# Statistical Inference and Machine Learning
# Homework 3

- This assignment can be solved in groups of 1 up to 5 students. You must mention the name of all the participants. Note that all the students in a group will get the same grade.
- Deadline: 14 December 2020, 23:59 (No late submissions will be accepted)
- Upload a single zip file on Moodle containing your solution and code. You can use any programming language.

## 1 PCA [40 pts]

One way to compress data point is to use the leading $k$ principal component to represent the data point. That is we first obtain the principal directions $w_1, w_2, \dots w_k$ of a dataset $D = \{x_1, x_2, \dots, x_n\}$ for $x_i \in R^d$, where $w_i^\top w_j = 0$ and $w_i^\top w_i = 1$, and $k \ll d$. Then we approximately represent each data point $x_i$ as

$$\widetilde{x}_i = \sum_{j=1}^{k} \underbrace{(w_j^\top x_i)}_{\alpha_j} w_j \tag{1}$$

Since $w_j$ are shared across the entire dataset, then we only need to remember those $\alpha_j$s for each data point. That is we only need to remember $k$ scalars for each data point even when the original dimension $d$ of the data point is very high. If $k$ is much smaller than $d$, this approximate representation scheme leads to huge saving in term of storage. Typically, when you increase the number $k$ of used principal directions $w_j$, the approximation gets better, meaning that the approximation error $\|x_i - \widetilde{x}_i\|_2$ is smaller. When $k = d$, the approximation error should be zero.

You will be using PCA to perform dimensionality reduction on the given dataset (leaf.csv). The present database comprises 40 different plant species. The data include 8 shape features and 6 texture features of leaves. Please refer to ReadMe.pdf for description of the dataset. You are to use Principal Component Analysis to perform Data Compression.

1. Submit a plot of the eigenvalues in ascending order (i.e., visualize the increase of computed eigenvalues).
2. Select a cut off to choose the top $k$ eigenvectors based on the your calculation. Discuss the reasoning for choosing this cut off.
   *Note:* this is an open problem. Any reasonable value of $k$ will be acceptable. State your cut off, reason, and report your corresponding $k$.
3. Now consider $k = 2$ and report the first $k = 2$ eigenvectors.
4. Visualize all the leaves in terms of the discovered $k = 2$ principle components in a 2-D scatterplot (i.e., for each leave, visualize its $(\alpha_1, \alpha_2)$).
5. For $k = 2$, compute the mean squared reconstruction error
   $$\|x_i - \tilde{x}_i\|_2^2$$
   using $x_1, x_2, \dots, x_n$ in the dataset.

## 2 EM algorithm [60 points]

In this problem, we will explore the use of EM algorithm for text clustering. Text clustering is a technique for unsupervised document organization, information retrieval. We want to find how to group a set of different text documents based on their topics. First we will analyze a model to represent the data.

**Bag of Words**

The simplest model for text documents is to understand them as a collection of words. To keep the model simple, we keep the collection unordered, disregarding grammar and word order. What we do is counting how often each word appears in each document and store the word counts into a matrix, where each row of the matrix represents one document. Each column of matrix represent a specific word from the document dictionary. Suppose we represent the set of $n_d$ documents using a matrix of word counts like this:

$$D_{1:n_d} = \begin{pmatrix} 2 & 6 & ... & 4 \\ 2 & 4 & ... & 0 \\ \vdots & & \ddots & \end{pmatrix} = T$$

This means that word $W_1$ occurs twice in document $D_1$ . Word $W_{n_w}$ occurs 4 times in document $D_1$ and not at all in document $D_2$.

**Multinomial Distribution**

The simplest distribution representing a text document is multinomial distribution. The probability of a document $D_i$ is:

$$p(D_i) = \prod_{j=1}^{n_w} \mu_j^{T_{ij}}$$

Here, $\mu_j$ denotes the probability of a particular word in the text being equal to $w_j$, $T_{ij}$ is the count of the word in document. So the probability of document $D_1$ would be $p(D_1) = \mu_1^2 \cdot \mu_2^6 \cdot ... \cdot \mu_{n_w}^4$.

**Mixture of Multinomial Distributions**

In order to do text clustering, we want to use a mixture of multinomial distributions, so that each topic has a particular multinomial distribution associated with it, and each document is a mixture of different topics. We define $p(c) = \pi_c$ as the mixture coefficient of a document containing topic $c$, and each topic is modeled by a multinomial distribution $p(D_i|c)$ with parameters $\mu_{jc}$, then we can write each document as a mixture over topics as

$$p(D_i) = \sum_{c=1}^{n_c} p(D_i|c)p(c) = \sum_{c=1}^{n_c} \pi_c \prod_{j=1}^{n_w} \mu_{jc}^{T_{ij}}$$

**EM for Mixture of Multinomials**

In order to cluster a set of documents, we need to fit this mixture model to data. In this problem, the EM algorithm can be used for fitting mixture models. This will be a simple topic model for documents. Each topic is a multinomial distribution over words (a mixture component). EM algorithm for such a topic model, which consists of iterating the following steps:

1. Expectation

   Compute the expectation of document $D_i$ belonging to cluster $c$:

$$\gamma_{ic} = \frac{\pi_c \prod_{j=1}^{n_w} \mu_{jc}^{T_{ij}}}{\sum_{c=1}^{n_c} \pi_c \prod_{j=1}^{n_w} \mu_{jc}^{T_{ij}}}$$

2. Maximization

   Update the mixture parameters, i.e. the probability of a word being $W_j$ in cluster (topic) $c$, as well as prior probability of each cluster.

$$\mu_{jc} = \frac{\sum_{i=1}^{n_d} \gamma_{ic} T_{ij}}{\sum_{i=1}^{n_d} \sum_{l=1}^{n_w} \gamma_{ic} T_{il}}$$

$$\pi_c = \frac{1}{n_d} \sum_{i=1}^{n_d} \gamma_{ic}$$

**Task 1 [30 pts]**

1. Verify the above-derived E-step and M-step.
2. Implement the EM algorithm on the toy dataset `data.csv`. You can treat `data.csv` as $D_{1:n_d}$, except that the last column represent the true cluster that this document belongs to. Run your algorithm and observe the results. Compare them with the provided true clusters each document belongs to. Report the evaluation (e.g. accuracy) of your implementation.

*Hint:* We already did the word counting for you, so the data file only contains a count matrix like the one shown above. For the toy dataset, set the number of clusters $n_c = 4$. You will need to initialize the parameters. Try several different random initial values for the probability of a word being $W_j$ in topic $c$, $\mu_{jc}$. Make sure you normalized it. Make sure that you should not use the true cluster information during your learning phase.

## MCMC algorithm

### Mixture of Gaussians

A finite Gaussian mixture of size $k$ has the following density:

$$f(x|k, \rho, \theta) = \sum_{k=1}^{K} \rho_k \mathcal{N}_k(x|\mu_k, \phi_k)$$

where $\mu_k$ and $\phi_k$ are the mean and precision (ie, inverse of covariance) of Gaussian density $\mathcal{N}_k(x|\mu_k, \phi_k)$ and $K$ is finite and known.

The likelihood of the data is

$$\mathcal{L} = \prod_{i=1}^{n} \sum_{k=1}^{K} \rho_k \mathcal{N}_k(x_i|\mu_k, \phi_k)$$

Assume we have $n$ observations $x = \{x_1, \ldots, x_n\}$ sampled iid from the finite Gaussian mixture model. We wish to make Bayesian inference for the model parameters $\theta := \{\rho_1, \mu_1, \phi_1, \ldots, \rho_K, \mu_K, \phi_K\}$ and evaluate the posterior distribution of the unknown parameters.

In order to simplify the likelihood, we introduce latent variables $Z_i$ such that

$$X_i|Z_i = k \ \sim \mathcal{N}_k(x|\mu_k, \phi_k) \ \text{ and } \ p(Z_i = k) = \rho_k.$$

These auxiliary variables allows us to identify the mixture component each observation has been generated from.

Therefore, for data $x = \{x_1, \ldots, x_n\}$, we assumes a missing dataset $z = \{z_1, \ldots, z_n\}$, which provide the labels indicating the mixture components from which the observations have been generated.

Using this missing dataset, the likelihood simplifies to

$$\mathcal{L} = \prod_{i=1}^{n} \rho_{z_i} \mathcal{N}_{z_i}(x_i|\mu_{z_i}, \phi_{z_i}) = \prod_{k=1}^{K} \rho_k^{n_k} \left( \prod_{i:z_i=k} \mathcal{N}_k(x_i|\mu_k, \phi_k) \right)$$

where $n_k = \#\{z_i = k\}$ and $\sum_{k=1}^{K} n_k = n$.

Then, the posterior probability that the observation $x_i$ has been generated from the $k$-th component is

$$\begin{aligned} p(z_i = k|x_i, \theta) &= \frac{\rho_k \mathcal{N}_k(x_i|\mu_k, \phi_k)}{\sum_{k=1}^{K} \rho_k \mathcal{N}_k(x_i|\mu_k, \phi_k)} \\ &= \frac{\rho_k \sqrt{\phi_k} \exp\{-\frac{\phi_k}{2}(x_i - \mu_k)^2\}}{\sum_{k=1}^{K} \rho_k \sqrt{\phi_k} \exp\{-\frac{\phi_k}{2}(x_i - \mu_k)^2\}} \end{aligned}$$

For the model parameters $\rho, \mu$ and $\phi$, we assume conjugate priors:

$$\begin{array}{ll} \text{Prior: } \rho \sim \mathit{Dirichlet}(\delta_1, \ldots, \delta_K) & \text{Posterior: } \rho|x, z \ \sim \mathit{Dirichlet}(\delta_1^*, \ldots, \delta_K^*) \\ \text{Prior: } \phi_k \sim \mathit{Gamma}(a/2, b/2) & \text{Posterior: } \phi_k|x, z \ \mathit{Gamma}(a_k^*/2, b_k^*/2) \\ \text{Prior: } \mu_k|\phi_k \ \sim \mathcal{N}(m_k, \frac{1}{\alpha_k \phi_k}) & \text{Posterior: } \mu_k|x, z, \phi_k \ \sim \mathcal{N}(m_k^*, \frac{1}{\alpha_k^* \phi_k}) \end{array}$$

where

$$\begin{aligned} \delta_k^* &= \delta_k + n_k \\ a_k^* &= a + n_k \\ b_k^* &= b + \sum_{i:z_i=k} (x_i - \mu_k)^2 \\ \alpha_k^* &= \alpha_k + n_k \\ m_k^* &= \frac{\alpha_k m_k + n_k \bar{x}_k}{\alpha_k + n_k} \quad \text{where: } \bar{x}_k = \frac{1}{n_k} \sum_{i:z_i=k} x_i \end{aligned}$$

Note that $\mathit{Dirichlet}(\delta_1, \ldots, \delta_K)$ is a Dirichlet distribution with density

$$f(\rho_1, \ldots, \rho_K) \propto \prod_{k=1}^{K} \rho_k^{\delta_k - 1}.$$

The usual prior choice is to take $(\delta_1, \ldots, \delta_K) = (1, \ldots, 1)$ to impose a uniform prior over the mixture weights. Note that this prior choice is equivalent to use the following reparametrization:

$$\begin{aligned} \rho_1 &= \eta_1 \\ \rho_k &= (1 - \eta_1) \cdots (1 - \eta_{k-1}) \eta_k \end{aligned}$$

assuming that $\eta_k \sim \mathit{Beta}(1, K - k + 1)$.

**MCMC algorithm**

1. Set initial values $\rho^{(0)}$, $\mu^{(0)}$ and $\phi^{(0)}$.

2. Update $z$ sampling from $z^{(j+1)} \sim z|x, \rho^{(j)}, \mu^{(j)}, \phi^{(j)}$.
3. Update $\rho$ sampling from $\rho^{j+1} \sim \rho|x, z^{(j+1)}$.
4. Update $\phi_k$ sampling from $\phi_k^{(j+1)} \sim \phi_k|x, z^{(j+1)}$.
5. Update $\mu_k$ sampling from $\mu_k^{(j+1)} \sim \mu_k|x, z^{(j+1)}, \phi_k^{(j+1)}$
6. $j = j + 1$. Go to 2.

**Task 2 [30 pts]**

1. Verify the above-derived $\delta_k^*, a_k^*, b_k^*, \alpha_k^*, m_k^*$.
2. Implement the MCMC algorithm on the toy dataset `data2.txt`. You can treat `data2.txt` as $x = \{x_1, \ldots, x_n\}$. Set $K = 2$ and run your MCMC algorithm and observe the results. As typically in MCMC, we may remove the initial burn-in phase. You need to:
   - Visualize the posterior distribution of your unknown parameters.
   - Compute the posterior mean of your unknown parameters.
   - Use your posterior mean as estimation of your unknown parameters. Generate 1000 samples $\tilde{x}$ from your estimated model. Visualize your original dataset $x$ and generated $\tilde{x}$ in terms of histogram respectively. Make a comparison.