Submission until: **10.05.** (late submissions will get a deduction)
Discussion on: 12.05.
Submission as upload to your groups stud.IP folder as *groupNumber_sheet3.zip*

General info: If you have questions concerning how to work with different matlab libraries/python modules send us a short mail. There is plenty of time in the practice session to discuss those issues.

Assignment 1 *(EM (10p))*

Info concerning this assignment:

> **MATLAB:** Using the matlab built-in normpdf, as well as functions like pdist2 and mathematical operations is of course allowed.
>
> **PYTHON:** You can use the normpdf from scipy.stats modules, as well as functions like cdist (scipy.spatial). The use of mathematical operations, file-handling and plotting is of course also allowed (see also sheet 2).
>
> **BUT** do not use built-ins for the fitting of the data or any other method that does the work of the EM-algo for you.
>
> **Do NOT** copy code from external sources and submit it as your own. If a group should happen to submit such code all group members will receive a serious deduction of points.

Let's implement the EM-algorithm. For simplicity we'll look at one-dimensional data only and three Gaussian distributions.

(a) Load the data provided in *data.mat*.

(b) (2p) Each Gaussian j is specified by three parameters: μ_j (the mean), σ_j (the standard deviation), α_j (the probability of this Gaussian distribution in the mixture ($\sum alpha_j = 1$)). Initialize these parameters using three random partitions of the data set. The α_j are initially calculated as the number of data samples in the partition divided by the number of data samples in the data set.

(c) (1p) Implement a loop that stops when the changes in the μ and σ fall below a specified epsilon (find a value for epsilon yourself).

(d) (3p) Implement the expectation step:
Basically what is done here is a soft classification of the data samples with the three Gaussian distributions. This is calculated as the probability of a data sample x_i belonging to the pdf_j given the parameters μ_j and σ_j. The result is weighted by the value $alpha_j$, i.e. the alpha value for the Gaussian j.
The result is an estimation of the probability that the data samples belong to one of the three Gaussians (i.e. a (200×3) matrix which we'll call p. The index i is used to identify the data samples, index j is used to identify the Gaussian).

(e) Normalize the previous results so that the probabilities for the three components/estimations) of every data sample sum up to 1.

(f) (4p) Maximization step:
For each of the three Gaussians estimate updated values for μ, σ and α.
$\mu_j = \frac{1}{\sum p_j} \sum_{i=1}^{n} p_{ij} x_i$ is the mean of all data samples weighted by their probability that they belong to this distribution.
σ is the standard deviation of the Gaussian with the μ that was just calculated. The standard

deviation is calculated as $\sigma^2 = \frac{1}{\sum p_j} \sum_{i=1}^{n} p_{ij}(x_i - \mu_j)^2$

The alphas are recaluclated as the mean value of p_j for each Gaussian j.

(g) (1p)After each E-step plot the data. Assign the color of the Gaussian distribution it belongs to to each data sample. You can use the following code to plot the Gaussians (if for some reason that doesn't work at least plot the means as lines)

```
for j =1:3
  ix = linspace(-4.5,4,200);
  plot(ix,normpdf(ix,mu(j),sigma(j))+1,color);
end
```

Figure 1: Matlab

```
for j in range (0,3):
    ix = numpy.linspace(-4.5,4,200);
    matplotlib.pyplot.plot(ix,scipy.stats.norm.pdf(ix,mu[j],sigma[j])+1,color);
```

Figure 2: Python

(You have to define the color matrix yourselves).

Hierarchical clustering

MATLAB: You are welcome to use the pdist2 function. You are also welcome to compare your result to the result of the matlab built-ins (like *clusterdata*).

PYTHON: You are welcome to use the cdist function (scipy.spatial). You are also welcome to compare your result to the result of the python built-ins like Hierarchical clustering form the *scipy.cluster.hierarchy* module or *sklearn.cluster.AgglomerativeClustering*

BUT everything that you submit must be an implementation of your own and must not use built ins for the clustering.

Assignment 2 *(Hierarchical clustering (6p))*

Implement a Matlab function which performs agglomerative clustering on the data set *points.mat*. The function should have the following features:

(a) Implement complete-linkage and single-linkage agglomerative clustering. (Define a boolean variable which switches between single and complete linkage clustering).

(b) Stop when 5 clusters are reached.

(c) Plot result, i.e. a scatterplot with different colors for each cluster.

Assignment 3 *(Hierarchical Clustering 2 (6p))*

Implement another Matlab function which performs agglomerative clustering on the data set *points.mat*.
The clustering should use the distance as the stop criterion (not the number of clusters).

(a) Implement average inkage clustering.

(b) Implement centroid clustering