Project 3 Algorithms Notes

STAT GU4243

Applied Data Science

Cynthia Rush
Columbia University

March 21, 2018

Second Project Task

Evaluate potential performance enhancements on the baseline momeory-based algorithm (Model 2 above) by considering changes to various components of the algorithm.

- [**All Groups**] Consider different similarity weights: (1) Spearman's correlation, (2) vector similarity, (3) entropy, (4) mean-square difference, and (5) SimRank. Most are discussed in section 5.1 of paper [2]. SimRank is discussed in paper [4].
- [**Groups 1, 2, 3**] Consider significance and variance weighting. Section 5.2 and 5.3 of paper [2].
- [**Groups 4, 5**] Consider selecting neighborhoods. Section 6 of paper [2].
- [**Group 6, 7**] Consider rating normalization. Section 7 of paper [2].

All Groups

Motivation

- Determination of the similarity weights is very important to the way the algorithm makes preditions.

Explore: Will prediction accuracy improve if we consider different similarity weights than that based on Pearson correlation?

Let's look at some examples...

Correlation-based Similarity

Baseline – Pearson correlation – what we coded today:

Let $r_{\mathrm{u,m}}$ be the rating by user 'u' for movie 'm' and $\bar{r}_{\mathrm{u}}$ be user u's average rating.

$$\hat{r}_{\mathrm{a,m}} = \bar{r}_{\mathrm{a}} + \frac{\sum_{\mathrm{u}\in\{\mathrm{users}\}}(r_{\mathrm{u,m}} - \bar{r}_{\mathrm{u}}) \times w_{\mathrm{u,a}}}{\sum_{\mathrm{u}\in\{\mathrm{users}\}} w_{\mathrm{u,a}}},$$

where $w_{\mathrm{u,a}}$ is the Pearson correlation coefficient

$$w_{\mathrm{u,a}} = \frac{\sum_{\mathrm{m}\in\mathrm{movies}}(r_{\mathrm{u,m}} - \bar{r}_{\mathrm{u}}) \times (r_{\mathrm{a,m}} - \bar{r}_{\mathrm{a}})}{\sqrt{\sum_{\mathrm{m}\in\mathrm{movies}}(r_{\mathrm{u,m}} - \bar{r}_{\mathrm{u}})^2 \sum_{\mathrm{m}\in\mathrm{movies}}(r_{\mathrm{a,m}} - \bar{r}_{\mathrm{a}})^2}}.$$

Spearman's Correlation

Motivation

- Pearson correlation measures a linear relationship.
- Spearman's correlation assesses monotonic relationships (whether linear or not).
- Spearman correlation is equal to the Pearson correlation between the rank values of those two variables.

Now $w_{\mathrm{u,a}}$ is calculated as

$$w_{\mathrm{u,a}} = \frac{\sum_{\mathrm{m}\in\mathrm{movies}}(\mathrm{rank}_{\mathrm{u,m}} - \bar{\mathrm{rank}}_{\mathrm{u}}) \times (\mathrm{rank}_{\mathrm{a,m}} - \bar{\mathrm{rank}}_{\mathrm{a}})}{\sqrt{\sum_{\mathrm{m}\in\mathrm{movies}}(\mathrm{rank}_{\mathrm{u,m}} - \bar{\mathrm{rank}}_{\mathrm{u}})^2 \sum_{\mathrm{m}\in\mathrm{movies}}(\mathrm{rank}_{\mathrm{a,m}} - \bar{\mathrm{rank}}_{\mathrm{a}})^2}}.$$

Cosine Similarity

Motivation

- In information retrieval, often compute similarity between two documents by considering vectors of word frequencies and computing cosine f the angle between the two vectors.
- Now users take the role of documents, items are words, and votes are frequencies.

Now $w_{\mathrm{u,a}}$ is calculated as

$$w_{\mathrm{u,a}} = \frac{\sum_{\mathrm{m}\in\mathrm{movies}} r_{\mathrm{u,m}} \times r_{\mathrm{a,m}}}{\sqrt{\sum_{\mathrm{m}\in\mathrm{movies}} (r_{\mathrm{u,m}})^2 \sum_{\mathrm{m}\in\mathrm{movies}} (r_{\mathrm{a,m}})^2}}.$$

Also want to compare entropy-based, mean-square difference, and SimRank similarity measures.

Component: Significance Weighting

Groups 1-3

Motivation

- How much do we trust the computed correlation values if it's based on very small amount of co-rated items.
- Neighbors based on a small number of co-rated samples not usually good predictors for the active user.
- More data points to compare, more we trust the computed correlation.

Explore: Will prediction accuracy improve if we add a correlation significance factor that devalues similarity weights based on a small amount of co-rated items?

Component: Variance Weighting

Groups 1-3

Motivation

- All similarity measures treat each item evenly in a user-to-user correlation calculation.
- In fact, a user's rating on certain items is more valueable than others.
- E.g. lots of people rank Titanic highly, so if two users both rank Titanic highly doesn't tell us much about shared interests.

Explore: Will prediction accuracy improve if we give distinguishing movies more influence in calculating correlation?

Component: Selecting Neighborhoods

Groups 4-5

Motivation

- After calculating similarity weights, we select which other users' data are used in computing the predictions (currently, all of them).
- There is evidence that selecting a subset of users improves accuracy.
- Moreover, when there are millions of users, using them all is infeasible.

Explore: Will prediction accuracy improve if we select the best neighbors of the active user to use in calculating predictions?

Component: Rating Normalization

Groups 6-7

Motivation

- Once a neighborhood is selected, ratings are combined to make a prediction.
- We've been computing a weighted average of the deviation of a neighbor's rating from her mean weighting.
- Could a simple weighted average be better, or maybe a weighted average of z-scores as opposed to simply deviations?

Explore: Will prediction accuracy improve if normalize the weighted average differently?

First Project Task

Implement and **evaluate** the performance of two collaborative filtering algorithms – one model-based and the other memory-based – on both datasets.

- [**Model 1**] Model-based algorithm: clustering discussed in paper [1] section 2.3.
- [**Model 2**] Memory-based algorithm: user-based neighborhood model using Pearson's correlation for similarity weight. This is introduced in equations (1) and equations (2) in paper [2].

Model-based Algorithm

We implemented Model 2 earlier, now let's look at the details of Model 1.

Notation

- N users and M movies.
- Let $I(i)$ be the set of movies users i has scored in the training set.
- Let $r_{i,m}$ be the score user i gave to movie m where $m \in I(i)$ and $r_{i,m} \in \{0, 1, 2, 3, 4, 5\}$.
- Let $R_{i,m}$ be the random variable representing the score of user i on movie m (as opposed to the realization $r_{i,m}$.)

Assume: Each user belongs to one of C different classes or groups. Denote the class of user i by G_i

Probablistic Perspective

CF task is estimating expected value of a vote or rating, given what we know about the user.

Main Idea

Let a be the active user and movie $m \notin I(a)$.

$$\hat{r}_{a,m} = \mathbb{E}(R_{a,m} \mid r_{a,j}, j \in I(a)) = \sum_{k=1}^{5} k \cdot P(R_{a,m} = k \mid r_{a,j}, j \in I(a)).$$

where

$$P(R_{a,m} = k \mid r_{a,j}, j \in I(a))$$

is the probability that user a gives movie m a k rating, given user a's other movie ratings.

Now we need to estimate $P(R_{a,m} = k \mid r_{a,j}, j \in I(a))$ for each k.

Bayesian Cluster Model

Can simplify...

$$P(R_{a,m} = k \mid r_{a,j}, j \in I(a))$$

-
-
-
-

Can simplify...

$$P(R_{a,m} = k \,|\, r_{a,j}, j \in I(a))$$
$$\stackrel{(a)}{=} \frac{P(R_{a,m} = k;\, R_{a,j} = r_{a,j}, j \in I(a))}{P(R_{a,j} = r_{a,j}, j \in I(a))}$$

- Step (a) follows by Bayes' rule.
-
-
-

Can simplify...

$$
\begin{aligned}
& P(R_{a,m} = k \,|\, r_{a,j}, j \in I(a)) \\
& \underset{=}{(a)} \frac{P(R_{a,m} = k;\ R_{a,j} = r_{a,j}, j \in I(a))}{P(R_{a,j} = r_{a,j}, j \in I(a))} \\
& \underset{=}{(b)} \frac{\sum_{c=1}^{C} P(R_{a,m} = k; R_{a,j} = r_{a,j}, j \in I(a);\ G_a = c)}{\sum_{c=1}^{C} P(R_{a,j} = r_{a,j}, j \in I(a);\ G_a = c)}
\end{aligned}
$$

- Step (a) follows by Bayes' rule.
- Step (b) includes the users' groups.
-
-

Can simplify...

$$
\begin{aligned}
& P(R_{a,m} = k \mid r_{a,j}, j \in I(a)) \\
& \overset{(a)}{=} \frac{P(R_{a,m} = k;\ R_{a,j} = r_{a,j}, j \in I(a))}{P(R_{a,j} = r_{a,j}, j \in I(a))} \\
& \overset{(b)}{=} \frac{\sum_{c=1}^{C} P(R_{a,m} = k; R_{a,j} = r_{a,j}, j \in I(a);\ G_a = c)}{\sum_{c=1}^{C} P(R_{a,j} = r_{a,j}, j \in I(a);\ G_a = c)} \\
& \overset{(c)}{=} \frac{\sum_{c=1}^{C} P(R_{a,m} = k; R_{a,j} = r_{a,j}, j \in I(a); \mid G_a = c) \cdot P(G_a = c)}{\sum_{c=1}^{C} P(R_{a,j} = r_{a,j}, j \in I(a) \mid G_a = c) \cdot P(G_a = c)}
\end{aligned}
$$

- Step (a) follows by Bayes' rule.
- Step (b) includes the users' groups.
- Step (c) uses Bayes' rule again.
-

Bayesian Cluster Model

Can simplify...

$$
\begin{aligned}
&P(R_{a,m} = k \,|\, r_{a,j}, j \in I(a)) \\
&\overset{(a)}{=} \frac{P(R_{a,m} = k;\; R_{a,j} = r_{a,j}, j \in I(a))}{P(R_{a,j} = r_{a,j}, j \in I(a))} \\
&\overset{(b)}{=} \frac{\sum_{c=1}^{C} P(R_{a,m} = k; R_{a,j} = r_{a,j}, j \in I(a);\; G_a = c)}{\sum_{c=1}^{C} P(R_{a,j} = r_{a,j}, j \in I(a);\; G_a = c)} \\
&\overset{(c)}{=} \frac{\sum_{c=1}^{C} P(R_{a,m} = k; R_{a,j} = r_{a,j}, j \in I(a);\; |\; G_a = c) \cdot P(G_a = c)}{\sum_{c=1}^{C} P(R_{a,j} = r_{a,j}, j \in I(a) \,|\, G_a = c) \cdot P(G_a = c)} \\
&\overset{(d)}{=} \frac{\sum_{c=1}^{C} P(R_{a,m} = k \,|\, G_a = c) \cdot \prod_{j \in I(a)} P(R_{a,j} = r_{a,j} \,|\, G_a = c) \cdot P(G_a = c)}{\sum_{c=1}^{C} \prod_{j \in I(a)} P(R_{a,j} = r_{a,j} \,|\, G_a = c) \cdot P(G_a = c)}.
\end{aligned}
$$

- Step (a) follows by Bayes' rule.
- Step (b) includes the users' groups.
- Step (c) uses Bayes' rule again.
- Step (d) uses a standard Naive Bayes formulation.

Bayesian Cluster Model

Now, in order to calculate

$$\hat{r}_{a,m} = \mathbb{E}(R_{a,m} \mid r_{a,j}, j \in I(a)) = \sum_{k=1}^{5} k \cdot P(R_{a,m} = k \mid r_{a,j}, j \in I(a)),$$

we need to estimate the following values:

$$P(G_a = c) \text{ for all } c \in \{1, 2, \dots C\},$$
$$P(R_{a,j} = k \mid G_a = c) \text{ for all } j \in I(a) \cup \{m\}, k \in \{0, 1, \dots, 5\}, c \in \{1, 2, \dots C\}.$$

Bayesian Cluster Model

Now, in order to calculate

$$\hat{r}_{a,m} = \mathbb{E}(R_{a,m} \mid r_{a,j}, j \in I(a)) = \sum_{k=1}^{5} k \cdot P(R_{a,m} = k \mid r_{a,j}, j \in I(a)),$$

we need to estimate the following values:

$$P(G_a = c) \text{ for all } c \in \{1, 2, \dots C\},$$
$$P(R_{a,j} = k \mid G_a = c) \text{ for all } j \in I(a) \cup \{m\}, k \in \{0, 1, \dots, 5\}, c \in \{1, 2, \dots C\}.$$

Cluster Assumption

We assume all users in the same class, c, have the same rating probabilities. Namely, for any two users, i_1 and i_2, we will assume:

1. $P(G_{i_1} = c) = P(G_{i_2} = c)$ for all $c \in \{1, 2, \dots, C\}$.
2. $P(R_{i_1,j} = k \mid G_{i_1} = c) = P(R_{i_2,j} = k \mid G_{i_2} = c)$ for all c, k, and j.

Therefore, approximately $C + 6CM$ parameters to estimate!

Parameters to Estimate

Because of the cluster assumption, we can simplify our notation:

$$\mu_c := P(G_a = c) \text{ for all } c \in \{1, 2, \dots C\},$$

$$\gamma_{j,c}^k := P(R_{a,j} = k \mid G_a = c) \text{ for all } j, k, c.$$

where

$$\sum_{c=1}^{C} \mu_c = 1, \quad \text{and} \quad \sum_{k=0}^{5} \gamma_{j,c}^k = 1.$$

Parameters to Estimate

Because of the cluster assumption, we can simplify our notation:

$$\mu_c := P(G_a = c) \text{ for all } c \in \{1, 2, \dots C\},$$

$$\gamma_{j,c}^k := P(R_{a,j} = k \,|\, G_a = c) \text{ for all } j, k, c.$$

where

$$\sum_{c=1}^{C} \mu_c = 1, \quad \text{and} \quad \sum_{k=0}^{5} \gamma_{j,c}^k = 1.$$

Using these notations,

$$P(R_{a,m} = k \,|\, r_{a,j}, j \in I(a)) = \frac{\sum_{c=1}^{C} \mu_c \gamma_{m,c}^k \prod_{j \in I(a)} \gamma_{j,c}^{r_{a,j}}}{\sum_{c=1}^{C} \mu_c \prod_{j \in I(a)} \gamma_{j,c}^{r_{a,j}}},$$

and

$$\hat{r}_{a,m} = \mathbb{E}(R_{a,m} \,|\, r_{a,j}, j \in I(a)) = \sum_{k=1}^{5} k \, \frac{\sum_{c=1}^{C} \mu_c \gamma_{m,c}^k \prod_{j \in I(a)} \gamma_{j,c}^{r_{a,j}}}{\sum_{c=1}^{C} \mu_c \prod_{j \in I(a)} \gamma_{j,c}^{r_{a,j}}}.$$

Parametric Models, Generally

Models

A **model** $\mathcal{P}$ is a set of probability distributions. We index each distribution by a parameter value $\theta \in \mathcal{T}$; we can then write the model as

$$\mathcal{P} = \{p(x|\theta)|\theta \in \mathcal{T}\} \ .$$

The set $\mathcal{T} \subset \mathbb{R}^d$, for some fixed dimension d, is called the **parameter space** of the model.

Parametric model

The model is called **parametric** if the number of parameters (i.e. the dimension of the vector θ) is (1) finite and (2) independent of the number of data points. Intuitively, the complexity of a parametric model does not increase with sample size.

Parametric Models, Generally

Models

A **model** $\mathcal{P}$ is a set of probability distributions. We index each distribution by a parameter value $\theta \in \mathcal{T}$; we can then write the model as

$$\mathcal{P} = \{p(x|\theta)|\theta \in \mathcal{T}\} .$$

The set $\mathcal{T} \subset \mathbb{R}^d$, for some fixed dimension d, is called the **parameter space** of the model.

Parametric model

The model is called **parametric** if the number of parameters (i.e. the dimension of the vector θ) is (1) finite and (2) independent of the number of data points. Intuitively, the complexity of a parametric model does not increase with sample size.

Our Problem

$$\theta = (\mu_c, \gamma_{j,c}^k; \text{ for all } j, k, c),$$
$$\mathcal{T} = \text{ space of all possible values of } \theta = [0,1]^{C+6CM}.$$

Maximum Likelihood Estimation

Setting

- Given: Data $x_1, \ldots, x_n$, parametric model $\mathcal{P} = \{p(x|\theta) \,|\, \theta \in \mathcal{T}\}$.
- Objective: Find the distribution in $\mathcal{P}$ which best explains the data. That means we have to choose a "best" parameter value $\hat{\theta}$.

Maximum Likelihood Estimation

Setting

- Given: Data $x_1, \ldots, x_n$, parametric model $\mathcal{P} = \{p(x|\theta) \,|\, \theta \in \mathcal{T}\}$.
- Objective: Find the distribution in $\mathcal{P}$ which best explains the data. That means we have to choose a "best" parameter value $\hat{\theta}$.

Maximum Likelihood approach

Maximum Likelihood assumes that the data is best explained by the distribution in $\mathcal{P}$ under which it has the highest probability of occurring (or highest density value).

Hence, the **maximum likelihood estimator** is defined as

$$\hat{\theta}_{ML} := \arg\max_{\theta \in \mathcal{T}} p(x_1, \ldots, x_n|\theta)$$

the parameter which maximizes the joint density of the data.

How do we estimate our parameters using ML?

Likelihood Function

For user i, write the **likelihood function** as:

$$L_i(\mu, \gamma \,|\, \text{data}) = P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j})$$

How do we estimate our parameters using ML?

Likelihood Function

For user i, write the **likelihood function** as:

$$\begin{aligned} L_i(\mu, \gamma \,|\, \text{data}) &= P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j}) \\ &= \sum_{c=1}^{C} P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j} \text{ and } G_i = c) \end{aligned}$$

How do we estimate our parameters using ML?

Likelihood Function

For user i, write the **likelihood function** as:

$$\begin{aligned} L_i(\mu, \gamma \,|\, \text{data}) &= P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j}) \\ &= \sum_{c=1}^{C} P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j} \text{ and } G_i = c) \\ &= \sum_{c=1}^{C} P(G_i = c) P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j} \,|\, G_i = c) \end{aligned}$$

How do we estimate our parameters using ML?

Likelihood Function

For user i, write the **likelihood function** as:

$$
\begin{aligned}
L_i(\mu, \gamma \,|\, \text{data}) &= P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j}) \\
&= \sum_{c=1}^{C} P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j} \text{ and } G_i = c) \\
&= \sum_{c=1}^{C} P(G_i = c) P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j} \,|\, G_i = c) \\
&= \sum_{c=1}^{C} P(G_i = c) \prod_{j \in I(i)} P(R_{i,j} = r_{i,j} \,|\, G_i = c)
\end{aligned}
$$

How do we estimate our parameters using ML?

Likelihood Function

For user i, write the **likelihood function** as:

$$
\begin{aligned}
L_i(\mu, \gamma \,|\, \text{data}) &= P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j}) \\
&= \sum_{c=1}^{C} P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j} \text{ and } G_i = c) \\
&= \sum_{c=1}^{C} P(G_i = c) P(\cap_{j \in I(i)} \; R_{i,j} = r_{i,j} \,|\, G_i = c) \\
&= \sum_{c=1}^{C} P(G_i = c) \prod_{j \in I(i)} P(R_{i,j} = r_{i,j} \,|\, G_i = c) \\
&= \sum_{c=1}^{C} \mu_c \prod_{j \in I(i)} \gamma_{j,c}^{r_{i,j}}.
\end{aligned}
$$

Likelihood Function

Then the likelihood of the full data is

$$L(\mu, \gamma \,|\, \text{data}) = \prod_{i=1}^{N} L_i(\mu, \gamma \,|\, \text{data}) = \prod_{i=1}^{N} \Big(\sum_{c=1}^{C} \mu_c \prod_{j \in I(i)} \gamma_{j,c}^{r_{i,j}} \Big).$$

Log-Likelihood Function

Often easier to work with the **log-likelihood function**:

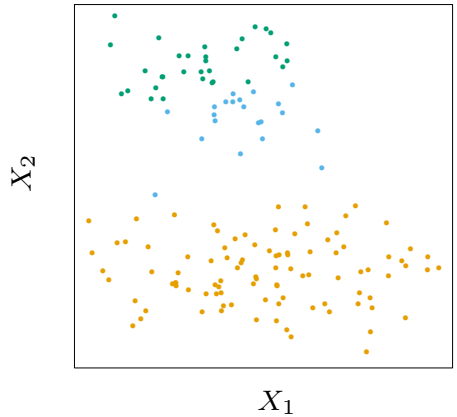
$$\begin{aligned} \ell(\mu, \gamma \,|\, \text{data}) = \log \Big[\prod_{i=1}^{N} L_i(\mu, \gamma \,|\, \text{data}) \Big] &= \sum_{i=1}^{N} \log L_i(\mu, \gamma \,|\, \text{data}) \\ &= \sum_{i=1}^{N} \log \Big(\sum_{c=1}^{C} \mu_c \prod_{j \in I(i)} \gamma_{j,c}^{r_{i,j}} \Big). \end{aligned}$$

EM Algorithm

Quick Review: Clustering

Problem

- Given: Data $x_1, \ldots, x_n$.
- Assumption: Each data point belongs to exactly one group or class. These groups are called **clusters**.
- Our task is to find the clusters, given only the data.



Representation

For C clusters, we encode assignments to clusters as a vector $\mathbf{G} \in \{1, \ldots, C\}^n$ as

$$G_i = c \qquad \Leftrightarrow \qquad x_i \text{ assigned to cluster } c$$

Clustering and classification

Clustering is the "unsupervised" counterpart to classification. There is no training data and no labels, only one, unlabeled data set.

A Very Simple Clustering Algo: K-means

K-means algorithm

- Randomly choose K "cluster centers" (the "means") $\mu_1^{(0)}, \ldots, \mu_K^{(0)} \in \mathbb{R}^d$
- Iterate until convergence (j = iteration number):
  1. Assign each x_i to the closest (in Euclidean distance) mean:

$$G_i^{(j+1)} := \arg \min_{k \in \{1,\ldots,K\}} \|x_i - \mu_k^{(j)}\|$$

  2. Recompute each $\mu_k^{(j)}$ as the mean of all points assigned to it:

$$\mu_k^{(j+1)} := \frac{1}{\left|\{i | G_i^{(j+1)} = k\}\right|} \sum_{i | G_i^{(j+1)} = k} x_i$$

Convergence Criterion

For example: Terminate when a the total change of the means satisfies

$$\sum_{k=1}^{K} \|\mu_k^{(j+1)} - \mu_k^{(j)}\| < \tau \ .$$

The threshold value τ is set by the user.

K-means: Gaussian Interpretation

K Gaussians

Consider the following algorithm:

- Suppose each μ_k is the expected value of a Gaussian density $p(x|\mu_k, \mathbb{I})$ with unit covariance.
- Start with K randomly chosen means and iterate:
  1. Assign each x_i to Gaussian under which it has the highest probability of occurence (more precisely: highest density value).
  2. Given assignments, fit $p(x|\mu_k, \mathbb{I})$ by MLE of μ_k from all points assigned to cluster k.

Comparison to K-means

- Since the Gaussians are spherical with identical covariance, density $p(x_i|\mu_k, \mathbb{I})$ is largest for mean μ_k that's closest to x_i in Euclidean distance.
- The MLE of μ_k is

$$\hat{\mu}_k := \frac{1}{\left|\{i|\, G_i = k\}\right|} \sum_{i|\, G_i = k} x_i$$

This is precisely the k-means algorithm!

What Next

- We will discuss a more sophisticated version of K-means called the *Expectation-Maximization (EM) algorithm.*
- EM gives
  1. A better statistical explanation of what is going on.
  2. A direct generalization to other distributions.
- EM is based on maximum likelihood estimation.

Mixture Models

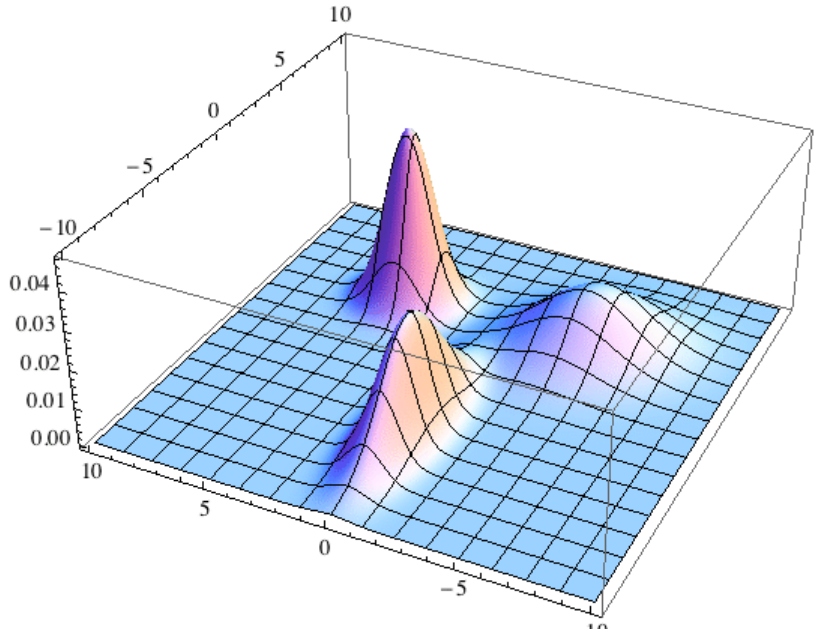
Mixture

For a parametric model $p(x|\theta)$ and a probability distribution μ, a distribution of the form

$$\pi(x) = \sum_{c=1}^{C} \mu_c \, p(x|\theta_c)$$

is called a **finite mixture model**. The distribution given by μ is called the **mixing distribution** and satisfies $\sum_c \mu_c = 1$ and $\mu_c \geq 0$.

Example: Finite mixture of Gaussians

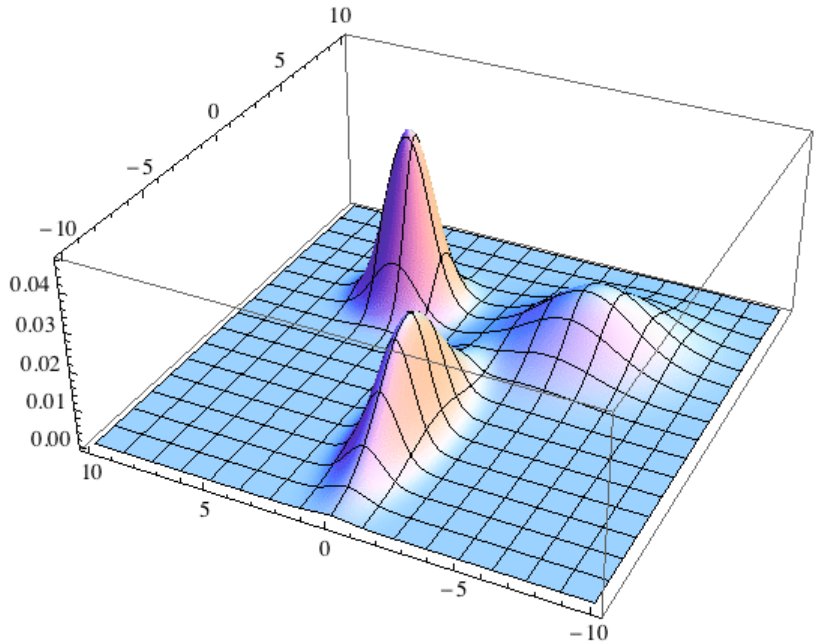


Sampling from a finite mixture

For a finite mixture with fixed parameters μ_c and θ_c, the two-step sampling procedure is:

1. Choose a mixture component at random. Each component c is selected with probability μ_c.
2. Sample x_i from $p(x|\theta_c)$.

Note: We always repeat both steps, i.e. for x_{i+1}, we choose again choose a (possibly different) component at random.



Clustering with finite mixtures

For a clustering problem with C clusters, weight μ_c is relative cluster size and

$$p(x|\theta_c) = \text{model of cluster } c.$$

Estimation problem

If C is fixed and given, unknown parameters of a mixture model are weights μ_c and cluster parameters θ_c. Parameters of finite mixtures are estimated using the *EM algorithm*.

Our Problem

Recall, for our model, for any user a, $\mu_c = P(G_a = c)$ is the relative frequency of cluster c and the model for cluster c is determined by parameters

$$\gamma_{j,c}^k = P(R_{a,j} = k \mid G_a = c).$$

Specifically,

$$p(x_i|\theta_c) = \prod_{j \in I(i)} \gamma_{j,c}^{r_{i,j}}.$$

General Model

In what follows, we consider a general mixture model with C clusters, where weight μ_c is relative cluster size and

$$p(x|\theta_c) = \text{model of cluster } c.$$

For our model, we think of

$$\theta_c = (\gamma_{j,c}^k \text{ for all } j \in \{1, 2, \ldots, M\},\ k \in \{0, 1, 2, 3, 4, 5\}).$$

Maximum likelihood for finite mixtures

Writing down the maximum likelihood problem for the general model is straightforward:

$$(\hat{\mu}, \hat{\theta}) := (\hat{\mu}_1, \ldots, \hat{\mu}_C, \hat{\theta}_1, \ldots, \hat{\theta}_C) = \arg\max_{\mu,\theta} \prod_{i=1}^{n} \Big(\sum_{c=1}^{C} \mu_c p(x_i|\theta_c) \Big)$$

(Compare with what we wrote for our problem a few slides back.) The maximality equation for the logarithmic likelihood is

$$\frac{\partial}{\partial(\mu,\theta)} \sum_{i=1}^{n} \log \Big(\sum_{c=1}^{C} \mu_c p(x_i|\theta_c) \Big) = 0$$

The component equation for each θ_c is:

$$\sum_{i=1}^{n} \frac{\mu_c \frac{\partial}{\partial \theta_c} p(x_i|\theta_c)}{\sum_{c=1}^{C} \mu_c p(x_i|\theta_c)} = 0$$

Solving this problem is analytically infeasible (we can't multiply out the denominator, since we sum over i). Even numerical solution is often difficult.

Reminder: Objective

Estimate θ and μ by (approximate) Maximum Likelihood for

$$\pi(x) = \sum_{c=1}^{C} \mu_c p(x|\theta_c) =: \pi(x|\mu, \theta) \ .$$

Latent Variables

Cluster assignments

- The mixture assumption implies that each x_i was generated from one component.
- For each x_i, we again use an **assignment variable** $G_i \in \{1, \ldots, C\}$ which encodes which cluster x_i was sampled from.

Latent Variables

Since we do not know which component each x_i was generated by, the values of the assignment variables are *unobserved*. Such variables whose values are not observed are called **latent variables** or **hidden variables**.

Estimation With Latent Variables

Latent variables as auxiliary information

If we knew the correct assignments G_i, we could:

- Estimate each component distribution $p(x|\theta_c)$ separately, using only the data assigned to cluster c.
- Estimate the cluster proportions μ_c as $\hat{\mu}_c := \frac{\#\text{points in cluster } c}{n}$.

Estimation With Latent Variables

Latent variables as auxiliary information

If we knew the correct assignments G_i, we could:

- Estimate each component distribution $p(x|\theta_c)$ separately, using only the data assigned to cluster c.
- Estimate the cluster proportions μ_c as $\hat{\mu}_c := \frac{\#\text{points in cluster } c}{n}$.

EM algorithm: Idea

The EM algorithm estimates values of the latent variables to simplify the estimation problem. EM alternates between two steps:

1. Estimate assignments G_i given current estimates of the parameters μ_c and θ_c ("E-step").
2. Estimate parameters μ_c and θ_c given current estimates of the assignments ("M-step").

These two steps are iterated repeatedly.

Representation of Assignments

We re-write the assignments as vectors of length C:

$$\mathbf{x}_i \text{ in cluster } c \qquad \text{as} \qquad M_i := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \longleftarrow c\text{th entry}$$

so $M_{ic} = 1$ if x_i in cluster c, and $M_{ic} = 0$ otherwise.
We collect the vectors into a matrix

$$\mathbf{M} = \begin{pmatrix} M_{11} & \dots & M_{1C} \\ \vdots & & \vdots \\ M_{n1} & \dots & M_{nC} \end{pmatrix}$$

Note: Rows = observations, columns = clusters
Row sums = 1, column sums = cluster sizes.

E-Step

Hard vs soft assignments

- Vectors M_i are "hard assignments" with values in $\{0, 1\}$ (as in k-means).
- EM computes soft assignments" a_{ic} with values in $[0, 1]$.
- Once the algorithm terminates, each point is assigned to a cluster by setting

$$G_i := \arg\max_c a_{ic}$$

The vectors M_i are the latent variables in the EM algorithm. The a_{ic} are their current estimates.

E-Step

Hard vs soft assignments

- Vectors M_i are "hard assignments" with values in $\{0, 1\}$ (as in k-means).
- EM computes soft assignments" a_{ic} with values in $[0, 1]$.
- Once the algorithm terminates, each point is assigned to a cluster by setting

$$G_i := \arg\max_c a_{ic}$$

The vectors M_i are the latent variables in the EM algorithm. The a_{ic} are their current estimates.

Assignment probabilities

If we have estimates of (μ_c, θ_c) for all c, the soft assignments are computed as

$$a_{ic} := \frac{\mu_c p(x_i|\theta_c)}{\sum_{\ell=1}^{C} \mu_\ell \, p(x_i|\theta_\ell)} \, .$$

They can be interpreted as

$$a_{ic} := \mathbb{E}[M_{ic}|x_i, \mu, \theta] = \Pr\{x_i \text{ generated by component } c \,|\, \mu, \theta\}$$

M-Step

Objective

The M-Step re-estimates μ and θ. In principle, we use maximum likelihood within each cluster, but we have to combine it with the use of weights a_{ic} instead of the hard assignments M_{ic}.

M-Step

Objective

The M-Step re-estimates μ and θ. In principle, we use maximum likelihood within each cluster, but we have to combine it with the use of weights a_{ic} instead of the hard assignments M_{ic}.

Cluster sizes

If we know which points belong to which cluster, we can estimate the cluster proportions μ_c by counting point:

$$\hat{\mu}_c = \frac{\#\text{ points in cluster } c}{n} = \frac{\sum_{i=1}^{n} M_{ic}}{n}$$

Since we do not know M_{ic}, we substitute our current best guess, which are the expectations a_{ic}:

$$\hat{\mu}_c := \frac{\sum_{i=1}^{n} a_{ic}}{n}$$

M-Step (2)

Multinomial special case

The estimation of the component parameters θ depends on which distribution we choose for p. For now, we assume a multinomial.

In our model, within each class c, the way any individual (in class c) rates a movie j is the outcome of a multinomial random variable, where the probability giving a rating k is $\gamma_{j,c}^{k}$.

M-Step (2)

Component parameters

We use maximum likelihood to estimate $\theta = (\gamma_{j,c}^k \text{ for all } c, j, k)$. **If we knew the class asssignments,** we could write the MLE of $\gamma_{j,c}^k$ as

$$\hat{\gamma}_{j,c}^k := \frac{\#\text{ people in cluster } c \text{ that gave a rating } k \text{ to movie } j}{\#\text{ points in cluster } c \text{ that rated movie } j}$$
$$= \frac{\sum_{\{i|G_i=c,\, j\in I(i)\}} \mathbb{I}(r_{i,j}=k)}{|\{i|G_i=c,\, j\in I(i)\}|} \quad \text{for } c, j, k.$$

The above is simply the relative frequency of rating k for movie j in class c.

By substituting current best guesses ($= a_{ic}$) again, we get:

$$\hat{\gamma}_{j,c}^k := \frac{\sum_{\{i|j\in I(i)\}} a_{i,c}\, \mathbb{I}(r_{i,j}=k)}{\sum_{\{i|j\in I(i)\}} a_{i,c}} \quad \text{for } c, j, k.$$

Notation Summary

Assignment probabilities

$$\mathbf{a} = \begin{pmatrix} a_{11} & \dots & a_{1C} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nC} \end{pmatrix} = \mathbb{E}\left[\begin{pmatrix} M_{11} & \dots & M_{1C} \\ \vdots & & \vdots \\ M_{n1} & \dots & M_{nC} \end{pmatrix}\right] = \begin{pmatrix} \mathbb{E}[M_{11}] & \dots & \mathbb{E}[M_{1C}] \\ \vdots & & \vdots \\ \mathbb{E}[M_{n1}] & \dots & \mathbb{E}[M_{nC}] \end{pmatrix}$$

Rows = observations, columns = clusters.

Mixture parameters

$\mathbf{tau} = (\mu, \theta)$ $\quad \mu =$ cluster proportions $\quad \theta =$ component parameters

Iterations

$\theta^{(j)}$, $\mathbf{a}^{(j)}$ etc = values in jth iteration

EM in Practice

Comparing solutions

- If (μ, θ) and (μ', θ') are two different EM solutions, we can always compute the log-likelihoods

$$\sum_i \log \pi(x_i|\mu, \theta) \quad \text{and} \quad \sum_i \log \pi(x_i|\mu', \theta')$$

  (no approximations or complications!).
- The solution with the higher likelihood is better.
- This is a very convenient feature of EM: Different solutions are comparable.

Random restarts

In practice, the best way to use EM is often:

- Restart EM repeatedly with randomly chosen initial values.
- Compute the log-likelihoods of all solutions and compare them.
- Choose the solution achieving maximal log-likelihood.

Recall: Bayesian Cluster Model

Parameters to Estimate

$$\mu_c := P(G_a = c) \text{ for all } c \in \{1, 2, \dots C\},$$
$$\gamma_{j,c}^k := P(R_{a,j} = k \,|\, G_a = c) \text{ for all } j, k, c.$$

where

$$\sum_{c=1}^{C} \mu_c = 1, \quad \text{and} \quad \sum_{k=0}^{5} \gamma_{j,c}^k = 1.$$

Likelihood Function

$$\ell(\mu, \gamma \,|\, \text{data}) = \sum_{i=1}^{N} \log \Big(\sum_{c=1}^{C} \mu_c \prod_{j \in I(i)} \gamma_{j,c}^{r_{i,j}} \Big).$$

EM in Our Case

Step 1: Initiate Parameter Estimates

Choose uniform values for the initial estimates $\hat{\mu}_c, \hat{\gamma}_{j,c}^k$, meaning

$$\hat{\mu}_c = \frac{1}{C} \quad \text{for all } c,$$
$$\hat{\gamma}_{j,c}^k = \frac{1}{6} \quad \text{for all } j, k, c.$$

Step 2: Expectation Step

Recompute assignment matrix a_{ic} for $c = 1, 2, \ldots, C$ and $i = 1, 2, \ldots, N$ as

$$a_{ic} := \frac{\hat{\mu}_c \prod_{j \in I(i)} \hat{\gamma}_{c,j}^{r_{i,j}}}{\sum_{\ell=1}^{C} \hat{\mu}_\ell \prod_{j \in I(i)} \hat{\gamma}_{\ell,j}^{r_{i,j}}} .$$

EM in Our Case

Step 3: Maximization Step

Recompute parameter estimates $\hat{\mu}_c, \hat{\gamma}_{j,c}^k$ as

$$\hat{\mu}_c = \frac{1}{N} \sum_{i=1}^{N} a_{ic} \quad \text{for } c = 1, 2, \ldots, C,$$

$$\hat{\gamma}_{c,j}^k = \frac{\sum_{\{i|j \in I(i)\}} a_{ic} \, \mathbb{I}(r_{i,j} = k)}{\sum_{\{i|j \in I(i)\}} a_{ic}} \quad \text{for } c, j, k,$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Note that this step uses the fact that the MLE of a weighted multinomial distribution is only the weighted frequency of each class.

Step 4: Iteration

Iterate steps 2 and 3 until convergence.