Theta}^{(t+1)}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \alpha_k^{(t+1)} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k^{(t+1)}, \boldsymbol{\Sigma}_k^{(t+1)}) \tag{6}$$

Check if the value of the log-likelihood has converged. If yes, terminate the algorithm. Otherwise, go back to step 2.

During the upcoming tasks, you will have to implement the following functions:

- Write a function that provides the initial values `alpha_0`, `mean_0`, and `cov_0`:
  `alpha_0, mean_0, cov_0 = init_EM(dimension, nr_components, scenario, X)`
- Write a function that implements the EM algorithm:
  `alpha, mean, cov, LL, labels = EM(X, K, alpha_0, mean_0, cov_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 consists of `alpha_0`, `mean_0`, and `cov_0`. The maximum number of iterations and the convergence tolerance are given by `max_iter` and `tol`. After convergence, the function returns the final parameter vector composed by `alpha`, `mean`, and `cov`, as well as the values of the log-likelihood `LL` over the iterations. The function further returns the labels that are obtained by performing soft classification of each sample according to the maximum value of (2)

  You may use the provided function `P = likelihood_multivariate_normal(X,mean,cov,log)` that estimates the likelihood, or the log-likelihood, of `X` given the specified multivariate Gaussian distribution.

2.1 EM for 2-dimensional features [9 Points]

This corresponds to `scenario` = 1. Here, we evaluate EM to two of the available features: sepal length and petal length, i.e., `x_2dim`. Specifically, you have to perform the following tasks:

- Implement EM and apply it to the data. In two separate plots, compare the soft classification results induced by EM to the original labeled data set (i.e., take `labels` into account). More concretely: 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. (6 P.)
- Make a scatter plot of the data and plot the obtained GMM over this plot. For that purpose, you can make use the helper function
  `plot_gauss_contour(mu,cov,xmin,xmax,ymin,ymax,nr_points,title)` for plotting each of the Gaussian components. The weights α_k are neglected for the visualization. (1 P.)

- For your evaluations above, you should have selected the correct number of components ($K = 3$), of course. Now provide the result when you use more or less components (e.g., $K = 2$ and $K = 5$). How do you choose your initialization $\boldsymbol{\Theta}^{(0)}$? Does this choice have an influence on the result? (1 P.)
- Also, plot the value of the log-likelihood function over the number of iterations! What is the behavior of this function? (1 P.)

2.2 EM for 4-dimensional features [2 Points]

Corresponds to `scenario` = 2. Here, we apply EM to the full data set with four features, i.e., `x_4dim`. The classification has to be performed for $L = 4$; it is sufficient, however, to visualize your results by plotting the same two features as in `scenario` 1. Specifically, you have to perform the following tasks:

- How do the properties of the convergence the classification change in comparison to `scenario` 1? (1 P.)
- Now, within your EM function, confine the structure of the covariance matrices to diagonal matrices. What is the influence on the results? (1 P.)

2.3 Sampling from a Gaussian Mixture Model [4 Bonus Points]

In this bonus subsection, we aim to manually sample from a GMM.

- Write a function `Y = sample_GMM(alpha, mean, cov, N)`, that draws `N` samples from a two-dimensional Gaussian Mixture distribution specified by the parameters `alpha`, `mean` and `cov`. Using a GMM of your choice ($K > 3$), demonstrate the correctness of your function.
  Hint: You can split the problem into two parts: first, sample from a particular distribution, then, based on these results, sample from another particular distribution. You may use the provided function `Y = sample_discrete_pmf(X, PM, N)` that generates `N` data points according to a discrete probability distribution `PM`, defined over the values `X`. (4 P.)

3 k-Means [9 Points]

This section is about the k-Means algorithm. You should implement k-Means as discussed in the lecture and compare the results with the ones obtained by EM. Recall that you can interpret k-Means as a modification of EM with certain simplifications: first, the classes are just represented by their means; second, there are no class weights; third, a hard classification is performed for all samples. The latter also implies that for the parameter update, only the points assigned to a particular cluster play a role.

During the upcoming tasks, you will have to implement the following functions:

- Write a function that provides the initial cluster centers `centers_0`:
  `centers_0 = init_k_means(dimension, nr_clusters, scenario)`.
- 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 optimal cluster centers `centers` and the values of the cumulative distance over the iterations in `D`. The `labels` are obtained by performing hard classification for each sample.

3.1 k-Means for 2-dimensional features [8 Points]

Corresponds to `scenario` = 1. Here, we evaluate k-Means on two of the available features: sepal length and petal length, i.e., `x_2dim`. Specifically, you have to perform the following tasks:

- Implement k-Means and apply it to the data. In two separate plots, compare the obtained hard classification result to the labeled data similar as we did it for EM. The way to plot the clusters, however, is different now: in the scatter plot, visualize the center of each class and plot all points assigned to one and the same cluster in the same color. Can k-Means describe the original class structure well? (5 P.)
- For your evaluations above, you should have selected the correct number of components ($K = 3$), of course. Now provide the result when you use more or less components (e.g., $K = 2$ and $K = 5$). (1 P.)
- Plot the evaluation of the cumulative distance over the number of iterations. What is the behavior of this function? (1 P.)
- In two separate plots, compare the hard classification results obtained by k-Means to the soft-classification results obtained by EM. (1 P.)

3.2 k-Means for 4-dimensional features [1 Point]

Corresponds to `scenario` = 2. Here, we apply k-Means to the full data set with four features, i.e., `x_4dim`. The classification has to be performed for $L = 4$; it is sufficient, however, to visualize your results by plotting the same two features as in `scenario` 2. Specifically, you have to perform the following task:

- How do the properties of the convergence and the classification change in comparison to `scenario` 1? (1 P.)

4 Principal Component Analysis [5 Points]

This section is about principal component analysis (PCA). You should implement PCA as discussed in the lecture and apply it to reduce the dimension of the data from $D = 4$ to $M = 2$.

During the upcoming tasks, you will have to implement the following functions:

- 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 describes the amount of variance that is explained by the transformation.
- If various features are differently scaled, it is 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 whitening and perform the according transformation if `whitening = True`.

4.1 Pre-processing the data with PCA [5 Points]

Corresponds to `scenario` = 3. Here we perform PCA first to reduce the dimension of the 4-dimensional data 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. Implement PCA and apply it to the data. How much of the variance in the data is explained in this way? [3 Points]

2. How does the performance of EM and k-Means compare to scenario `scenario` 1 and scenario `scenario` 2? [1 Point]

3. Apply PCA with whitening, so that the transformed data has zero mean and a identity covariance matrix. Again apply EM and k-Means to the so pre-processed data and visualize the results. [1 Point]