Machine Learning Group
Fakulty IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Problem Set 2: k-Means, Gaussian Mixture Models, Agglomorative Clustering

Part 1: Implementation

Assignment 1 (5 point)

Implement k-means clustering as a function

```
mu, r, loss = kmeans(X, k, max_iter=100)
```

which, with respect to the rows of the $n \times d$ matrix `X`, calculates the $k \times d$ matrix for the `k` cluster centroids `mu` as well as the n-dimensional vector `r` of cluster membership: the i-th entry of `r` should contain the index of the clusters to which the i-th datapoint belongs. It also returns the `loss` (k-means criterion).

The algorithm should terminate when the membership no longer changes or after `max_iter` (optional parameter with default value 100) number of steps, depending on which comes first. The function should print the following information after each iteration:

- The number of iterations performed so far.
- The number of cluster memberships which changed in the preceding step.
- The loss function value (see handbook).

Think about how you want to handle the case when clusters become empty and document this in your report.

Assignment 2 (10 point)

Implement stepwise optimal hierarchichal agglomerative clustering with the k-means criterion as a function,

```
R, kmloss, mergeidx = kmeans_agglo(X, r)
```

which given the columns of the $n \times d$ matrix `X` and the initial clustering solution given by the length n membership vector `r` calculates a hierarchical clustering solution. The result should be returned in the following format:

- `R` is a $(k-1) \times n$ matrix which contains the cluster membership *before* each agglomeration step. That is, the l-th row of `R` contains the cluster indices for each data point where the total number of clusters is $k - l + 1$. The first row of `R` is the initial clustering `r`, and the last row is a clustering with two clusters.
- `kmloss` is a lenght k vector, which contains the loss function value after each agglomeration step, where the first entry is the loss of the initial clustering `r`. The loss function is the sum of the Euclidean distances from each data point to its cluster center.
- `mergeidx` is a $(k-1) \times 2$ matrix, which contains the indices of the two clusters that were merged at each step. That is, the l-th row of `mergeidx` contains the two indices that were unified in the l-th step. The index of the new (joint) cluster is the highest index plus one.

You should implement this yourself, do not use functions like `scipy.cluster.hierarchy.linkage`.

Assignment 3 (5 point)

Implement a function which given a hierarchical clustering sets up a dendrogram plot:

```
agglo_dendro(kmloss, mergeidx)
```

The parameters `kmloss` and `mergeidx` correspond to the the results of `kmeans_agglo`. See the guide for an example dendrogram plot.
You may use the function `scipy.cluster.hierarchy.dendrogram`.

Assignment 4 (5 points)

Write a function that computes the probability density function for a multivariate Gaussian distribution

```
y = norm_pdf(X, mu, C)
```

It takes as input the datapoints `X` as a $n \times d$ matrix, the center of the Gaussians distribution `mu` as a length d vector, and the covariance matrix of the Gaussian distribution `C` as a $d \times d$ matrix. The output is the probability density function evaluated at all points in `X` as a length n vector:

$$\mathtt{y}_i = \frac{1}{(2\pi)^{d/2}\det(\mathtt{C})^{1/2}} \cdot \exp\left(-\tfrac{1}{2}(\mathtt{X}_i - \mathtt{mu})^\top \mathtt{C}^{-1}(\mathtt{X}_i - \mathtt{mu})\right).$$

Think about how you want to handle the covariance inverse. Do you use `solve` or `inv`? Can the covariance function become singular? When? What do you do to remedy this issue? Discuss all this in your report. Note that you do not need `for`-loops in this function.
You obviously can not use a numpy/scipy/scikit-learn implementation for the probability density function.

Assignment 5 (15 points)

Implement the EM algorithm for Gaussian Mixture Models (GMM) as a function:

```
pi, mu, sigma, loglik = em_gmm(X, k, max_iter=100, init_kmeans=False, tol=1e-5)
```

where the parameters have the following definitions:

- `pi`: length k vector of $\widehat{\pi}_k$
- `mu`: $k \times d$ matrix of $\widehat{\boldsymbol{\mu}}_k$ (center points)
- `sigma`: list of length k of the $d \times d$ covariance matrices $\widehat{\boldsymbol{\Sigma}}_k$
- `loglik`: the log-likelihood after the last iteration
- `X`: $n \times d$ array of datapoints
- `k`: number of normally distributed components
- `max_iter`: maximal number of iterations (default: 100)
- `init_kmeans`: initialisation by means of k-means cluster solution (default: `False`)

After every step the function should print the number of the iteration and the log-likelihood. The algorithm should terminate when the maximal number of iterations `max_iter` has been reached or the log-likelihood does not change; i.e. when a local maximum has been reached.

Can a cluster become too small here? What is the equivalent to degenerate clusters in k-means? How does this show? Document all this in your report.

Assignment 6 (5 point)

Write a function that visualizes the GMM for two-dimensional data:

```
plot_gmm_solution(X, mu, sigma)
```

The figure should show:

- the data as a scatter plot,
- the mean vectors as red crosses, and
- the covariance matrices as ellipses (centered at the mean).

Part 2: Application

Please clarify your answers to the following questions with suitable plots.

Assignment 7 (15 points)

Analyse the `5gaussians` dataset with both methods (k-means and GMM) for $k = 2, \ldots, 7$ clusters.

1. Did you see obvious local (i.e. non-global) optima? How do you avoid them?
2. Do both methods find the 5 clusters reliably?
3. What role does the initialisation of the GMM with a k-means solution play in the number of necessary iterations and the quality of the solution?
4. What does the dendrogramm of the hierarchical clustering look like and is it possible to pick a suitable value of k from the dendrogramm?

Assignment 8 (10 points)

Analyse the `2gaussians` dataset with k-means and GMM. Explain exactly what you tested.

1. Compare the cluster centers. Which algorithm works better and why?
2. How does the GMM depend on the initialisation?

Assignment 9 (15 points)

Use GMM and k-means clustering on the USPS dataset with $k = 10$.

1. Compare the cluster centers. Which algorithm delivers better results?
2. Set up a dendrogramm to the hierarchical clustering solution and also a plot which displays the cluster centroids as a 16×16 image at every agglomerative step.

Assignment 10 (15 points)

A colleague of yours noticed your experience with outlier detection and clustering algorithms. His laboratory analyzed probes and recorded some data which your colleague think is distributed by a *mixture of three Gaussians*. One day they realized that part of the equipment was contaminated and they recorded data they have to filter out (*outliers*).

1. Load dataset `lab_data.npz`. It contains the observations in `X` and whether this data point is an outlier in `Y` (-1 for outlier, $+1$ for inlier).
2. Run your `em_gmm` function on the data points with $k = 3$ clusters and initialize with `kmeans`. Make sure you do not just get a bad local optimum. Report the log-likelihood and the cluster centers.
3. Also compute the `gammaidx` from the last sheet for `X`. Which number of nearest neighbors gives highest AUC?
4. Compute and report the AUC for both methods. Does the custom tailored outlier detection outperform the `gammaidx`? Plot the ROCs for both methods into one plot.

If you can have trouble getting your `gammaidx` or `auc` function to work, please contact us.