Clustering, K-means and EM

Morteza Haghir Chehreghani

March 24, 2011

Overview

Clustering, K-means and EM

- Clustering
- K-means
- Mixture Models and the EM algorithm
- Matrix Factorization

Pen & Paper

Assignment

Clustering

Problem

Given N data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of dimension D. Find a partitioning of the space using K prototypes $\mathbf{u}_k$ (sometimes $\boldsymbol{\mu}_k$).

Assignment

A data point $\mathbf{x}_n$ is assigned to its closest prototype:

$$z_{k^*(\mathbf{x}_n),n} = 1 \quad \text{and} \quad z_{k,n} = 0, \forall k \neq k' \forall n$$

with

$$k^*(\mathbf{x_n}) = \underset{k}{\operatorname{argmin}} \left\{ \|\mathbf{x}_n - \mathbf{u}_1\|_2^2, \ldots, \|\mathbf{x}_n - \mathbf{u}_k\|_2^2, \ldots, \|\mathbf{x}_n - \mathbf{u}_K\|_2^2 \right\}.$$

Problem in $\boldsymbol{U}$ and $\mathbf{Z}$.

K-means

Objective, cost function

K-means considers the following cost function:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{k,n} \|\mathbf{x}_n - \mathbf{u}_k\|_2^2.$$

Constraints

Without constraints: Minimum for $\mathbf{z}_n = \mathbf{0}$. However, also have the constraint: $z_{k,n} = \{0, 1\}$, $\sum_{k=1}^{K} z_{k,n} = 1$. In other words: we need to assign a data point to one and only one cluster.

Mixture models

Different objective, same task

The Gaussian mixture model considers the following function:

$$\ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}.$$

Which we aim at maximizing. Generative, probabilistic model with the following parameters:

- $\boldsymbol{\mu}_k$: centroids, same as $\mathbf{u}_k$ in K-means.
- $\boldsymbol{\Sigma}_k$: covariance of the k-th cluster.
- $\boldsymbol{\pi}$: mixture weights of the clusters.

EM algorithm

Hard assignment vs. soft assignment

- K-means: A data point is always assigned to one cluster.
- EM: probabilities of a data point being assigned to a cluster.

Why might probabilities be useful?

Better estimation of the means!

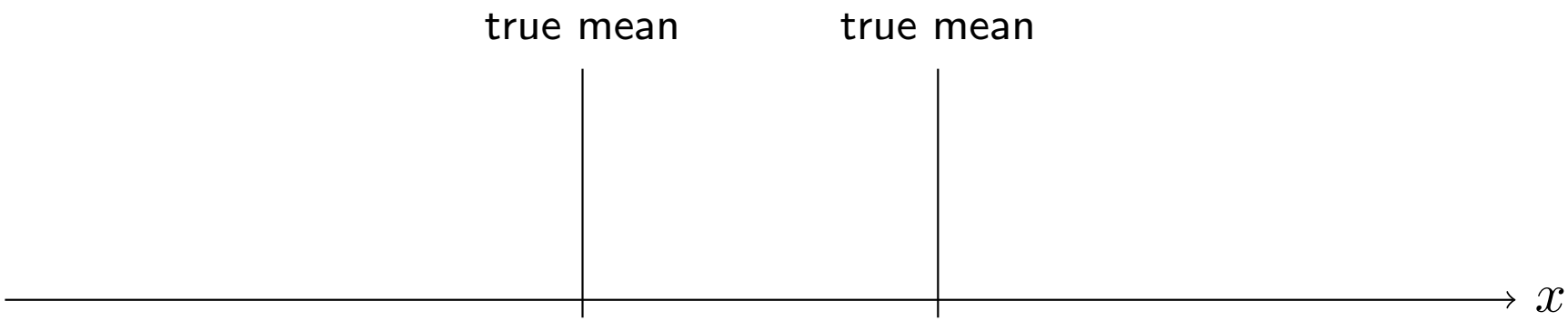

Figure: Means estimated by the K-means algorithm are pushed outside as the overlap part is missing.

Formal Description of EM

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters.

1. Initialize the means $\boldsymbol{\mu}_k$, and mixing coefficients π_k. Set the $\boldsymbol{\Sigma}_k$ to the given covariances.
2. **E-step.** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{k,n}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

3. **M-step.** Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\texttt{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{k,n}) \mathbf{x}_n$$

$$\pi_k^{\texttt{new}} = \frac{N_k}{N} \qquad \texttt{where} \qquad N_k = \sum_{n=1}^{N} \gamma(z_{k,n})$$

4. Evaluate the log likelihood and check for convergence of either the parameters or the log likelihood.

Matrix Factorization

Main theme of the lecture!

Approximate a matrix by

$$\mathbf{X} \approx \mathbf{U}\mathbf{Z}$$

or variants thereof.

K-means algorithm again

Can rewrite the K-means objective

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{k,n}\|\mathbf{x}_n - \mathbf{u}_k\|_2^2 = \sum_{n=1}^{N}\|\mathbf{x}_n - \sum_{k=1}^{K}\mathbf{u}_k z_{k,n}\|_2^2 = \|\mathbf{X} - \mathbf{U}\mathbf{Z}\|_2^2.$$

with $z_{k,n} = \{0, 1\}$, $\sum_{k=1}^{K} z_{k,n} = 1$. Only true for these special constraints, not in general! GMM objective can not be rewritten in this way.

Pen & Paper: Question 1

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

Consider a GMM with $\boldsymbol{\Sigma}_k = \sigma_k^2 \mathbf{I}$ and one of the component means equal to a data point: $\boldsymbol{\mu}_j = \mathbf{x}_n$.

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.
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?

Pen & Paper: Question 2

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.

Assignment Description

- 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.
- 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 Pipeline

The pipeline for the given matrix $\mathbf{X}$ is:

1. Loading the data and splitting it into training/testing sets
2. Imputing missing values
3. Clustering the training matrix
4. Prediction
5. Evaluation

The structure of the implementation

In the given template, there are three main scripts:

1. `CollabFilteringEvaluation:`
2. `Xpred = PredictMissingValues(X, nil):`
3. `[z, U, score] = K-means(data, nClusters, varargin):`

Efficient coding

You will only need to fill in one `for` loop (over the clusters), no more. Everything else can be easily expressed by matrix operations!

```
while (change>threshold && iterations < maxIter),
    iterations = iterations+1;
    disp(sprintf('Iterations = %d  Change = %0.5g.', \
                    iterations, change));

    % Assignment step: estimate class indices

    % Update step: estimate means

    % estimate change
    score_old = score;
    score = sum(sum((data - U*Z).^2, 1));
    change = score_old - score;
end

% convert assignments to vector representation
[foo, z] = max(Z, [], 1);
```

Functions you will need to use

- `min`: Computing minimum, especially the second output is interesting for you.
- `sub2ind`: Going from matrix indices to vector indices.
- `repmat`: Repeat vectors/matrices.
- `./` and `.^2`: Elementwise operations.

If you have problems with division-by-zero leading to `NaN` consider the use of `eps`.

Efficient code: Assignment step

First version (not tested):

```
for n=1:nExamples
    for k=1:nClusters
        Z(k,n) = sum((data(:,n)-U(:,k)).^2);
    end
    [foo, ind] = min(Z(:,n));
    Z(:,n) = 0;
    Z(ind,n) = 1;
end
```

Faster version:

```
% compute distances
for k=1:nClusters,
    Z(k,:) = sum((data-repmat(U(:,k),1,nExamples)).^2,1);
end
% TODO: do something with these distances!
% Can be expressed as matrix operations.
```