Conjugate-Prior-Regularized Multinomial PLSA for Collaborative Filtering

Marcus Klasson

Master's thesis
2016:E12

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Master's Thesis, 2016
Mathematics

Conjugate-Prior-Regularized Multinomial pLSA for Collaborative Filtering

Marcus Klasson

Supervisors: Søren Vang Andersen & Stefan Ingi Adalbjörnsson

Lund University
Faculty of Engineering
Centre for Mathematical Sciences

Abstract

Collaborative filtering is a method for making predictions about consumer interests by collecting preferences or information about opinions from other consumers. For this purpose statistical modeling techniques are applied to learn personalized models for each consumer based on every purchase or provided rating to the available items. Such a technique is *probabilistic Latent Semantic Analysis* (pLSA), which within this thesis attempts to model consumers into groups based on similarities in movie preferences to improve personalized rating predictions on unseen movies. The main challenge with pLSA in collaborative filtering is the overfitting problem, which results in model parameters that are strictly determined by the past ratings and thus gives unreliable predictions for unknown data. To counteract the overfitting a regularization method called *conjugate-prior-regularization* is proposed to introduce additional information about the proportions of the model parameters. It is shown that the proposed regularization provides more robust learning from sparse data sets and also improves the recommendation performance on discrete ratings.

Acknowledgments

First and foremost, I want to express my gratitude to my supervisors Stefan Ingi Adalbjörnsson and Søren Vang Andersen for all the encouragement and trust they have given to me during these two semesters. They have inspired me to always do my best and also to challenge myself when it comes to learning and understanding new knowledge. I look back on this time, even the difficult moments during the work, with great joy and I hope that I get the opportunity to work with them again in the future.

Secondly, I want to thank Andreas Jakobsson and Johan Swärd for their welcoming and positive presence during my thesis work and Magnus Örn Berg for inspiring me to write this thesis and allowing me to use and extend parts of his code. I would also like to thank Carl-Gustaf Werner for giving me a better computer and for helping me with rookie problems in Linux.

Last but by no means least, I want to express my gratitude to the other Master's thesis students I have met at the Department of Mathematics - Lea, Linus, Gabrielle, David, Edvard, Carolina, Jonas, Matilda, Anna, Kajsa, Teo, Erik and Josephine. Thank you for motivating me to have a hard working attitude and for keeping up a happy and cheerful atmosphere during both working hours and coffee breaks. Studying side by side with you has helped me a lot in writing this thesis.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Some Main Challenges With Recommender Systems . . . 2
- 1.3 Scope of the Thesis . . . 3

2 Theoretical Background **4**
- 2.1 Formal Framework for Collaborative Filtering . . . 4
- 2.2 Probabilistic Latent Semantic Analysis . . . 5
  - 2.2.1 Model Definition . . . 5
  - 2.2.2 Drawbacks with pLSA . . . 7
- 2.3 Expectation Maximization Algorithm . . . 9
  - 2.3.1 Maximum Likelihood Estimation . . . 9
  - 2.3.2 Derivation of the EM Algorithm . . . 9
- 2.4 Rating Predictions and Evaluation Metrics . . . 14
  - 2.4.1 Metrics for Prediction Accuracy . . . 14
  - 2.4.2 Leave-one-out Algorithm . . . 14
  - 2.4.3 Baselines in Collaborative Filtering . . . 15
- 2.5 Regularization Methods . . . 15
  - 2.5.1 Early Stopping . . . 16
  - 2.5.2 Maximum a Posteriori (MAP) Estimation . . . 16
  - 2.5.3 Conjugate Prior Distributions . . . 17
  - 2.5.4 Conjugate-Prior-Regularized Learning . . . 18
  - 2.5.5 Selection of the Hyperparameters $\gamma_{u,z}$ and $\gamma_{i,r,z}$ . . . 20

3 Methodology **22**
- 3.1 The EachMovie Data set . . . 22
- 3.2 Evaluation Metrics . . . 22
- 3.3 First Experiment - pLSA . . . 23
- 3.4 Second Experiment - ES pLSA . . . 23
- 3.5 Third Experiment - Conjugate-Prior-Regularized pLSA . . . 23
  - 3.5.1 Grid Search Procedure . . . 23
  - 3.5.2 Finding the Optimal State Size k . . . 24

4 Results **25**
- 4.1 Experiment With pLSA . . . 25
- 4.2 Experiment With ES pLSA . . . 27

4.3 Experiment With Conjugate-Prior-Regularized pLSA 29
4.3.1 Grid Search Procedure Experiment 29
4.3.2 Varying the Latent State Size k Experiment 32
4.4 Comparing Prediction Errors of pLSA Models 34

5 Discussion 37
5.1 Data sets 37
5.2 Implementation Issues 38
5.3 pLSA With Tolerance Level Condition Only 38
5.4 pLSA With Early Stopping Condition 39
5.5 Conjugate-Prior-Regularized pLSA 39
5.5.1 Selection of Conjugate Prior Hyperparameters 40
5.5.2 The Grid Search Procedure 40
5.5.3 Varying the Latent State Size k 41
5.5.4 Extensions of the Model 42

6 Conclusion 44

Bibliography 45

Appendix A Proofs and Derivations 48
A.1 Proof of Jensen's Inequality 48
A.2 Derivation of $Q^*(z; u, i, r; \theta)$ 49
A.3 Derivation of $P(z|u)$ and $P(r|i, z)$ 50
A.4 Derivation of the Conjugate Prior Distribution $P(\theta)$ 53
A.5 Derivation of the MAP Parameter Estimations 54

Appendix B Case Study 57
B.1 Introduction 57
B.2 MovieLens 1M Data set 57
B.3 Ranking Accuracy Metrics 58
B.4 Line Search for Optimal Hyperparameters 59
B.5 Experimental Setup 59
B.6 Results and Discussion 61

1 Introduction

1.1 Background

Since streaming sites like Netflix and Spotify were launched the number of users has increased rapidly. Also, e-commerce has skyrocketed in recent years. The purchase of media or items has thus been more accessible, e.g., by introduction of new services such as mobile apps, which in turn probably lead to an increase of objects to select. With the extended range of items users seem to request recommendations for items they have not observed. For instance, a user whose favorite movies are Interstellar and The Deer Hunter may request for other movies that the user would enjoy, or need service in finding which accessories that would fit to a certain garment. However, people tend to be disturbed by getting "bad" recommendations.

It is indeed difficult to recommend something to a person who has not purchased anything from a particular service, or to purchasers who occasionally or often change their consumption pattern. Good tools for this are hard to find, but recommender systems that are employed in the correct way may actually aid good statistical predictions of the most likely choice by the consumer.

A very important question to ask is how the recommender system is supposed to handle the data collection, i.e., what kind of information should the recommendations be based upon? There are different ways to do so, but the two techniques most commonly used are (i) content-based filtering and (ii) collaborative filtering. *Content-based filtering* learns to recommend items that the purchaser has liked in the past and calculates the similarity of items based on features associated with the compared items. *Collaborative filtering* (CF) on the other hand, recommends items to the purchaser that others with similar tastes liked in the past and the calculations are based on the similarity in the rating history of the purchasers. CF techniques' main advantage is their simplicity, because only past ratings are needed in the learning process [2, p. 13][3, p. 11-12].

Collaborative filtering approaches can be divided in two categories; namely *memory-based* and *model-based* methods, where the latter will be focused on within this thesis. Memory-based methods uses neighborhood-based techniques, e.g., *K-nearest-neighbours (K-NN)*, to compute a similarity between pairs of users and items based on known ratings. These methods directly transform stored preference data into predictions, but the whole data set has to be accessible in order to make recommendations. Model-based methods use all known ratings to learn a personalized model for each user and then utilizes the model to predict the user's degree of interest for particular items [2, p. 14], [3, p. 784]. A refinement of model-based methods is to use statistical methods relying on probabilistic modeling in both learning and prediction phases. A generalization of such probabilistic approaches is the *probabilistic Latent Semantic Analysis* (pLSA), which was developed in the context of information retrieval and first presented by Thomas Hofmann in 1999 [1, p. 90].

1.2 Some Main Challenges With Recommender Systems

Sparseness. In any recommender system the amount of users and items are typically large. Still, most users consume only a small fraction of the available items and rarely provide a rating for the purchased item if they are requested to. Therefore, the data collected may not contain enough information about particular users and items. An insufficient set of data with respect to the modeling purpose can of course seriously affect the accuracy of recommendations. Several techniques has been proposed to deal with the data sparsity, such as dimensionality reduction methods and user/item-clustering techniques [1, p. 91], [2, p. 9], but so far no technique have been proven better than the other.

Unbalancedness. There occurs significant differences in levels of activity between users and the popularity of items. Every user has its personalized consumption behavior in how often and how much they consume items. If providing ratings to items is optional, another important aspect is how often they end up rating a consumed item and how they use the rating scale. The user activities corresponds thus to various popularity of items. Popular items also tend to be more easily accessible, since they are often more visible in the systems application or web site.

Cold start. A "cold start" is when a new user or item is registered by the system. The recommendation engine cannot give any suggestions to new users, since the recommendations are based on user preferences. Moreover, an unknown item that no potential consumer has purchased yet can be difficult to relate to other items. To deal with this issue new users could be asked to rate a list of items or to select some favorite genres in the registration phase. Another approach to the deal with cold start is to consider both ratings and additional information, such as demographic information about users; i.e., age, gender, and item features such as genre, rating, or year of release. Using more specific information will probably make it easier to find relationships between new users or items and already registered users and items in the system catalog. [2, p. 9].

Scalability. The recommender system needs to rapidly generate high-quality suggestions among a large collection of items. This calls for the adoption of well-defined data structures for data analysis and scalable methods, which could be focusing on item-to-item relationships or tend to model individual users in communities. Scalability is typically measured by experimenting with growing data sets, showing how the speed and resource consumption behave as the task scales up [2, p. 9].

It is very important to measure how fast the system can generate recommendations, e.g., by investigating the number of recommendations the system can provide per second. Thus, evaluations of the computational complexity of algorithms is an essential part in the construction of recommender systems. The possibilities for an algorithm to be successful is preferably enhanced by changing some parameters, such as the complexity of the model or the sample size [3, p. 293]. Another aspect of scalability is the usage of multiple processors in order to parallelize the algorithms computational tasks.

Privacy. Collection of user preferences can of course be exploited to identify information about particular consumers. Therefore, recommender systems pose a serious threat to individual privacy. Analysis of a person's private integrity tends to focus on a worst case scenario, such that private information from all the users is revealed. For recommender systems in general, the disclosure of information gathered on a single user is considered very inappropriate. However, experimental studies have shown that accurate recommendations are possible while user privacy is preserved [2, p. 9], [3, p. 291-292].

1.3 Scope of the Thesis

This thesis proceeds from Hofmann's article [1] in order to understand and reproduce his proposed pLSA method used for predicting movie ratings. The multinomial pLSA model is considered, so that the predicted ratings are modeled with a multinomial distribution, and with an extended regularization method called *conjugate-prior-regularized learning*. The experiments will investigate if the multinomial distribution can do a better job at modeling discrete rating data with the proposed regularization technique rather than modeling the ratings as continuous variables described by a Gaussian distribution, which is proposed in [1]. As for pLSA and most model-based CF techniques, the main drawback when making predictions is the overfitting phenomenon, which results in unreliable parameter estimations. Introducing a prior distribution, such as a conjugate prior, to the model has been proven to be successful in mitigating overfitting in [11], [12]. For these experiments, the retired EachMovie data set is used in order to make comparisons to the results in [1], since the EachMovie data was used there as well. All implementations are made in Matlab.

For future studies, a case study is included in Appendix B where a thresholding or quantization will be made on the rating scale in order to represent the ratings as a like or dislike. This can be viewed as a discrete version of the normalization of the Gaussian-emission pLSA in [1]. Three different rating scale setups will be utilized by the conjugate-prior-regularized multinomial pLSA model to investigate and compare their prediction and recommendation performances. The 1M MovieLens data set is used for these experiments.

2 Theoretical Background

This chapter explains in general how a recommender system with collaborative filtering could be designed. The main discussion is about the pLSA model and the essential mathematics behind it, such as parameter estimations, prediction metrics and regularization methods.

2.1 Formal Framework for Collaborative Filtering

Further in the report the terms *users* and *items* will be used thoroughly, where user means a registered consumer in a community and item is a certain movie within that community's movie collection. Let $U = \{u_1, \dots, u_m\}$ be a set of m users and $I = \{i_1, \dots, i_n\}$ a set of n items. It is possible for users to rate items with a preference value, such as a degree of appreciation. That user preference or rating data $r_{u,i}$ is stored in a $m \times n$ matrix called $\mathbf{R}$, where Figure 2.1 shows an example of such a matrix. In real-world applications, $\mathbf{R}$ is characterized by an exceptional sparseness which is due to the fact that individual users tend to only rate a small amount of the available items. In a way, it is also because the number of users and items often are very large (typically with $m >> n$) [1, p. 91][2, p. 2].

The user preferences are considered either as *implicit* or *explicit*. Implicit data correspond to the observation of the co-occurrence of u and i, and can be collected from, e.g., clicks, likes/dislikes or check-ins. Hence, the entry $r_{u,i}$ in the rating matrix $\mathbf{R}$ is a binary value, i.e., $r_{u,i} = 0$ if there are no co-occurrence and $r_{u,i} = 1$ if the co-occurrence exists. Entries $r_{u,i}$ could also be a count variable, such that the variable value is the number of clicks that a user has made on an item. Explicit data are actual ratings from individual users on experienced items from which users are asked to proactively evaluate the purchased/selected item on a given rating scale $\mathcal{R}$, where $r \in \mathcal{R}$. The corresponding feedback offers thus the users a chance to explicitly express their preferences. These ratings $r_{u,i}$ may be represented as a positive or negative integer corresponding to like/dislike or as a score on a fixed rating scale [1, p. 91], [2, p. 2-3].

In collaborative filtering frameworks using explicit ratings, there are two different settings for the prediction problem; *forced prediction* and *free prediction.* Forced prediction involves predicting the preference value or rating for a particular item that a certain user is able to purchase. Thus, one may state the conditional probability $P(r|u,i)$ that user u will rate item i with r. Forced prediction mimics an experimental setup where a person is presented with various items and it is compulsory to give a response to the items. This is a typical scenario when items are presented as recommendations to a user and one is interested in foreseeing the user's response [1, p. 91-92], [2, p. 26-27].

	i_1	i_2	i_3	i_4	i_5
u_1	4	3	3		1
u_2	4	4	5		
u_3	3	3		3	
u_4	4	1		2	
u_5					5

Figure 2.1: Example of a 5×5 user preference matrix $\mathbf{R}$ with explicit ratings on a 5-level scale.

In free prediction, the item selection is part of the predictive model and the goal is to learn the probabilities $P(r, i|u)$ in order to predict both which item the user will select and (optionally) the associated rating. The probability $P(r, i|u)$ may be rewritten as $P(r, i|u) = P(r|i, u)P(i|u)$. Thus, the problem is partitioned into predicting which item the user will select and then predicting the provided rating to the hypothetically selected item. This setting mimics a scenario where the user is free to select an item by choice and - in the case of explicit ratings - provides a rating for it [1, p. 91-92], [2, p. 26-27].

2.2 Probabilistic Latent Semantic Analysis

Probabilistic Latent Semantic Analysis is a statistical method for analysis of co-occurrence data. It was presented in 1999 by Thomas Hofmann and is influenced by *Latent Semantic Analysis* (LSA), which is based on *Singular Value Decomposition* (SVD). Unlike those methods pLSA offers statistical techniques for inference, such as maximum likelihood (ML) estimation, and utilizes model selection and complexity control for model fitting to enhance the reliance in resulting calculations. The applications of pLSA ranges from information retrieval and filtering, natural language processing and machine learning from text in addition to recommender systems [1, p. 90-91]. The idea is to discover meaningful and semantic relationships in the given data, such that different topics are distinguished from the data. Within this thesis, preference data is exploited to the model in order to find the latent structure of topics or concepts, which are supposed to capture "abstract preference patterns" about users and items [2, p. 45-46].

2.2.1 Model Definition

Initially, pLSA is specified by a co-occurrence data model, where the data is thought of as implicit preferences represented as user-item pairs $\langle u, i\rangle$ [1, p. 94]. This model is called *the aspect model* [5, p. 290] and all observation pairs $\langle u, i\rangle$ that are exploited by the model are assumed to have been generated independently. The key idea is to introduce *hidden variables* Z which associates an unobserved latent state $z \in \{z_1, \ldots, z_k\}$ with each user-item pair $\langle u, y\rangle$ (observation) [1, p. 94][8, p. 180]. This is assumed in most statistical latent variable models in order to make user u and item i conditionally independent on the state z [5, p. 290]. The maximum number of possible states is finite and of size k.

The aspect model is defined by a *mixture model* which aims to model the joint probability $P(u, i)$ with a set of parameters θ. A mixture model is a general probabilistic approach that introduces a set of latent factor variables to represent a generative process in which data points may be generated from a finite set of probability distributions. The distributions may be quite complex, but the introduction of latent variables allows simpler modeling of the distribution components [2, p. 32]. The aspect model explains both components in an observation pair $\langle u, i\rangle$ in terms of a latent variable, i.e., associating both user u and item i in the pair to a latent state z [2, p. 38], such that

$$P(u, i; \theta) = \sum_z P(u, i, z) = \sum_z P(i|z)P(z|u)P(u), \tag{2.1}$$

where the summation symbol over z means summing over all possible k states [1, p. 94]. Note that the parameter vector θ contains every probability distribution in the model and that $P(z|u)$ and $P(i|z)$ are proper conditioned probabilities, i.e., $\sum_z P(z|u) = 1$ and $\sum_i P(i|z) = 1$. Equation (2.1) is the most commonly used structure of the aspect model, but it may also be parameterized equivalently by applying Bayes' rule such that $P(u, i; \theta') = \sum_z P(z)P(u|z)P(i|z)$ and $P(u, y; \theta'') = \sum_z P(u|z)P(z|i)P(i)$ [1, p. 94], [5, p. 290]. Since the latent states z will be modeled based on the user preferences within this thesis, it may be assumed that no further information about the users is provided in the probability to purchase a certain item. This assumption states the conditional independence between u and i through z.

The typical situation in collaborative filtering is to make personalized, user-specific recommendations. The recommendation engine's main goal is thus to find which items the consumer would most likely purchase and therefore the conditioned probability $P(i|u)$ is considered instead of $P(u, i)$ [1, p. 94]. By also applying the law of total probability, the outcome of this is the mixture model

$$P(i|u; \theta) = \sum_z P(i|z, u)P(z|u). \tag{2.2}$$

Since the conditional independence described above is assumed then $P(i|z, u) = P(i|z)$, which leads to

$$P(i|u; \theta) = \sum_z P(i|z)P(z|u), \tag{2.3}$$

where $\theta = \{P(i|z), P(z|u)\}$ is the parameter vector containing all $P(i|z)$ and $P(z|u)$ in a total of $k|I| + k|U|$ probabilities, where $|I|$ and $|U|$ denotes the total number of items and users in the sets I and U respectively.

By taking the response variable or rating r into account and considering the forced prediction case, $P(i|z)$ is replaced with $P(r|i, z)$. Item i is then treated as a fix conditioned variable and the observations becomes triplets $\langle u, i, r\rangle$. If r is a categorical (discrete) variable, e.g., a score on a rating scale, one can parameterize the conditional probability $P(r|i, z)$ by introducing so called success probability parameters $\pi_{i,z}^{(r)} \in [0; 1]$ and defining $P(r|i, z) \equiv \pi_{i,z}^{(r)}$. Then the properties for $P(r|i, z)$ will be [1, p. 99]

$$P(r|i,z) = \pi_{i,z}^{(r)}, \quad \text{where} \quad \sum_{r \in \mathcal{R}} \pi_{i,z}^{(r)} = 1. \tag{2.4}$$

Finally, this and the earlier assumptions leads to the following mixture model

$$P(r|u,i;\theta) = \sum_{z} P(r|i,z)P(z|u). \tag{2.5}$$

Note that $\theta = \{P(r|i,z), P(z|u)\}$ consists of $k|I||\mathcal{R}| + k|U|$ variables now, where $|\mathcal{R}|$ denotes the number of possible ratings on the rating scale $\mathcal{R}$.

The latent states z of a hidden variable Z may seem unclear and there are actually no priori or theoretically built meaning associated to them [1, p. 95]. Moreover, the latent state itself is not directly observable, but the outcome will hopefully be some interesting data structure about user communities and groups of related or similar items. A state z associated to an observation $\langle u, i, r\rangle$ is intended to model a hidden cause for the occurrence/situation when user u has selected and rated item i with r. Each z provides thus a hypothetical explanation to the given rating if the user or item clusters are grouped in a reasonable way. Also, since the number of possible states k is typically selected to be much smaller than both the number of users and items, the model encourages grouping users into user communities and, correspondingly, groups together related items. Due to the sparseness of available data in practice, the size of k should be selected with great care as the k-factor immediately adjusts the model complexity [1, p. 95].

The pLSA model may seem very similar to probabilistic clustering models, but it is important not to confuse them with each other [1, p. 98]. In clustering techniques each user would be associated with a single latent state, while in pLSA each observation $\langle u, i, r\rangle$ is connected to a latent state. In other words, different observations for the same user are explained with different latent causes z in the pLSA model, whereas in a user clustering model all the ratings from a certain user are related to the same underlying cluster/community [2, p. 47].

The pLSA and aspect model shares many similarities, such as; single observations are associated to latent states rather than to users, and also that the probability for a co-occurrence is governed by a distribution over all possible items. But the aspect model is focused on free prediction and aims directly at modeling the joint probability $P(u, i)$, while pLSA considers the probability $P(i|u)$ [2, p. 47]. Also, pLSA may apply free or forced prediction mode when ratings are included to the model. Within this thesis the forced prediction case is considered, since the interest lays in modeling how users tend to rate particular items and not their consumption behavior. This is because the aim is to recommend movies that will receive high ratings from the consumers and not suggest movies that they just usually would watch.

2.2.2 Drawbacks with pLSA

A major drawback with pLSA (and most other model-based approaches to recommender systems) is *overfitting*, which is a model selection problem. A rough definition of overfitting is to say that the model has an exceptionally good capability to describe the observations in the training set, but has poor prediction accuracy for unseen observations [2, p. 55]. The model selection and complexity is controlled by the latent state size k in pLSA and the model is trained, usually with some iterative procedure, to generate parameter values with respect to a known data set called *training set.* Since the goal with the recommender system is to achieve good prediction accuracy for new or missing data, then the model must be validated on some independent data referred to as a *validation set.* From a machine learning perspective, this evaluation phase is part of the training by validating, e.g., the prediction error, model order selection or a particular hyperparameter. The prediction accuracy is measured for both the training and validation set with some statistical metric, such as the root mean squared error (RMSE) or mean absolute error (MAE). The prediction error of the training set is typically non-increasing, while the validation set error often shows a decrease at first followed by an increase as the number of iterations grows. The final evaluation of the model is made with a second unknown data set, which is usually called a *test set.* The test set is exposed to the model to the selected prediction error metrics after the training phase in order to strengthen the validated model's credibility.

A way to reduce overfitting is to increase the size of the training data set, if such factor is available. Since the pLSA model is quite complex (some may say "more flexible") with many parameters to estimate, the model needs a lot of data points in the training data set in order to fit the parameters correctly according to the data [4, p. 9]. Another technique to control overfitting involves adding a coefficient to an optimization function; the technique called *regularization.* There are different approaches to achieve regularization, such as introducing *Lagrange multipliers* or using *conjugate-prior-penalization* [11], [12], where the latter mainly will be used within this thesis. There is also alternatives to regularization, such as *early stopping*, which will be tested to mitigate overfitting [1], [4]. These methods will be explained in more detail later.

Another, but rather minor, disadvantage with pLSA is the sensitivity to the initial conditions in the learning procedure. Model sensitivity can be tested by performing several training runs using different initial configurations on the parameters, or just by applying some fair setting to the initial parameter values [2, p. 52]. The initial $P(z|u)$ was selected uniformly in the experiments within this thesis, since it seemed reasonable not to force a user to belong in a specific state in the initialization phase. $P(r|i,z)$ was initialized by selecting a random Normal distributed number to all possible ratings for a fixed item and state and then normalizing each number into probabilities by dividing with the sum of the randomly selected numbers. This initial setting could of course be selected the other way around with $P(z|u)$ being selected randomly and $P(r|i,z)$ uniformly. But this should not have a major impact on the outcome, since the parameter values depends more on the training data set than their initialized values. It should be mentioned that both parameters should not be selected uniformly, because it will make the parameter values converge back to the initial values after a couple of iterations.

2.3 Expectation Maximization Algorithm

In latent variable models, such as pLSA, the parameter learning optimization procedure for ML estimation is the *Expectation Maximization* (EM) algorithm. EM is alternated between two steps: (i) the expectation (E) step, where the variational probabilities for the latent states given the current parameter estimates are computed and (ii) the maximization (M) step, where the parameters are updated based on the likelihood function, which depends on the posterior probabilities computed in the previous E-step [6, p. 4-5], [8, p. 182].

2.3.1 Maximum Likelihood Estimation

Before deriving the EM algorithm, it may be appropriate to recall the definition of ML estimation. The parameter vector θ consisting of the conditional probabilities $P(z|u)$ and $P(r|i,z)$ is supposed to be fitted to the observed data, where the data is assumed to be independent and identically distributed. The probability for the parameters in θ given the observation triplets $\langle u,i,r\rangle$ is called "likelihood". It is defined as the product of $P(r|u,i;\theta)$ for all observations given by

$$L(\theta) = \prod_{\langle u,i,r\rangle} P(r|u,i;\theta), \tag{2.6}$$

where $\langle u,i,r\rangle$ under a summation or product symbol is used as shorthand notation referring to all observation triplets. To make calculations easier and numerically more tractable, it is preferable to use the log-likelihood

$$\ell(\theta) = \log L(\theta) = \log \prod_{\langle u,i,r\rangle} P(r|u,i;\theta) = \sum_{\langle u,i,r\rangle} \log P(r|u,i;\theta). \tag{2.7}$$

The log-likelihood $\ell(\theta)$ should be maximized in order to obtain the optimal parameters in θ, i.e., the model parameters that are most suitable to regenerate the given data. Thus, the ML estimator is obtained and given by

$$\theta^{*}_{\mathrm{ML}} = \arg\max_{\theta}\, \ell(\theta) = \arg\max_{\theta} \sum_{\langle u,i,r\rangle} \log P(r|u,i;\theta). \tag{2.8}$$

The above equation can be solved by taking the derivative of $\ell(\theta)$ with respect to the model parameters, setting the equations to zero and solving them directly. It may though be difficult to find such analytic expressions in estimation problems involving latent variables. Consequently, elaborated techniques, such as the EM algorithm, has to be used for parameter estimations of this kind [9, p. 1].

2.3.2 Derivation of the EM Algorithm

This section will show the derivations of the EM algorithm for multinomial $P(r|i,z)$ only, since the rating data will be considered as discrete. The pLSA model's log-likelihood function is given by

$$\ell(\theta) = \sum_{\langle u,i,r \rangle} \log P(r|u,i;\theta) = \sum_{\langle u,i,r \rangle} \log \sum_z P(r|i,z)P(z|u). \tag{2.9}$$

In optimization problems there is typically some cost or loss function that is to be minimized. Hence, it is appropriate to minimize the negative log-likelihood, such that the best fit parameters are given by

$$\theta^* = \arg\min_\theta \, (-\,\ell(\theta)) = \arg\min_\theta \left(-\sum_{\langle u,i,r \rangle} \log \sum_z P(r|i,z)P(z|u) \right). \tag{2.10}$$

The fact that the latent states z are unknown, and also because the negative log-likelihood includes the logarithm of a sum over the latent states, will make the minimization of $-\ell(\theta)$ quite difficult. To circumvent this problem, *variational probability distributions* $Q(z;u,i,r;\theta)$ are introduced for every observation triplet, where $Q(z;u,i,r;\theta) \geq 0$ and $\sum_z Q(z;u,i,r;\theta) = 1 \; \forall \langle u,i,r \rangle$. Intuitively, the Q-distribution models the probability for a latent state z to be associated with a certain observation $\langle u,i,r \rangle$, such that there are $k \cdot |\mathbf{R}|$ number of parameters $Q(z;u,i,r;\theta)$, where $|\mathbf{R}|$ is the total number of observations. Extending the negative log-likelihood function with the Q-distributions results in

$$-\,\ell(\theta) = -\sum_{\langle u,i,r \rangle} \log \sum_z P(r|i,z)P(z|u) \frac{Q(z;u,i,r;\theta)}{Q(z;u,i,r;\theta)} \tag{2.11a}$$

$$= -\sum_{\langle u,i,r \rangle} \log \sum_z Q(z;u,i,r;\theta) \frac{P(r|i,z)P(z|u)}{Q(z;u,i,r;\theta)} \tag{2.11b}$$

$$\leq -\sum_{\langle u,i,r \rangle} \sum_z Q(z;u,i,r;\theta) \log \frac{P(r|i,z)P(z|u)}{Q(z;u,i,r;\theta)} \tag{2.11c}$$

where the last step holds due to Jensen's inequality [1, p. 95-96]. For a convex function f, Jensen's inequality states that

$$f\left(\sum_{n=1}^N \lambda_n x_n \right) \leq \sum_{n=1}^N \lambda_n f(x_n), \tag{2.12}$$

for the constants $\lambda_n \geq 0$ with $\sum_{n=1}^N \lambda_n = 1$ [6, p. 3]. By letting the constants λ_n to be Q-distributions, because $Q(z;u,i,r;\theta)$ is a probability measure and they have the same properties, and since the negative logarithm is a convex function, it is clear that Jensen's inequality is applicable between Equation (2.11b) and (2.11c) [6, p. 5-6]. A proof for Jensen's inequality from [6] can be found in appendix A.1.

The upper bound in Equation (2.11c), which further on will be called $\mathcal{L}(Q,\theta)$, may be derived to

$$\mathcal{L}(Q,\theta) = -\sum_{\langle u,i,r\rangle}\sum_{z} Q(z;u,i,r;\theta)\left[\log P(r|i,z)P(z|u) - \log Q(z;u,i,r;\theta)\right] \tag{2.13a}$$

$$= -\sum_{\langle u,i,r\rangle}\left(\sum_{z} Q(z;u,i,r;\theta)\log P(r|i,z)P(z|u) - \sum_{z} Q(z;u,i,r;\theta)\log Q(z;u,i,r;\theta)\right) \tag{2.13b}$$

$$= -\sum_{\langle u,i,r\rangle}\left(\mathrm{E}_{z\sim Q(z;u,i,r;\theta)}\left[\log P(r|i,z)P(z|u)\right] + H\left[Q(z;u,i,r;\theta)\right]\right), \tag{2.13c}$$

where $z \sim Q(z;u,i,r;\theta)$ means that z was drawn from that particular Q-distribution and H is the *entropy*[1] with respect to $Q(z;u,i,r;\theta)$ [2, p. 155]. Therefore, the Expectation step (E-step) includes evaluating the expected value of $\log P(r|i,z)P(z|u)$ with respect to z drawn from $Q(z;u,i,r;\theta)$. The term entropy is a measure for uncertainty in the outcome of a discrete random variable. In this setup, it can be interpreted as some uncertainty in the estimated Q-distributions. When the entropy is zero, the "true" Q-distributions, i.e., the $Q(z;u,i,r;\theta)$ that describe every observations association to all possible states z best, have been found and then equality holds in Equation (2.11).

Suppose that θ^{old} is the current parameter estimates. In the E-step, the upper bound $\mathcal{L}(Q,\theta^{\text{old}})$ is minimized with respect to the Q-distributions while θ^{old} is held fixed [4, p. 451]. Thus, one has to find an expression for the optimal $Q(z;u,i,r;\theta^{\text{old}})$, denoted by $Q^*(z;u,i,r;\theta^{\text{old}})$. Introducing a Lagrange multiplier to $\mathcal{L}(Q,\theta^{\text{old}})$, such that the normalization constraints $\sum_z Q(z;u,i,r;\theta) = 1 \ \forall \ \langle u,i,r\rangle$ holds, a Lagrange function can be formed. Minimizing that Lagrange function with respect to $Q(z;u,i,r;\theta)$, an expression for the optimal variational distributions $Q^*(z;u,i,r;\theta)$ can be derived, such that

$$Q^*(z;u,i,r;\theta) = \frac{P(r|i,z)P(z|u)}{\sum_{z'} P(r|i,z')P(z'|u)}, \tag{2.14}$$

where $\theta = \theta^{\text{old}}$ such that $P(z|u)$ and $P(r|i,z)$ are the current parameter estimates. The latent state z denotes the single state whose variational probability is computed, while the sum over z' still means summing over all possible latent states [1, p. 96]. For the complete derivation of (2.14) see appendix A.2. The computed Q^*-distributions will then be applied to the upper bound, such that $\mathcal{L}(Q^*,\theta)$ will represent an approximation of the negative log-likelihood $-\ell(\theta)$ for any given parameter vector θ [2, p. 156]. In other words, the equality will hold in Equation (2.11)

In the subsequent M-step, the distribution $Q^*(z;u,i,r;\theta^{\text{old}})$ is held fixed and the upper bound $\mathcal{L}(Q^*,\theta^{\text{old}})$ is minimized with respect to θ in order to give a new parameter estimate θ^{new}. This will cause $\mathcal{L}$ to decrease further, unless $\mathcal{L}(Q^*,\theta^{\text{old}}) = -\ell(\theta^{\text{old}})$ already is the upper bounds minimum. Since $\mathcal{L}$ and $-\ell$ are coupled together in the E-step through $Q(z;u,i,r;\theta)$, any minimization of the upper bound will make the negative

[1] $H\left[Q(z;u,i,r;\theta)\right] = -\sum_z Q(z;u,i,r;\theta)\log Q(z;u,i,r;\theta)$

log-likelihood follow it constantly. Also, because the Q^*-distributions are determined using old parameter values and are held fixed during the M-step, they will not equal the new Q^*-distributions computed with the next parameter updates. Then through the entropy in Equation (2.13c), there will be a difference between the upper bound and the negative log-likelihood, which causes $-\ell(\theta)$ to decrease at least as much as $\mathcal{L}(Q^*, \theta^{\text{old}})$ does [4, p. 451].

After the E-step, the upper bound takes the form

$$\mathcal{L}(Q^*, \theta^{\text{old}}) = -\sum_{\langle u,i,r\rangle} \sum_{z} Q^*(z; u, i, r; \theta^{\text{old}}) \log P(r|i, z)P(z|u) + \text{constant}, \tag{2.15}$$

where the constant is the entropy of the Q-distribution, since it is independent of θ. Thus, in the M-step, the quantity that is being minimized is the expectation of $\log P(r|i, z)P(z|u)$ in Equation (2.13c), since the parameter vector θ, which is optimized, only appears inside the logarithm. Minimizing Equation (2.15) with respect to the parameters in θ separately yields the new vector of optimal parameters θ^{new}. In order to achieve this, some constrained optimization problems for the parameters using Lagrange multipliers need to be solved. The derivations, which can be found in appendix A.3, give the set of equations

$$P(z|u) = \frac{\sum_{\langle u',i,r\rangle : u'=u} Q^*(z; u, i, r; \theta)}{\sum_{z'} \sum_{\langle u',i,r\rangle : u'=u} Q^*(z'; u, i, r; \theta)}, \tag{2.16a}$$

$$P(r|i, z) = \frac{\sum_{\langle u,i',r'\rangle : i'=i,\, r'=r} Q^*(z; u, i, r; \theta)}{\sum_{\langle u,i',r\rangle : i'=i} Q^*(z; u, i, r; \theta)}, \tag{2.16b}$$

where the prime signs under the summations denote a fixed variable for the conditional probability computed.

To summarize the EM algorithm, the procedure can be viewed graphically through Figure 2.2, which shows a two-dimensional space consisting of $-\ell(\theta)$ with respect to θ. The negative log-likelihood function $-\ell(\theta)$ is represented as a convex function with a local minimum. Starting with the initial parameter set θ^{old}, the first E-step evaluating $Q(z; u, i, r; \theta^{\text{old}})$ gives rise to the blue upper bound $\mathcal{L}_{\text{old}}(Q, \theta)$ in Figure 2.2. The blue bound is a convex approximation in the point $-\ell(\theta^{\text{old}})$, such that equality only holds for $\mathcal{L}_{\text{old}}(Q, \theta^{\text{old}}) = -\ell(\theta^{\text{old}})$. Thus, the upper bound makes tangential contact with $-\ell(\theta)$ at θ^{old}, so that both curves have the same gradient. In the M-step, the upper bound is minimized with respect to θ resulting in the new parameter set θ^{new}, which gives a smaller value of the negative log-likelihood than θ^{old}. Since $\mathcal{L}_{\text{old}}(Q, \theta^{\text{new}})$ is not a convex approximation of $-\ell(\theta^{\text{new}})$, i.e., $\mathcal{L}_{\text{old}}(Q, \theta^{\text{new}}) \neq -\ell(\theta^{\text{new}})$, the upper bound need to be updated for the new optimal parameter vector θ^{new}. Hence, the following E-step constructs the green upper bound $\mathcal{L}_{\text{new}}(Q, \theta)$, which is a convex approximation in the point $-\ell(\theta^{\text{new}})$, such that $\mathcal{L}_{\text{new}}(Q, \theta^{\text{new}}) = -\ell(\theta^{\text{new}})$ [4, p. 452-453].

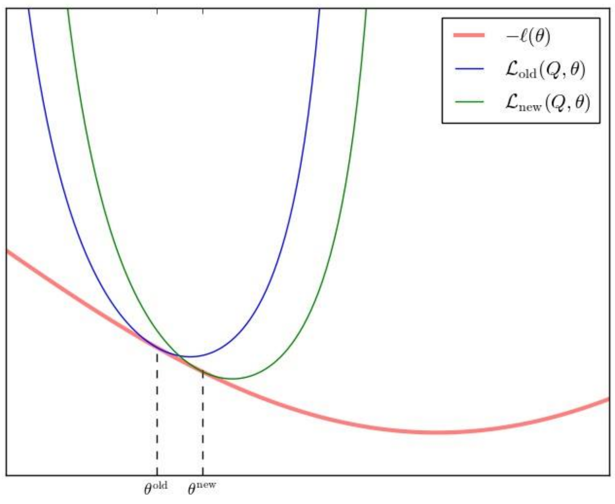

Figure 2.2: The EM algorithm described graphically with the negative log-likelihood function $-\ell(\theta)$, shown as the red curve, with respect to the parameter values in θ. The blue and green curves are the upper bounds $\mathcal{L}_{\text{old}}(Q,\theta)$ and $\mathcal{L}_{\text{new}}(Q,\theta)$ constructed from the parameter vectors θ^{old} and θ^{new} respectively.

The EM procedure alternates between the two steps until the optimal parameters resulting in the smallest negative log-likelihood value has been found. The EM algorithms optimization procedure, where the upper bound $\mathcal{L}(Q,\theta)$ is optimized, thus consists of [2, p. 156]:

- Initialize $\theta^{(0)}$
- Repeat until convergence, for increasing EM step t:

  E-step: Given $\theta^{(t)}$, compute $Q^*(z;u,i,r;\theta^{(t)}) = \arg\min_Q \mathcal{L}(Q,\theta^{(t)})$

  M-step: Given $Q^*(z;u,i,r;\theta^{(t)})$, compute $\theta^{(t+1)} = \arg\min_{\theta^{(t)}} \mathcal{L}(Q^*,\theta^{(t)})$

As explained above, for any given parameter vector θ there is a unique minimum of the upper bound $\mathcal{L}(Q,\theta)$ with respect to $Q = Q^*(z;u,i,r;\theta)$ and this choice of Q results in $\mathcal{L}(Q,\theta) = -\ell(\theta)$ for one specific value of θ. This concludes that since the algorithm converges to the global minimum of $\mathcal{L}(Q,\theta)$, it will also find a value of θ that gives the global minimum of the negative log-likelihood $-\ell(\theta)$ [4, p. 454].

The requirement of minimization may be relaxed to simply decreasing $-\ell(\theta)$, instead of necessarily finding its local minimum [6, p. 8]. This approach is called *Generalized Expectation Maximization* (GEM) algorithm and is useful in cases where the M-step optimization is intractable. Since $\mathcal{L}(Q,\theta)$ is still an upper bound on $-\ell(\theta)$, each complete EM step of GEM is guaranteed to decrease (or be unchanged if the parameters already correspond to the local minimum) the negative log-likelihood [4, p. 454].

2.4 Rating Predictions and Evaluation Metrics

The goal in collaborative filtering in recommender systems is to predict ratings or preferences as accurately as possible. Thus, the recommendation problem can be interpreted as a missing value prediction problem [2, p. 5], such that for a given active user, the system is asked to predict the user's preferences among a set of items. Let the predicted rating be denoted by $\hat{r}_{u,i}$ for the observation pair $\langle u, i \rangle$. When computing the predicted rating the formula for the expected rating is used, which is given by [1, p. 100]

$$\hat{r}_{u,i} = \mathrm{E}[r|u,i] = \sum_{r \in \mathcal{R}} rP(r|u,i) = \sum_{r \in \mathcal{R}} r \sum_{z} P(r|i,z)P(z|u), \tag{2.17}$$

where $\mathcal{R}$ is the given discrete rating scale. Note that if the rating scale would be continuous, then the summation symbol over $r \in \mathcal{R}$ is replaced with integration.

2.4.1 Metrics for Prediction Accuracy

In the evaluation of a ratings-based recommender system a preferred metric is to measure how close the predicted ratings are to the actual ratings. The aim is then to minimize the difference between the provided rating and the predicted one. Some widely known statistical metrics for prediction accuracy are [2, p. 5]:

- The **Root Mean Squared Error (RMSE).** Measures the deviation of observed ratings from predicted values and emphasizes large errors. RMSE is defined as

$$\mathrm{RMSE} = \sqrt{\frac{1}{|S|} \sum_{\langle u,i \rangle \in S} (r_{u,i} - \hat{r}_{u,i})^2}. \tag{2.18}$$

- The **Mean Absolute Error (MAE).** Measures the average absolute deviation between the predicted and actual rating. MAE is defined as

$$\mathrm{MAE} = \frac{1}{|S|} \sum_{\langle u,i \rangle \in S} |r_{u,i} - \hat{r}_{u,i}|. \tag{2.19}$$

S is the set containing all real ratings $r_{u,i}$ in the user preference matrix $\mathbf{R}$. The estimate in Equation (2.17) minimizes the RMSE and MAE if the model is correct.

2.4.2 Leave-one-out Algorithm

The training, validation and test data sets are obtained with the *leave-one-out* algorithm. The main idea is to randomly pick one rating from every user and remove it from the entire data set. The model is then trained on the reduced entire set, which is called the training set, and the left out ratings are used in a validation or test phase [1, p. 104]. Left out ratings are thus mostly for measuring and evaluating the recommender systems ability to predict ratings [3, p. 517].

The leave-one-out demands that users possess a minimal number of $M \geq 2$ observed ratings. Otherwise, if a particular user has only provided a single preference to $\mathbf{R}$ this user's preference will not be considered during the model training phase. When using both validation and test data sets the minimal number has to be $M \geq 3$, since the validation and test sets are picked out separately. The minimal number of observations M may be varied to investigate the prediction accuracy for users with M available ratings [1, p. 104]. By increasing M the prediction accuracies would probably improve for the valid users with enough observed ratings, since the trained model has only considered users who has provided a sufficient amount of preference information to the model.

2.4.3 Baselines in Collaborative Filtering

As explained earlier, collaborative filtering techniques are based on past ratings and their main feature is their capability to summarize experiences on multiple users and items. However, there are other less robust methods that still rely on the preference matrix, but only focuses on individual entries while ignoring their collaborative characteristics. Such methods are [2, p. 13]:

- The **item average**, defined as

$$\text{itemAvg}(i) \equiv \bar{r}_i = \frac{1}{|U(i)|} \sum_{u \in U(i)} r_{u,i}, \tag{2.20}$$

  where $U = \{u_1, \dots, u_m\}$ is the set of all m users and $U(i)$ denotes a set of all users that have rated item i.

- The **user average**, defined as

$$\text{userAvg}(u) \equiv \bar{r}_u = \frac{1}{|I(u)|} \sum_{i \in I(u)} r_{u,i}, \tag{2.21}$$

  where $I = \{i_1, \dots, i_n\}$ is the set of all n items and $I(u)$ denotes a set of all items rated by user u.

These averages can be used for comparisons in prediction accuracy between different recommendation techniques and are typically called baselines. The item average, which is also referred to as *pop*, is a non-personalized algorithm that ranks every item according to their popularity of the items in the training set that the users have rated [13, p. 5]. Within this thesis, pop will mainly be used as baseline, since it was also used as baseline for the experiments in [1, p. 104].

2.5 Regularization Methods

Regularization is a technique regularly practiced to reduce overfitting. Usually, it involves adding a penalty term to an error function in order to discourage the parameters from reaching large or undesirable values [4, p. 10]. This section discusses the *maximum a posteriori* (MAP) regularization approach, but also an alternative to regularization called *early stopping*. Both methods are widely known in the field of machine learning when trying to avoid the overfitting problem.

2.5.1 Early Stopping

Since the negative log-likelihood is decreasing for every EM iteration, the prediction errors of the training set will typically decrease as well. The validation error will also decrease in the beginning of the training, but will flatten out or even increase, if the number of model parameters are larger than the available data points, after some iterations. Therefore, the training can be stopped when the validation sets smallest possible error has been reached. This condition is called early stopping [4, p. 259] and is used for controlling overfitting to a minor extent.

In [1, p. 107] it is proposed that after the stopping condition is reached to run one last EM iteration using the full data set, i.e., training plus validation set. This is a quite different approach from the general early stopping setting as the training and validation sets are merged together. Thus, there will probably occur some hassles when the model is exploited with new independent data from the test set. The latter early stopping approach will still be tested in the experiments to make comparisons with the regularized pLSA model.

2.5.2 Maximum a Posteriori (MAP) Estimation

Since there are no assumptions made on the prior likelihood of θ in ordinary ML estimation, all kinds of values on the parameters within the constraints, e.g., $\sum_z P(z|u) = 1$, are equally probable. Optimal parameter values will thus be uniquely identified by the observed data. The following approach allows the parameter estimations to include some prior knowledge by weighting the parameters with a prior distribution $P(\theta)$. The prior knowledge is then combined with the observed data to obtain the optimal parameter vector θ^*. Hence, instead of optimizing the likelihood $P(D|\theta)$, the posterior probability $P(\theta|D)$ is maximized to find the best fitted parameters given the data and the prior [2, p. 53-54]. Note that D is a set containing all observation triplets: $\langle u, i, r\rangle \in D \; \forall \; \langle u, i, r\rangle$. This new setting may be introduced by expressing the likelihood through Bayes' rule [10, p. 2], given by[2]

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}. \tag{2.22}$$

The corresponding terminology is

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}.$$

[2]Derivation: $P(\theta|D) \cdot P(D) = P(D, \theta) = P(D|\theta) \cdot P(\theta)$.

As the name *maximum a posteriori*, the objective is to maximize the posterior with respect to the parameters in θ in order to obtain new optimal parameter estimates

$$\theta^*_{MAP} = \arg\max_{\theta} P(\theta|D), \tag{2.23}$$

which is proportional to

$$\theta^*_{MAP} = \arg\max_{\theta} \frac{P(D|\theta)P(\theta)}{P(D)} \propto \arg\max_{\theta} P(D|\theta)P(\theta). \tag{2.24}$$

It is also preferred in MAP estimation to take the negative logarithm of the posterior, or product of the likelihood and the prior, which is given by [12, p. 200]

$$-\log P(\theta|D) \propto -\log P(D|\theta)P(\theta) = -\log P(D|\theta) - \log P(\theta). \tag{2.25a}$$

Since $\log P(D|\theta)$ represents the log-likelihood, then $-\log P(D|\theta) = -\ell(\theta)$ from Equation (2.11). Hence, an upper bound on the log-posterior is constructed in the same fashion as for the EM case, such that

$$\begin{aligned} -\log P(\theta|D) &\propto -\ell(\theta) - \log P(\theta) && (2.26a)\\ &\leq -\sum_{\langle u,i,r\rangle}\sum_{z} Q^*(z;u,i,r;\theta)\log\frac{P(r|i,z)P(z|u)}{Q^*(z;u,i,r;\theta)} - \log P(\theta). && (2.26b)\end{aligned}$$

The best fit of the regularized parameters is thus given by minimizing the upper bound in Equation (2.26b) with respect to θ,

$$\theta^*_{MAP} = \arg\min_{\theta} -\sum_{\langle u,i,r\rangle}\sum_{z} Q^*(z;u,i,r;\theta)\log\frac{P(r|i,z)P(z|u)}{Q^*(z;u,i,r;\theta)} - \log P(\theta). \tag{2.27}$$

The MAP approach allows a smoothing in the estimation procedure, such that the regularization constrains the optimal parameter values in a specific range. Including the prior in the estimations allows thus better control of the overfitting problems [2, p. 54]. The next section discusses how to select the prior distribution $P(\theta)$.

2.5.3 Conjugate Prior Distributions

In Bayesian statistics, the model is allowed to have some prior belief about the parameters that are to be estimated. In MAP estimation, and therefore Bayesian inference, it is a practical choice to represent the prior distribution $P(\theta)$ with a *conjugate prior* [11, p. 288], [12, p. 200]. Each possible likelihood function is said to have a conjugate prior distribution, which in fact has the same distribution as the posterior but with different parameters. Since the posterior and the prior then would belong to the same "family of distributions", EM update rules can be derived to obtain the optimal parameter estimates [11, p. 288], [15, p. 641].

Conjugate priors are constructed through derivations and partitioning of a likelihood function that are based on finding parts that are dependent and independent of the model parameters. The conjugate prior is defined proportional to the derived function segment that is dependent of the model parameters of interest. This function needs to be derived further with elaborated techniques for transformation and reparameterization in order to find the prior hyperparameters, see e.g., [14, p. 3-8]. These derivations will yield a tractable posterior distribution with the same distributional form as the obtained prior [11, p. 287].

Since the conjugate priors belongs to different distributional families, their parametric structures looks different depending on which likelihood function they have been derived from. Advantageously, probably most of the available likelihood distributions have already been derived earlier in literature studies, such as [14], to determine which particular distribution their conjugate prior belongs to. In pLSA models using MAP estimations, the commonly used conjugate priors are for the Gaussian distribution, which is the product of a Gaussian and a Wishart density, and the multinomial distribution, which is a Dirichlet density [11, p. 288], [12, p. 200].

A Dirichlet distribution of order $m \geq 2$ with parameters $\mathbf{w} = (w_1, \dots, w_m)$ where $w_i \geq 0$ with $\sum_{i=1}^{m} w_i = 1$ has a probability density function given by

$$\mathrm{Dir}(\{\mathbf{w}\}|\{\gamma\}) = \frac{\Gamma\left(\sum_{i=1}^{m} \gamma_i\right)}{\prod_{i=1}^{m} \Gamma(\gamma_i)} \prod_{i=1}^{m} w_i^{\gamma_i - 1}, \tag{2.28}$$

where the hyperparameters $\gamma = (\gamma_1, \dots, \gamma_m)$ and $\Gamma(\cdot)$ is the Gamma function[3] [4, p. 76-77].

The hyperparameters of the Dirichlet distribution can be selected with relatively great freedom. They may be updated iteratively in the parameter estimation stage (M-step) [12, p. 202], or they may be set constant for their corresponding distribution [11, p. 289]. Optionally, the hyperparameters could have a constant parameter-specific value based on some particular prior belief on the user or item. Within this thesis, every hyperparameter in the Dirichlet distributions will have the same value for their individual distribution for simplicity.

2.5.4 Conjugate-Prior-Regularized Learning

In MAP approaches the prior distribution $P(\theta)$ is seen as a penalty term added to the log-likelihood function. The main advantage when this penalty term is chosen as the logarithm of a conjugate prior is that a variant of the M-step can be derived to obtain optimal regularized parameter estimates. This method is called *conjugate-prior-regularized learning*, which was proposed in [11] to improve the learning ability of Gaussian mixture models (GMMs). Note that in [11] the regularization method is referred to as conjugate-prior-penalized learning, but within this thesis emphasizing the actual regularization was preferred.

[3]The Gamma function is defined by $\Gamma(x) \equiv \int_0^\infty t^{x-1} e^{-t}\, dt$ [4, p. 62].

Since the model consists of two multinomial distributions, consequently the prior distribution is thus given by

$$\begin{aligned} P(\theta) &= \prod_{\langle u,i,r\rangle} \prod_{z} \mathrm{Dir}(\{P(r|i,z)\}|\{\gamma_{i,r,z}\}) \cdot \mathrm{Dir}(\{P(z|u)\}|\{\gamma_{u,z}\}) && (2.29a) \\ &\propto \prod_{z} \left[\prod_{i,r} P(r|i,z)^{\gamma_{i,r,z}-1} \prod_{u} P(z|u)^{\gamma_{u,z}-1} \right], && (2.29b) \end{aligned}$$

where $\varphi = \{\gamma_{i,r,z}, \gamma_{u,z}\}$ are the hyperparameters of each Dirichlet densities [12, p. 200]. The product over both i and r denotes the product for all possible items and ratings. For simplicity, the normalization constant with Gamma functions of the Dirichlet densities has been omitted, because the Gamma functions are independent of the parameters in θ. The logarithm of $P(\theta)$ is thus proportional to

$$\begin{aligned} \log P(\theta) &= \sum_{z} \left[\sum_{i,r} \log \mathrm{Dir}(\{P(r|i,z)\}|\{\gamma_{i,r,z}\}) + \sum_{u} \log \mathrm{Dir}(\{P(z|u)\}|\{\gamma_{u,z}\}) \right] && (2.30a) \\ &\propto \sum_{z} \left[\sum_{i,r} (\gamma_{r,i,z} - 1) \log P(r|i,z) + \sum_{u} (\gamma_{u,z} - 1) \log P(z|u) \right]. && (2.30b) \end{aligned}$$

By inserting Equation (2.30) in (2.26), the negative log-posterior function is expressed as

$$\begin{aligned} -\log P(\theta|D) &= -\,\ell(\theta) - \log P(\theta) && (2.31a) \\ &\leq -\sum_{\langle u,i,r\rangle} \sum_{z} Q^{*}(z;u,i,r;\theta) \log \frac{P(r|i,z)P(z|u)}{Q^{*}(z;u,i,r;\theta)} \\ &\quad - \sum_{z} \left[\sum_{i,r} \log \mathrm{Dir}(\{P(r|i,z)\}|\{\gamma_{i,r,z}\}) + \sum_{u} \log \mathrm{Dir}(\{P(z|u)\}|\{\gamma_{u,z}\}) \right]. && (2.31b) \end{aligned}$$

As for the EM algorithm, the upper bound in the above equation is minimized with respect to θ in order to find the optimal MAP parameter estimates,

$$\begin{aligned} \theta^{*}_{MAP} = \arg\min_{\theta} & -\sum_{\langle u,i,r\rangle} \sum_{z} Q^{*}(z;u,i,r;\theta) \log \frac{P(r|i,z)P(z|u)}{Q^{*}(z;u,i,r;\theta)} \\ & - \sum_{z} \left[\sum_{i,r} (\gamma_{r,i,z} - 1) \log P(r|i,z) + \sum_{u} (\gamma_{u,z} - 1) \log P(z|u) \right]. \end{aligned} \qquad (2.32)$$

A more thorough derivation of the prior distribution is shown in appendix A.4.

The optimization is performed using Lagrange multipliers subject to the constraints $\sum_r P(r|i,z) = 1$ and $\sum_z P(z|u) = 1$ [12, p. 200]. Hence, the set of equations for the regularized parameters is given by

$$P(z|u) = \frac{\sum_{\langle u',i,r\rangle : u'=u} Q^*(z;u,i,r;\theta) + (\gamma_{u,z} - 1)}{\sum_{z'} \sum_{\langle u',i,r\rangle : u'=u} Q^*(z';u,i,r;\theta) + (\gamma_{u,z'} - 1)} \tag{2.33a}$$

$$P(r|i,z) = \frac{\sum_{\langle u,i',r'\rangle : i'=i,\, r'=r} Q^*(z;u,i,r;\theta) + (\gamma_{i,r,z} - 1)}{\sum_{\langle u,i',r\rangle : i'=i} Q^*(z;u,i,r;\theta) + (\gamma_{i,r,z} - 1)}. \tag{2.33b}$$

These derivations are found in appendix A.5. It is worth mentioning that conjugate-prior-regularization only affects the M-step. Since the prior distribution $P(\theta)$ is independent of the Q-distributions, minimizing the negative log-posterior with respect to $Q(z;u,i,r;\theta)$ will result in the same expression for the optimal $Q^*(z;u,i,r;\theta)$ as in Equation (2.14). This implies that extending the model with the prior $P(\theta)$ will have no influence on the E-step.

2.5.5 Selection of the Hyperparameters $\gamma_{u,z}$ and $\gamma_{i,r,z}$

The hyperparameters for the conjugate priors may be interpreted as *sufficient statistics* of an additional set of artificial data observations [4, p. 117], [15, p. 642]. Since, e.g., $P(z|u)$ depends on the real observations $\langle u,i,r\rangle$ plus the value for the hyperparameter $\gamma_{u,z}$, therefore these two factors are called the sufficient statistics for the Dirichlet distribution $\mathrm{Dir}(\{P(z|u)\}|\{\gamma_{u,z}\})$ [4, p. 93, 117]. Consequently, $\gamma_{i,r,z}$ is a sufficient statistic for $\mathrm{Dir}(\{P(r|i,z)\}|\{\gamma_{i,r,z}\})$. By including the artificial observations, the parameter estimates are given extra knowledge beyond the real data used in the training procedure. This results in softer user clustering within $P(z|u)$ and, respectively within $P(r|i,z)$, a greater chance for providing different ratings on the rating scale. Thus, the selection of hyperparameter values is quite crucial, because they adjust the randomness of the pLSA parameters [12, p. 202].

By introducing the same hyperparameters for each Dirichlet distribution, the prior belief is that each multinomial is generated from an equal number of samples. The size of the artificial data is then defined by adopting the notation of an *equivalent sample sizes* $\omega_{\{P(z|u)\}}$ and $\omega_{\{P(r|i,z)\}}$ [11, p. 289], [15, p. 642], which leads to the following equations

$$\gamma_{u,z} = 1 + \omega_{\{P(z|u)\}}, \tag{2.34a}$$

$$\gamma_{i,r,z} = 1 + \omega_{\{P(r|i,z)\}}. \tag{2.34b}$$

The degree of regularization is thus determined by varying the equivalent sample sizes $\omega_{\{P(z|u)\}}$ and $\omega_{\{P(r|i,z)\}}$ [11, p. 289]. Note that the original EM algorithm is obtained by setting $\omega_{\{P(z|u)\}}$ and $\omega_{\{P(r|i,z)\}}$ equal to zero. These few artificial data points are thus uniformly distributed over each state and rating. Hence, one could interpret that by increasing the equivalent sample size, one is increasing the belief that each state, or rating, is equally probable.

3 Methodology

This chapter discusses the different parts in the experiments of the pLSA model, such as data sets, evaluation metrics and practical methods.

3.1 The EachMovie Data set

The EachMovie data set has been collected by DEC Research, today known as HP/Compac Research, from 1995 to 1997 [1, p. 103][17]. The number of users and items in the data set used in the experiments differs a bit from the ones described in [1] and [17], probably because the references has different end dates of the data collection. The data set that was used for these experiments contains $2,811,718$ ratings entered by $61,265$ users for 1623 movies (items). The average amount of ratings per user is 45.9. The ratings are given from a discrete six-star scale with values from the lowest rating 1 to the highest rating 6. The mean rating over all observation triplets is ≈ 4.04 and the overall rating variance is ≈ 2.44.

3.2 Evaluation Metrics

The EachMovie data set was divided into training, validation and test data sets using the leave-one-out algorithm. This procedure was performed twice on the complete data set by first picking out the test set and thereafter the validation set. The remaining data set was used for training and consisted of about $2,692,585$ ratings, while the test and validation sets consisted of about $60,087$ and $59,044$ ratings, respectively. The minimum amount of observations for the users was set to $M = 2$ and users with less than M ratings were removed from the data sets. Since one randomly picked rating is removed from the valid users, this will lead to that some users that are included in the test set are removed from the training and validation set because these users may only have one rating left after the first leave-one-out procedure. This may lead to worse prediction performances when evaluating the model on the test set.

During each EM iteration the user ratings were predicted with the expected value of the rating from equation (2.17). The prediction error, i.e., the deviation between the predicted rating and the real observed rating, was then measured with RMSE and MAE, since both were used for model evaluation in [1]. The pop (item average) defined by equation (2.20) was used as baseline measure [1, p. 104] and was calculated based on the training set ratings only. RMSE and MAE for the pop baseline was computed separately for training, validation and test sets.

3.3 First Experiment - pLSA

The first setup investigates how the pLSA models prediction accuracy is affected when the number of possible latent states is varied. The same amount of states used for the pLSA in [1, p. 104-108] will be used to compare the results. Ten training, validation and test sets are obtained with different random seeds from the EachMovie data set using leave-one-out algorithm in order to gain some statistical significance and reduce the variance in the results.

For every EM step the negative log-likelihood is evaluated for the current parameter estimates. The EM algorithm stops the training when the log-likelihood has decreased with less than 10^{-3}, according to the condition

$$|\,\ell(\theta^{(t)}) - \ell(\theta^{(t-1)})\,| < 10^{-3}, \tag{3.1}$$

where t is the current iteration step in the EM algorithm. The parameter values are thus said to have converged when the decrease of the negative log-likelihood is that small.

The stopping condition in Equation (3.1) will depend relatively on the amount of data, i.e., the more data available in the training set, the smaller absolute overall log-likelihood. This is due to that the likelihood is calculated from the product of probability mass functions over all observations. Smaller values of the log-likelihood will lead to smaller differences between the iteration steps, which may result in that the stopping criteria is reached too quickly. Another factor that also could lead to small values of the log-likelihood is the total number of probability mass functions in the model, which is dependent on the number of available latent states z and the possible rating levels.

3.4 Second Experiment - ES pLSA

This setup is an attempt to reproduce the pLSA using the early stopping (ES) method described in Section 2.5.1, which will further on be called ES pLSA. The data sets picked for the first experiment are also used here and the model is trained for the same states sizes k as earlier. The ES condition is reached if the validation sets RMSE increases at any iteration and then performs one last EM step with the training plus validation data. The algorithm will also be terminated if the log-likelihood condition in Equation (3.1) occurs, but then the extra EM step is not taken.

3.5 Third Experiment - Conjugate-Prior-Regularized pLSA

The conjugate prior hyperparameters $\gamma_{u,z}$ and $\gamma_{i,r,z}$ are held constant during the training procedures for simplicity and each hyperparameter value is varied with Equation (2.34).

3.5.1 Grid Search Procedure

In order to find beneficial hyperparameter values, or degrees of regularization on each multinomial parameter $P(z|u)$ and $P(r|i,z)$, that could decrease the prediction errors, a cross-validation method was used. By executing the training with various values on each hyperparameter and evaluating the RMSE, a 3-D shaded surface plot was constructed for the RMSE values with respect to the different $\gamma_{u,z}$ and $\gamma_{i,r,z}$. Hence, the hyperparameters which may lead to the best prediction accuracies are obtained by studying the grid. As earlier, the training procedure was terminated when the stopping condition in Equation (3.1) is reached. The grid procedure was performed with $k = 200$ states, since the lowest prediction errors were obtained with 200 latent states in the second experiment setup. It is worth mentioning that it may be advantageous to run some pre-experiments for different intervals and combinations of the hyperparameters to see for which intervals the RMSE seems to have its minimum, because the grid search is a time-costly procedure.

3.5.2 Finding the Optimal State Size k

When appropriate hyperparameter intervals have been found based on the grid search, investigations to see if the same hyperparameters combinations fit well for other latent state sizes than $k = 200$ will be made. The selected state sizes was $k = 10, 50, 100$. The computational time was due in large part to the number of points in both intervals for $\gamma_{u,z}$ and $\gamma_{i,r,z}$. Thus, the number of differently picked data sets with the leave-one-out algorithm, such that some statistical significance may be achieved, would depend on how long a simulation takes for one set of training, validation and test data. The purpose with this experiment was to find which ranges for k that are appropriate to use when the main goal is to mitigate overfitting.

4 Results

4.1 Experiment With pLSA

The training procedure of the pLSA model required around 40-50 EM iterations before the log-likelihood tolerance level was reached, and one single EM step took on average about 90 seconds. Note that the computational time for training and number of iterations grows proportional with larger k.

As seen in Figure 4.1, the prediction errors for the training set decreases when the state size increases. But the errors for the validation and test sets begin to increase a bit after $k \geq 20$ and become quite constant for $k \geq 40$. Their prediction errors are almost the same with the validation set errors a little bit smaller than the test errors, which could be due to their similarity in data size. The large gap between the training and test set errors indicates that the model suffers from severe overfitting. The model's prediction capacity is best using 10 latent states according to the results for the test set.

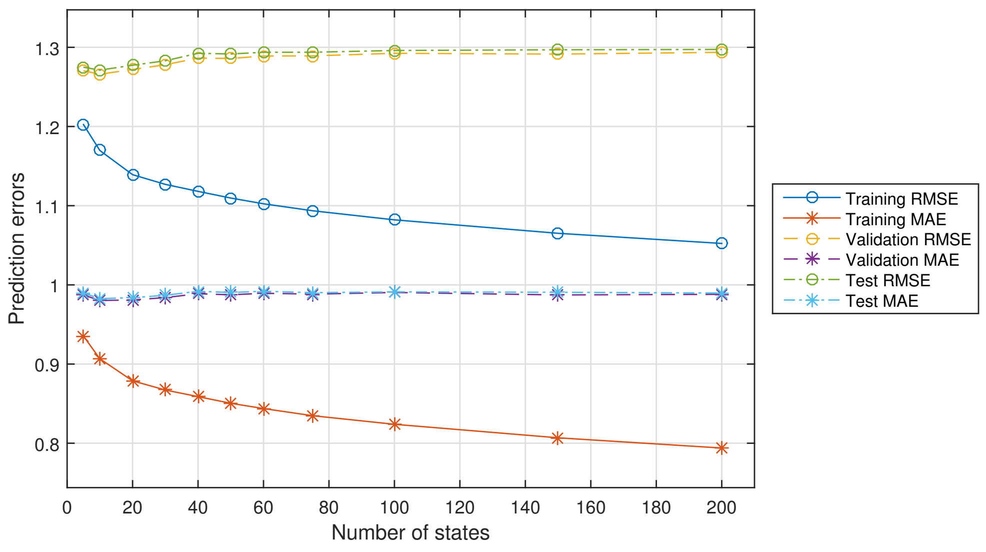

Figure 4.1: Predictive performance averaged over 10 runs in terms of RMSE and MAE for training, validation and test sets for the pLSA model as a function of the number of latent states k.

Figure 4.2 shows the EM iteration procedure for the data split that resulted in the largest RMSE difference from training to validation and test sets when using $k = 100$ states. After about 15 iterations, the validation and test error begins to slowly increase, while the training error strictly decreases. This is a typical phenomenon for model training procedures affected by overfitting.

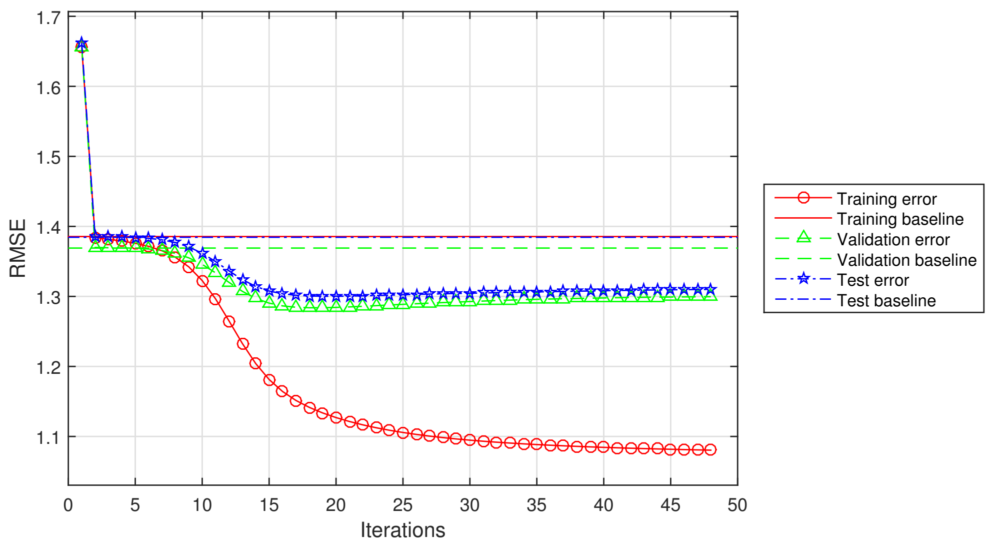

Figure 4.2: RMSE for the EM iteration procedure for the pLSA model with state size $k = 100$. The popularity baselines for the data sets are also included.

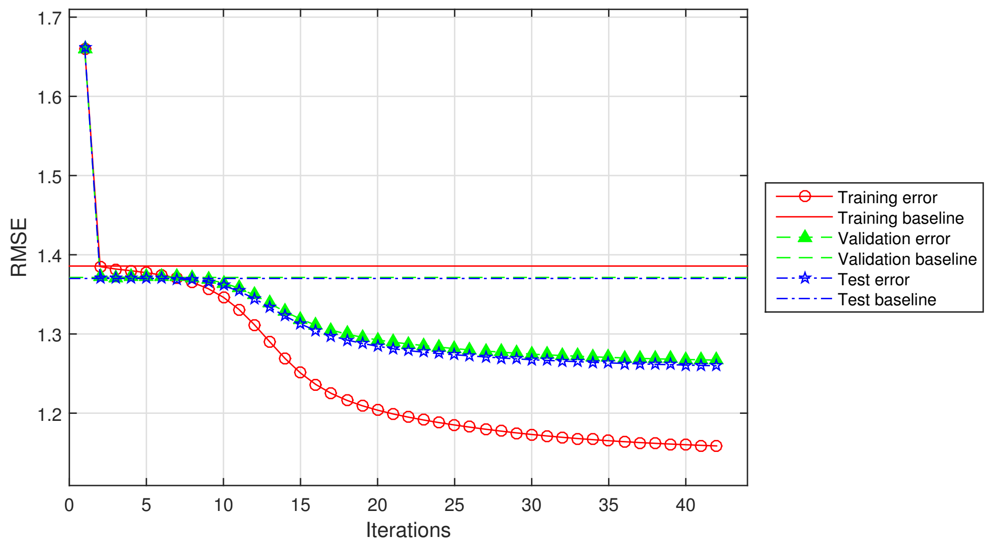

Figure 4.3: RMSE for the best performing EM iteration procedure for the multinomial pLSA model with state size $k = 10$. The popularity baselines for the data sets are also included.

The best prediction performances were achieved when the model used $k = 10$ states and such procedure with the data split that gave the lowest RMSE is shown in Figure 4.3. The validation and test errors actually decreases for every iteration and the gap between the errors is not as wide as for the model with 100 states. Also, the log-likelihood as a function of iterations from this pLSA model is included in Figure 4.4, where it is shown that the negative log-likelihood decreases monotonically.

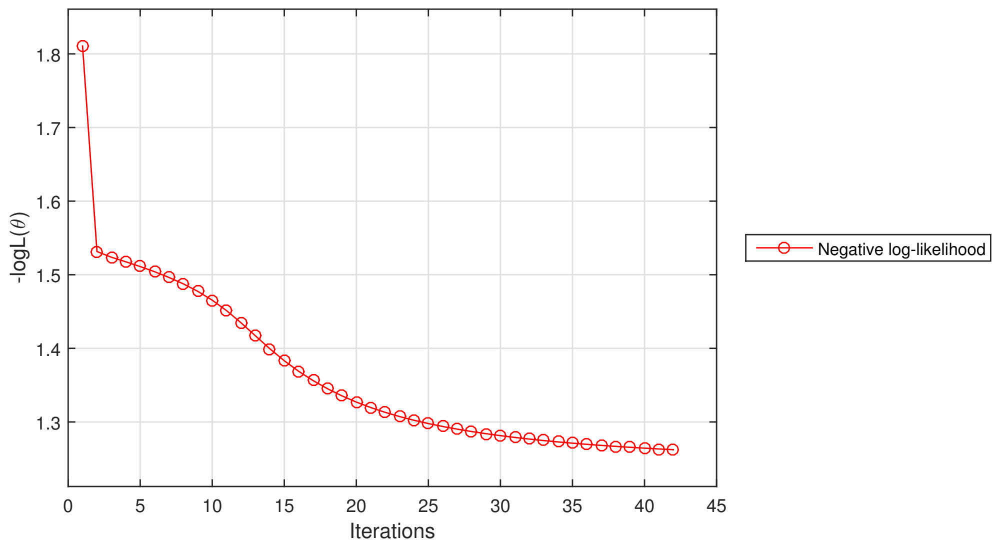

Figure 4.4: The negative log-likelihood as a function of iterations for the best performing EM iteration procedure with $k = 10$ states.

4.2 Experiment With ES pLSA

When applying the ES condition described in Section 2.5.1 to the pLSA model, the number of iterations for the model training has decreased to around 20 for state sizes $k \geq 30$, while it takes 4-10 iterations for smaller state sizes. The results from this experiment are shown in Figure 4.5. All three mean errors have a decreasing trend when the state size increases, so the best averaged prediction performance occurs when the model uses $k = 200$ states. For most state sizes k the validation set errors are much lower than the test set errors. This is due to the validation set being exploited to the training in the extra EM step after stopping, which leads to an improvement in the prediction errors for the validation set. The test set is still held unknown, thus leading to worse prediction performances compared to the validation set.

Figure 4.6 shows the EM training for the best performing data split when $k = 150$. The validation and test set RMSE follow each other until to the last iteration, but after the extra EM step the validation error is pushed down towards the training error. This shows the controversy of including data in the training procedure which is also used for evaluating the model.

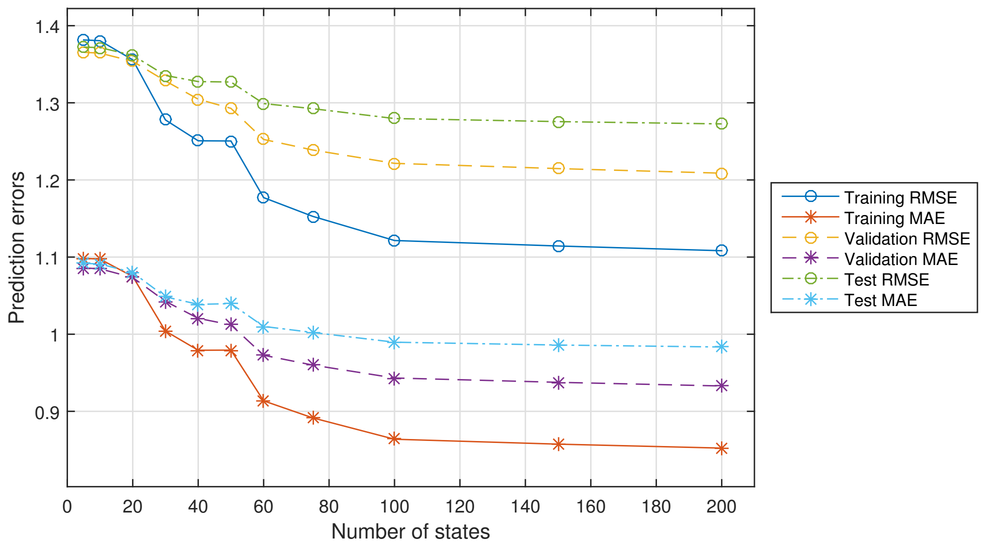

Figure 4.5: Average predictive performance over 10 runs in terms of RMSE and MAE for training, validation and test sets for the pLSA model with early stopping as a function of the number of latent states k.

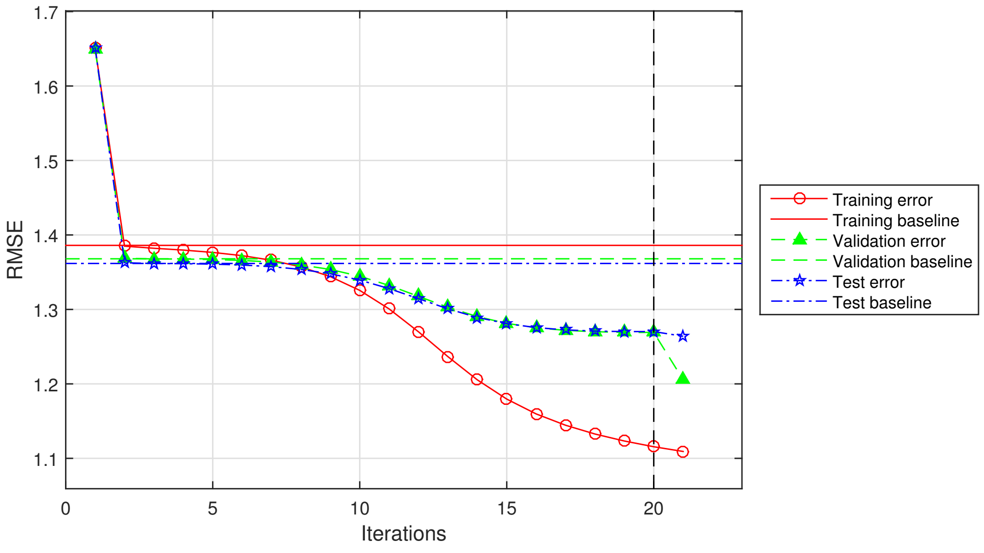

Figure 4.6: RMSE for the best performing EM iteration procedure using state size $k = 150$ for the pLSA model with early stopping. The popularity baselines for the data sets are also included. The black dashed vertical line at iteration 20 is where the early stopping condition is reached. The last iteration trains the model using the training plus validation set.

4.3 Experiment With Conjugate-Prior-Regularized pLSA

This experiment began with the grid search procedure in order to find favorable values on the hyperparameters for $k = 200$ that reduces the prediction errors. After this, the experiments focused on finding the best suitable latent state size k for suitable hyperparameter values from the grid search procedure.

4.3.1 Grid Search Procedure Experiment

After some test simulations, the hyperparameter intervals which would be inspected in the grid search procedure were selected to be

$$1.0 \leq \gamma_{u,z} \leq 1.2, \tag{4.1a}$$

$$1.0 \leq \gamma_{i,r,z} \leq 6.0. \tag{4.1b}$$

For $\gamma_{u,z} = 1.2$ and $\gamma_{i,r,z} = 6.0$ it was found that the stopping condition in Equation (3.1) was reached after very few iterations. The pre-experiments indicated that the value of $\gamma_{u,z}$ resulted in lower RMSE than for only adjusting $\gamma_{i,r,z}$. Why the parameters converges faster for some hyperparameters will be investigated further later on by using a stopping condition based on a certain amount of EM iterations instead of Equation (3.1).

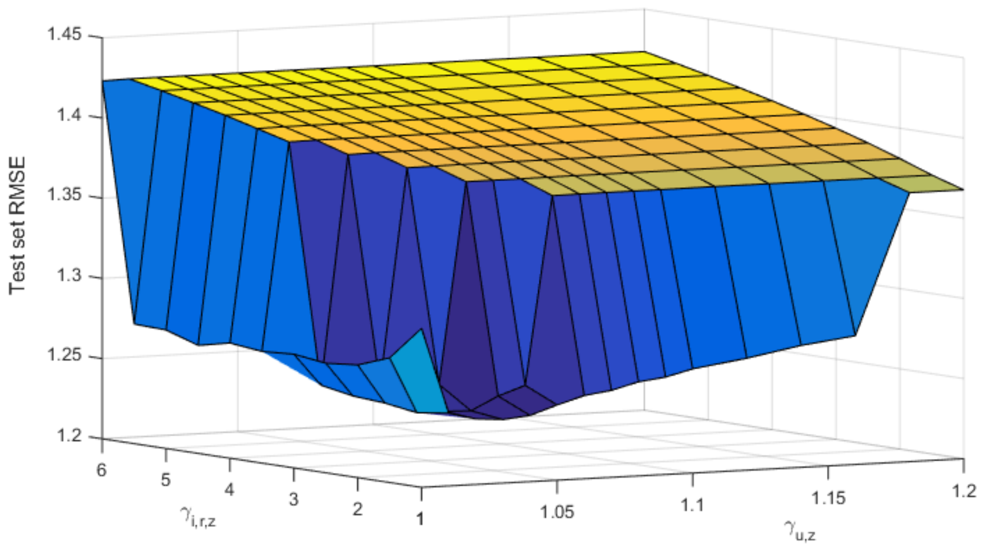

Figure 4.7: Grid over RMSE for test set with respect to different combinations of hyperparameter values for $\gamma_{u,z}$ and $\gamma_{i,r,z}$.

In the grid search procedure $\gamma_{u,z}$ used eleven equidistant points in $[1.0; 1.10]$ followed by five equidistant points in $[1.12; 1.20]$. The varied distance between the intervals was chosen because the RMSE showed an increasing trend when $\gamma_{u,z} > 1.04$. The other hyperparameter, $\gamma_{i,r,z}$, used eleven equidistant points in $[1.0; 6.0]$. Hence, the grid search was run for a total of 176 combinations of the hyperparameters which corresponds to the same amount of EM algorithm simulations. Due to the costly computational time, the procedure was only simulated once for one set of training, validation and test data. The resulting grid is shown in Figure 4.7 and the blue area in the grid is where the RMSE is the lowest. In this area the parameter estimates converges after about 50 EM iterations. The largest RMSE values are found in the yellow area, where the parameter estimates converges after 4-5 iterations.

For the first values in both hyperparameter intervals the RMSE grid begins to decrease, but the lowest RMSE value is achieved when only $\gamma_{u,z}$ is varied. The other hyperparameter, $\gamma_{i,r,z}$, has a rather stable RMSE value on the interval, while the RMSE for $\gamma_{u,z}$ decreases fast but then starts to slowly increase. If both priors are combined with the first values on their individual intervals, the grid takes the shape of a small valley with low RMSE. However, the RMSE suddenly increases for growing hyperparameter values and retains an increasing trend further on the grid. Since $\gamma_{u,z}$ seems to have the best impact in terms of decreasing the RMSE, the grid was plotted in two dimensions with respect to this hyperparameter only. The resulting plot is shown in Figure 4.8 and demonstrates the RMSE's behavior when combining $\gamma_{u,z}$ with a fixed $\gamma_{i,r,z}$. When both conjugate priors are applied, the hyperparameters need to be small in order to not reach the stopping condition after few EM iterations. Also, the lowest possible RMSE value increases a bit for increasing $\gamma_{i,r,z}$. The lowest RMSE value was obtained for the hyperparameters $\{\gamma_{u,z}, \gamma_{i,r,z}\} = \{1.03, 1.0\}$ and was found to be 1.24. Hence, the conjugate prior for the probability over the user clustering seems to be the most beneficial one.

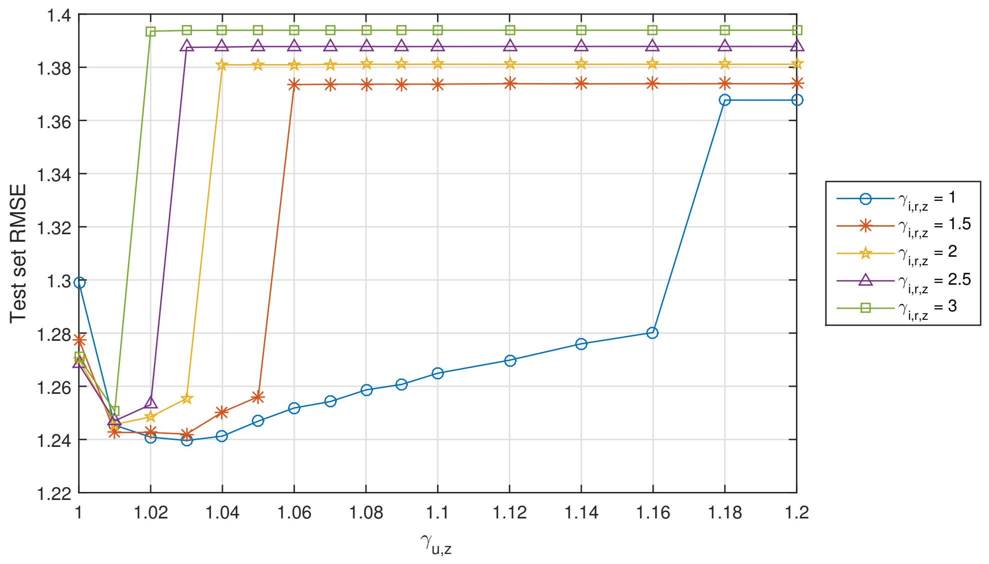

Figure 4.8: RMSE on test set from the grid in Figure 4.7 with respect to $\gamma_{u,z}$. The RMSE values is plotted for the first five values of the interval for $\gamma_{i,r,z}$.

The yellow area in Figure 4.7 is where the negative log-likelihoods stopping condition in Equation (3.1) is reached after few iterations. Since the increase of RMSE is very rapid it may be interesting to see what happens to the RMSE if the EM algorithm is run for some hyperparameter combinations in the yellow area with a larger amount of iterations. This led to the next experiment where the maximum number of iterations was chosen to 60, because the blue area of the grid had been trained for about 50 EM steps. The chosen grid points were selected to be five equidistant points in the intervals $\gamma_{u,z} = [1.04; 1.2]$ and $\gamma_{u,z} = [1.0; 5.0]$. The state size was still $k = 200$ and the results is shown in Figure 4.9. The grid still retains its increasing shape in terms of RMSE for increasing hyperparameter values, but will increase more smoothly when the training is terminated based on a reasonable maximum amount of iterations. The hyperparameter $\gamma_{u,z}$ is still the most beneficial prior in keeping the RMSE low.

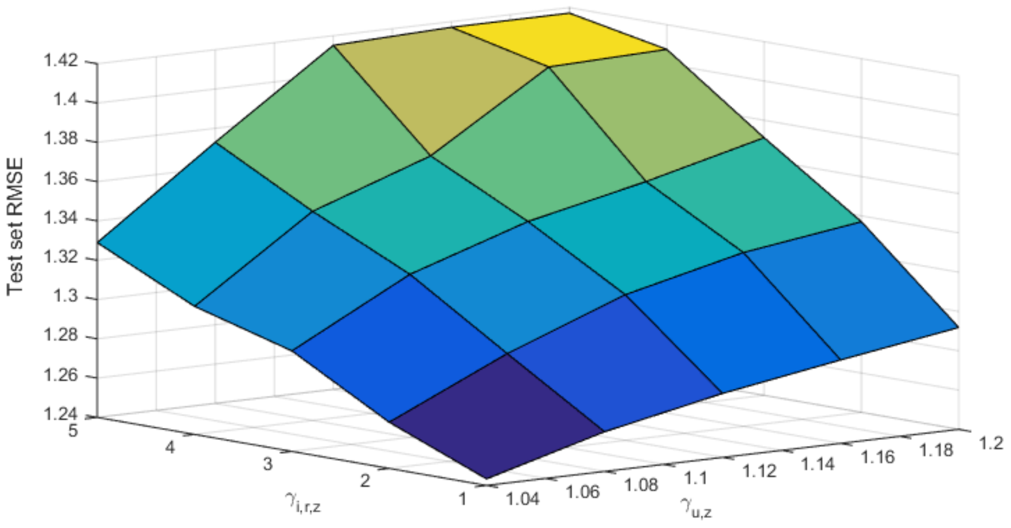

Figure 4.9: Grid over RMSE for test set with respect to various grid points from Figure 4.7. The EM algorithm stopped after 60 iterations for every hyperparameter combination.

The negative log-likelihood was also plotted for the hyperparameter combinations $\gamma_{u,z} = \{1.04, 1.12, 1.20\}$ and $\gamma_{i,r,z} = \{1, 3, 5\}$ and Figure 4.10 shows this graph. By increasing any of the hyperparameters the negative log-likelihood gets a slower change, which corresponds to less variations of the estimated parameter values. When $\gamma_{i,r,z}$ increases, the negative log-likelihood gets larger at the second iteration for each EM procedure. Furthermore, it seems that both hyperparameters have an impact on how many iterations that are needed for the log-likelihood to start decreasing, since the training needs more iterations for the log-likelihood to decrease when increasing the hyperparameter values. By inspecting the parameter values for the first EM iterations it appears that the parameters are distributed equally, since the log-likelihood is fairly constant. This means that a user is equally probable to be in any state or cluster, or that an item within any state could be provided with any rating on the rating scale. The conjugate priors thus generates an uncertainty or "fuzziness" to the parameter estimates when adding more artificial data points, i.e., that the estimates need more EM iterations to learn which their optimal values are. Unfortunately, this can be seen as a drawback of using conjugate-prior-regularization with too much regularization, since the optimization procedure is not aware of the slow training or uncertainty that the priors yield to the parameter estimates in the beginning.

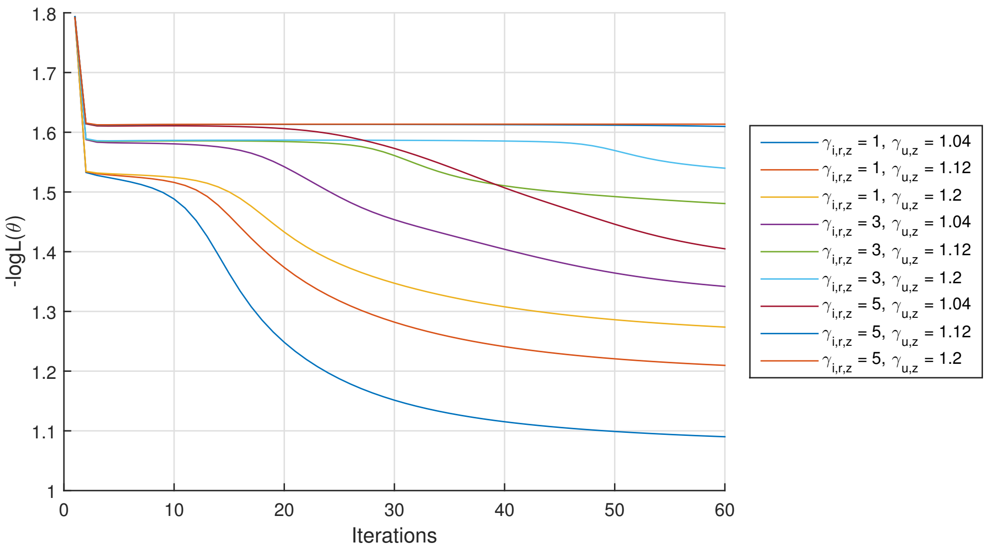

Figure 4.10: Negative log-likelihood with respect to number of iterations. The log-likelihoods comes from pLSA models with different combinations of prior hyperparameters, such as $\gamma_{i,r,z} = \{1, 3, 5\}$ and $\gamma_{u,z} = \{1.04, 1.12, 1.2\}$. The EM algorithm stopped after 60 iterations for every combination of the hyperparameters.

4.3.2 Varying the Latent State Size k Experiment

The following experiment was made to show how the conjugate-prior-regularized pLSA performs for various state sizes k. The selected state sizes were k = 10, 50, 100 and 200. After some test simulations it was found that the intervals for the hyperparameters would be set to

$$1.0 \leq \gamma_{u,z} \leq 1.8, \tag{4.2a}$$
$$1.0 \leq \gamma_{i,r,z} \leq 1.5. \tag{4.2b}$$

Based on the grid in Figure 4.7, varying $\gamma_{u,z}$ appeared to benefit the decrease of prediction errors more than $\gamma_{i,r,z}$. Therefore, it was decided to use many more points on the interval of $\gamma_{u,z}$ than $\gamma_{i,r,z}$. Since the smallest RMSE values was found for $\gamma_{u,z} \leq 1.1$

when $k = 200$, the interval $1 \leq \gamma_{u,z} \leq 1.2$ had the most points. In total 18 points was used for the interval of $\gamma_{u,z}$ with gradually increasing distances between the points. The values for the interval of $\gamma_{i,r,z}$ was chosen to consist of only 1 and 1.5 in order to avoid that the log-likelihood stopping criteria would be reached after a small amount of iterations. Since this experiment was very time consuming, only three differently picked training, validation and test sets were used to make an average over RMSE. The results are shown in Figure 4.11.

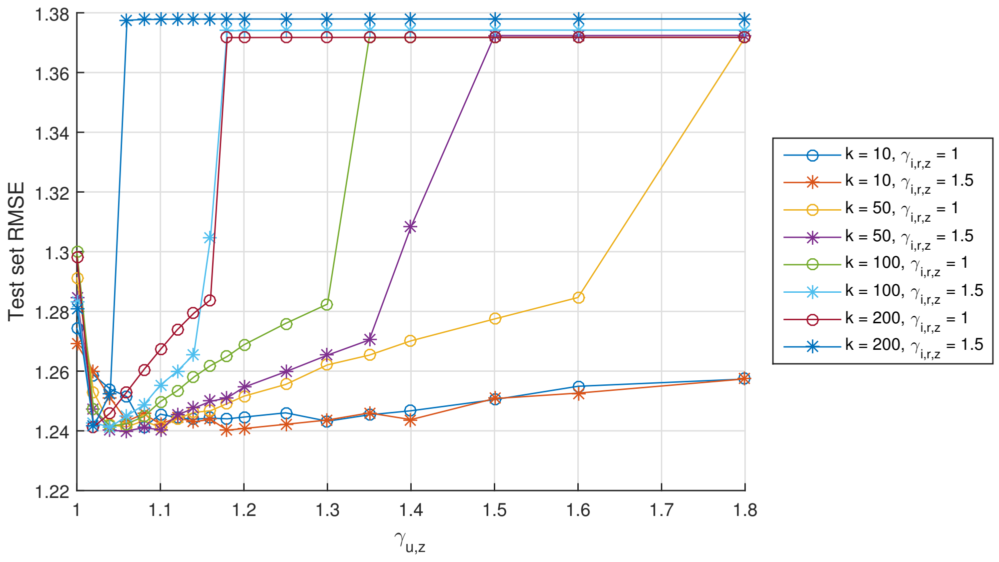

Figure 4.11: Average RMSE on three differently picked test sets with respect to $\gamma_{u,z}$, different state sizes k and $\gamma_{i,r,z} = 1$ and 1.5.

All state sizes k have the ability to decrease the RMSE using some hyperparameter combination in the range $1.0 < \gamma_{u,z} \leq 1.1$. By decreasing k, the RMSE will increase more slowly when $\gamma_{u,z}$ gets larger. Also, $\gamma_{i,r,z}$ plays a role for this case, where it is more beneficial for every state size to have $\gamma_{i,r,z} = 1.0$. It appears that for larger k the hyperparameter $\gamma_{i,r,z}$ is not really worth using, because it increases the sensitivity to select $\gamma_{u,z}$ in a favorable way. The RMSE for every model grows for increasing $\gamma_{u,z}$, but the model with $k = 10$ seems to be the most robust since it has a slow increase of RMSE and it is approximately the same independent of $\gamma_{i,r,z}$.

The benefits from small state sizes, e.g., reduced model complexity and shorter computational time, suggests that the conjugate-prior-regularized pLSA should use 10 or 50 latent states for the EachMovie data set. The next experiment will therefore be to investigate these state sizes further. The interval of $\gamma_{i,r,z}$ will also be extended with one more point at $\gamma_{i,r,z} = 2$, in order to investigate if this prior could have different impact on the RMSE for these state sizes than for larger ones. To cut down on the simulation time, the points for $\gamma_{u,z}$ was shortened to 16 points. Only three training, validation and test sets were picked with different seeds and the results are shown in Figure 4.12. The models with $k = 10$ still shows the robustness in terms of RMSE independent of $\gamma_{i,r,z}$ when both priors are included. For $k = 50$, the lowest RMSE can be found for $\gamma_{u,z} \leq 1.1$, but the RMSE increases fast right after the lowest value has been found. The RMSE when $k = 10$ increases slowly when $\gamma_{u,z} \leq 1.2$.

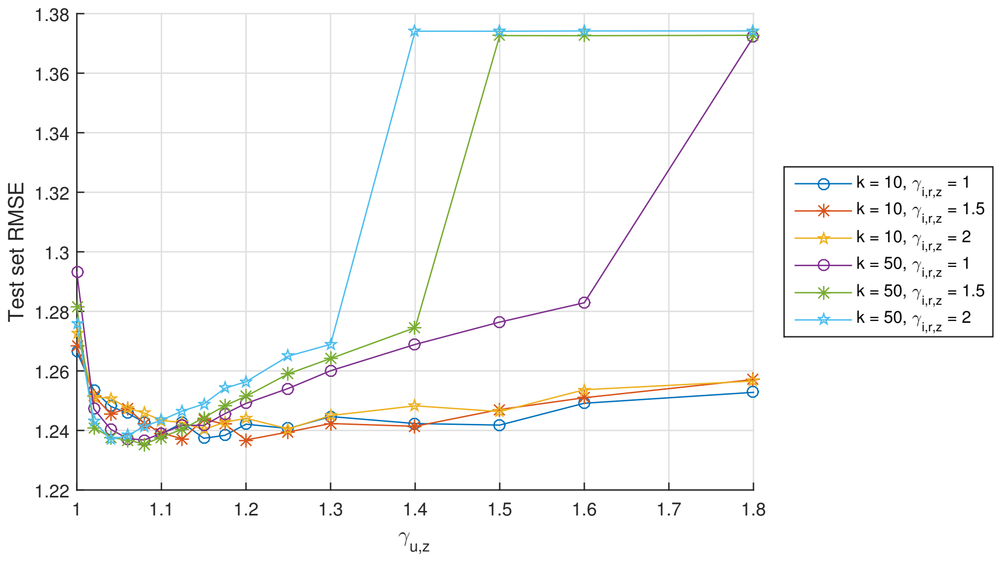

Figure 4.12: Average RMSE on three differently picked test sets with respect to $\gamma_{u,z}$ for state sizes $k = 10, 50$ and $\gamma_{i,r,z} = 1, 1.5, 2$.

Since the hyperparameter $\gamma_{i,r,z}$ appears to have less impact on the RMSE for lower values on $\gamma_{u,z}$, it was decided to plot the RMSE with respect to an interval of $\gamma_{i,r,z}$ and fixed values for $\gamma_{u,z} = \{1.0, 1.1, 1.2\}$. To get a picture of when the RMSE increases for these hyperparameter combinations, the interval of $\gamma_{i,r,z}$ was selected to be 10 equidistant points in $[1; 19]$ after some test simulations. The resulting plot is shown in Figure 4.13 and with some respects it is similar to the earlier figures. The lowest RMSE values are achieved when $1 \leq \gamma_{i,r,z} \leq 3$ and also applying $1.1 \leq \gamma_{u,z} \leq 1.2$ for $k = 10$ or $\gamma_{u,z} = 1.1$ for $k = 50$. The figure clearly shows that it is not possible to achieve low RMSE for both $k = 10$ and 50 by only varying $\gamma_{i,r,z}$.

4.4 Comparing Prediction Errors of pLSA Models

To establish a concluding result that shows that the conjugate-prior-regularized pLSA can mitigate the overfitting and reduce the prediction error, two models were selected with $k = 10$ and $\{\gamma_{u,z}, \gamma_{i,r,z}\} = \{1.2, 1.5\}$ respectively $k = 50$ and $\{\gamma_{u,z}, \gamma_{i,r,z}\} = \{1.08, 1.5\}$. These models were trained with the same 10 data sets as for the first and second pLSA experiments. Table 4.1 shows the best performing models from the made experiments and the pop baseline in terms of RMSE and MAE based on the averaged prediction errors on the 10 test sets.

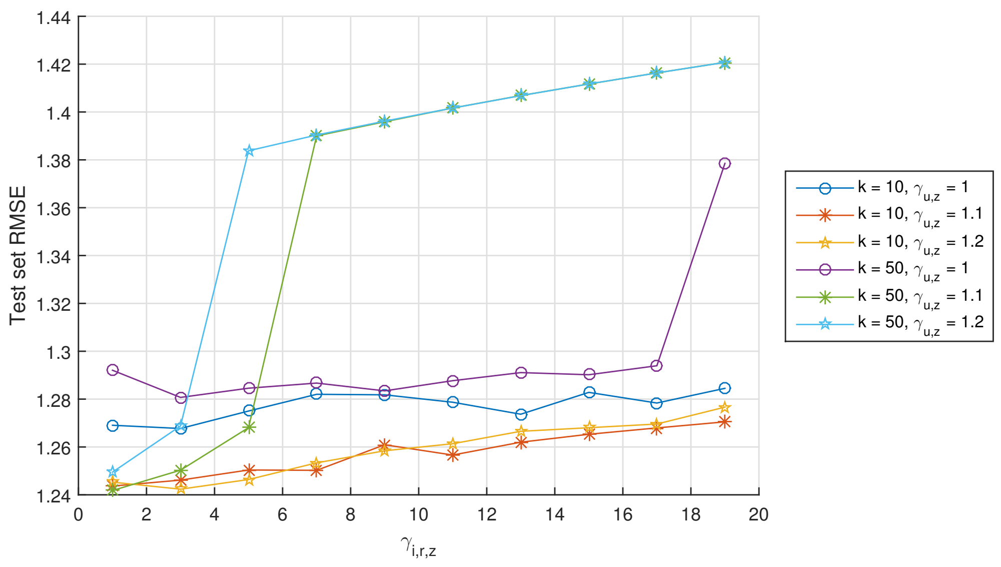

Figure 4.13: Average RMSE on three differently picked test sets with respect to $\gamma_{i,r,z}$ for state sizes $k = 10, 50$ and $\gamma_{u,z} = 1, 1.1, 1.2$.

	Error		Relative improvement	
Method	RMSE	MAE	RMSE	MAE
Popularity baseline	1.371	1.091	±0	±0
pLSA, $k = 10$	1.271	0.983	7.3%	9.8%
ES pLSA, $k = 200$	1.273	0.983	7.2%	9.8%
CPR pLSA, $k = 10$, $\{\gamma_{u,z}, \gamma_{i,r,z}\} = \{1.2, 1.5\}$	1.240	0.971	9.6%	11.0%
CPR pLSA, $k = 50$, $\{\gamma_{u,z}, \gamma_{i,r,z}\} = \{1.08, 1.5\}$	1.238	0.971	9.7%	11.0%

Table 4.1: Prediction errors for the "Pop" baseline (item average) and various methods for the pLSA averaged over the 10 different data sets.

The best performing EM procedure with the conjugate-prior-regularized pLSA model using $k = 10$ states is shown in Figure 4.14. The error for validation and test set has been decreased, but the training error has increased. Thus, the gap between the training error and validation and test error is not as wide as in the earlier figures showing the EM procedures. The same happens with the model using $k = 50$, where the best performing EM procedure is shown in Figure 4.15. The training error gets smaller when increasing k, which was seen in the first experiments. It is clear that the conjugate-prior-regularization does mitigate the overfitting problem to pLSA, although it only improves the prediction errors with about 2% from the standard and ES pLSA.

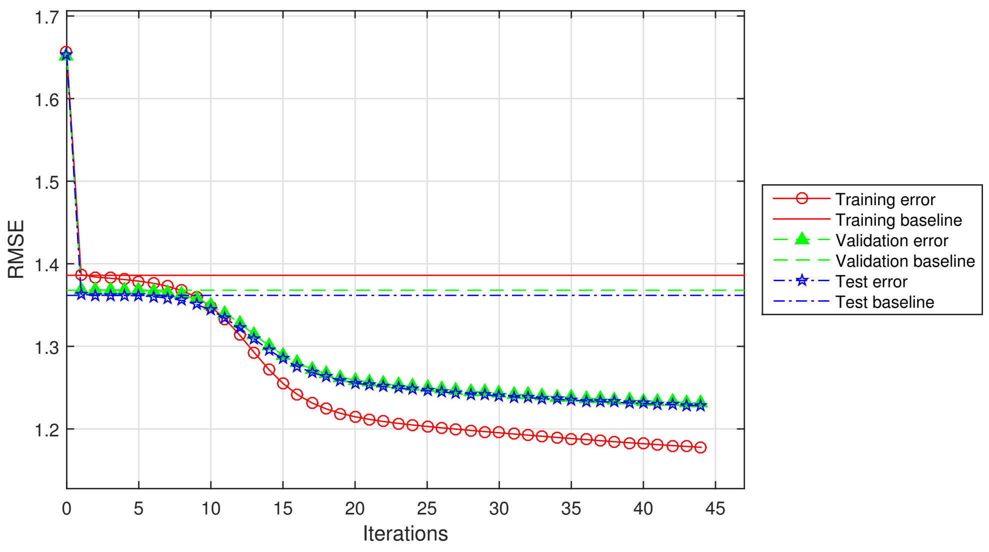

Figure 4.14: RMSE for the best performing EM iteration procedure for the conjugate-prior-penalized pLSA model with state size $k = 10$ and $\{\gamma_{u,z}, \gamma_{i,r,z}\} = \{1.2, 1.5\}$. The popularity baselines for the data sets are also included.

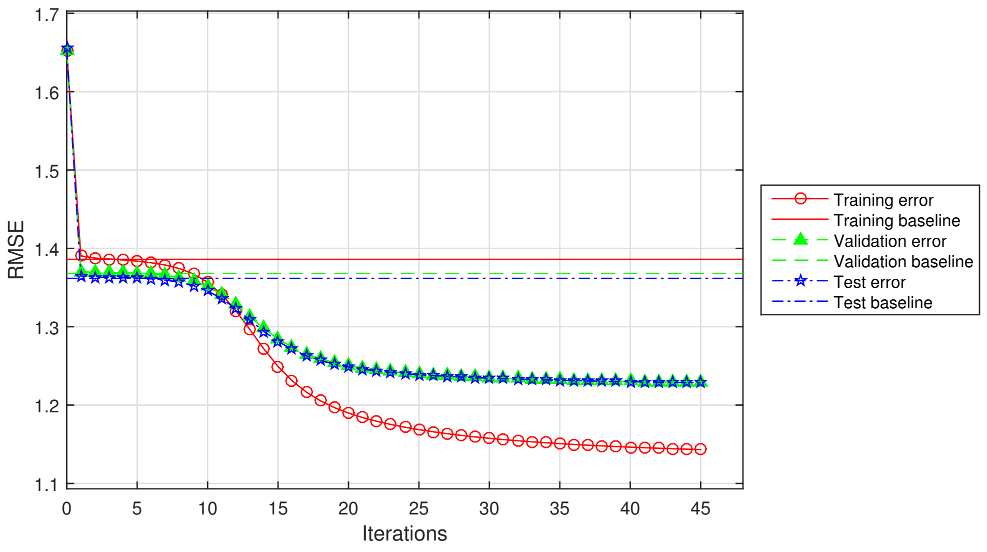

Figure 4.15: RMSE for the best performing EM iteration procedure for the conjugate-prior-penalized pLSA model with state size $k = 50$ and $\{\gamma_{u,z}, \gamma_{i,r,z}\} = \{1.08, 1.5\}$. The popularity baselines for the data sets are also included.

5 Discussion

5.1 Data sets

In the leave-one-out procedure, all users with less than $M = 2$ ratings are removed from the complete data set twice when splitting the data set into training, validation and test sets. The test set is picked before the validation set, so there might be some users with less than 2 ratings in the temporary data which is partitioned into training and validation set. This means that the test set may contain ratings from users that are not included in the training and validation set, such as users with only 2 ratings in the complete data set. In this case, the prediction errors for the test set may be larger than for the validation set, since the model is never trained with observations from these particular users. This could have been solved by removing all the users with $M = 3$ ratings in the first leave-one-out procedure and then use $M = 2$ for the second procedure. But since $M = 2$ happened to be used on both leave-one-out procedures for the experiments with pLSA and ES pLSA, the same setting was used for every experiment in order to have equivalent initialization between them.

It is difficult to guess a consumer's preference based on few rated items. It may thus be better to remove these users from the training and validation procedures to minimize the prediction errors. But on the other hand, maybe users with few observations are the ones that are the most important consumers to provide with recommendations. In long-term, it could be advantageously that the recommender system tries to give recommendations to users with few rated items in order to encourage the users to provide the system with more ratings. It would hopefully make the users get more involved in rating items, so that the recommender system acquires more knowledge about the user preferences and can give better recommendations. However, this really depends on the user activity and also if the system's movie rating phase is forced or optional.

In the Eachmovie data set, there are some items that only have been given a rating once, and these items are removed before the leave-one-out procedure. Because no statistical significance, such as variance, is provided for these particular items, it is unnecessary to make predictions for items that only one user has rated. If item i' has only been rated once with r', the probabilities $P(r \neq r'|i', z)$ will be equal to zero for every possible state z. Thus, the only predicted rating the model could give to item i' would be the deterministic value r', which simply is a non-reasonable prediction by the recommendation engine. In other words, the model would overfit this item to rating r'.

5.2 Implementation Issues

The multinomial pLSA was implemented in Matlab. All ratings from the data sets have been stored in sparse matrices due to the computational advantages, e.g., that only the non-zero elements are stored and operations on zero elements are eliminated [19]. The parameter values of $P(z|u)$ and $P(r|i,z)$ were stored in a matrix and a cell array respectively. Since $P(r|i,z)$ depends of three variables, all the items are linked to a cell that contains a matrix with probabilities for every rating r on the rating scale in the rows and all possible latent states in the columns. The variational probability distributions Q were separated into two cell arrays in the E-step, because the parameter estimates are summed over Q-distributions for fixed items or fixed users and ratings in the M-step.

Most of the simulations or experiments have been very time consuming, so there has been lots of thoughts about optimizing the computational speed of the algorithms. Allocating memory takes a lot of time due to the vast amount of data, so it is good to do the allocation before the algorithm starts running. Since the prediction accuracy only can be evaluated from already provided ratings, all the indices for viewed items and their ratings were stored in cell arrays for every single user. This speeded up the prediction metrics, because the rating matrix was filled with predicted ratings for items that had already been observed by the users.

If the pLSA program uses `for` loops where the involved variables are independent of the loop steps, then Matlab's `parfor` could be used to parallelize the loops to reduce the computational time. The `parfor` function starts up a number of Matlab worker stations by the programmer's choice and the independent worker runs the separate loop steps. But it is important to be aware of how much memory that is used for all the started Matlab workers, since the risk for "out of memory" errors increases because of the multiple worker activities. This problem, and also the computational time issue, could probably have been dealt with using the computer clusters at Lunarc[1]. However, there were a lot of troubles starting programs and running code in Matlab on the cluster, which the staff at Lunarc could not solve at the time.

To be humble, there is most certainly lots of modifications that can be made in the code to optimize the computational speed of the algorithms. It would also be very interesting to investigate how effectively the pLSA works if it is implemented in an object oriented programming language, such as C++ or Python.

5.3 pLSA With Tolerance Level Condition Only

It was expected that the results in Figure 4.1 would display the overfitting problem. However, it was not foreseen but rather obvious that the model would perform almost equally bad for state sizes $k \geq 40$ despite the strictly decreasing errors for the training set. A typical situation when overfitting occurs is when the number of model parameters is larger than the training data set. Assuming that $k = 40$ and that no users or items are removed from the data set, the number of parameters is given by

[1]Center for scientific and technical computing at Lund University, `http://www.lunarc.lu.se/` .

$$k|I||\mathcal{R}| + k|U| = 40 \cdot 1623 \cdot 6 + 40 \cdot 61,265 = 2,840,120 \,\text{parameters},$$

which is larger than the amount of observations in the EachMovie data set. The model thus reaches an upper limit in prediction accuracy for models with 40 or more states. The prediction errors could be decreased by adding more data to the model training and validation procedures if that is available. Still, the main conclusion drawn from these results is that some regularization method should be applied to the model in order to mitigate the severe overfitting.

5.4 pLSA With Early Stopping Condition

Applying the early stopping condition results in a decreasing trend for the prediction errors when the state size increases, which can be seen in Figure 4.5. Also, the validation and test set errors follow each other until $k \geq 40$, which is due to the fact that the number of parameters is larger than the available observations. The difference in prediction errors is very small for $k = 100$, 150 and 200 and it is neither better nor worse than the prediction performance from the pLSA model using $k = 10$ states from the first experiment. The decreasing prediction error by using larger state sizes may principally be due to that the EM algorithm is stopped after fewer iterations for small k. Thus, the models with small k may have gained better performance if there were some other condition, e.g., forcing the algorithm to run for a certain number of iterations before the early stopping condition is allowed to kick in. This is the main drawback with early stopping, namely how and when the algorithm should continue or terminate the training if the early stopping is reached.

Figure 4.6 shows that the RMSE for both the validation and test set follow each other until the stopping condition is reached. Thanks to the extra EM step after stopping, which includes training plus validation data, the RMSE for the validation data will be pushed down towards the training error. The RMSE for the test set will also decrease, but not as much as the RMSE for the validation set. If the standard pLSA and ES pLSA were evaluated by the validation data sets, the ES pLSA would obviously perform better due to the extra EM step. It is doubtful that any unknown data set would perform as good as when the validation set is exploited to this model, which makes this setting of early stopping quite problematic.

5.5 Conjugate-Prior-Regularized pLSA

This section discusses the experiments with the conjugate-prior-regularization technique, such as the impact from the conjugate priors and their hyperparameters, the grid search procedure, the prediction performances when varying the latent state size k and also some potential extensions to the regularized model.

5.5.1 Selection of Conjugate Prior Hyperparameters

In [11], [12], [15] about MAP estimation and conjugate priors the hyperparameter for a multinomial distributions, sometimes called mixing probability, is greater or equal to 1. Due to these earlier studies, the experiments on the hyperparameters have only been for $\gamma_{\{\cdot\}} \geq 1$ and hence can the analogy between priors and artificial data points be made. But perhaps adding "imagined" observations to the model is not the *only* way a prior can contribute to the modeling and learning procedure. The Dirichlet distribution is allowed to have hyperparameters less than 1 and this was also tested briefly. Then the training procedure went on well in the beginning, but after some iterations the prediction error values exploded. Since this setting did not seem to be profitable at the time, these experiments were not continued. It would though be interesting to investigate this further to make some reliable assumptions on using $\gamma_{\{\cdot\}} \leq 1$, because these hyperparameter values could maybe allow new approaches to the conjugate priors in helping to reduce overfitting.

The range in magnitudes of the hyperparameters differs quite much in order for them to establish a good fit of the estimated parameters. By increasing the hyperparameter value the corresponding parameter estimates become more equally distributed, e.g., $P(z|u)$ for user u becomes uniformly distributed over all possible states z, since the corresponding normalization constant gets larger. In the case for $P(z|u)$ to achieve the equal probabilities, the normalization constant depends on both the hyperparameter value and the state size k, such that the range for the hyperparameter value shrinks for growing k. Since $P(r|i, z)$ always consists of six probabilities in the case for the EachMovie data set, then a large hyperparameter value is needed to reach a uniformly distributed rating scale. This may imply why the hyperparameters have such different ranges in values compared to each other and also explains that some parameters need more regularization than others. Note that this explanation holds for the EachMovie data set, so if another data set would have been used, such as MovieLens, the hyperparameters could have different values compared to the ones in this thesis.

5.5.2 The Grid Search Procedure

The grid in Figure 4.7 displays that $\gamma_{u,z}$ has more beneficial impact in reducing the RMSE and mitigating overfitting than $\gamma_{i,r,z}$. Both conjugate priors reduce the RMSE, but $\gamma_{u,z}$ should always be preferred or a combination of both priors with a not too high value on $\gamma_{i,r,z}$ in this case. Using only one conjugate prior is advantageously when trying to find an optimal hyperparameter, since the grid search is scaled down to a 1-D grid over RMSE on $\gamma_{u,z}$.

Regarding the yellow area in the grid where the parameters converges after only a few iterations, there are still some questions that need to be answered. For the hyperparameter combinations which reaches the stopping condition quickly, the parameter estimates do not vary much in the beginning of the training procedure according to Figure 4.10. This is due to that the absolute differences of the decreasing negative log-likelihood are so small, which makes it look like the log-likelihood has converged. This shows the problem that comes from using large values on the hyperparameters, because the uncertainty in the model parameter values increases in the beginning of the training phase when lots of artificial data points are added through the hyperparameters. Since the initial $P(z|u)$ parameters are uniformly distributed, this may also contribute to the slow variations in the log-likelihood of the parameters. Perhaps the log-likelihood would be less static in the beginning of the EM training by using randomly selected initial parameters, e.g., that the initialization of $P(z|u)$ is treated in the same manner as for the initialization of $P(r|i,z)$. Also, the steep RMSE increase in Figure 4.7 is actually growing more smoothly if the EM algorithm is allowed to run for some larger number of iterations than 4-5 in the training procedure and therefore, maybe, another stopping condition should be considered. The RMSE seemed to always have been increased for hyperparameter combinations in the yellow area, so hopefully it does not occur a combination that decreases the RMSE further in the yellow area.

Another consideration between the log-likelihood and fast convergence is the fact that the log-likelihood is the product of every available model parameter. A latent variable model using $k = 200$, or any large k for that matter, will obviously result in a vast amount of multiplications of probability densities. This will result in a small log-likelihood value, since a probability density has a value between 0 and 1. The outcome of that may be that the negative log-likelihood is very close to the tolerance level ($= 10^{-3}$) in the beginning of the training. Moreover, since the variation of the model parameters is small in the beginning, especially when the priors are included, then the risk for reaching the convergence condition is more possible. This could be avoided to a certain limit by excluding or using a small value of $\gamma_{i,r,z}$.

5.5.3 Varying the Latent State Size k

It is clear in Figures 4.11 and 4.12 that $\gamma_{u,z}$ has a promising ability to decrease the prediction error for different state sizes k. Also, the conjugate prior for $P(r|i,z)$ has noticeable less impact on adding the uncertainty to the model parameters estimates for smaller state sizes, since the RMSE is approximately the same independent of the value of $\gamma_{i,r,z}$. Thus, it seems that it does not really matter how many artificial data points are added to the estimate of $P(r|i,z)$ when using few states, which could have something to do with that the amount of model parameters is larger than the available training data. The rating emission prior therefore becomes not as necessary as the user clustering prior when the goal is to decrease the prediction errors.

In Figure 4.12, the RMSE curves remain relatively low and flat over $\gamma_{u,z}$ when $k = 10$, which tells that the model is then more robust to the extrapolation from training data to test data. Also, from the grid in Figure 4.7 it was seen that by only varying $\gamma_{u,z}$ gave rise to the smallest RMSE. Thus, $\gamma_{u,z}$ has a greater influence on decreasing the prediction errors for larger k. However, using more states will add more sensitivity in the selection of an optimal $\gamma_{u,z}$. This makes sense because the complexity of θ, which is dependent of k, enforces overfitting. The smoothing effect of the prior, which is achieved with MAP estimation, gets more limited with a complex parameter vector θ [2, p. 55]. Therefore, the hyperparameters must be selected with greater care when k grows. Another flaw with using large k is that the computational time increases both by more parameters to estimate and also longer EM procedures with more iterations.

The experiment in Figure 4.13 gives a clearer picture of how $\gamma_{i,r,z}$, for $k = 10$ and $k = 50$, affects the RMSE. The error gets the same increasing trend as in the earlier experiments, while $\gamma_{i,r,z}$ can have a large value for $k = 10$ before the RMSE clearly increases. This adds trust in the statement that $\gamma_{i,r,z}$ gets less impact on the prediction errors for models with small k, especially when $\gamma_{u,z} > 1$ is applied as well. Also note that rather low RMSE values can be established by using $1 \leq \gamma_{i,r,z} \leq 3$ together with $\gamma_{u,z} > 1$ for both k in the experiment.

The RMSE curves in Figures 4.11 and 4.12 are very sharp and edgy, so it would be good to establish more statistical significance in the results by running more simulations with differently picked data sets. Also, in the further experiments it could have been beneficial to run the grid procedure for the latent state sizes that are to be investigated. Since the courses become more flat for smaller state sizes, the hyperparameter intervals, and therefore the grids, should look a lot different than the grid in Figure 4.7.

For a future project about pLSA, it would be interesting to analyze more about the associations between observations and the latent states and also investigate why some state sizes are more beneficial than others. There is probably more information that can be extracted from the states, e.g., how does the rating history influence the probability of a user belonging to a certain state. Maybe this could help when trying to find which state size k that is optimal for mitigating the overfitting.

5.5.4 Extensions of the Model

Further work on conjugate-prior-regularization could be to use or elaborate some methods in the selection and variations of the prior hyperparameters instead of just having constant hyperparameter values for both multinomial distributions. One approach with the prior on $P(z|u)$ could be to add more regularization to users with few preferences and correspondingly less to the ones who have rated many movies. This means that the users with few ratings would be less controlled in being associated with a certain state. Intuitively, the more ratings a user has provided the more the system knows about the user's movie preferences. This approach relies on if the users have provided possible ratings from the whole scale. For certain users with a typical behavior of only providing high or low ratings, it could be inappropriate to push them farther away from the state that describes their characteristics best by establishing softer clustering to the model.

Considering $P(r|i, z)$, one approach could be to regularize user's towards belonging more to a particular state. Assume that an item in a state is overfitted to the highest rating. One user who clearly belongs to the state becomes less probable in being connected to the state, because the user rated the overfitted item with a lower rating than the highest. Thus, it could be beneficial to add regularization to that item's rating scale in order for the user to regain the possibility probability to belong in that state. The added regularization thus makes the possible rating probabilities to become more distributed over the rating scale for that item in the state.

Another approach regarding the prior for $P(r|i, z)$ is to regularize depending on the items mean rating in a state. For now the regularization biases the rating scale towards the middle, i.e., $\bar{r} = 3.5$ for the case using the EachMovie data set. Thus, it could be beneficial to add some bias to $P(r|i,z)$ towards the items average rating in the certain state in order to make the predictions more accurate.

The rating probability $P(r|i,z)$ may be replaced with a Gaussian probability density function $P(r;\mu_{i,z},\sigma_{i,z})$[2], which showed good improvements in prediction accuracies in [1]. In that setting, the M-step updates the mean $\mu_{i,z}$ and standard deviation $\sigma_{i,z}$ in order to compute the current ratings scale probabilities with a Gaussian distribution. The Gaussian pLSA was also elaborated by having a user normalization on the rating scale, since all users utilizes the rating scale in different ways in reality, i.e., everyone does not interpret an overall movie rating with 3-stars similarly. Thus, it was proposed to broaden the scale in order to model the user preferences arbitrary equivalent by using a Gaussian normalization method per user, which involves the user-specific mean ratings and normalizing the variance of ratings for each user to one [1, p. 100].

Another approach would be to quantize all ratings into like or dislike, which would be made by putting a threshold on the rating data. By investigating every user's rating behavior and drawing a conclusion on what rating that corresponds to each user's preference, then all user preferences are bonded together in the modeling and the same thing happens for the disliked items. This can be viewed as the discrete version of the normalization scheme proposed for the Gaussian pLSA, which showed significant improvement over the non-normalized Gaussian model. Thus, the hope is that the multinomial pLSA model with a good choice of normalization strategy could improve the prediction performance further. The conjugate priors would of course be applied to that model as well. The issue with giving suggestions to users that only have rated movies they think are bad still holds, but as before it could possibly be fixed in some sense with a user-specific hyperparameter selection as regularization. The quantization would also reduce the number of model parameters in $P(r|i,z)$, which was the case for the Gaussian pLSA model.

[2] $P(r;\mu_{i,z},\sigma_{i,z}) = \frac{1}{\sqrt{2\pi\sigma_{i,z}^2}} \exp\left\{-\frac{(r-\mu_{i,z})^2}{2\sigma_{i,z}^2}\right\}$ [4, p. 78].

6 Conclusion

The thesis work began with reproducing the multinomial pLSA with the generalized EM algorithm for training and learning of the model parameters. The implemented model is however affected with severe overfitting, which is a common problem in this kind of statistical modeling setup. The first try to mitigate the overfitting was to apply the early stopping method from [1]. This technique worked best for models with a large amount of latent states ($k \geq 100$) and could decrease the computational time for the EM training. But in terms of prediction error, the ES pLSA was almost exactly as good as the pLSA that only used a log-likelihood tolerance level as stopping condition. Thus, it was proposed to apply a regularization method to the pLSA in order to improve the prediction performance on unseen data by reducing the overfitting problem.

The proposed regularization method, conjugate-prior-regularization, is done by extending the pLSA model with a conjugate prior distribution. By using conjugate priors in the regularization it is allowed to develop the EM algorithm with a MAP approach and therefore new expressions for parameter estimates can be derived. The conjugate prior for a multinomial distribution is the Dirichlet distribution, which yielded two hyperparameters corresponding to each conjugate prior of the pLSAs multinomial probabilities. To find the optimal hyperparameters, the RMSE grid search procedure was used for $k = 200$, which was the best k to use with the ES pLSA. This is a slow variant of cross validation, but it is intuitive since the differences in prediction errors between various hyperparameter combinations are very clear in the grid. The RMSE grid will probably look very different for various latent state sizes k since the hyperparameter intervals vary in length depending on k, but also because a log-likelihood tolerance condition was used for terminating the training procedure. The drawback with a log-likelihood tolerance controlling the EM algorithm is that the training is usually slow in the beginning of the training, which results in small differences in the evaluated log-likelihood. This sometimes results in that the training is terminated earlier than what is desired. Still it is common to use the evaluated log-likelihood for stopping conditions, but perhaps that could be evaluated on the differences in the parameter values instead.

Both hyperparameters have the ability to reduce the prediction errors, but the user clustering prior is the more beneficial one for this purpose. However, in most cases $\gamma_{u,z}$ has to be chosen with greater care than $\gamma_{i,r,z}$, since $\gamma_{u,z}$ is usually slightly larger than 1. The choice of hyperparameter values also differ for different latent state sizes k, but usually results in an increase of possible hyperparameter combinations that will decrease the prediction errors significantly. This do not translate to always use small state sizes to the regularized pLSA model, since the state sizes used in the experiments have been able to achieve the same low prediction errors, more or less.

The final results of the conjugate-prior-regularized pLSA showed promising results in mitigating the overfitting problem and decreasing the prediction error. Thus, it has been shown that modeling discrete ratings with a multinomial distribution may be favorable in describing such data with an applied conjugate prior distribution. There is most likely more research that could be done on these methods, especially in comparing the multinomial with the Gaussian rating emission pLSA. The pLSA model for collaborative filtering have proven to be a promising method and it would be interesting to apply the conjugate-prior-regularization technique in a setting different from recommender systems.

Bibliography

[1] T. Hofmann, "Latent Semantic Models for Collaborative Filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89-115, Jan. 2004.

[2] N. Barbieri, G. Manco, and E. Ritacco, "Probabilistic Approaches to Recommendations," *Synthesis Lectures on Data Mining and Knowledge Discovery*, Lecture no. 9, Morgan & Claypool Publishers, 2014.

[3] F. Ricci, L. Rokach, B. Shapira, P. Kantor, "Recommender Systems Handbook," Springer Science+Business Media (Springer), 2011.

[4] C. Bishop, "Pattern Recognition and Machine Learning," Springer Science+Business Media (Springer), 2006.

[5] T. Hofmann, "Probabilisitc Latent Semantic Analysis", pp. 289-296, 1999, `http://arxiv.org/ftp/arxiv/papers/1301/1301.6705.pdf` [2016-03-25].

[6] S. Borman, "The Expectation Maximization Algorithm - A short tutorial," June 28 2006, `https://www.cs.utah.edu/~piyush/teaching/EM_algorithm.pdf` [2016-03-24].

[7] R. Neal, G. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," *Learning in Graphical Models*, pp. 355-368, 1998.

[8] T. Hofmann, "Unsupervised Learning by Probabilistic Latent Semantic Analysis," *Machine Learning*, vol. 42, pp. 177-196, Kluwer Academic Publishers (Springer), 2001.

[9] J. Bilmes, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," International Computer Science Institute, Berkeley CA, April 1998.

[10] G. Heinrich, "Parameter estimation for text analysis," ver. 2.4, vsonix GmbH + University of Leipzig, Germany, Aug. 2008.

[11] J. Chu, Y. Lee, "Conjugate-Prior-Penalized Learning of Gaussian Mixture Models for Multifunction Myoelectric Hand Control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 17, no. 3, pp. 287-297, June 2009.

[12] J. Chien, M. Wu, "Adaptive Bayesian Latent Semantic Analysis," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 198-207, Jan. 2008.

[13] K. Verstrepen, B. Goethals, "Unifying Nearest Neighbors Collaborative Filtering," *University of Antwerp*, 2014.

[14] D. Fink, "A Compendium of Conjugate Priors," Environmental Statistics Group, Department of Biology, Montana State University, May 1997, `http://www.johndcook.com/CompendiumOfConjugatePriors.pdf` [2016-03-24].

[15] D. Ormoneit, V. Tresp, "Averaging, Maximum Penalized Likelihood and Bayesian Estimation for Improving Gaussian Mixture Probability Density Estimates," *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 639-650 July 1998.

[16] T. Hofmann, J. Puzicha, "Latent Class Models for Collaborative Filtering," *Proceedings of IJCAI'99*, 1999, `http://www.eecs.berkeley.edu/~russell/classes/cs294/f05/papers/hofmann+puzicha-1999.pdf` [2016-03-25].

[17] GroupLens, "EachMovie dataset", *GroupLens Research*, `http://grouplens.org/datasets/eachmovie/` [2016-02-01].

[18] GroupLens, "MovieLens 1M Dataset", *GroupLens Research*, `http://http://grouplens.org/datasets/movielens/1m/` [2016-02-19].

[19] MathWorks, "Computational Advantages of Sparse Matrices", *The MathWorks, Inc.*, `http://se.mathworks.com/help/matlab/math/computational-advantages-of-sparse-matrices.html` [2016-03-02].

A Proofs and Derivations

A.1 Proof of Jensen's Inequality

Jensen's inequality is used for the derivation of the EM algorithm to create an upper bound on the negative log-likelihood function, which is a convex function. The proof of the theorem comes from [6, p. 3-4].

Theorem 1 (Jensen's inequality)**.** *Let f be a convex function defined on an interval I. If $x_1, x_2, \ldots, x_n \in I$ and $\lambda_1, \lambda_2, \ldots, \lambda_n \geq 0$ with $\sum_{i=1}^n \lambda_i = 1$,*

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i).$$

Proof. For $n = 1$, this is trivial. The case $n = 2$ corresponds to the definition of convexity. To show that this is true for all natural numbers, we proceed by induction. Assume the theorem is true for some n then,

$$\begin{aligned}
f\left(\sum_{i=1}^{n+1} \lambda_i x_i\right) &= f\left(\lambda_{n+1} x_{n+1} + \sum_{i=1}^{n} \lambda_i x_i\right) \\
&= f\left(\lambda_{n+1} x_{n+1} + (1-\lambda_{n+1}) \frac{1}{1-\lambda_{n+1}} \sum_{i=1}^{n} \lambda_i x_i\right) \\
&\leq \lambda_{n+1} f(x_{n+1}) + (1-\lambda_{n+1}) f\left(\frac{1}{1-\lambda_{n+1}} \sum_{i=1}^{n} \lambda_i x_i\right) \\
&= \lambda_{n+1} f(x_{n+1}) + (1-\lambda_{n+1}) f\left(\sum_{i=1}^{n} \frac{\lambda_i}{1-\lambda_{n+1}} x_i\right) \\
&\leq \lambda_{n+1} f(x_{n+1}) + (1-\lambda_{n+1}) \sum_{i=1}^{n} \frac{\lambda_i}{1-\lambda_{n+1}} f(x_i) \\
&= \lambda_{n+1} f(x_{n+1}) + \sum_{i=1}^{n} \lambda_i f(x_i) \\
&= \sum_{i=1}^{n+1} \lambda_i f(x_i)
\end{aligned}$$

□

Note that if f is a concave function, such as $f = \log x$, and apply Jensen's inequality the following result is obtained

$$\log \sum_{i=1}^{n} \lambda_i x_i \geq \sum_{i=1}^{n} \lambda_i \log(x_i). \tag{A.1}$$

This allows to set a lower bound on the logarithm of a sum, which is used in the derivation of the EM algorithm.

A.2 Derivation of $Q^*(z; u, i, r; \theta)$

In Section 2.3.2 the optimal Q was stated in equation (2.14) to be

$$Q^*(z; u, i, r; \hat{\theta}) = \frac{P(r|i,z)P(z|u)}{\sum_{z'} P(r|i,z')P(z'|u)}.$$

Q^* is obtained by minimizing the upper bound on the negative log-likelihood Equation (2.11c) with respect to $Q(z; u, i, r; \hat{\theta})$, which is given by

$$Q^*(z; u, i, r; \hat{\theta}) = \underset{Q(z;u,i,r;\hat{\theta})}{\arg\min} \; - \sum_{\langle u,i,r \rangle} \sum_{z} Q(z; u, i, r; \theta) \log \frac{P(r|i,z)P(z|u)}{Q(z; u, i, r; \theta)}. \tag{A.2}$$

In order to ensure the normalization constraint $\sum_z Q(z; u, i, r; \theta) = 1$, the Lagrange multiplier $\lambda_{u,i,r}$ is introduced in order to form the Lagrangian [1, p. 113]

$$\begin{aligned} \mathcal{L}(Q, \theta, \lambda_{u,i,r}) = &- \sum_{\langle u,i,r \rangle} \sum_{z} Q(z; u, i, r; \theta) [\log P(r|i,z)P(z|u) - \log Q(z; u, i, r; \theta)] \\ &+ \sum_{\langle u,i,r \rangle} \lambda_{u,i,r} \left(\sum_{z} Q(z; u, i, r; \theta) - 1 \right), \end{aligned} \tag{A.3}$$

where Q denotes $Q(z; u, i, r; \theta)$. Taking the derivative of $\mathcal{L}(Q, \theta, \lambda_{u,i,r})$ with respect to $Q(z; u, i, r; \theta)$ for fixed u, i, r and z and setting the result to zero gives

$$\frac{\partial \mathcal{L}(Q, \theta, \lambda_{u,i,r})}{\partial Q(z; u, i, r; \theta)} = -\log P(r|i,z)P(z|u) + \log Q(z; u, i, r; \theta) + 1 + \lambda_{u,i,r} = 0 \tag{A.4}$$

Note that the upper equation is when only state z and the observation triplet $\langle u, i, r \rangle$ are considered in $Q(z; u, i, r; \theta)$ when $\mathcal{L}(Q, \theta, \lambda_{u,i,r})$ is derived. Equation (A.4) is equivalent to

$$\begin{aligned}\log Q(z;u,i,r;\theta) = \log P(r|i,z)P(z|u) - 1 - \lambda_{u,i,r} &\iff \\ e^{\log Q(z;u,i,r;\theta)} = e^{\log P(r|i,z)P(z|u) - 1 - \lambda_{u,i,r}} &\iff \\ e^{\log Q(z;u,i,r;\theta)} = e^{\log P(r|i,z)P(z|u)} e^{-(1+\lambda_{u,i,r})} &\iff \\ Q(z;u,i,r;\theta) = P(r|i,z)P(z|u) \cdot e^{-(1+\lambda_{u,i,r})}.\end{aligned}$$

Thus, $Q(z;u,i,r;\theta)$ is given by

$$Q(z;u,i,r;\theta) = \frac{P(r|i,z)P(z|u)}{e^{1+\lambda_{u,i,r}}}. \tag{A.5}$$

Finding an expression for $\lambda_{u,i,r}$ is done by taking the derivative of the Lagrangian with respect to $\lambda_{u,i,r}$ for fixed u, i and r and setting it equal to zero

$$\frac{\partial \mathcal{L}(Q, \theta, \lambda_{u,i,r})}{\partial \lambda_{u,i,r}} = 1 \cdot \left(\sum_z Q(z;u,i,r;\theta) - 1 \right) = 0. \tag{A.6}$$

Inserting Equation (A.5) in (A.6) is equivalent to

$$\begin{aligned}\sum_z \frac{P(r|i,z)P(z|u)}{e^{1+\lambda_{u,i,r}}} - 1 = 0 &\iff \\ \sum_z P(r|i,z)P(z|u) = e^{1+\lambda_{u,i,r}} &\iff \\ \log \sum_z P(r|i,z)P(z|u) = 1 + \lambda_{u,i,r}\ ,&\end{aligned}$$

which leads to the solution

$$\lambda_{u,i,r} = \log \sum_z P(r|i,z)P(z|u) - 1. \tag{A.7}$$

Inserting the upper expression for $\lambda_{u,i,r}$ in Equation (A.5) results in the optimal solution of Q that minimizes the upper bound of $-l(\theta)$

$$Q^*(z;u,i,r;\theta) = \frac{P(r|i,z)P(z|u)}{e^{1+\log \sum_{z'} P(r|i,z')P(z'|u) - 1}} = \frac{P(r|i,z)P(z|u)}{\sum_{z'} P(r|i,z')P(z'|u)}, \tag{A.8}$$

where the sum over z' denotes summing over every possible z.

A.3 Derivation of $P(z|u)$ and $P(r|i,z)$

The model parameters are obtained by minimizing the upper bound of negative log-likelihood in Equation (2.15) with respect to the parameters $P(z|u)$ and $P(r|i,z)$ separately. Start with deriving the expression for $P(z|u)$, the optimal $P(z|u)$ is denoted by $P^*(z|u)$ and is given by

$$P^*(z|u) = \underset{P(z|u)}{\arg\min} - \sum_{\langle u,i,r\rangle} \sum_z Q^*(z;u,i,r;\theta) \log P(r|i,z)P(z|u) \tag{A.9a}$$

$$= \underset{P(z|u)}{\arg\min} - \sum_{\langle u,i,r\rangle} \sum_z Q^*(z;u,i,r;\theta) \left[\log P(r|i,z) + \log P(z|u)\right] \tag{A.9b}$$

$$= \underset{P(z|u)}{\arg\min} - \sum_{\langle u,i,r\rangle} \sum_z Q^*(z;u,i,r;\theta) \log P(z|u). \tag{A.9c}$$

$P(r|i,z)$ is omitted, since the above equation is minimized with respect to $P(z|u)$ only. By introducing the Lagrange multiplier λ_u to enforce the normalization constraint $\sum_z P(z|u) = 1$ for all u, one forms the Lagrangian function

$$\begin{aligned} \mathcal{L}_1(Q^*, P(z|u), \lambda_u) = &- \sum_{\langle u,i,r\rangle} \sum_z Q^*(z;u,i,r;\theta) \log P(z|u) \\ &+ \sum_u \lambda_u \left(\sum_z P(z|u) - 1 \right). \end{aligned} \tag{A.10}$$

Taking the derivative of the Lagrangian with respect to $P(z|u)$ for fixed u and z and setting it equal to zero gives

$$\frac{\partial \mathcal{L}_1(Q^*, P(z|u), \lambda_u)}{\partial P(z|u)} = - \sum_i \sum_r Q^*(z;u,i,r;\theta) \frac{1}{P(z|u)} + \lambda_u = 0. \tag{A.11}$$

Solving $P(z|u)$ results in

$$P(z|u) = \frac{1}{\lambda_u} \sum_i \sum_r Q^*(z;u,i,r;\theta). \tag{A.12}$$

The Lagrange multiplier λ_u is obtained by taking the derivative of the Lagrangian with respect to λ_u for a fixed u and setting it equal to zero

$$\frac{\partial \mathcal{L}_1(Q^*, P(z|u), \lambda_u)}{\partial \lambda_u} = 1 \cdot \left(\sum_z P(z|u) - 1 \right) = 0. \tag{A.13}$$

Inserting Equation (A.12) in (A.13) is equivalent to

$$\sum_z \left(\frac{1}{\lambda_u} \sum_i \sum_r Q^*(z;u,i,r;\theta) \right) - 1 = 0$$

and gives the solution

$$\lambda_u = \sum_z \sum_i \sum_r Q^*(z;u,i,r;\theta). \tag{A.14}$$

Inserting λ_u in Equation (A.12) results in the optimal parameter $P^*(z|u)$ given by

$$P^*(z|u) = \frac{\sum_i \sum_r Q^*(z; u, i, r; \theta)}{\sum_{z'} \sum_i \sum_r Q^*(z'; u, i, r; \theta)} = \frac{\sum_{\langle u', i, r\rangle : u'=u} Q^*(z; u, i, r; \theta)}{\sum_{z'} \sum_{\langle u', i, r\rangle : u'=u} Q^*(z'; u, i, r; \theta)}. \tag{A.15}$$

Finding the expression for $P(r|i, z)$ is done in the same manner as for $P(z|u)$. The derived function that is being minimized is given by

$$P^*(r|i, z) = \underset{P(r|i,z)}{\arg\min} - \sum_{\langle u,i,r\rangle} \sum_z Q^*(z; u, i, r; \theta) \log P(r|i, z). \tag{A.16}$$

Introducing the Lagrange multiplier $\lambda_{i,z}$ to enforce the normalization constraint $\sum_r P(r|i, z) = 1$ for all i and z results in the second Lagrangian function, which is given by

$$\begin{aligned} \mathcal{L}_2(Q^*, P(r|i, z), \lambda_{i,z}) = &- \sum_{\langle u,i,r\rangle} \sum_z Q^*(z; u, i, r; \theta) \log P(r|i, z) \\ &+ \sum_i \sum_z \lambda_{i,z} \left(\sum_r P(r|i, z) - 1 \right). \end{aligned} \tag{A.17}$$

Taking the derivative of the Lagrangian with respect to $P(r|i, z)$ for fixed i, r and z and setting it equal to zero results in

$$\frac{\partial \mathcal{L}_2(Q^*, P(r|i, z), \lambda_{i,z})}{\partial P(r|i, z)} = - \sum_u Q^*(z; u, i, r; \theta) \frac{1}{P(r|i, z)} + \lambda_{i,z} = 0. \tag{A.18}$$

Solving $P(r|i, z)$ gives

$$P(r|i, z) = \frac{1}{\lambda_{i,z}} \sum_u Q^*(z; u, i, r; \theta). \tag{A.19}$$

Solving the Lagrange multiplier $\lambda_{i,z}$ is done by taking the derivative of the Lagrangian with respect to $\lambda_{i,z}$ for fixed i and z and setting it equal to zero gives

$$\frac{\partial \mathcal{L}_2(Q^*, P(r|i, z), \lambda_{i,z})}{\partial \lambda_{i,z}} = 1 \cdot \left(\sum_r P(r|i, z) - 1 \right) = 0. \tag{A.20}$$

Inserting Equation (A.19) in (A.20) is equivalent to

$$\sum_r \left(\frac{1}{\lambda_{i,z}} \sum_u Q^*(z; u, i, r; \theta) - 1 \right) = 0$$

and solving $\lambda_{i,z}$ gives

$$\lambda_{i,z} = \sum_u \sum_r Q^*(z; u, i, r; \theta). \tag{A.21}$$

Now the optimal parameter $P^*(r|i,z)$ is obtained by inserting the upper expression for $\lambda_{i,z}$ in Equation (A.19), such that

$$P^*(r|i,z) = \frac{\sum_u Q^*(z;u,i,r;\theta)}{\sum_u \sum_r Q^*(z;u,i,r;\theta)} = \frac{\sum_{\langle u,i',r'\rangle : i'=i,\, r'=r} Q^*(z;u,i,r;\hat{\theta})}{\sum_{\langle u,i',r\rangle : i'=i} Q^*(z;u,i,r;\hat{\theta})}. \tag{A.22}$$

A.4 Derivation of the Conjugate Prior Distribution $P(\theta)$

The prior distribution $P(\theta)$ is chosen to be the conjugate prior of the two multinomial densities in $\theta = \{P(r|i,z), P(z|u)\}$. Consequently, $P(\theta)$ is expressed by the product of two Dirichlet distributions over all available observations $\langle u,i,r\rangle$ and possible latent states z,

$$\begin{aligned} P(\theta) &= \prod_{\langle u,i,r\rangle} \prod_z \mathrm{Dir}(\{P(r|i,z)\}|\{\gamma_{i,r,z}\}) \cdot \mathrm{Dir}(\{P(z|u)\}|\{\gamma_{u,z}\}) \\ &= \prod_z \left[\prod_{i,r} \mathrm{Dir}(\{P(r|i,z)\}|\{\gamma_{i,r,z}\}) \prod_u \mathrm{Dir}(\{P(z|u)\}|\{\gamma_{u,z}\}) \right], \end{aligned} \tag{A.23}$$

where the Dirichlet density is given by

$$\mathrm{Dir}(\{\mathbf{w}\}|\{\gamma\}) = \frac{\Gamma\left(\sum_{i=1}^m \gamma_i\right)}{\prod_{i=1}^m \Gamma(\gamma_i)} \prod_{i=1}^m w_i^{\gamma_i - 1}.$$

Applying the probability parameters in θ to the Dirichlet distribution leads to the following expressions

$$\mathrm{Dir}(\{P(z|u)\}|\{\gamma_{u,z}\}) = \frac{\Gamma\left(\sum_u \sum_z \gamma_{u,z}\right)}{\prod_u \prod_z \Gamma(\gamma_{u,z})} \prod_u \prod_z P(z|u)^{\gamma_{u,z}-1}, \tag{A.24a}$$

$$\mathrm{Dir}(\{P(r|i,z)\}|\{\gamma_{i,r,z}\}) = \frac{\Gamma\left(\sum_{i,r} \sum_z \gamma_{i,r,z}\right)}{\prod_{i,r} \prod_z \Gamma(\gamma_{i,r,z})} \prod_{i,r} \prod_z P(r|i,z)^{\gamma_{i,r,z}-1}. \tag{A.24b}$$

The normalization constants with the Gamma functions Γ only depends on its own hyperparameter γ. Therefore, these constants can be omitted when optimizing with respect to the parameters in θ. The prior $P(\theta)$ is thus proportional to

$$P(\theta) \propto \prod_z \left[\prod_{i,r} P(r|i,z)^{\gamma_{i,r,z}-1} \prod_u P(z|u)^{\gamma_{u,z}-1} \right] \tag{A.25}$$

The logarithm of $P(\theta)$ is then given by

$$
\begin{aligned}
\log P(\theta) \propto & \sum_z \left[\sum_{i,r} \log{(r|i,z)}^{\gamma_{r,i,z}-1} + \sum_u \log P(z|u)^{\gamma_{u,z}-1} \right] \\
= & \sum_z \left[\sum_{i,r} (\gamma_{r,i,z}-1) \log P(r|i,z) + \sum_u (\gamma_{u,z}-1) \log P(z|u) \right].
\end{aligned}
\tag{A.26}
$$

Hence, a simpler expression for the prior distribution $P(\theta)$ has been derived.

A.5 Derivation of the MAP Parameter Estimations

The MAP parameter estimates are obtained by minimizing Equation (2.33) with respect to the parameters in θ

$$
\begin{aligned}
\theta^*_{MAP} = & \arg\min_\theta - \sum_{\langle u,i,r\rangle} \sum_z Q^*(z;u,i,r;\theta) \log \frac{P(r|i,z)P(z|u)}{Q^*(z;u,i,r;\theta)} \\
& - \sum_z \left[\sum_{i,r} (\gamma_{r,i,z}-1) \log P(r|i,z) + \sum_u (\gamma_{u,z}-1) \log P(z|u) \right].
\end{aligned}
$$

The derivations are similar to the ones in Appendix A.3 and the steps are almost the same. The object function for $P(z|u)$ is given by

$$
\begin{aligned}
P^*(z|u) = & \arg\min_{P(z|u)} - \sum_{\langle u,i,r\rangle} \sum_z Q^*(z;u,i,r;\theta) \log P(z|u) \\
& - \sum_z \sum_u (\gamma_{u,z}-1) \log P(z|u)
\end{aligned}
\tag{A.27}
$$

Using the same normalization constraints $\sum_z P(z|u) = 1$, the Lagrange multiplier λ_u is introduced to Equation (A.5), such that a Lagrange function can be formed which is given by

$$
\begin{aligned}
\mathcal{L}_1(Q^*, P(z|u), \lambda_u) = & - \sum_{\langle u,i,r\rangle} \sum_z Q^*(z;u,i,r;\theta) \log P(z|u) \\
& - \sum_u \sum_z (\gamma_{u,z}-1) \log P(z|u) + \sum_u \lambda_u \left(\sum_z P(z|u) - 1 \right).
\end{aligned}
\tag{A.28}
$$

Taking the derivative of $\mathcal{L}_1$ with respect to $P(z|u)$ for fixed u and z and setting it equal to zero results in

$$\frac{\partial \mathcal{L}_1(Q^*, P(z|u), \lambda_u)}{\partial P(z|u)} = -\sum_i \sum_r Q^*(z; u, i, r; \theta) \frac{1}{P(z|u)} - (\gamma_{u,z} - 1) \frac{1}{P(z|u)} + \lambda_u = 0 \tag{A.29}$$

and solving $P(z|u)$ gives the expression

$$P(z|u) = \frac{1}{\lambda_u} \left[\left(\sum_i \sum_r Q^*(z; u, i, r; \theta) \right) + (\gamma_{u,z} - 1) \right]. \tag{A.30}$$

The Lagrange multiplier λ_u is obtained by taking the derivative of $\mathcal{L}_1$ with respect to λ_u for fixed u and setting it equal to zero

$$\frac{\partial \mathcal{L}_1(Q^*, P(z|u), \lambda_u)}{\partial \lambda_u} = 1 \cdot \left(\sum_z P(z|u) - 1 \right) = 0 \tag{A.31}$$

Inserting Equation (A.30) in (A.31) and solving λ_u gives

$$\lambda_u = \sum_z \left[\left(\sum_i \sum_r Q^*(z; u, i, r; \theta) \right) + (\gamma_{u,z} - 1) \right]. \tag{A.32}$$

Plugging λ_u into Equation (A.30) results in the optimal MAP estimate $P^*(z|u)$

$$P^*(z|u) = \frac{\left(\sum_i \sum_r Q^*(z; u, i, r; \theta)\right) + (\gamma_{u,z} - 1)}{\sum_{z'} \left[\left(\sum_i \sum_r Q^*(z'; u, i, r; \theta)\right) + (\gamma_{u,z'} - 1) \right]} \tag{A.33a}$$

$$= \frac{\sum_{\langle u', i, r\rangle : u' = u} Q^*(z; u, i, r; \theta) + (\gamma_{u,z} - 1)}{\sum_{z'} \sum_{\langle u', i, r\rangle : u' = u} Q^*(z'; u, i, r; \theta) + (\gamma_{u,z'} - 1)}, \tag{A.33b}$$

where $\gamma_{u,z'}$ indicates that the component is added to the sums over i and r for every possible z'.

The optimal solution for $P(r|i, z)$ has the objective function

$$P^*(r|i, z) = \underset{P(r|i,z)}{\arg\min} - \sum_{\langle u,i,r\rangle} \sum_z Q^*(z; u, i, r; \theta) \log P(r|i, z) - \sum_{i,r} \sum_z (\gamma_{i,r,z} - 1) \log P(r|i, z). \tag{A.34}$$

The Lagrangian constructed by introducing the Lagrange multiplier $\lambda_{i,z}$ under the constraint $\sum_r P(r|i, z) = 1$ is given by

$$\begin{aligned} \mathcal{L}_2(Q^*, P(r|i, z), \lambda_{i,z}) = &- \sum_{\langle u,i,r\rangle} \sum_z Q^*(z; u, i, r; \theta) \log P(r|i, z) - \sum_{i,r} \sum_z (\gamma_{i,r,z} - 1) \log P(r|i, z) \\ &+ \sum_i \sum_z \lambda_{i,z} \left(\sum_r P(r|i, z) - 1 \right). \end{aligned} \tag{A.35}$$

Taking the derivative of $\mathcal{L}_2$ with respect to $P(r|i,z)$ with fixed r, i and z and setting it equal to zero results in

$$\frac{\partial \mathcal{L}_2(Q^*, P(r|i,z), \lambda_{i,z})}{\partial P(r|i,z)} = -\sum_u Q^*(z;u,i,r;\theta)\,\frac{1}{P(r|i,z)} - (\gamma_{i,r,z}-1)\,\frac{1}{P(r|i,z)} + \lambda_{i,z} = 0, \tag{A.36}$$

which is equivalent to the expression

$$P(r|i,z) = \frac{1}{\lambda_{i,z}}\left[\left(\sum_u Q^*(z;u,i,r;\theta)\right) + (\gamma_{i,r,z}-1)\right]. \tag{A.37}$$

Solving the Lagrange multiplier $\lambda_{i,z}$ is performed by taking the derivative of $\mathcal{L}_2$ with respect to $\lambda_{i,z}$ with fixed i and z and setting it equal to zero, such that

$$\frac{\partial \mathcal{L}_2(P(r|i,z), \lambda_{i,z})}{\partial \lambda_{i,z}} = 1 \cdot \left(\sum_r P(r|i,z) - 1\right) = 0. \tag{A.38}$$

By inserting the expression in Equation (A.37) in (A.38), the solution for $\lambda_{i,z}$ is derived to

$$\lambda_{i,z} = \sum_r \left[\left(\sum_u Q^*(z;u,i,r;\theta)\right) + (\gamma_{i,r,z}-1)\right]. \tag{A.39}$$

Plugging $\lambda_{i,z}$ into Equation (A.37) results in the optimal MAP estimate $P^*(r|i,z)$

$$P^*(r|i,z) = \frac{\left(\sum_u Q^*(z;u,i,r;\theta)\right) + (\gamma_{i,r,z}-1)}{\sum_{r'}\left[\left(\sum_u Q^*(z;u,i,r';\theta)\right) + (\gamma_{i,r',z}-1)\right]} \tag{A.40a}$$

$$= \frac{\sum_{\langle u,i',r'\rangle : i'=i,\, r'=r} Q^*(z;u,i,r;\theta) + (\gamma_{i,r,z}-1)}{\sum_{\langle u,i',r\rangle : i'=i} Q^*(z;u,i,r;\theta) + (\gamma_{i,r,z}-1)}, \tag{A.40b}$$

where $\gamma_{i,r',z}$ in Equation (A.40a) indicates that the component is added to the sum over u for every possible r'.

B Case Study

B.1 Introduction

Consumers tend to use a provided rating scale differently, e.g., everyone does not have the same opinion on how to interpret the score 3 on a five-graded scale. This case study proposes a quantization of the scale, such that higher and lower ratings are discretized or thresholded into a group of ratings separately. Two quantized rating scales will be tested and compared with a standard rating scale in order to see if the recommendation performance can be improved by thresholding the scale and thereby reducing the model complexity. The applied rating scales are:

- **M1:** 5 star-rating scale $\mathcal{R} = \{1, 2, 3, 4, 5\}$, where 1 and 5 are the lowest and highest given rating respectively.
- **M2:** Thresholding the 5 star-rating scale, such that the mapping of the scale is $\mathcal{R} = \{1, 2, 3, 4, 5\} \rightarrow \{-1, +1\}$ where 3 is considered as a non-observation.
- **M3:** Thresholding the 5 star-rating scale, but 3 is considered as a "do not care"-rating which is still available for the modeling. The rating scale is thus mapped to $\mathcal{R} = \{1, 2, 3, 4, 5\} \rightarrow \{1, 2, 3\}$.

The modeling and learning procedure will be made by the Conjugate-Prior-Regularized pLSA and a line search strategy will be introduced for finding optimal prior hyperparameters. The MovieLens 1M data set [18] will be used for the experiments and the data is partitioned into training, validation and test data sets with two leave-one-out procedures.

Comparing and evaluating the models on prediction error does not work in this setting. Since the rating scales in **M2** and **M3** only have been reduced in range of possible values, the prediction errors for these models will be smaller due to that the range of possible predicted values have become smaller. Therefore, a ranking scenario will be used for evaluating which model that has the best prediction and recommendation performance.

B.2 MovieLens 1M Data set

MovieLens rating data sets are collected and given out by GroupLens, a research lab at the University of Minnesota. The 1M data set was released in February 2003 and the ratings have been collected over various periods of time. It consists of $1,000,209$ million ratings entered by $6,040$ users for $3,706$ movies. The average amount of ratings per user is 165.6. The rating scale is a discrete five-star scale with the highest rating 5. The mean rating based on all observations is ≈ 3.58 and the overall rating variance is ≈ 1.25.

B.3 Ranking Accuracy Metrics

A distinct way of showing the usefulness of a recommender system is to generate personalized recommendation lists by utilizing the model to predict the ratings for unobserved items and ranking them for each user. By comparing the predicted ranking list with the users true ordering, i.e., what recommendations the user actually wishes to receive, yields a measure to the recommender system's suitability in generating personalized item orderings [2, p. 7]. For this purpose, the following ranking metrics was proposed in [13, p. 5-6]; *hit rate* (HR), *average reciprocal hit rate* ($ARHR$) and *area under curve* (AUC). Although the measurements are slightly different, the preparatory work with the data is the same. When comparing algorithms independently of the application AUC is a useful measure. But when selecting an algorithm to use for a particular task it is preferable to use a measure that reflects the specific needs at hand, which HR and $ARHR$ does [3, p. 276].

The data sets are converted into a binary, positive-only data set, where high ratings are interpreted as preferences and low ratings as dislikes. Ratings that are in the middle of the rating scale, such as a 3, are ignored and assumed to be unknowns. Both unknowns and dislikes are represented as zeros in the training data, but are allowed to be distinguished when evaluating the recommendations. Preferences are represented as ones, thus the data set has been converted into a binary rating matrix. By using the leave-one-out algorithm, one preference is picked randomly from every user with a minimum of two preferences and put in a test set. Every user in the test set has a hit set H_u, which contains all test preferences h_u from user u. All users with a test preference are contained in the set $U_t \equiv \{u \in U | \, |H_u| > 0\}$. When the model has been trained, the possible ratings for all unobserved items in the training data set are predicted. The predicted ratings for every user $u \in U_t$ are then ranked in a list.

HR and $ARHR$ measures the percentage of test users whose test preferences were predicted to end up in the top N recommendations list, where N is a natural number less than or equal to the total amount of available items. The difference between the measures is that $ARHR$ takes into account the test preference ranking in the top recommendations list. The HR and $ARHR$ are given by [13, p. 5]

$$HR@N = \frac{1}{|U_t|} \sum_{u \in U_t} |H_u \cap \text{top}N(u)| \tag{B.1}$$

$$ARHR@N = \frac{1}{|U_t|} \sum_{u \in U_t} |H_u \cap \text{top}N(u)| \cdot \frac{1}{\text{rank}\{h_u\}}, \tag{B.2}$$

where $\text{top}N(u)$ is the top-N recommendations list and $\text{rank}\{h_u\}$ is the ranking for the test preference h_u in $\text{top}10(u)$. The measure for AUC is the $AMAN$ version, where $AMAN$ stands for "All Missing As Negative" which means that a missing preference is treated as a dislike. It is given by

$$AUC^{AMAN} = \frac{1}{|U_t|} \sum_{u \in U_t} \frac{|I| - \text{rank}\{h_u\}}{|I| - 1}. \tag{B.3}$$

Like $ARHR$, AUC also takes into account the rank of the user's test preference in their recommendation list. However, AUC^{AMAN} decreases slower than $ARHR$ when rank$\{h_u\}$ is found lower in the list [13, p. 6]. Note that if $AUC = 0.5$ then the prediction performance is as good as random guessing which items that will end in the recommendation lists, since all items are equally likely to get picked.

B.4 Line Search for Optimal Hyperparameters

The proposed line search method consists of running the EM algorithm with a given latent state size k for different hyperparameter values and evaluating alternately on HR, $ARHR$ and AUC^{AMAN}. Thus, there will be optimal hyperparameter combinations for each of the three metrics for state size k in the end. The EM is terminated when the log-likelihood condition in Equation (3.1) is reached and a training, validation and test data set are picked by using two leave-one-out procedures on the complete data set. Since the evaluations are made on the metrics in Section B.3, the validation and test sets will only include ratings that are considered as preferences in the different models.

At first, the EM algorithm runs for the initial hyperparameter values $\{\gamma_{u,z}, \gamma_{i,r,z}\} = \{1.0, 1.0\}$ and then the evaluation metric value M_{val} on the validation set is computed for the trained model. The next step is to increase $\gamma_{u,z}$ with some predetermined value and run the EM algorithm again. When the current model has been trained, its evaluation metric value M_{val} is compared to the previous model's value. If M_{val} has increased, then $\gamma_{u,z}$ is increased further and the EM runs for the next hyperparameter combination. But if M_{val} has decreased, the line search switches direction to $\gamma_{i,r,z}$ by increasing $\gamma_{i,r,z}$ and setting $\gamma_{u,z}$ constant with the last value that gave rise to an increase of M_{val} for the next EM training. The line search procedure continues until M_{val} decreases and then it is assumed that the optimal $\gamma_{i,r,z}$, denoted by $\gamma^*_{i,r,z}$, has been found. If $\gamma^*_{i,r,z} > 1$, it may be that M_{val} has a larger value for a smaller value of $\gamma_{u,z}$, thus $\gamma_{u,z}$ is decreased until M_{val} reduces again. When the optimal hyperparameters have been found, the procedure is terminated and the evaluation metric M_{test} on the test set is computed for evaluating the recommendation performance. This line search procedure runs separately for the three evaluation metrics HR, $ARHR$ and AUC^{AMAN}. The pseudocode of the line search is shown in Algorithm 1 and an illustration of the procedure is shown in Figure B.1 .

Since $\gamma_{u,z}$ had the greater influence in decreasing the prediction error in the previous experiments on the EachMovie data set, it was decided to begin with increasing this hyperparameter in the line search as well. It seems fair to conclude that by decreasing the prediction error, then the model should do a better job in recommending items. Therefore, the main task will be to find a good value for $\gamma_{u,z}$ and then hopefully $\gamma_{i,r,z}$ can improve the recommendation measure a bit further.

B.5 Experimental Setup

The experiments included finding optimal hyperparameters for **M1**, **M2** and **M3** with latent state sizes $k = 10, 25$ and 60 with only one run of the line search method for

Algorithm 1 Line Search

```
1: Initiate $\gamma_{u,z} = 1$, $\gamma_{i,r,z} = 1$ **and** select evaluation metric $M$
2: **while** $\gamma^*_{u,z}$ and $\gamma^*_{i,r,z}$ are not found **do**
3:     Run EM Algorithm with training data
4:     Compute evaluation metric $M^{(i)}_{val}$ at iteration $i$
5:     **if** $i > 1$ **then**
6:         **if** $M^{(i)}_{val} \leq M^{(i-1)}_{val}$ **then**
7:             **if** Searching for $\gamma_{u,z}$ **then**
8:                 Search for $\gamma_{i,r,z}$
9:             **else if** Searching for $\gamma_{i,r,z}$ **then**
10:                $\gamma^*_{i,r,z}$ is found
11:            **else if** Searching for smaller $\gamma_{u,z}$ **then**
12:                **return** $\gamma^*_{u,z}$ and $\gamma^*_{i,r,z}$
13:            **end if**
14:            **if** $\gamma^*_{i,r,z}$ is found **and** $\gamma_{u,z} > 1$ **then**
15:                Search for smaller $\gamma_{u,z}$
16:            **else**
17:                $\gamma^*_{u,z}$ is found
18:                **return** $\gamma^*_{u,z}$ and $\gamma^*_{i,r,z}$
19:            **end if**
20:        **end if**
21:    **end if**
22:    **if** Searching for $\gamma_{u,z}$ **then**
23:        Increase $\gamma_{u,z}$
24:    **else if** Searching for $\gamma_{i,r,z}$ **then**
25:        Increase $\gamma_{i,r,z}$
26:    **else if** Searching for smaller $\gamma_{u,z}$ **then**
27:        Decrease $\gamma_{u,z}$
28:    **end if**
29: **end while**
30: Compute evaluation metric $M_{test}$
```

each model. If the evaluated metric on the validation set increases, then the varying hyperparameter value is increased with 0.01, or decreased with the same value for $\gamma_{u,z}$ if $\gamma^*_{i,r,z}$ has been found. The training, validation and test sets were picked differently whenever the state size is changed for all the models. This means that the same data sets were used even though the line search switched cost function, i.e., HR, $ARHR$ and AUC^{AMAN}, for a given k. The resulting evaluations and hyperparameter values are only based on one test set for every model and corresponding state size. The predicted ranking lists are made with $N = 20$ top ranked items.

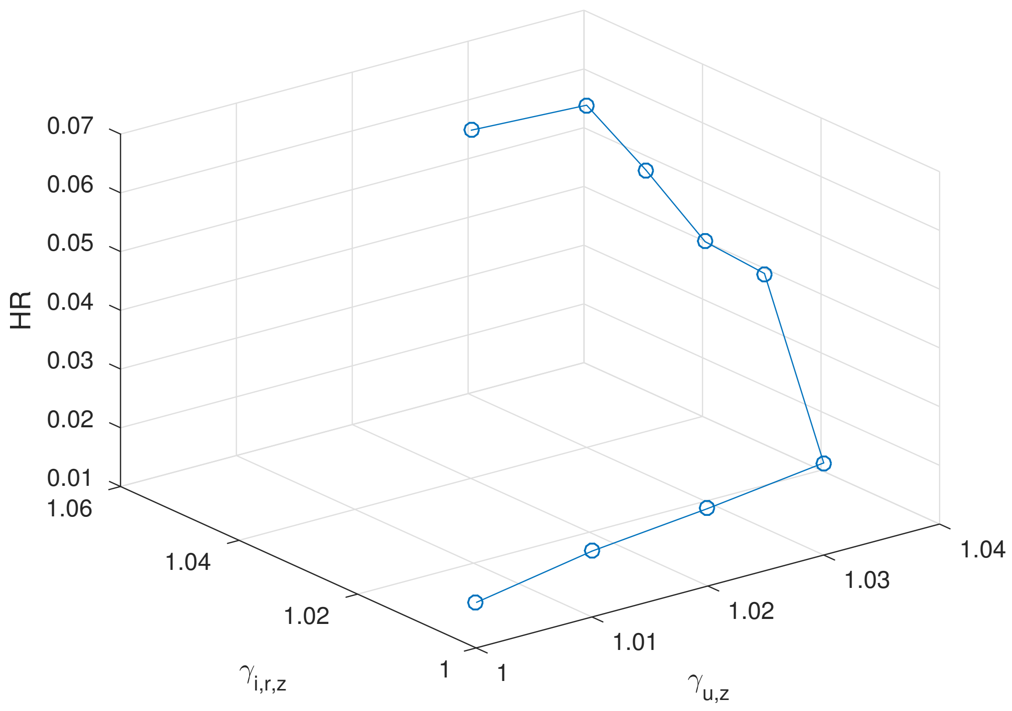

Figure B.1: Illustration of the line search procedure for **M1** with $k = 25$ and HR as evaluation metric.

B.6 Results and Discussion

The rating scale in **M2** for $k = 60$ has the best recommendation performance between the models, since it has the highest HR, $ARHR$ and AUC^{AMAN}. It is preferable to use $k > 10$ for every model and the value for $\gamma^*_{i,r,z}$ usually grows for the two larger state sizes. Also, combining both priors seems to be preferable in most cases. According to these experiments, the quantization of the rating scale improves the recommendation performance, especially when every given $r = 3$ is considered a missing rating. Reducing the model complexity, i.e., less quantities of $P(r|i, z)$ to estimate, thus results in better predictions of the missing preferences.

Since the quantization procedure improved the recommendation metrics, there will be some further investigations on how the reduction of parameters in $P(r|i, z)$ can be utilized. The suggested normalization strategies thus showed that they have a good chance in overcoming the problems that comes from different rating behaviors among the users. Also, it would be interesting to look more into which latent state size that is preferred to use.

The next step would be to develop the line search, such that it is more flexible in switching between which optimal hyperparameter that is to be found. Another proposal

		k								
		10			25			60		
Model	**Measure**	Value	$\gamma^*_{u,z}$	$\gamma^*_{i,r,z}$	Value	$\gamma^*_{u,z}$	$\gamma^*_{i,r,z}$	Value	$\gamma^*_{u,z}$	$\gamma^*_{i,r,z}$
M1	HR@20	**.044**	1.02	1.03	.061	1.02	1.04	.058	1.05	1.02
	$ARHR$@20	.007	1.02	1.02	.012	1.01	1.02	.009	1.02	1.01
	AUC^{AMAN}	.773	1.00	1.01	**.802**	1.01	1.01	.802	1.01	1.10
M2	HR@20	.042	1.00	1.02	**.091**	1.00	1.08	**.103**	1.00	1.05
	$ARHR$@20	**.009**	1.00	1.01	**.017**	1.00	1.01	**.024**	1.00	1.03
	AUC^{AMAN}	.765	1.01	1.02	.797	1.04	1.08	**.819**	1.02	1.09
M3	HR@20	.041	1.02	1.03	.066	1.02	1.06	.073	1.02	1.05
	$ARHR$@20	.007	1.01	1.03	.015	1.03	1.05	.018	1.01	1.04
	AUC^{AMAN}	**.776**	1.00	1.02	.795	1.01	1.09	.809	1.03	1.09

Table B.1: Model evaluations with HR, $ARHR$ and AUC^{AMAN} and hyperparameters $\gamma_{u,z}$ and $\gamma_{i,r,z}$ for the proposed models and different latent state sizes k. The largest evaluation measurement for state size k is bold typed.

could be to replace the line search with a Nelder-Mead method for finding suitable hyperparameters. Even though Nelder-Mead does not guarantee finding an optimal solution, it might be worth using a more advanced optimization technique rather than the line or grid search in order to reduce the computational time.

Master’s Theses in Mathematical Sciences 2016:E12
ISSN 1404-6342

LUTFMA-3290-2016

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/