Exercises
**Computational Intelligence Lab**
SS 2011

**Machine Learning Laboratory**
Dept. of Computer Science, ETH Zürich
**Prof. Dr. Joachim M. Buhmann**
Web http://ml2.inf.ethz.ch/courses/cil

# Series 4, Mar 24th, 2011 (K-means and Mixture Models)

**Problem 1 (Singularities in Mixture of Gaussians Models):**

In this exercise we study the problem of singularities in the mixture of Gaussians model. Consider the data set $\mathbf{X}$ consisting of $N$ i.i.d observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. The goal is to cluster this data set using a mixture of $K$ Gaussians.

1. Write down the log of the likelihood function (i.e., $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$).

Consider a Gaussian mixture model whose components have covariance matrices given by $\boldsymbol{\Sigma}_k = \sigma_k^2\mathbf{I}$, where $\mathbf{I}$ is the unit matrix. Suppose that one of the components of the mixture model, let's say the $j$-th component, has its mean $\boldsymbol{\mu}_j$ exactly equal to one of the data points so that $\boldsymbol{\mu}_j = \mathbf{x}_n$ for some value of $n$.

2. Give the log of the likelihood for this data point (i.e. $\ln p(\mathbf{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$)
3. Calculate $p(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, the probability of $\mathbf{x}_n$ given the $j$-th component.
   Hint: The multivariate Gaussian distribution is defined as
$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \frac{1}{(\sqrt{2\pi})^D|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$
4. In the limit $\sigma_j \to 0$, how does the probability in Question 3 change? Discuss about the influence of this issue on the maximization of the likelihood function.
5. Can this situation occur in the case of a single Gaussian distribution (i.e., $K = 1$)? Motivate your answer.
6. We can hope to avoid the singularities by using suitable heuristics. Can you propose such a heuristic?

**Problem 2 (Identifiability):**

Another issue in finding maximum likelihood solutions arises from *identifiability*. In this section we study *identifiability* in mixture models.

1. Suppose that we have solved a mixture of $K$ Gaussians problem, and have obtained the values of the parameters. For this given solution, how many equivalent solutions do exist?
2. This problem is known as *identifiability*. Explain why this problem is irrelevant for the purpose of data clustering.

**Problem 3 ($K-$means for Collaborative Filtering):**

As you already know, a recommender system is concerned with presenting items (e.g. books on Amazon or movies at Movielens) that are likely to interest the user. In collaborative filtering, we base our recommendations on the (known) preference of the user towards other items, and also take into account the preferences of other users. You have already examined collaborative filtering by applying SVD to the data. The data comes as a matrix $\mathbf{X}$, where each row represents a user and each column represents an item, i.e. each entry $\mathbf{X}_{i,j}$ indicates the rating of item $j$ by user $i$. In the provided dataset, we observe that not all users rated all items, and hence, such ratings are not available. The goal is to predict missing values from the observed ones.

In this assignment, we will apply $K-$means clustering to build the recommender system. The idea is to treat the items as dimensions and the users as observations, and then cluster the users based on the items. The co-clustered members are then used to predict the missing values. We begin with first setting up the environment on your local machine for sending the recommender system to the submission system, then we go step by step through the implementation pipeline.

**Submission system environment setup:**

1. The web page for the collaborative filtering application can be found here:

   http://cil.inf.ethz.ch/applications/collaborative_filtering.

2. Download the provided templates for the function `PredictMissingValues.m`, the evaluation script `CollabFilteringEvaluation.m` and the data `data.mat` into a local folder.

3. To simplify the implementation, we have provided an updated version of `PredictMissingValues.m` and a function called `K-means` which identify the structure of the implementation. Using these function files is arbitrary but recommended. You can find them in the course web page.

To submit your solution to the online evaluation system, we require you to prepare a ".zip" file containing at least `PredictMissingValues.m` (i.e. `X_pred = PredictMissingValues(X, nil)`) and any other function file (e.g. `kmeans.m`).

Similar to the SVD method, your submission is evaluated according to the following criteria:

1. Root mean squared error of prediction.

2. Run time of `PredictMissingValues.m`.

**Clustering for collaborative filtering:**
To cluster the users, we treat the items as features (or dimensions) and use the cluster members to predict the missing ratings. The assumption is that the users in the same cluster share similar interests. The pipeline for the given matrix $\mathbf{X}$ is:

1. **Load data:** This loads a matrix $\mathbf{X}$ of 7834 users by 100 items given in `Data.mat`. Each entry $\mathbf{X}_{i,j}$ contains a rating of item $j$ by user $i$ (scaled on the $[-10, 10]$ interval) or a special symbol (`nil` =99) for an absent rating (missing value). The goal is to predict the missing values $(99)$ using the observed ones.

2. **Splitting data into training/testing sets:** Being a genuine dataset, some of the ratings are missing and therefore can not be used for evaluation. The remaining known values are then used for both training and testing. We split $\mathbf{X}$ into Train/Test partitions, by splitting the existing ratings in the data $\mathbf{X}$ into two new matrices (of the same size) $\mathbf{X}^{trn}$ and $\mathbf{X}^{tst}$ which have no ratings in common. If a rating $\mathbf{X}_{i,j}$ is observed in $\mathbf{X}^{trn}$, then it is a missing in $\mathbf{X}^{tst}$. The omitted known (observed) values in $\mathbf{X}^{tst}$ are used for evaluation.

3. **Imputing missing values:** Performing $K-$means on $\mathbf{X}^{trn}$ requires a full matrix. Therefore, for each missing value, we calculate the average of the means by user and by item and replace it by this quantity. In this way we initialize the full matrix $\mathbf{X}^{pred}$.

4. **Clustering the training matrix:** We cluster the full matrix $\mathbf{X}^{pred}$ using $K-$means algorithm. Note that according to the lecture matrix orientations the observations are represented in columns. We therefore transpose $\mathbf{X}^{pred}$ and send it to the $K-$means routine.

5. **Prediction:** We consider the missing values of $\mathbf{X}^{trn}$ and update the corresponding entries in $\mathbf{X}^{pred}$ by setting them to the average over all observed (non-missing) items of the cluster which contains the missing entry. Therefore, $\mathbf{X}^{pred}$ becomes a full matrix whose observed entries are identical to the observed ratings in $\mathbf{X}^{trn}$ and its missing values are estimated by averaging over observed items of the co-clustered users.

6. **Evaluation:** We calculate the prediction error using Eq. 1 which compares the predicted values in $\mathbf{X}^{pred}$ to the true observed values in $\mathbf{X}^{tst}$ using the root mean squared error:

$$J = \sqrt{\frac{\sum_{\mathbf{X}_{i,j}^{tst} \neq \texttt{nil}} (\mathbf{X}_{i,j}^{pred} - \mathbf{X}_{i,j}^{tst})^2}{\sum_{\mathbf{X}_{i,j}^{tst} \neq \texttt{nil}} 1}} \tag{1}$$

7. **Model Selection:** Calculate the error (Eq. 1) for different choices of number of clusters $K$ and observe what happens to your error rates.

**The structure of the implementation:**
In the given template, there are three main scripts that provide the different steps of the pipeline:

1. `CollabFilteringEvaluation:` This script is the base evaluation script which loads the data matrix, splits known values into training and testing sets, and computes the prediction error rate. It loads the data from `Data.mat` and calls `PredictMissingValues(`$X^{trn}$`, nil)`.

2. `Xpred = PredictMissingValues(X, nil):` Predicts missing entries in matrix `X` based on known entries. Missing values in `X` are denoted by the special constant value `nil (=99)`. In the updated version, this function calls `[z, U, score] = K-means(data, nClusters, varargin)` to perform a clustering. The output `X_pred` is a matrix with the missing `nil` entries replaced by hypothesized ratings.

3. `[z, U, score] = K-means(data, nClusters, varargin):` The implementation of the $K-$means clustering appears here. The first argument is `data` whose columns represent the observations (i.e. the users). The other argument, i.e. `nClusters`, specifies the number of clusters. The other arbitrary arguments can be given by `varargin`. This function returns the vector `z` of cluster indices , the matrix `U` whose columns are the estimated centroids, and the parameter `score` which indicates the negative log-likelihood of the estimated solution.

**Extensions:** You can improve your solution in different ways. For example:

- You can employ a better optimization method to minimize the $K-$means cost function.
- Yo can find a suitable way to tune the number of clusters.
- Is the $K-$means clustering the best solution for this problem?