Handed out: April 21 (Wed)

Hints for solution: Exercise class on April 23 (Fr)

Hand in: April 28 (Wed), by email to doris.entner@helsinki.fi. Please submit your report as a single PDF containing figures, relevant program output and **discussion**. *Submit the source code as separate files.*

Solving the exercises below gives you points, which will, at the end, determine your grade for the computer project. Each exercise in all the assignments gives you an equal amount of points. These are the last exercises!

Ex. 1 — K-means and vector quantization

1. Implement the K-means algorithm and test it on some simple two-dimensional data like in Fig. 10.1 in the lecture hand-out. Make plots which show the clustering of the data as in Figs 10.2 to 10.4. Also make plots which show the value of the log10 of the objective function in Eq 10.3 at each iteration. Note:
   1. Initialize the means randomly within the range of your data.
   2. Your algorithm should stop when the change in the objective is smaller than some threshold.
   3. Your algorithm should work for data of any dimension with any number of clusters.
   4. The algorithm should return the means of the clusters $\mathbf{w}_c$, and, for each data point, indicate to which cluster it belongs (variable $i(t)$ in Eq 10.3). Also, it should return the objective function in Eq 10.3.

   (30 %)
2. Load the `image.txt` and cut it into pieces of size $d \times d$ pixels as in session 3, ex 2. Run your k-means on it with $d = 2$ and $d = 5$ and for each with $k = 5$ and $k = 100$ clusters. Visualize the cluster means as $d \times d$ images. Also, plot the objective function and make a histogram which shows how often each cluster is used (make a histogram of $i(t)$). Note:
   1. Visualize the cluster means on the same scale so that you can see the differences in the gray level between the different means.
   2. K-means might get stuck in local minima, so you might have to run the algorithm from several starting points.

   (30 %)
3. Replace each patch with the corresponding mean, and glue the patches together as in session3, ex2. Show us this image. This kind of image compression is called vector quantization. (30 %)

4. The pixels in the original image were encoded in $8 = \log_2(256)$ bits, so that 256 different gray levels were used. How many bits are thus needed to encode the whole image like that? How many bits do you need to encode the image after vector quantization if the block size is $d \times d$ and you use K different means? Calculate the ratio between the number of bits needed for the original image and the vector quantizised image for the values of d and k above.(10 %)

Ex. 2 — Clustering for binary data
We implement here a clustering algorithm for binary data. The algorithm "Bernoulli-EM" is summarized at the very end of math exercise 2.

1. Generate $T = 10000$ samples from a mixture of $C = 2$ multivariate Bernoulli distributions with the parameters

```
mu1 =

    0.3542
    0.3328
    0.2199
    0.8008
    0.7973

mu2 =

    0.5073
    0.6762
    0.7672
    0.2749
    0.8812
```

and $\pi_1 = 0.3$, $\pi_2 = 0.7$. Sampling from this kind of mixture is done in similar way as sampling from a Gaussian mixture (p. 111, lecture hand-out): First decide from which cluster $c \in 1 \dots C$ you sample, then sample from the multivariate Bernoulli distribution describing that cluster. A multivariate Bernoulli distribution for the *binary* random vector $\mathbf{x} = (x_1 \dots x_n)^T$ with mean $\boldsymbol{\mu_c} = (\mu_{c1} \dots \mu_{cn})^T$ is defined as

$$p(\mathbf{x}; \boldsymbol{\mu_c}) = \prod_{i=1}^{n} p(x_i; \mu_{ci}) \quad (1)$$

$$= \prod_{i=1}^{n} \mu_{ci}^{x_i} (1 - \mu_{ci})^{1-x_i}. \quad (2)$$

The binary x_i are thus independent form each other, being

distributed according to $p(x_i; \mu_{ci}) = \mu_{ci}^{x_i}(1-\mu_{ci})^{1-x_i}$. Sampling $\mathbf{x}$ can thus be done by sampling each x_i independently. *Hint:* The probability that $x_i = 1$ is μ_{ci}.
Check your sampling process by calculation of the sample mean of your data and comparison with the theoretical mean

$$\mathbb{E}(\mathbf{x}) = \sum_{c=1}^{C} \pi_c \boldsymbol{\mu_c}. \tag{3}$$

(20 %)

2. Implement the algorithm "Bernoulli-EM". It should return an estimate for the π_c and $\boldsymbol{\mu_c}$, $c = 1 \dots C$. Apply it to the data. Does the algorithm find the correct values for π_c and $\boldsymbol{\mu_c}$? *Hint:* We initialized the π_c all with $1/C$ (0.5 here) and the $\boldsymbol{\mu_{ci}}$ uniformly in [0.25 0.75]. (40 %)
3. Load the file `binDigits.txt` and extract all the samples showing the digits 0 and 1 (the first 200 data points). Run your algorithm on that data for $C = 2$ and $C = 4$. Visualize the cluster means $\boldsymbol{\mu_c}$ as an $28px \times 28px$ image and for each $\boldsymbol{\mu_c}$, indicate on the plot the cluster probability π_c. Comment on your results! Show us at least two different runs of the algorithm since the point of convergence of the algorithm depends on the initialization. (20 %)
4. Run the algorithm on the whole data with $C = 10$. Show us the estimates as before. (10 %)
5. Run the algorithm on the digits of your choice with the number of clusters of your choice. Show us the estimates as before. (10 %)

Ex. 3 — MDS and Kernel PCA

1. Load the two dimensional data `data_ex3.txt` and make a scatter plot. Check that the variables are zero mean.(5 %)
2. Perform MDS on the data by calculation of the eigenvectors of $X^T X$ where $X = (\mathbf{x}_1 \dots \mathbf{x}_N)$ and $\mathbf{x}_i$ is two dimensional and $N = 1000$. The first eigenvector contains projections of all the data points. Visualize this projection by coloring each data point by the value of its projection. (Make a color plot as Fig.11.2) How well does the projection encode the structure of the data? (30 %)
3. Perform MDS on the data via PCA and visualize the projection in the same way as above. Why is it the same? (20 %)
4. Perform Kernel PCA on the data, and visualize it in the same way as above. Define the projection by using the eigenvector with the eigenvalue which has the largest absolute value. For us, $\sigma^2 = 1$ in

Eq. 11.8 worked fine. How does this projection encode the data? What are the differences to the previous cases? (45 %)