Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

WHEN AUTOENCODERS MEET RECOMMENDER SYSTEMS: COFILS APPROACH

Julio César Barbieri Gonzalez de Almeida

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Geraldo Zimbrão da Silva
Leandro Guimarães Marques Alvim

Rio de Janeiro
Março de 2017

WHEN AUTOENCODERS MEET RECOMMENDER SYSTEMS: COFILS APPROACH

Julio César Barbieri Gonzalez de Almeida

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2017

Almeida, Julio César Barbieri Gonzalez de

When autoencoders meet recommender systems: COFILS approach/Julio César Barbieri Gonzalez de Almeida. – Rio de Janeiro: UFRJ/COPPE, 2017.

XII, 72 p.: il.; 29, 7cm.

Orientadores: Geraldo Zimbrão da Silva
Leandro Guimarães Marques Alvim

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Referências Bibliográficas: p. 62 – 72.

1. Recommender Systems. 2. Collaborative filtering. 3. Autoencoder. 4. Deep learning. 5. COFILS. I. Silva, Geraldo Zimbrão da *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

For all those who would like to
be scientists someday

Acknowledgments

I would like to thank everyone that have contributed to this work in some way. My family and my girlfriend for all the support given during those two years, my advisors Geraldo Zimbrão and Leandro Alvim for advices and knowledges passed during the last year and my former professor Filipe Braida do Carmo for the encouragment to follow in this area with this problem in specific.

Beyond that I would like to thank some friends and classmates that contributed with ideas, advices or just some talking during these last two years: Alexsander Andrade de Melo, Hugo Diniz Rebelo, Luiz Fernando dos Reis de Oliveira, Kleyton Pontes Cotta, Raul Sena Ferreira, Victor Marinho Furtado and Ygor de Mello Canalli.

Even though many years have passed, I have to thank my former professor Ronaldo Ribeiro Goldschmidt for having given me the first opportunity to work on a research project, one of the reasons that I have followed this path.

Also, I can't forget to thank the examination board, the professors, staff members and my colleagues from PESC. Their contributions were as important as those of the people mentioned above.

And finally, special thanks to CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for funding this research.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

AUTOENCODERS E SISTEMAS DE RECOMENDAÇÃO: UMA ABORDAGEM COFILS

Julio César Barbieri Gonzalez de Almeida

Março/2017

Orientadores: Geraldo Zimbrão da Silva
Leandro Guimarães Marques Alvim

Programa: Engenharia de Sistemas e Computação

Collaborative Filtering to Supervised Learning (COFILS) transforma um problema de filtragem colaborativa (CF) em um problema clássico de aprendizado supervisionado (SL). Sua aplicação reduz a esparsidade e torna possível a utilização de variados algoritmos de SL em oposição aos métodos de decomposição de matrizes. Primeiramente, a Decomposição em Valores Singulares (SVD) gera um conjunto de variáveis latentes a partir da matriz de avaliações. Na fase de mapeamento, um novo conjunto de dados é gerado, do qual cada amostra contém um conjunto de variáveis latentes de um usuário e do item avaliado; e um valor que corresponde a avaliação que o usuário atribuiu a esse item. Por fim, o algoritmo de SL é aplicado. Um ponto negativo do COFILS é sua dependência ao SVD, incapaz de extrair características não-lineares e sem robustez à dados ruidosos. Nesse caso, propomos a troca do SVD por um Stacked Denoising Autoencoder (SDA). Com o uso de um SDA, representações mais úteis e complexas podem ser aprendidas em uma rede neural profunda com um critério local de remoção de ruído. Executamos nossa técnica, chamada Deep Learning COFILS (DL-COFILS), nos conjuntos de dados MovieLens, R3 Yahoo! Music e Movie Tweetings comparando os resultados com o COFILS padrão, como baseline, e demais técnicas de estado da arte de CF. Com os resultados obtidos, é possível mencionar que DL-COFILS supera COFILS para todos os conjuntos de dados, com uma melhora de até 5.9%. Além disso, o DL-COFILS alcança o melhor resultado para o MovieLens 100k e se encontra entre os três melhores algoritmos nos demais conjuntos de dados. Dessa forma, mostraremos que DL-COFILS representa um avanço na metodologia COFILS, melhorando seus resultados e se mostrando um método adequado para CF.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

WHEN AUTOENCODERS MEET RECOMMENDER SYSTEMS: COFILS APPROACH

Julio César Barbieri Gonzalez de Almeida

March/2017

Advisors: Geraldo Zimbrão da Silva
Leandro Guimarães Marques Alvim

Department: Systems Engineering and Computer Science

Collaborative Filtering to Supervised Learning (COFILS) transforms a Collaborative Filtering (CF) problem into classical Supervised Learning (SL) problem. Applying COFILS reduce data sparsity and make it possible to test a variety of SL algorithms rather than matrix decomposition methods. It main steps are: extraction, mapping and prediction. Firstly, a Singular Value Decomposition (SVD) generates a set of latent variables from a ratings matrix. Next, on the mapping phase, a new data set is generated where each sample contains a set of latent variables from an user and it rated item; and a target that corresponds the user rating for that item. Finally, on the last phase, a SL algorithm is applied. One problem of COFILS is it's dependency on SVD, that is not able to extract non-linear features from data and it is not robust to noisy data. To address this problem, we propose switching SVD to a Stacked Denoising Autoencoder (SDA) on the first phase of COFILS. With SDA, more useful and complex representations can be learned in a Deep Network with a local denoising criterion. We test our novel technique, namely Deep Learning COFILS (DL-COFILS), on MovieLens, R3 Yahoo! Music and Movie Tweetings data sets and compare to COFILS, as a baseline, and state of the art CF techniques. Our results indicate that DL-COFILS outperforms COFILS for all the data sets and with an improvement up to 5.9%. Also, DL-COFILS achieves the best result for the MovieLens 100k data set and ranks on the top three algorithms for these data sets. Thus, we show that DL-COFILS represents an advance on COFILS methodology, improving it's results and that is a suitable method for CF problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we introduce and motivate this work, a summary of the achieved results, the main goals of the work, the questions evaluated and how this work is structured

1.1 Motivation

The use of latent factors-based models in Collaborative Filtering field has grown massively in recent years (KOREN *et al.*, 2009). This type of technique is used constantly to explain the ratings by characterizing users and items in a determined number of factors, e. g., in movie recommendation, the discovered factors can be related to the genres most liked by the user (KOREN *et al.*, 2009). It consists in the idea that there are non-observed features for each user in the data that explain their relationship with determined item or vice versa (SARWAR *et al.*, 2000).

To achieve this, there are a wide variety of linear techniques that apply dimensionality reduction to extract latent factors from raw data. Singular Value Decomposition (SVD), initially proposed in SARWAR *et al.* (2000), is the most common of them and since then vastly used in different Collaborative Filtering techniques. The same technique was used for similar purposes in Collaborative Filtering to Supervised Learning (COFILS), a domain-independent methodology that aims to transform the Collaborative Filtering problem in a Supervised Learning problem (BRAIDA *et al.*, 2015), with reliable results and outperforming most of the classical Collaborative Filtering algorithms.

Another research direction is deep learning applied to Collaborative Filtering. In recent years, Deep learning has been successfully applied in several areas such text mining (RANZATO & SZUMMER, 2008), images processing (LEE *et al.*, 2009a) and audio recognition (LEE *et al.*, 2009b), outperforming state-of-the-art baselines. Deep learning is an unsupervised learning technique that attempts to learn, in a faster way, distributed representations of data as a set of non-linear features and combinations of them. Deep learning covers approaches such as Autoencoders (HINTON & SALAKHUTDINOV, 2006) and Convolutional Networks (LECUN & BENGIO, 1998).

Autoencoder is a deep learning approach and it was introduced by HINTON & SALAKHUTDINOV (2006) in order to perform a non-linear dimensionality reduction in raw data. The technique, as deep learning in general, has been applied in many areas (LE *et al.*, 2012, SALAKHUTDINOV & HINTON, 2009, VISHNUBHOTLA *et al.*, 2010) with reliable results. Autoencoder is an artificial neural network which uses a gradient descent algorithm for learning features from data. The main idea is to set the input data equal to the target, and limiting the number of neurons or forcing sparsity of the neuron activations in the hidden layer. This procedure can be stacked producing high-level linear and non-linear composition of features.

In this work, we propose Deep Learning COFILS (DL-COFILS), a COFILS improvement that uses an Autoencoder in order to extract non-linear features. We experiment DL-COFILS at MovieLens 100k, MovieLens 1M, R3 Yahoo! Music and MovieTweetings data sets and compare to COFILS and the state-of-the-art techniques.

Table 1.1: Summary of the MAE results.

System	**COFILS**	**DL-COFILS**	**Error Reduction (%)**
MovieLens 100k	0.702	0.697	0.717
MovieLens 1M	0.667	0.661	0.900
R3 Yahoo! Music	0.948	0.895	5.921
MovieTweetings	1.315	1.304	0.843

Our results, described in Table 1.1, indicate that DL-COFILS outperforms COFILS for all data sets and achieves 2.095% and 1.509% of average error reduction regarding MAE and RMSE, respectively. Moreover, DL-COFILS achieves one of the best results for MovieLens 100k and MovieLens 1M data sets regarding all other techniques and third best for MovieTweetings and R3 Yahoo! Music data sets.

1.2 Objectives

This work objectives are as follows:

- Autoencoder in COFILS: Propose an Autoencoder for the latent variable extraction step in COFILS;

- Modify COFILS: Modeling Autoencoder and modify which is necessary in COFILS to improve the original work obtained results;

- Comparison and Analysis: Compare the proposed method performance with COFILS using SVD as latent variable extraction technique and some Collaborative Filtering state-of-the-art approaches from literature.

1.3 Questions

Several questions are investigated in this work:

- Quantity of trees: Random Forest Regression has better quality results using higher number of trees;
- Latent Variable Extraction technique: Switching SVD to Autoencoders improves COFILS results;
- Number of layers and neurons: Autoencoders are sensitive to number of layers and neurons;
- Type of Autoencoder: Denoising Autoencoders are better than ordinary ones for this task.

1.4 Document structure

This work is organized as follows: In chapter 2, we describe the theoretical concepts that will give a basis to understand this work. In chapter 3, we present DL-COFILS, our proposed method. Next, in chapter 4, we detail our methodology in order to conduce the experiments, results and empirical findings. Finally, in the last section, we conclude our work and we point out some promising research directions.

Chapter 2

Theoretical Foundation

This chapter presents theoretical background about recommendation systems, machine learning, neural networks and autoencoders that will provide a basis for understanding the methodology used for the experiments performed. Finally, the literature review will be presented, describing the main works of the literature that use the mentioned techniques in the context of recommendation systems.

2.1 Recommender Systems

Origins of recommender systems date back to the beginning of the computing history (EKSTRAND *et al.*, 2011). The first system that incorporate some concepts that today are intrinsically linked to the recommendation was *Grundy* in 1979 (RICH, 1998). This system relies on pre-codified stereotypes of book preferences, modelling users in each one through a short interview, thus being possible to make recommendations using these same stereotypes.

Later, in the early 1990s, the first recommendation system, know as *Tapestry* (GOLDBERG *et al.*, 1992), emerged. This email systems was created to handle the excess of online information and allowed users to manually create filters so they could receive only emails according to their preferences. Through the relationship between different documents, precisely the message and its response, this system was the first to use the concept known as collaborative filtering, which would later become reference in recommendation systems.

In the late 1990s there has been an increase in popularity and commercial implementations of recommendation systems began to emerge. Systems such as *Amazon.com*[1] which uses the purchase history, browsing history and the item that the given user is currently browsing to make recommendations of new items to him (EKSTRAND *et al.*, 2011). Studies have shown that, since they were introduced,

[1]http://www.amazon.com

Table 2.1: Example of a ratings matrix fragment for a movie recommender system.

User	Star Wars	Harry Potter	The Shinning	The Avengers
John Smith	5	3	∅	1
Jane Doe	∅	4	5	4
João da Silva	3	5	∅	5

commercial recommendation systems have impacted positively on the consumption of items (ZHOU *et al.*, 2010), especially in sales of less popular products, such as technical books (CHEN *et al.*, 2004). Nowadays, in addition to *Amazon.com*, many other web sites began to use the recommendation as a differential in its business, such as *Netflix*[2] and *Youtube*[3].

Recommendation problem was formally defined by ADOMAVICIUS & TUZHILIN (2005) as: Being U the set of all system users and I the set of all possible items that can be recommended, such as movies, books or musics. Being r the utility function capable of measuring the utility of the item i for the user u, that is, $r : U \times I \rightarrow R$, where R is the sorted set of user preferences for each one of the available items. This way, for each $u \in U$ user, its needed to choose the $i' \in I$ item that maximizes the utility function for this user. In equation 2.1 its possible to see it more formally.

$$\forall u \in U, i'_u = \operatorname*{argmax}_{i \in I} r(u,i) \tag{2.1}$$

In recommendation systems, the utility of an item is often represented by a rating, that indicates how much the user liked the specific rated item, for example, an user rated the movie "Star Wars" by a value of 3 (out of 5). For identification purposes each element of user space U must include at least one unique user ID, which does not prevent the definition of a profile including additional features of each user, such as age, gender, month income, among others. Similarly, each element of item space I can include features beyond item ID, such as title, description, genre and etc.

The main problem of recommendation systems is that the utility function r is often not defined throughout the $U \times I$ space, but for a subset of this, that is, r must be extrapolated to the entire $U \times I$ space. In the table 2.1 is shown an example of user-item preference matrix for a movie recommendation system, where the ratings are presented in a range of 1 to 5 and the symbol "∅" indicates that the user did not rate the corresponding movie. The recommender should then be able to predict the ratings that each user would assign to the movies not evaluated by them and present their recommendations based on these predictions.

[2]http://www.netflix.com
[3]http://www.youtube.com

Predictions can be made through two different approaches: Use of heuristics that define and validate the utility function or estimating the utility function that optimizes a given performance criterion (ADOMAVICIUS & TUZHILIN, 2005).

According to RICCI *et al.* (2010), having extended the previous taxonomy presented by BURKE (2007), there are 6 different classes of recommendation systems, which varies according to the domain, used knowledge and, above all, the recommendation algorithm itself.

- **Content-based:** Learning of the system is given by the recommendation of items that are similar to those that the user liked in the past. The similarity between items is calculated according to characteristics associated with the items in question, for example, if the user tends to react positively to movies of a certain genre, it is very likely that the system will introduce new movies of this genre to him.
- **Collaborative filtering:** Learning is done based on items that other users similar to the user in question liked. The similarity among users tastes is calculated based on the similarity between the rating given previously by each user.
- **Demographic:** System recommends items according to the demographic profile of each user, that is, the system assumes that people from the same demographic niche will have similar preferences.
- **Knowledge-based:** Learning is based on domain-specific knowledge about how certain characteristics of the item satisfy user needs and preferences. The similarity is calculated as a function that estimates how the user needs match the possible recommendations.
- **Community-based:** System models social relationships and recommends items based on the user's friends preferences. The preferences of each user will be extracted according to the previously assigned ratings.
- **Hybrid recommender systems:** Learning for this kind of system will be done according with some combination of the classes listed above.

2.1.1 Recommendation approaches

In this section will be presented and detailed the main recommendation approaches (collaborative filtering, content-based and hybrid), as well as its main algorithms.

2.1.1.1 Collaborative Filtering

Collaborative filtering process is based on a concept that has been used by humans for centuries: seeking opinions from others for recommendations that will aid them in their own decision-making process. According to SCHAFER *et al.* (2007), collaborative filtering is the process of filtering or evaluating items using opinions provided by others.

For example, if enough John acquaintances liked a certain product, John may be inclined to think that would be interesting to consume it, and, if many didn't like it, John may not want to consume it. Another scenario is the fact that different people may recommend different types of products in which John may or may not be interested. In that case John will notice that some people have similar tastes to yours, while others do not, and the rest may have mixed tastes between what John likes and what he does not like. This way John will learn to identify people who have similar tastes to his and know who he should listen for recommendations.

More formally, the utility function $r(u, i)$ of an item i for an user u is estimated based on the utilities $r(u_j, i)$ assigned to the same item i by all users $u_j \in U$ which are most similar to the user u (ADOMAVICIUS & TUZHILIN, 2005).

As advantages of this approach, its possible to point that it don't require additional knowledge about users and items, so the quality of the recommendation does not depend on this information as content-based recommendation. Another important point is that collaborative filtering may present unexpected recommendations, avoiding obvious indications, through a concept known as serendipity (GE *et al.*, 2010).

Despite the advantages, collaborative filtering also has its disadvantages. User-item matrix sparsity is one of the most frequent problems since the recommendation systems data sets are usually large, but users generally rates less than 1% of the items, making the data extremely sparse (SARWAR *et al.*, 2001).

Sparsity issue leads to another common problem called cold-start which refers to users (or items) that still not evaluate any item in the system yet. As there is no information related to the preference of a new user, there is a clear difficulty in predicting ratings for these cases (SCHEIN *et al.*, 2002).

Another common problem is the recommendation algorithms scalability issue, since the computational cost increases as more users and items are in the system. In this way, an online recommendation system with 1 million users and items may suffer great scalability problems (SARWAR *et al.*, 2001).

To address the challenges described above, there are several approaches in the literature. These approaches are divided into two categories: memory-based and model-based techniques (ADOMAVICIUS & TUZHILIN, 2005).

2.1.1.1.1 Memory-based

Memory-based algorithms, also known as neighborhood-based, are heuristics that compute the rating prediction for an user u based on the previous ratings of his closest neighbors, previously identified by similarity. In others words, the unknown rating value $r_{u,i}$, which represent the value that an user u would give to an item i, will be computed as an aggregation of the most similar user's ratings for the item i (ADOMAVICIUS & TUZHILIN, 2005).

This way, once the k nearest neighbours of user u are denoted by $N_i(u)$, the unknown rating $r_{u,i}$ can be predicted by the average rating given to i by these neighbours, as shown in equation 2.2.

$$\hat{r}_{ui} = \frac{1}{|N_i(u)|} \sum_{v \in N_i(u)} r_{vi} \tag{2.2}$$

There are two different ways to generate predictions in memory-based methods for collaborative filtering: user-based and item-based variants. In the user-based variant, the system uses the most similar users to an user u to perform the prediction of this user rating for the item i. Similarlry, for the item-based variant, the system uses the most similar items to the i item to predict the user u preference for this item.

Despite representing with simplicity this approach, the equation 2.2 does not take into account the different similarity levels that neighbours can have with the user u. This issue can be solved by using a weighted average rather than a simple average (RICCI *et al.*, 2010).

For the user-based method, let w_{uv} be the similarity between two users u and v and considering that the item i has already been evaluated previously by a set of users. Recommendation then, will be the weighted average of the ratings given by the k most similar users $N_i(u)$ to the user u that rated the item i as the equation 2.3 shows.

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{2.3}$$

Similarly, for the item-based method, let w_{ij} be the similarity between two items i e j, considering that the user u already rated some items and $N_u(i)$ as the set of the k nearest neighbours of item i that have been evaluated by the user u. The recommendation will be the weighted average of the ratings given to the most similar items to i as shown in equation 2.4.

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u(i)} w_{ij} r_{uj}}{\sum_{v \in N_u(i)} |w_{ij}|} \tag{2.4}$$

Equations 2.3 and 2.4 do not take into account that the users can use different rating values to quantify the same level of appreciation for a given item. This issue can be solved by converting each neighbour ratings to a normalized value $h(r_{vi})$ (RICCI *et al.*, 2010) when making predictions, as is shown in the equations 2.5 e 2.6.

$$\hat{r}_{ui} = h^{-1}\left(\frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|}\right) \tag{2.5}$$

$$\hat{r}_{ui} = h^{-1}\left(\frac{\sum_{j \in N_u(i)} w_{ij} r_{uj}}{\sum_{v \in N_u(i)} |w_{ij}|}\right) \tag{2.6}$$

Most common normalization forms reported in the literature are the mean centering and the Z-score (RICCI *et al.*, 2010). Mean centering seeks to determine if the rating is positive or negative in relation to the average rating of the nearest neighbors set of ratings, that is, the algorithm will predict this deviation instead of the rating itself.

$$h(r_{ui}) = r_{ui} - \bar{r}_u$$

$$h(r_{ui}) = r_{ui} - \bar{r}_i$$

Prediction will then be obtained through the equations 2.7 for the user-based and 2.8 for the item-based.

$$\hat{r}_{ui} = \bar{r}_u + \left(\frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|}\right) \tag{2.7}$$

$$\hat{r}_{ui} = \bar{r}_i + \left(\frac{\sum_{j \in N_u(i)} w_{ij} r_{uj}}{\sum_{v \in N_u(i)} |w_{ij}|}\right) \tag{2.8}$$

Z-score works in a similar way when compared with mean centering, however, in addition to considering that different users have different average rating perceptions, it also takes into account the propagation of each user individual scales. This can be achieved by dividing the mean centering by the ratings standard deviation given by u or i.

$$h(r_{ui}) = \frac{r_{ui} - \bar{r}_u}{\sigma_u}$$

$$h(r_{ui}) = \frac{r_{ui} - \bar{r}_i}{\sigma_i}$$

Prediction will then be obtained through the equations 2.9 for the user-based and 2.10 for the item-based.

$$\hat{r}_{ui} = \bar{r}_u + \sigma_u \left(\frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|} \right) \tag{2.9}$$

$$\hat{r}_{ui} = \bar{r}_i + \sigma_i \left(\frac{\sum_{j \in N_u(i)} w_{ij} r_{uj}}{\sum_{v \in N_u(i)} |w_{ij}|} \right) \tag{2.10}$$

Similarity is a critical part when it comes to building a memory-based recommender. These metrics, in general, have a significant impact on accuracy and performance as it will be directly responsible for selecting the most reliable neighbors (RICCI *et al.*, 2010).

The similarity measure $sim(u, u')$ between an user u and another user u' will be a distance measure used as a weight in the weighted average during the recommendation process. This measure is usually calculated using only rating values rated in common between the pair of items or users (ADOMAVICIUS & TUZHILIN, 2005). Among the most common measures for this task, we highlight the Cosine similarity and Pearson correlation (RICCI *et al.*, 2010).

Cosine is a similarity measure between two objects s and b which consists of its representation in the form of two vectors x_a and x_b. Hence, similarity is then computed between these two vectors.

$$cos(x_a, x_b) = \frac{x_a^T x_b}{||x_a||||x_b||}$$

In recommendation systems context, its possible to consider the user u or the item i as a vector $x_u \in \mathbb{R}^{|I|}$, where $x_{ui} = r_{ui}$ if the user u rated the item i, or if the item i has been rated by u, and 0 if not, obtaining, in this way, the similarity between two users, or items, u and v according with the equation 2.11.

$$CV(u, v) = cos(x_a, x_b) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2 \sum_{j \in I_v} r_{vj}^2}} \tag{2.11}$$

A problem with the cosine similarity is that this measure does not consider the difference between the ratings average and variance. Pearson correlation is a measure capable of removing these undesirable effects from the cosine. Thus, by Pearson, similarity is given by equation 2.12.

$$PC(i, j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r_i})(r_{uj} - \bar{r_j})}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r_i})^2 \sum_{u \in U_{ij}} (r_{uj} - \bar{r_j})^2}} \tag{2.12}$$

2.1.1.1.2 Model-based

Model-based algorithms seek to use the ratings set to learn a model that can be used to predict the rating that an user u would give to an item i. Machine learning, data mining, dimensionality reduction, among others techniques are, usually, applied in order to make a model that can improve the accuracy over the regular memory-based methods.

In BREESE *et al.* (1998), authors proposed a probabilistic approach that sens collaborative filtering task as the calculation of the rating based on what is known about the user. More formally, being u a given user, i a given item and having the ratings as an integer value in a range between 0 and m, we have that $p_{u,i}$ will be the conditional probability of the user u assign a certain value as a rating to the item i given the previous ratings. This model is shown in equation 2.13.

$$p_{u,i} = E(r'_{u,i}) = \sum_{r=0}^{m} Pr(r'_{u,i} = r | r'_{u,k}, k \in I_u) r \tag{2.13}$$

Author presented two different techniques for estimating this conditional probability: cluster models and Bayesian networks. Cluster model consists in the idea that there are certain user groups or classes with similar preferences. Given the class to which the user is assigned, the ratings given by him are considered independent. A naïve Bayesian classifier was used to train the model and learn its parameters. Bayesian network model is a network where each node corresponds to each item in the domain and its state represents the different values that the user assigned rating can assume (BREESE *et al.*, 1998).

Subsequently, clustering idea continued to be vastly explored. In UNGAR *et al.* (1998) was proposed a statistical model model were each user belongs to a given class and several algorithms, such as K-means and Gibbs sampling, were compared in order to estimate the model parameters. In the work of HSIN CHEN & GEORGE (2002) the proposal was a Bayesian model that sought to cluster the system users into groups where the ratings follow similar probability distributions. To estimate the missing ratings in this model were used a hybrid search algorithm in conjunction with a Markov Chain.

In contrast with some more complicated model approaches for collaborative filtering, LEMIRE & MACLACHLAN (2005) proposes a simple model for online and offline recommendation. Namely Slope One, this approach uses a principle called "popularity differential", that determine how much better an item is liked than another in a pairwise function between the two. In practice the computed difference between these items for an user can be used to predict the rating value that another user would give to one of them.

A simple example of how this technique works is simply consider this difference by subtracting the average rating of the two items. For this, consider two users U and V and two items I and J. Suppose that user U rates item I by 1.5 and item J by 2.5, while user V rates it by 2. So, item J is rated more than item I, by a difference of $2.5 - 1.5 = 1$, this way, the prediction of the value that the user V will rate item J will be $2 + 1 = 3$. This example is shown in figure 2.1.

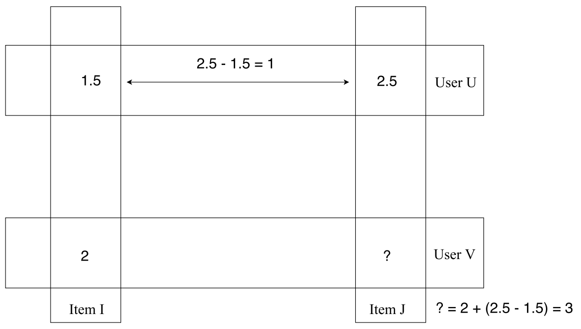


Figure 2.1: An illustration of the main idea used in Slope One schemes (LEMIRE & MACLACHLAN, 2005).

More formally, consider v_i and w_i as evaluation arrays, where $i = 1, ..., n$, the technique minimize $\sum_i (v_i + b - w_i)^2$ in order to find the best predictor like $f(x) = x + b$ that predicts w from v. Derivative with respect to b is $b = \frac{\sum_i w_i - v_i}{n}$, i.e., the constant b must be the average difference between v_i and w_i.

Considering χ the training set, i and j two items and u_i and u_j its respective ratings based on some user avaliation $u \in S_{j,i}(\chi)$, the average deviation of item i in relation to item i is shown in equation 2.14.

$$dev_{j,i} = \sum_{u \in S_{j,i}(\chi)} \frac{u_j - u_i}{card(S_{j,i}(\chi))} \tag{2.14}$$

This way, considering that $dev_{j,i} + u_i$ is one prediction for u_j given u_i, the predictor will be an average of all these predictions. Equation 2.15 shows this process, where the set of all relevant items is $R_j = \{i | i \in S(u), i \neq j, card(S_{j,i}(\chi)) > 0\}$.

$$P(u)_j = \frac{1}{card(R_j)} \sum_{i \in R_j} (dev_{j,i} + u_i) \tag{2.15}$$

The simply scheme of Slope One doesn't consider the number of ratings observed, i.e., if one wants to predict an user U rating value to an item I based on the existing user U ratings given to items J and K, where 2000 users rated both items J and L, while 20 rated both K and L. It is fairly reasonable to think that a better predictor would be given by the one that have more ratings, that is, the given user I rating for item J. This could be arranged using the equation 2.16, where $c_{j,i} = card(S_{j,i(\chi)})$.

$$P(u)_j = \frac{\sum_{i \in S(u)-\{j\}} (dev_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u)-\{j\}} c_{j,i}} \tag{2.16}$$

Over the years, new model-based techniques have emerged in the literature. Among others, regression-based models was introduced (SARWAR *et al.*, 2001) in order to approximate the rating using this kind of approach. Considering i as an target item and a similar item N, we have the vectors R_i and R_N respectively, being possible to represent the regression model as shown in equation 2.17. The parameters β and ϵ were determined by a regression between both ratings vectors. This approach is often used in conjunction with weighted sum memory-based technique, were the approximations obtained are used in place of the raw ratings.

$$\bar{R'}_N = \alpha \bar{R}_i + \beta + \epsilon \tag{2.17}$$

Using a rating approximation slightly similar to that seen in regression models, the proposal developed by FUNK (2011) uses SVD to construct its model and approximate the ratings using the factors extracted from the user-item matrix. Funk's work was followed and improved by many others, inaugurating a new class of collaborative filtering algorithms known as latent factor models or matrix factorization. This kind of model assumes that there is latent knowledge in the user-item matrix that can be extracted in a number of factors to explain a user relationship to certain items. For movies, the latent factors extracted may explain whether the genre is more female or male oriented or whether the characters are better constructed or not, for example. For users, each factor will measure how he identifies himself with movies that have certain values in the corresponding factor (KOREN *et al.*, 2009). Figure 2.2 illustrate this idea.

In Funk proposal, later named Regularized Singular Value Decomposition (RSVD) (PATEREK, 2007), each item i is associated to a vector $q_i \in \mathbb{R}^f$, in the same way that each user u is associated to a vector $p_u \in \mathbb{R}^f$. The initialization of both vectors is obtained through the latent factor decomposition of the user-item matrix applying the previously mentioned SVD technique, which decomposes the matrix in PSQ^T, where P is the matrix of latent factors of the users, Q^T is the matrix of latent factors of the items and S the singular matrix, discarded during the process. Figure 2.3 shows an example of SVD applied to Collaborative Filtering.

In this way, rating predictions are performed according to the equation 2.18 in these approach, where q_i and p_j are the K-dimensional factors vectors and the dot product $q_i^T p_u$ among these factors of the user u with the factors of the item i is the prediction for the rating that the user u would assign to the item i.

$$\hat{r}_{iu} = q_i^T p_u \tag{2.18}$$

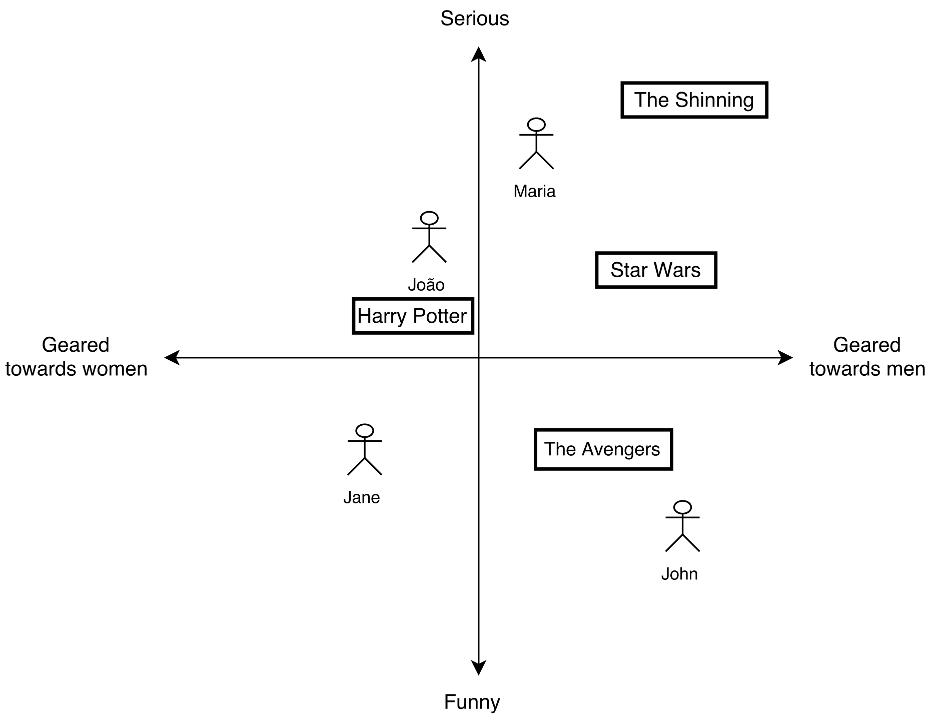


Figure 2.2: Simplified illustration based on the idea of representation by latent factors (KOREN *et al.*, 2009).

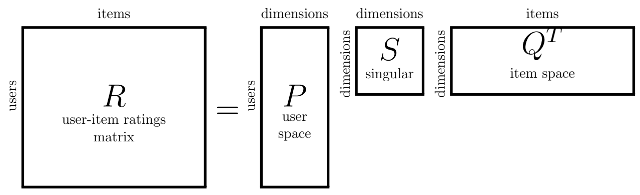


Figure 2.3: Example of SVD in a ratings matrix. Note that the S matrix is discarded for Regularized SVD technique.

Cost function is the squared error between the predicted rating and its actual value. Thus, the system learns the predictions by minimizing this function, as shown in equation 2.19, where λ is a regularization constant of a proposed value 0.02, in order to prevent overfitting.

$$\min_{q*,p*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{2.19}$$

The model is trained by stochastic gradient descent algorithm, where its iterate over the training set ratings r_{ui} and calculate the error in relation with the predicted rating r'_{ui}. Equation 2.20 illustrate this process.

$$e_{iu} = r_{iu} - \hat{r}_{iu} \tag{2.20}$$

The adjustment in the parameters q_i and p_u is then performed for each training example according to the equations 2.21 and 2.22, where γ is the learning rate with the proposed value of 0.001.

$$q_{ik} += \gamma(e_{iu}p_{uk} - \lambda q_{ik}) \tag{2.21}$$

$$p_{uk} += \gamma(e_{iu}q_{ik} - \lambda p_{uk}) \tag{2.22}$$

After this work, the popularization of latent factor models contributed to the appearance of several improvements extending what was initially proposed by Funk (KOREN, 2008, PATEREK, 2007, TAKÁCS *et al.*, 2008).

Equation 2.18 seeks to acquire the interaction between items and users to try to predict correctly for each pair of then. In spite of this, each user has a different way of evaluating and can be more or less critical, for example, as each item has different characteristics, such as good critical reception, which can interfere in different ways with the general rating pattern of each user. These biases can be incorporated into the model and learned along with latent factor vectors to achieve superior results.

In this way, the system will attempt to identify the portion of the values of $q_i^T p_u$ that the user or item biases will be able to explain in relation to the global mean μ. Equation 2.23 demonstrates this process.

$$b_{ui} = \mu + b_i + b_u \tag{2.23}$$

Given that μ is the global average rating, b_i is the item bias and b_u is the user bias, the average rating suffers the interference of how the user sees all items and how the item is viewed by all users. For example, suppose that the recommender system needs to predict the rating of a certain user John for the movie *The Hobbit*. Also consider that the global average μ is 3.5 stars. *The Hobbit* is a better than average movie, therefore, tends to be rated 0.6 above the average. John is an uncritical user, who often likes all movies he watches, giving ratings 0.5 above the average. Thus, the prediction of John rating for *The Hobbit* movie then will be 4.6 stars $(3.5 + 0.6 + 0.5)$. The new information about biases extend the equation 2.18 according to the equation 2.24.

$$\hat{r}_{iu} = \mu + b_i + b_u + q_i^T p_u \tag{2.24}$$

With the prediction method modified, the cost function showed in equation 2.19 will also be updated to incorporate the user and item biases. Note that the predicted rating is now made up of four elements: global average, user bias, item bias and the interaction between user and item factors. Equation 2.25 shows the updated cost function.

$$\min_{q*,p*,b*} \sum_{(u,i)\in K} (r_{ui} - \mu + b_u + b_i + q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2 + b_u^2 + b_i^2) \tag{2.25}$$

For the learning of the biases, the gradient descent algorithm is also used, as can be observed in the updated equations 2.26 e 2.27.

$$b_u + = \gamma(e_{iu}p_{uk} - \lambda b_u) \tag{2.26}$$

$$b_i + = \gamma(e_{iu}q_{ik} - \lambda p_{uk}) \tag{2.27}$$

Subsequently, several variations of this technique were proposed, adding information such as time, context, implicit data, among other kind of improvements, to the model in order to get better predictions. One of the most successful extensions, namely SVD++, was proposed by KOREN (2008) which adds the use of implicit feedback to improve the quality of recommendations.

Different sources of implicit data could be used to improve the predictions of the model, and the most natural way would be to use the rental history for each movie. However, since this source of information is not always easy to obtain, it is possible to use a less obvious source available in the most common collaborative filtering data sets, the items that each user rated. This data type reduces the original user-item matrix to a binary matrix, where "1" means rated item and "0" unrated item. While this kind of implicit data is not as vast as other implicit data sources, its incorporation into predictions often significantly improves results (KOREN, 2008).

Equation 2.24 will be extended according to the equation 2.28.

$$\hat{r}_{iu} = \mu + b_i + b_u + q_i^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j\in R(u)} y_j \right) \tag{2.28}$$

Equations related to the learning of the parameters 2.21, 2.22, 2.26 and 2.27 will also be modified for the inclusion of the term related to the using of implicit data, as is possible to see in equations 2.29, 2.30, 2.31 and 2.32.

$$q_{ik} + = \gamma_2(e_{iu}(p_{uk} + |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} y_j) - \lambda_7 q_{ik}) \tag{2.29}$$

$$p_{uk} + = \gamma_2(e_{iu}q_{ik} - \lambda_7 p_{uk}) \tag{2.30}$$

$$b_u + = \gamma_1(e_{iu}p_{uk} - \lambda_6 b_u) \tag{2.31}$$

$$b_i + = \gamma_1(e_{iu}q_{ik} - \lambda_6 p_{uk}) \tag{2.32}$$

This class of algorithms gained popularity and importance from the Netflix Prize, a competition organized by online movie rental company Netflix, where the goal was to develop a recommendation technique that would improve the performance in relation to the technique used by the company by 10%. In this competition, the best results were obtained by algorithms belonging to this class (KOREN *et al.*, 2009).

While most of the proposed latent factor models are based on the Funk's model described below, another approaches emerged incorporating different techniques in order to learn the model. Probabilist Matrix Factorization (PMF) model (SALAKHUTDINOV & MNIH, 2007), is a technique largely similar to the already described RSVD, but don't use SVD to initialize the latent factors.

From that model, were proposed a fully bayesian treatment of it namely Bayesian Probabilistic Matrix Factorization (BPMF) (SALAKHUTDINOV & MNIH, 2008). This work uses Gibbs sampling algorithm for sampling each user and item latent variable from its conditional distributions. In general, Markov Chain Monte Carlo methods, like Gibbs sampling, are slow, so the authors sought to speed up the sampler by sampling the variables in parallel.

Another way to factorize the ratings matrix that will be used to learn the model, is using a different kind of matrix factorization method. One technique that can be used with this purpose is the Non-Negative Matrix Factorization (NMF) (LEE & SEUNG, 2000). This technique consists in, given a non-negative $u \times i$ ratings matrix R, where u is the numbers of users and i is the number of items, factorize R into an $i \times n$ non-negative P matrix and an $n \times u$ non-negative Q matrix, where n is the number of factors. In resume, we want $R \approx PQ$.

Collaborative filtering techniques based on NMF generally uses the algorithms Expectation-Maximization or Weighted Non-Negative Matrix Factorization to train the model and then make its predictions (ZHANG *et al.*, 2006).

More recently, another latent factor model-based algorithm was introduced in order to offer high recommendation accuracy and a better scalability. This technique relies on a Bayesian Network to joint modelling users and items as we can see in figure 2.4.

The model prediction is given by the equation 2.33, where the constants ϕ_u and ψ_i are, respectively, the average rating of user u and average rating of item i, while ϵ represents the variation in the ratings that the model cannot explain. Weight vectors w_i and v_u represents item i disposition towards different user groups and user u preferences respectively. Note that this model also don't use SVD to initialize the latent factors vectors.

$$\hat{r}_{iu} = p_u w_i^T + q_i v_u^T + \phi_u + \psi_i + \epsilon \tag{2.33}$$

The original work presenting this model proposes an algorithm called Expecta-

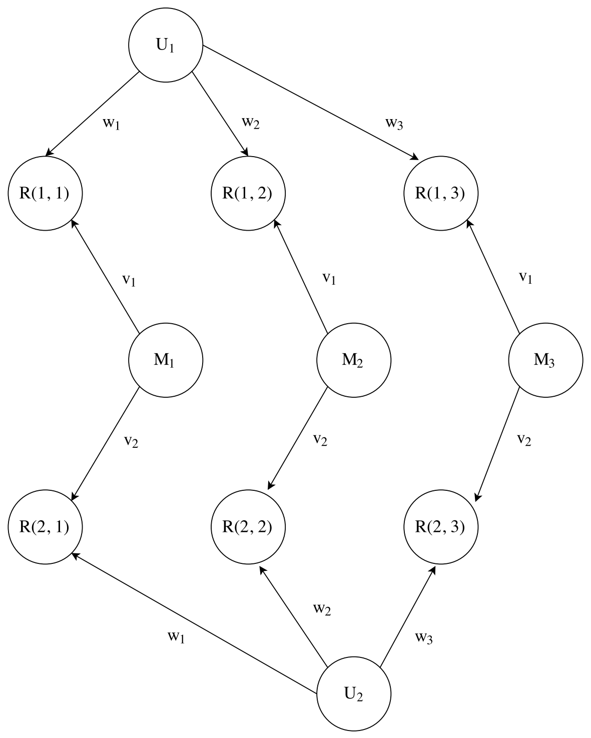


Figure 2.4: An example of the bayeasian network model, with 2 users and 3 items (LANGSETH & NIELSEN, 2012).

tion–Maximization for train the model (LANGSETH & NIELSEN, 2012). However, this algorithm requires the calculation of the full covariance matrix for all latent variables of users and items, approach that not scale very well for larger data sets. The subsequent work proposed mean-field approximation of the variational distribution, technique based on the Variational Bayes framework (LANGSETH & NIELSEN, 2015).

2.1.1.2 Content-based

Content-based recommendation systems seek to recommend items to the user based on the items that they have evaluated positively in the past. What is commonly achieved by comparing the attributes of an user profile where their preferences and interests are stored (RICCI *et al.*, 2010).

More formally, the utility function $r(u, i)$ of an item i for an user u will be estimated based on the utilities $r(u, i_j)$ attributed by the user u to the items $i_j \in I$ which are similar to the item i (ADOMAVICIUS & TUZHILIN, 2005). As an example, in a movie recommentation system, in order to recommend movies to an user u, the recommender will get its recommendation by identifying the user behavior pattern when rating each movie in the past, taking into accoun the similarity between some characteristics of the movies, such as starring actors, director, film genre, among others. Thus, only movies that have a high level of similarity to well-evaluated movies by u will be recommended.

Having its roots in information retrieval and information filtering (ADOMAVICIUS & TUZHILIN, 2005), content-based recommendation systems often focus on recommending items with textual information such as documents, web sites, news, and more. A user profile is where all the user explicit or implicit reactions to the items consumed are stored. Initialization of the profile can be done by letting the users select their areas of interest on their first visit to the system. In this way, it is possible to construct a model capable of predicting the relevance of new items to the user based on their reactions to the past items and item descriptions.

According to ADOMAVICIUS & TUZHILIN (2005) content-based recommendation methods have a series of advantages. One of the then lies in the fact that the recommendations made by it depend only on ratings provided by the user to whom the recommendation is made, while collaborative filtering methods also depends on ratings provided by other users.

Another benefit is that this system is not affected when new items are added to it, since the recommendation is completely based on the profile built for the user. In collaborative filtering it is necessary for other users to evaluate the new item so that it can begin to be recommended.

Recommendations made by this kind of system tend to be more transparent and reliable for users, since it is possible to explain why it was concluded that such an item should be recommended to such a user. Collaborative filtering systems are not able to explain their recommendations since they are predicted according to other users who have similar tastes to the user under review.

Although it has advantages, according to ADOMAVICIUS & TUZHILIN (2005) this class also has its disadvantages and limitations. One of then is the amount of user properties and items that are required, since it is not always possible to get all user-related information, which can reduce the accuracy of the recommendations. In the same way, it is usually necessary to know the domain to which the system applies so that it can capture good attributes of the items to be analyzed.

Another disadvantage, especially in relation to collaborative filtering, is the lack of mechanisms capable of making unexpected recommendations, a property called serendipity. Since the focus is an excessive specialization on the user characteristics and what he likes, the recommendations will always be focused on items similar to what the user liked, which makes it impossible to recommend something new and unexpected.

Finally, there is the new user problem, where the system is not able to recommend items to a newly registered user. A number of ratings are required to provide a precise recommendation, since the system needs to build the user profile and this task is not possible with only a few number of ratings.

2.1.1.3 Hybrid methods

Hybrid recommendation systems often combine techniques from different classes in order to compensate each approach limitations while maximizing its advantages. In general a new technique is created with the blend of concepts from different algorithm classes or even combining different techniques predictions using the most different approaches (ADOMAVICIUS & TUZHILIN, 2005).

One of the most common ways to build a hybrid system consists in the separate implementation of two systems from different recommendation classes for later combine their predictions using some pre-determined criterion. Predictions between these systems can be combined according to some pattern, for example, a weighted average between the two method predictions (CLAYPOOL *et al.*, 1999), or choosing the prediction from one of the systems by using some quality metric criterion, such as TRAN & COHEN (2000), which chooses the recommendation that most reflects the user past actions on the system.

Another way to build hybrid systems is by introducing some system class characteristics into an algorithm from another class. As an example, its possible to highlight BALABANOVIĆ & SHOHAM (1997), PAZZANI (1999), which seeks to introduce content-based characteristics in systems based in collaborative filtering. Both works applies traditional collaborative filtering approaches, but including a content-based profile rather than rated items when calculating the similarity between the users. This solution brings some benefits, such as reducing the sparsity problem impact, since the use of content-based profiles waives the use of common ratings between users, which often do not exist in sufficient quantity for some user pairs.

There are another works that uses this approach to build hybrid systems, among then, its possible to cite the work of MELVILLE *et al.* (2002). The method, called Content-Boosted Collaborative Filtering, uses a Naïve Bayes text classifier extended to support a bag of words vector to construct a user profile by using existing ratings given by him, where each bag of word represents an item attribute. In this way, the profile is used to complete the user-item matrix from which the predictions will be performed through the classical memory approach.

Still within the scope of the techniques presented previously, there are techniques that aims to do the reverse, that is, introduce collaborative filtering characteristics into content-based methods. The most common idea by works that use these approach is applying dimensionality reduction techniques on a group of content based profiles. In the work of NICHOLAS & NICHOLAS (1999), the authors proposed a technique that applies Latent Semantic Indexing (LSI) in order to create a collaborative view of user profiles. These profiles are then represented by terms vectors, obtained from documents considered relevant to the information needed by the user, which will be projected in a new space where the similarities between users can be computed.

Finally, the last way to construct hybrid systems is the construction of a unique system that contemplates characteristics from different systems classes. These kind of approach uses the most varied techniques. The proposal of BASU *et al.* (1998), for example, seeks to integrate collaborative and content-based characteristics into a single classification model. There are another proposals, such as ANSARI *et al.* (2000), CONDLIFF *et al.* (1999) which focuses on the use of a bayesian methodology that integrates different characteristics into a single probabilistic model that uses the Markov Chain method to estimate the parameters.

2.2 Machine Learning

From the earliest days of computer science, humanity believed in the potential that computers would have to be explicitly programmed to learn. Possible advances in computer self-learning could help different people in their daily lives in most diverse ways, such as learning from medical records which treatments would be effective for new diseases, learning how to manage energy costs for a home according to the usage pattern of the people living in it, learn from past ratings what would be the best movie to be recommended for a particular user, among others.

The first advances in this direction came with Arthur Samuel, a pioneer in the field of computer games and artificial intelligence. During the 1950s, Samuel developed a chess program capable of learning to improve its own playing style with only the rules of the game, a sense of direction, and a redundant list of game parameters (SAMUEL, 1959). Samuel also coined one of the earliest definitions of machine learning that has been reported:

Definition 1. *Field of study that gives computers the ability to learn without being explicitly programmed.*

Subsequently, computer scientist Tom Mitchell formalized machine learning into a more complete definition (MITCHELL, 1997):

Definition 2. *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

For example, a movie recommendation system needs to learn the preferences of its users to be able, through its learning, to perform its task of recommending movies correctly to them. According to MITCHELL (1997), to possess a well-formulated learning problem, it is necessary to identify three characteristics: the task, the performance measure and the training experience. Using the example given above, it is possible to identify these three characteristics as follows:

- Task T: recommend movies
- Performance measure P: error between the user actual rating and the system predicted rating
- Training experience E: learning from the ratings given by the system existing users

The purpose of the learning task is to determine a hypothesis that seeks to generalize the problem over all instances of the training set. Thus, without any other kind of information, the assumption is that the best hypothesis obtained is the hypothesis that best fits the data of the training set.

There are several categories by which machine learning algorithms can be grouped together. According to MITCHELL (1997), the main four are:

- Supervised Learning: Learning method that aims to use data with already defined answers to learn a model capable of generalizing the answers to the other examples.
- Unsupervised Learning: Learning method where there are no answers for each example. The goal is to find patterns in the data to form groups according to similar characteristics between the examples.
- Reinforcement Learning: Learning method that uses delayed or indirect information during the training phase.
- Evolutionary Learning: Learning that uses methods based on nature, where the model evolves according to the training. The goal is not to find the right value but to minimize the cost function.

2.2.1 Supervised Learning

Supervised learning is a machine learning task where the set of training examples has defined their corresponding target values. Its purpose is to learn a function, from these data, able to predict new elements, i.e., indicate what they represent in the data set. Thus, based on the training data, the learned function will be adjusted to learn from actual data answers and can be used for mapping new examples, generalizing to any given example type.

More formally, according to VAPNIK (1999), a general supervised learning model can be formulated from three components:

- A generator G of random vectors $x \in \mathbb{R}^n$ generated independently from an unknown and fixed probability distribution function $F(x)$.
- A supervisor S that returns a value y for each input vector x, according to an unknown and fixed probability distribution function $F(y|x)$.
- A learning machine LM able to implementing a set of funtions $f(x, \alpha), \alpha \in \Lambda$, where Λ is the set of parameters.

Learning problem is, then, choose from a set of functions $f(x, \alpha), \alpha \in \Lambda$, the one that best closely approximates to the response of S given a training set in the format $(x_1, y_1), ..., (x_n, y_n)$.

Thus, as can be seen in figure 2.5, during the training phase, the learning machine looks at each pair (x, y) of the training set and learns the best function to apply in it. After the training phase, for every x given, the machine must return a value $\bar{y}$. To obtain the best possible result, the machine must return a value $\bar{y}$ that is as close as possible to the supervisor response y.

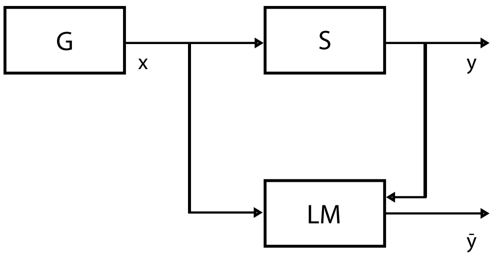


Figure 2.5: Supervised learning process (VAPNIK, 1999).

Depending on the target variable type that the algorithm must predict, supervised learning problems can be classified as classification problems, or regression problems. When the target variable has a small amount of discrete values, that is $Y \in \mathbb{N}$, the problem is classification. When the target variable has a continuous value, the problem is a regression, that is, $Y \in \mathbb{R}$. In this way, it is possible to say that a classification problem can be transformed into a regression problem, since $\mathbb{N} \subset \mathbb{R}$, although the reverse is not possible.

Regression problems seek to interpolate a function so that it fits well with the data. In this way, it would be possible to predict the value of a new point by applying the function learned in it.

Classification problems have as objective, given an input data vector and a set of N possible classes, the classification of the input data vector into one, and only one, of the possible N classes. For example, having a set of characteristics of a geometric figure, the learning algorithm should classify it between circle, triangle or rectangle.

2.2.1.1 Algorithms

2.2.1.1.1 Decision Trees

Decision tree-based learning is a simple and efficient technique that aims to classify the input data into discrete values. Generally, the function learned by this technique can be represented by two different ways: trees or a collection of if-then rules. Although simple, decision trees are vastly applied across a wide range of tasks, such as learning medical diagnostic cases and credit risk analysis (MITCHELL, 1997).

Each tree node represent a rule that will test an instance attribute from the input data and each path below the node represents the values that the same attribute can assume. In this way, the classification is performed with the instance starting at the root of the tree and being tested by each node, descending according to the values of its attributes. Leaf nodes represent the classes that the instance can assume (MITCHELL, 1997). Figure 2.6 illustrates a decision tree that shows the conditions under which a particular user would like to watch a particular movie.

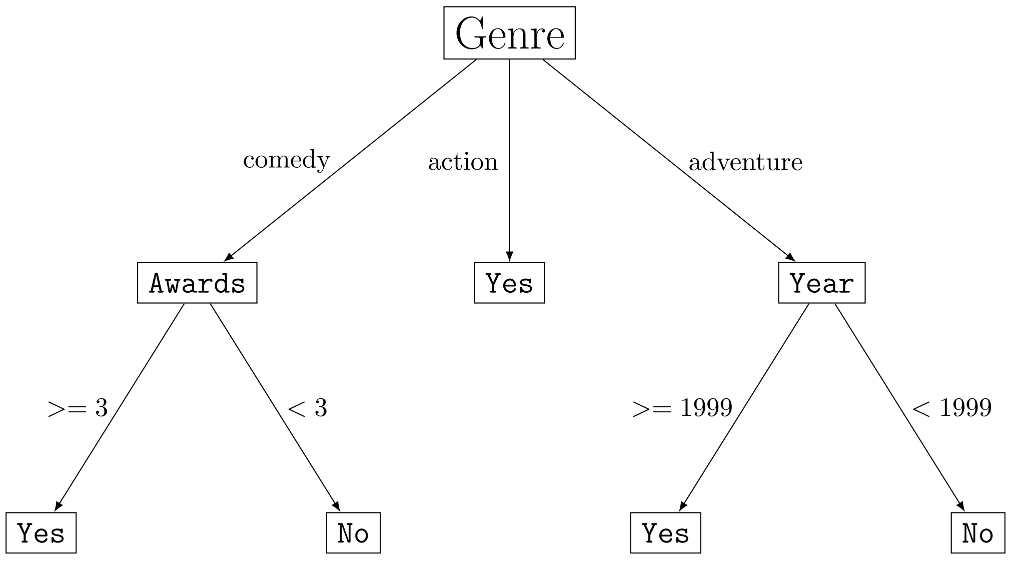


Figure 2.6: Example of a decision tree.

Alternatively, a decision tree can be represented as a disjunction of constraint conjunctions over the attributes of the instances, where each path from the root to a leaf will be a conjunction of rules between attributes, while the tree corresponds to the disjunction between these conjunctions. Hence, the tree of the figure 2.6 can be represented as the equation 2.34.

$$\text{(Watch film)} = ((\text{Awards} >= 3 \wedge \text{Comedy}) \vee (\text{Action}) \vee (\text{Year} >= 1999 \wedge \text{Adventure})) \tag{2.34}$$

In general, algorithms for learning decision trees are greedy, traversing the space from top to bottom in search of possible decision trees. Since the search for an optimal tree is usually computationally costly, because the size of the space is too large, heuristics are often appied so that the search is performed in a reasonable time.

Thus, the well know algorithms for building the decision tree are the ID3 (QUINLAN, 1986) and its successor C4.5 (QUINLAN, 1993). In both algorithms the tree is built top down performing statistical tests to find out which attribute would be best suited to be the root of the sub tree. The statistical tests performed aims to find out how well the attribute itself can classify the training examples. When the best attribute is chosen, the descendant nodes of the root are created for each of their possible values. Finally, the training examples are ordered for each of their appropriate descendant nodes and the process is repeated on its descending nodes with each of their training examples sent to them (MITCHELL, 1997). The test for choosing the best attribute is usually performed with the optimization of some metrics, such as entropy or classification error.

2.2.1.1.2 Random Forests

The model based on random forests is a learning method that applies *ensemble learning*, i.e., the grouping of several decision trees. Initially proposed by HO (1995), the main idea consists in the training of several decision trees with different random training set samples and finally combining the predictions of each of then to improve the results achieved by using only one tree. The figure 2.7 illustrates this learning process.

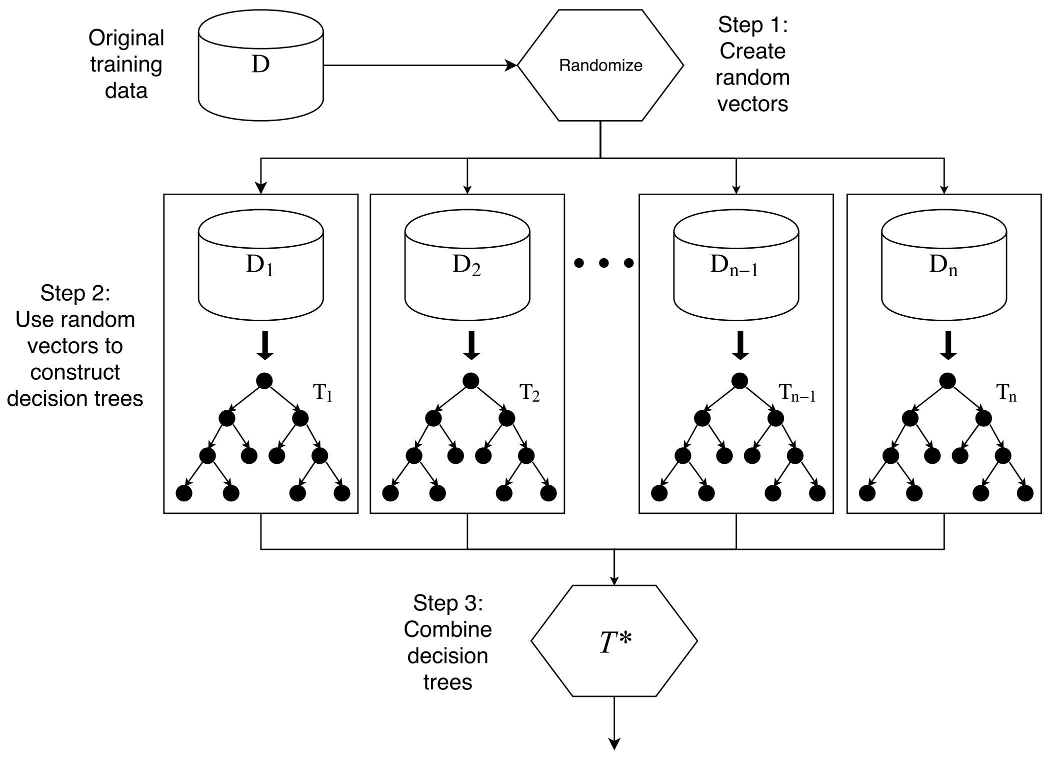

Original training data
D
Randomize
Step 1: Create random vectors
Step 2: Use random vectors to construct decision trees
D_1
D_2
D_{n-1}
D_n
T_1
T_2
T_{n-1}
T_n
Step 3: Combine decision trees
T^*

Figure 2.7: Schematization of the learning process based on random forests (TAN *et al.*, 2005).

2.2.1.1.3 Neural networks

Artificial neural networks are a computational model composed of interconnected units called neurons, where each neuron outputs the input of the other. The main concept of this technique was inspired, in part, by the observation of the human biological learning model.

The origin of this model goes back to MCCULLOCH & PITTS (1943) works, who were the first ones to try to model an artificial neuron. Subsequently, Perceptron, a simple artificial neuron model for binary classification, capable of expressing linearly separable classes, was created in ROSENBLATT (1958). In figure 2.8 its possible to see a Perceptron example.

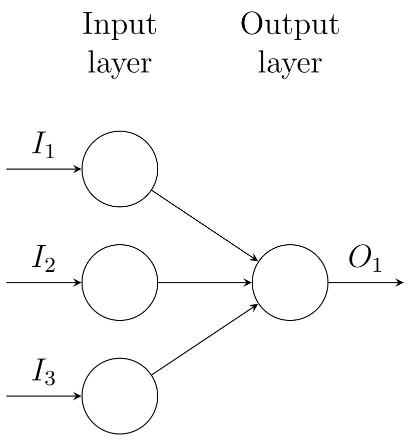


Figure 2.8: An exemple of perceptron.

A perceptron is a model that receive a real valued-input vector from x_1 to x_n and calculate a linear combination of these inputs. Hence, the output $o(x_1, ..., x_n)$ will be 1 if the resulting value is above a certain threshold and -1 otherwise, as its possible to observe in equation 2.35, where each w_i is a real value, called weight, which determines the input x_i contribution level to the value that will be obtained at the output.

$$o(x_1, ..., x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n > 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.35}$$

In order to simplify the equation 2.35, its possible to imagine a constant input $x_0 = 1$, namely bias, which allow to write this equation as $\sum_{i=0}^{n} w_ix_i > 0$, or, in its vectorized form, $\vec{w}.\vec{x} > 0$ (MITCHELL, 1997). This way, the activation function can be written according to

$$o(\vec{x}) = sgn(\vec{w}.\vec{x})$$

where

$$sgn(y) = \begin{cases} 1 & \text{if } y > 0 \\ -1 & \text{otherwise} \end{cases}$$

The perceptron learning process consists in chosing values for the weights

$w_0, ..., w_n$. There are different ways to achieve this goal, one of then is initialize the weights by random values and iterate over the training set modifying the weights each time the training example is wrongly classified (MITCHELL, 1997). This procedure is shown in equations 2.36 and 2.37, where t is the expected value for the training example output, o is the perceptron predicted output value for the same training example and γ is the learning rate.

$$\Delta\ w_i = \gamma(t - o)x_i \tag{2.36}$$

$$w_i \leftarrow w_i + \Delta\ w_i \tag{2.37}$$

The perceptron neuron model is only able to represent linearly separable classes, such as some boolean functions like AND and OR. However, with a set of different multiple interconnected neurons, it is possible to learn more complex functions. In the figure 2.9 we can see an example of a multilayer neural network or multilayer perceptron.

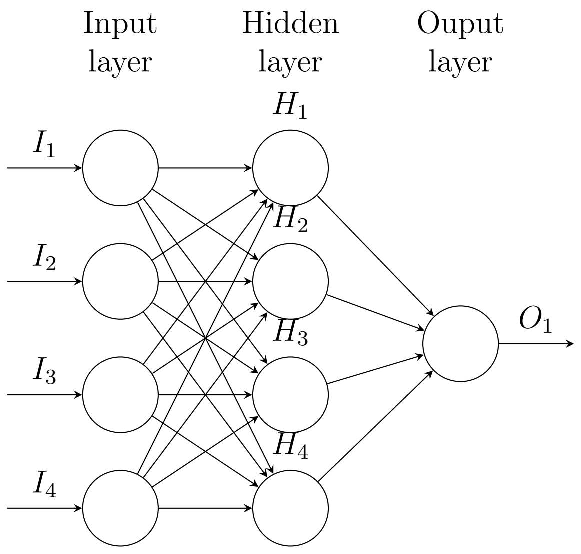


Figure 2.9: An example of multilayer artificial neural network.

Another important improvement, besides the more complex architecture, is the use of a non-linear activation function and a different training algorithm. The activation function used in perceptron, even with the use of multiple neurons and multiple layers, would still be able to perform the learning of only linear functions. Thus, a possible solution would be the use of the sigmoid function, shown in the equation 2.38.

$$f(z) = \frac{1}{1 + exp(-z)} \tag{2.38}$$

The most popular training algorithm for multi-layer neural networks training is Backpropagation. This algorithm uses the gradient descent to minimize the cost function between the network output and the expected value (MITCHELL, 1997).

2.2.2 Deep Learning

Usually requiring a human expert and domain knowledge, learning complex high-level representations from raw data is a non-trivial and challenging problem today. Based on this problem, several methods based on deep learning have emerged, which seek to learn hierarchical features, where features of higher levels are formed from a composition of features of lower levels. That is, this kind of automatic learning at different abstraction levels allows the learning of complex functions that map raw data directly without the need for a human expert (BENGIO, 2009).

According to BENGIO (2009), the concept of deep learning originated from several failed attempts to train multi-layer Perceptrons (MLP) with several hidden layers, namely Multi-Layer Deep Neural Networks, inspired by the architectural depth of the human brain. However, the Backpropagation algorithm, commonly used to train this type of network, does not work well for learning networks with more than a small number of hidden layers. Generalized presence of local optimums and optimization challenges in the non-convex cost function have become the obstacles to the use of this type of learning (DENG & YU, 2014). Only in 2006, with the introduction of the Deep Beliefs Networks (HINTON *et al.*, 2006), composed of stacked Restricted Boltzmann Machines (RBM) and using a learning algorithm that trains each layer at a time, good results began to be obtained. Later, exploring the same learning principle, algorithms based on Autoencoders (HINTON & SALAKHUTDINOV, 2006) were proposed.

Since then, deep learning techniques have been applied successfully in different tasks, such as classification (VINCENT *et al.*, 2008), regression (SALAKHUTDINOV & HINTON, 2007), dimensionality reduction HINTON & SALAKHUTDINOV (2006), faces detection (LE *et al.*, 2012), natural language processing (VISHNUBHOTLA *et al.*, 2010), information retrieval (SALAKHUTDINOV & HINTON, 2009) e collaborative filtering (SALAKHUTDINOV *et al.*, 2007).

Deep learning is based on intersections between the areas of neural networks, artificial intelligence, graphical modeling, optimization, pattern recognition and signal processing (DENG & YU, 2014).

2.2.2.1 Autoencoders

An Autoencoder is a three layer neural network, i. e., a composition of interconnected small units called neurons, where each output of a neuron can be the input to another neuron. A neural network is trained with samples $(x^{(i)}, y^{(i)})$. In the specific case of Autoencoders $x^{(i)} = y^{(i)}$ making the examples unlabeled, and allowing the network to be trained in the same way to learn a latent representation of this data in an unsupervised way.

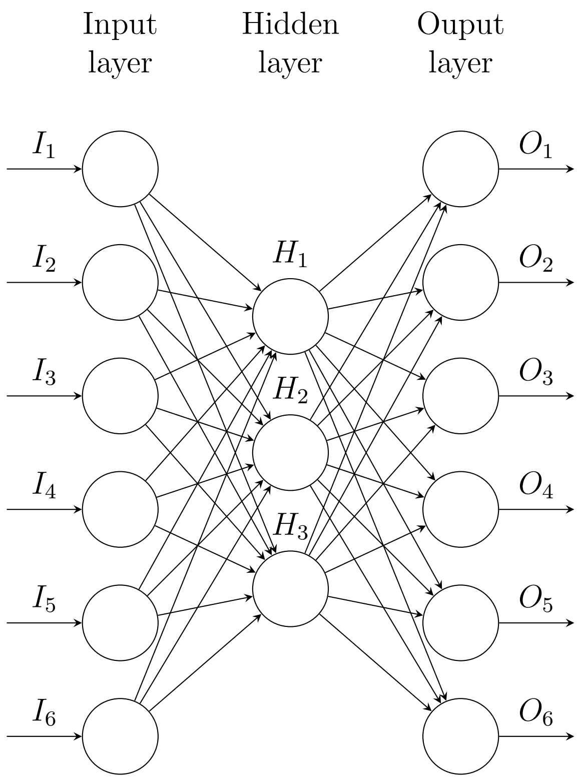


Figure 2.10: An example of autoencoder.

Therefore an Autoencoder is a non-linear generalization of PCA consisting of a multilayer encoder, which transform the high dimensional data in a low dimensional code, and a similar decoder, which try to reconstruct the data from the code, learning the identity function (HINTON & SALAKHUTDINOV, 2006). Thus, it is possible to learn a smaller representation of the original data, that is, to allow the extraction of latent variables in an unsupervised and non-linear way. Figure 2.10 shows an example of an Autoencoder with one hidden layer, where the leftmost layer is the input layer, the center layer is the hidden layer and the rightmost layer is the output layer.

Encoder layer consists in

$$y = f_\theta(x) = s(Wx + b)$$

Parametrized by $\theta = W.b$ where W is a $d' \times d$ weight matrix and b is the bias unit.

Decoder layer can be expresses by

$$z = g_\theta(y) = s(W'y + b')$$

Also parametrized by $\theta = W'.b'$.

The parameters of the model can be optimized by minimizing the *average reconstruction error* through the *backpropagation* algorithm:

$$\theta^*, \theta'^* = \underset{\theta,\theta'}{\operatorname{argmin}} = \frac{1}{n}\sum_{i=1}^{n}\left(x^{(i)}, z^{(i)}\right)$$

$$= \underset{\theta,\theta'}{\operatorname{argmin}} = \frac{1}{n}\sum_{i=1}^{n}\left(x^{(i)}, g_{\theta'}(f_\theta(x^{(i)}))\right)$$

where L is the loss function such as *squared error* or *cross entropy cost*, shown in equation 2.39 and 2.40 respectively (VINCENT *et al.*, 2008).

$$L(x, z) = ||x - z||^2 \tag{2.39}$$

$$L(x, z) = -\sum_{k=1}^{d}[x_k \log z_k + (1 - x_k)\log(1 - z_k)] \tag{2.40}$$

2.2.2.2 Denoising Autoencoders

A Denoising Autoencoder is an Autoencoder that, instead of only coding in the encoder phase, corrupt randomly some components of it in order of obtain more robust features. This prevents the Autoencoder from simply learn the identity function.

Corruption of data can be done by corrupting the input x to get a partially destroyed version $\widetilde{x}$ by a stochastic mapping $\widetilde{x} \sim q_D(\widetilde{x}|x)$. This process can be parametrized by v to regulate the inserted noise. Thus, for each x, a fixed number vd of components of the data is chosen randomly and forced to zero, while the other remains untouched. The corrupted input x is mapped, such as in the standard Autoencoder, to the hidden representation $y = f_\theta(\widetilde{x}) = s(W\widetilde{x} + b)$ from which is reconstructed a $z = g_{\theta'}(y) = s(W'y + b')$.

Then the parameters are trained by minimizing the *average reconstruction error* $L_{\mathbb{H}}(x, z) = \mathbb{H}(B_x|B_z)$ over a training set(VINCENT *et al.*, 2008). Figure 2.11 illustrate this process.

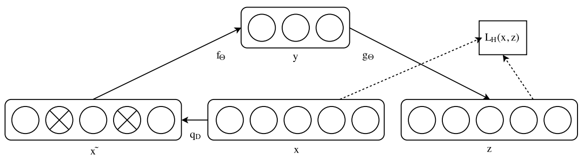


Figure 2.11: An example where can one see x being corrupted to $\widetilde{x}$. The autoencoder will map the corrupted version to y and will attempt to reconstruct x in z (VINCENT *et al.*, 2008).

2.2.2.3 Stacked Autoencoders

Stacking of autoencoders can be performed in order to extract more meaningful and robust latent variables. Stacked Autoencoders lead to a deep network which is trained by a learning algorithm that greedily trains one layer at a time, i.e., after the first level Autoencoder is trained, the encoding function learned is used in the clean input and the resulted representation can be used to train a second Autoencoder that will learn another encoding function (VINCENT *et al.*, 2010). After that, the process can be repeated how much is necessary. Figure 2.12 shows this process.

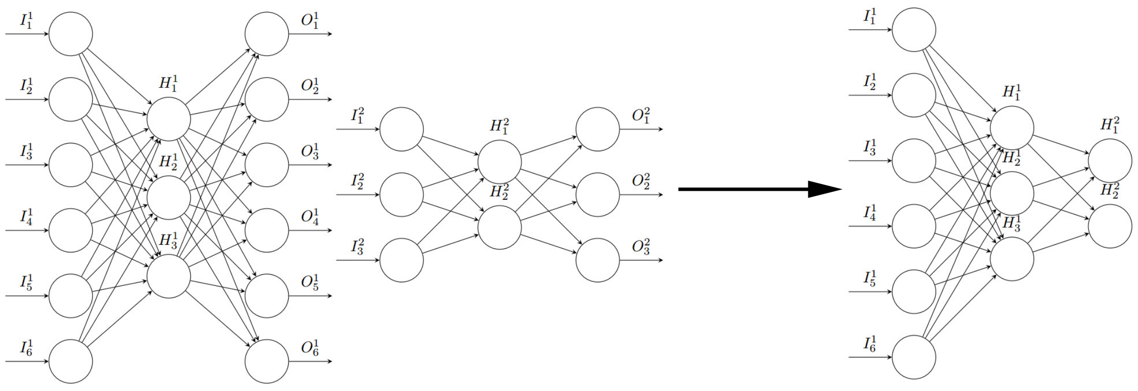


Figure 2.12: An example of two autoencoders that can be stacked to form a sightly deeper architecture.

2.3 Related Work

Deep learning was introduced in collaborative filtering during Netflix Prize, when SALAKHUTDINOV *et al.* (2007) proposes a model based in a Restricted Boltzmann Machine (RBM), where was used a different RBM for each user with binary hidden units representing if the user rated a movie or not and softmax visible units for the movies that the user has rated. They trained it with a new learning procedure called Contrastive Divergence successfully outperforming classical matrix factorization based techniques in Netflix Prize data set.

Since then, another Boltzmann Machines-based works have been introduced. For example, the work of TRUYEN *et al.* (2009) proposed an Ordinal Boltzmann Machine for joint modelling users and items, assuming that both play an equal role in the data. Making use of ordinal features to express the user preferences and also using Contrastive Divergence sampling strategy to learn the model.

Another work in the same line was that of GEORGIEV & NAKOV (2013). They proposed an model based on a RBM that considers both item-item and user-user correlations where the whole user-item matrix is treated as a single training example. Firstly they formulated an unique RBM for all users, using each of them as a single training case, in order to learn the representation of each user as a binary features set. This feature set, allows the generation of ratings for all items. Later, an item-based version was formulated and was combined with the user-based version to form the final model.

More recently some authors started to use approaches based on Autoencoders obtaining good results as well.

OUYANG *et al.* (2014) proposed a model which uses an Autoencoder for each user, where the inputs was the ratings of each movie that this user rated, meaning that the Autoencoder would have very few connections if the user rated few movies. Since that each Autoencoder have only a single training case, weights and biases units are tied together, that is, they are shared between each user models. They used a RBM to pre-train the model, taking the RBM parameters to initialize the Autoencoder and then fine-tune the model with Backpropagation.

SEDHAIN *et al.* (2015) proposes AutoRec, an Autoencoder based model that extracts latent variables only of the users or the items and minimizes directly RMSE, regularizing the learned parameters to prevent overfitting on the observed ratings and only updating the weights associated with observed inputs. The results showed the approach is promising, outperforming some collaborative filtering methods.

WANG *et al.* (2015) proposes a model named Collaborative Deep Learning (CDL) based on concepts of Stacked Denoising Autoencoders and Collaborative Topic Regression (CTR). They used a bayesian formulation of Stacked Denoising

Autoencoders, training the model by drawing samples for CDL and learning features only for items. Similarly LI *et al.* (2015) proposes an deep architecture integrating matrix factorization with deep feature learning through an Marginalized Denoising Autoencoder, learning features for both users and items.

In the meantime, some studies have tried to deal with the sparsity problem treating it as a supervised learning problem, by derivating features from the sparse matrix and training a classifier/regressor with these features (HSU *et al.*, 2007, O'MAHONY & SMYTH, 2009, O'MAHONY *et al.*, 2010). The main drawback of this approach is that the proposed transformations rely on the domain information, making it difficult.

In this scenario emerged COFILS, a domain-independent methodology aims to convert collaborative filtering problem into a supervised learning problem. The main idea behind COFILS is the use of a feature vector extracted directly of the ratings matrix associated with each corresponding rating. After that, a supervised learning algorithm can be used to train a model with the extracted data (BRAIDA *et al.*, 2015).

2.3.1 COFILS

COFILS methodology was developed to make benefit from the vast and already stabilised supervised learning methods available. One advantage of treating collaborative filtering problems as supervised learning is that we can extract high level features from data instead of raw attributes such as ratings from user-item. Hence, the possibility to extract high level features can give rise to improve model results.

As illustrated in figure 2.13, COFILS methodology is divided in three main steps: extraction, mapping and prediction. In the first one, an SVD builds latent variables from data. Next, on the mapping process, a new feature space is built, where the collaborative filtering problem is converted into a supervised learning problem. Thus, each target represents a rating of an user for a specific item and each sample is a composition of latent variables from an user with latent variables from an item. In the last phase, a supervised learning method is applied in order to learn a model.

2.3.2 Transformation

2.3.2.1 Extracting latent variables

The main idea behind the latent variables extraction is that some knowledge relative to the users' preferences is hidden in raw data and dimensionality reduction techniques can exploit this. Hence, extracting the data needed to explain the outcomes of the ratings matrix into users and items representations. More formally, an

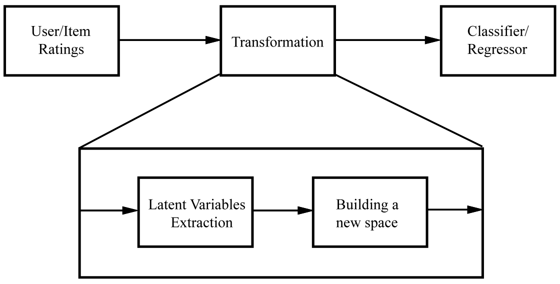


Figure 2.13: COFILS methodology steps to transform the classic collaborative filtering setting into a supervised machine learning problem (BRAIDA *et al.*, 2015).

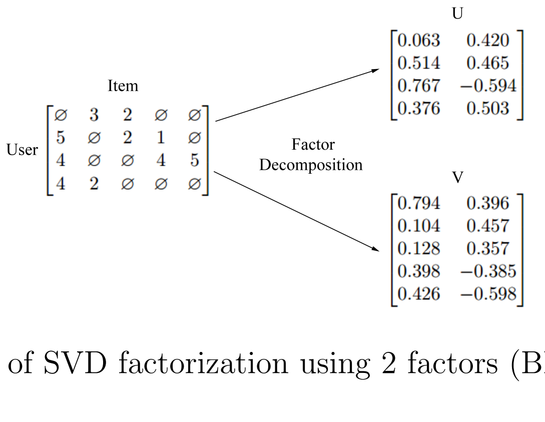
observation is represented by a rating $r_{ij} \in R$ of the rating matrix R for an user i and a particular item j. Moreover, we can represent each pair of user i and item j by vectors of latent variables u_i and v_j, respectively.

In that case, users in the vector space U, as $u_i \in U$, and items in vector space V, as $v_j \in V$.

Another drawback of extracting features from raw data, as a collaborative filtering problem, is the sparsity. There are several ways to deal with this. A standard approach is to insert values into unfilled rating positions with a value outside of the rating values range. For instance, assign a zero value when the ratings range varies from 1 to 5. Another way to circumvent this is to fill with a global average.

The introductory COFILS approach applies an SVD for latent variable extraction (BRAIDA *et al.*, 2015). Nevertheless, SVD may not learn more robust features such as non-linear methods.

In figure 2.14 we illustrate an example of factors decomposition with SVD, as well as it's factors U and V. Note that, the use of two factors are just for illustration purposes, good results are achieved generally with eight or more factors.



$$\text{User}\begin{bmatrix} \varnothing & 3 & 2 & \varnothing & \varnothing \\ 5 & \varnothing & 2 & 1 & \varnothing \\ 4 & \varnothing & \varnothing & 4 & 5 \\ 4 & 2 & \varnothing & \varnothing & \varnothing \end{bmatrix} \quad U = \begin{bmatrix} 0.063 & 0.420 \\ 0.514 & 0.465 \\ 0.767 & -0.594 \\ 0.376 & 0.503 \end{bmatrix} \quad V = \begin{bmatrix} 0.794 & 0.396 \\ 0.104 & 0.457 \\ 0.128 & 0.357 \\ 0.398 & -0.385 \\ 0.426 & -0.598 \end{bmatrix}$$

Figure 2.14: Example of SVD factorization using 2 factors (BRAIDA *et al.*, 2015).

2.3.2.2 Building a new space

After the extraction of latent variables, the next step consists of building an new input feature space from the variables extracted. To achieve this, each row from U

and V matrices, corresponding to an user u_i and an item v_j respectively, need to be mapped into the filled positions of the matrix for their corresponding ratings.

At the end of the process we will have a supervised training set $D = \{(x, y)_k\}_{k=1}^{n}$, where $x = (u_i, v_j)$, $y = r_{ij}$, and k corresponds to each position of the original matrix filled with a rating and n is the number of filled positions. With this training set it is possible to train a supervised learning model and make predictions from it. Figure 2.15 illustrate this process.

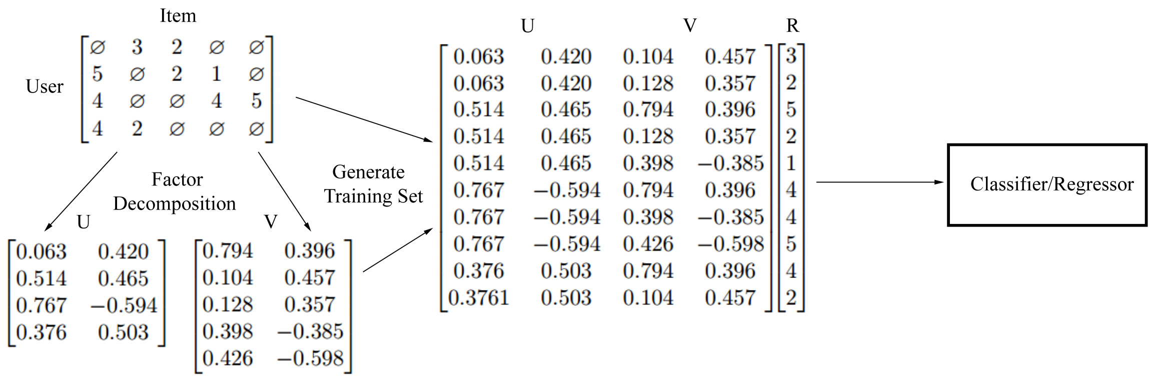


Figure 2.15: Procedure for transforming the collaborative filtering problem in a supervised learning problem (BRAIDA *et al.*, 2015).

Chapter 3

Deep Learning COFILS

This chapter will describe the proposed solution for the latent variable extraction issue in COFILS. Initially we will motivate for the SVD to be replaced by an Autoencoder, later we will define the problem and, finally, we will present each step of our proposal.

3.1 Motivation

Here, we will introduce the main issue of using linear techniques for latent variables extraction, which prevent learning of more complex architectures.

Some functions types failed to be efficiently represented with too shallow architectures, making the use of deep architectures essential to overcome this issue. As mentioned in the previous chapter, deep learning seeks the learning of features hierarchies, where features from higher levels are a composition of features of lower levels. This process allows the learning of complex functions directly from data and makes it unnecessary to use expensive hand-crafted features, that relies on expert knowledge (BENGIO, 2009).

Deep learning was tested in a wide range of fields for different tasks with reliable results (LE *et al.*, 2012, SALAKHUTDINOV & HINTON, 2009, VISHNUBHOTLA *et al.*, 2010). For digits and faces unsupervised tasks, Deep Learning performed better than a linear technique which is largely similarly to SVD, a technique extensively used in collaborative filtering problems. The work of HINTON & SALAKHUTDINOV (2006) showed that an Autoencoder, a deep learning technique, can give a much better reconstruction than Principal Component Analysis (PCA), a linear technique.

Given the motivation that deep learning methods can overcome linear latent variables extraction techniques, such as SVD and PCA, we will be followed by a definition of the problem that we are addressing.

3.2 Problem setting

In this section, we will define the problem presented in the previous section. As discussed earlier, the main goal is extract better representations from the user-item matrix than the ones extracted by linear techniques such SVD. That is, its possible to view this problem as the field of representation learning (BENGIO *et al.*, 2013).

Features chosen to train a machine learning model directly affect the method used for learning. This way, the results achieved by COFILS can be vastly influenced by the latent variable extraction technique chosen for interpret and extract representation from raw data.

When it comes to the representations extracted, its possible to illustrate this problem with an example from digits classification. Suppose an image showing an A alphabetic letter. Depending on the extracted representation from this image, the supervised learning algorithm may be able to better generalize this training example for the other letters of the same class in the test set. That is, maybe the features extracted by one algorithm perform better than the features extracted by other approaches. Examples of features that can be extracted for an A letter includes: distance between edges of the letter and bend information, showed in figure 3.1. These features would be different from a letter I, for example.

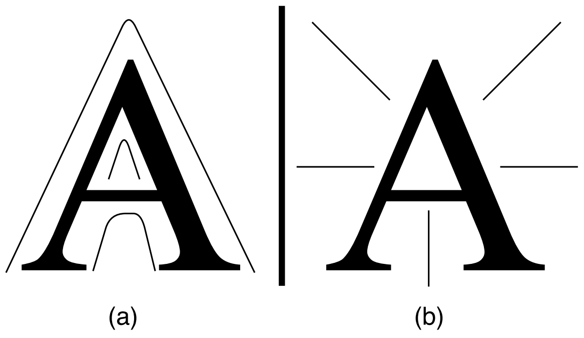


Figure 3.1: Example of different types of features that can be used to represent a letter in a digits classification problem. The left figure shows bend information, while the right figure shows the distance between edges.

The oldest feature extraction algorithm known is PCA, which can be expressed as a linear transformation that, from an input $x \in \mathbb{R}^{d_x}$, learns $h = f(x) = W_T x + b$, where the $d_x \times d_h$ matrix W columns form an orthogonal basis for the d_h orthogonal directions of greatest variance in the raw data. However, the linear features learned from PCA have a limited representation power, for example, due to linearity of these techniques, its not possible to stack then and form deeper and more abstract features to better represent the raw data (BENGIO *et al.*, 2013). This type of architecture for feature extraction is called *shallow architecture* (BENGIO, 2009).

According to BENGIO (2009), differently from shallow architectures, deep architectures can automatically discover abstractions such as the discussed earlier, that is, concepts from the lowest to the highest level. In this context, deep learning methods emerged with the goal of training deep architectures and bring the results of feature extraction algorithms to a next level.

Thus, the main challenge and goal of this work, is propose a deep learning technique for first step of COFILS methodology, in order to build more meaningful representations and overcome the achieved results so far. Hence, showing that the methodology is robust and can be upgraded as new techniques emerge in the future.

3.3 Proposal

In this section, we present our DL-COFILS architecture based on COFILS methodology. The proposed method is based in COFILS methodology and thus can be divided in three steps, however, to improve understanding, we divide the proposed method into five steps: matrices generation, normalization, autoencoder architecture, generation of supervised learning data set and appying regressor.

3.3.1 Matrices Creation

First step is the creation of the user-item matrix based on the ratings of the training set. This matrix and its transpose is used, respectively, as user and item matrices where the latent variables will be extracted.

3.3.2 Normalization

Next step is the choose of the ratings matrices normalizations. Each of the constructed matrices, user and item, can be normalized with different normalization types, such as normalization by the average user or item rating. Additionally, instead of using the actual user rating as the target for the regression task, it can also be normalized with normalization by the average user or item rating (BRAIDA *et al.*, 2015).

Note that a major difference between COFILS and this work is the untying of the matrix normalization from the target normalization. This flexibilized the methodology and was able to improve the results even using SVD as latent variable extraction technique for some data sets.

3.3.3 Autoencoder Architecture

After the normalization process, it is needed to adjust the users and items matrix range to the used Autoencoder activation function range. Later, an Autoencoder is trained for the users features using the normalized user matrix and another is trained for the item features used the normalized item matrix, as shown in the figure 3.2.

3.3.4 SL Data Set Generation

Then the features obtained from both Autoencoders are mapped just like in the regular COFILS methodology process mentioned early. To do that, the respective vector of features from an user u are combined with the feature vector of an item i and the rating r, or the normalized rating, given by the user u for the item i are used as the target for regression.

3.3.5 Applying Regressor

In the last phase, as in supervised learning problems, the generated supervised learning data set and the normalized regression targets are used to train a regressor model that can be used to make predictions.

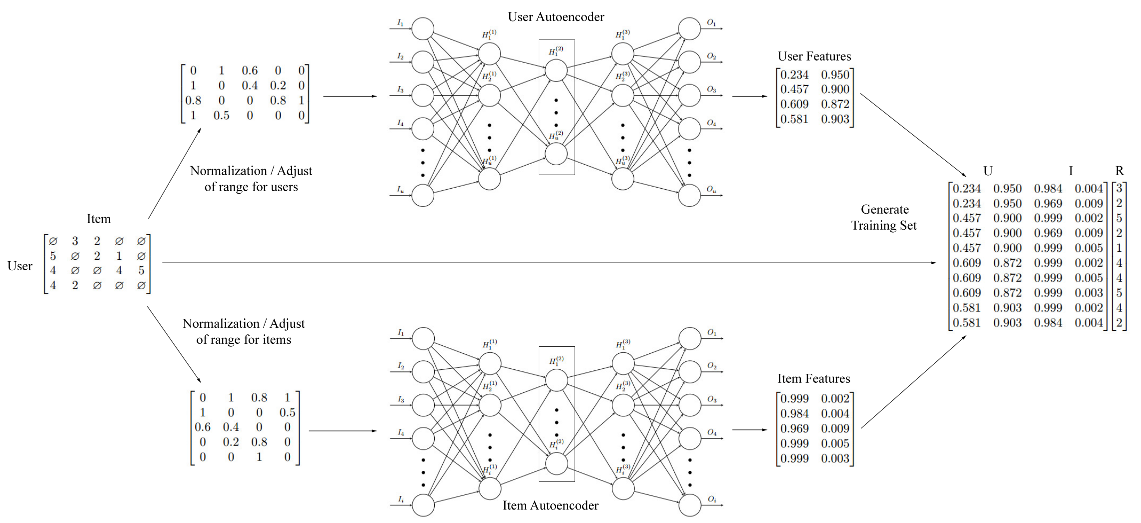


Figure 3.2: The proposed architecture based in COFILS methodoloy. Note that the last step was suppressed for being exactly as in the standard COFILS.

Chapter 4

Experimental Evaluation

This chapter presents the objectives of the experiments carried out for the validation of the proposal, the used data sets, the methodology for conducting experiments, the experiment results and a brief discussion of each achieved result.

4.1 Experiment Goals

In the previous section was presented the proposal of this work, that consists in switch the SVD for an Autoencoder in the latent variable extraction step of COFILS methodology. Besides that, was also proposed that the ratings matrix and the regression targets were normalized separately in the last step of the original methodology. The resulted method from these modifications was namely DL-COFILS.

An Autoencoder is characterized by its nonlinear extraction of features, differing greatly from the technique previously used, which may change the normalization type applied in the ratings matrix. Furthermore, being a neural network, this technique has different parameters, such as learning rate, L2 regularization, number of stacked layers and its respectives neurons, which may affect directly the obtained results. Thus, the main goals of the experiments conducted are the evaluation of the normalization and each one of the listed Autoencoder parameters, and the validation of its performance and stability in relation to the distinct data sets used. Another important point is the comparison of the proposed method against several others state-of-the-art collaborative filtering techniques in the literature, including Slope One (LEMIRE & MACLACHLAN, 2005), Regularized SVD (FUNK, 2011), Improved Regularized SVD (PATEREK, 2007), SVD++ (KOREN, 2008), Non-Negative Matrix Factorization (LEE & SEUNG, 2000), Bayesian Probabilistic Matrix Factorization (BPMF) (SALAKHUTDINOV & MNIH, 2008), RBM-CF (SALAKHUTDINOV *et al.*, 2007), AutoRec (SEDHAIN *et al.*, 2015), Mean Field (LANGSETH & NIELSEN, 2015).

4.2 Data sets

Several and distinct data sets were used in the present work to carry out the experiments to validate the proposal. Two of them were collected from the online recommender Movie Lens[1] (HARPER & KONSTAN, 2015). The first data set, MovieLens 100k, has 100,000 ratings given by 943 users to 1682 movies. The second data set, MovieLens 1M, has 1,000,209 ratings given by 6040 users to 3952 movies. In both data sets, the ratings are integers that vary between 1 and 5. Besides the ratings, both data sets have other information regarding user, such as age and sex, and items, such as title and genre. These information are discarded in order to maintain aligned with the collaborative filtering proposal. In figures 4.1 and 4.2 it's possible to see, respectively, MovieLens 100k and MovieLens 1M histograms. Note that both shares similar distributions where most ratings are between the average rating, 3.53 for MovieLens 100k and 3.58 for MovieLens 1M, and have a behavior of a normal distribution.

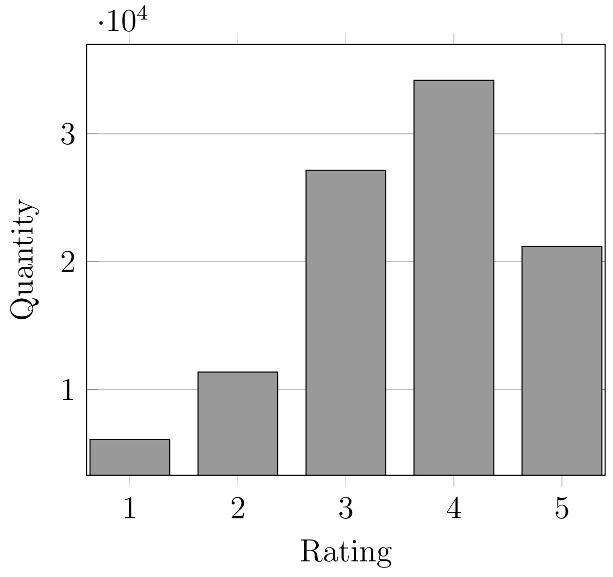


Figure 4.1: MovieLens 100k histogram.

The third one, *MovieTweetings 10k* (DOOMS *et al.*, 2013), consists of movie ratings collected from well structured tweets on Twitter. The justification for the propose of this new data set, is that older and classic data sets, such as *Netflix Prize* and *MovieLens* present dated information, failing to include new and relevant items. This version of the data has 10,000 ratings given by 3794 users to 3096 movies and the ratings vary between 1 and 10. In figures 4.3 it's possible to see MovieTweetings 10k histogram which, similarly, to other movie recommendation data sets, have most of ratings between the average rating, which is 7.34. In the same way as MovieLens

[1]https://movielens.org/

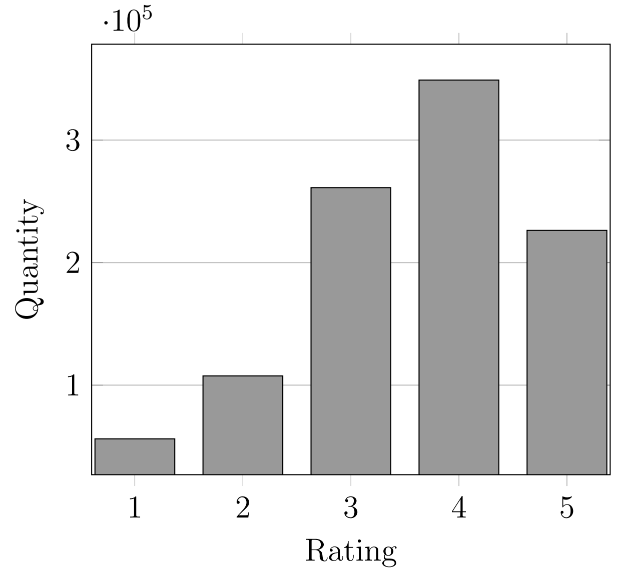


Figure 4.2: MovieLens 1M histogram.

data sets distributions, this distribution have a behavior of a normal distribution.

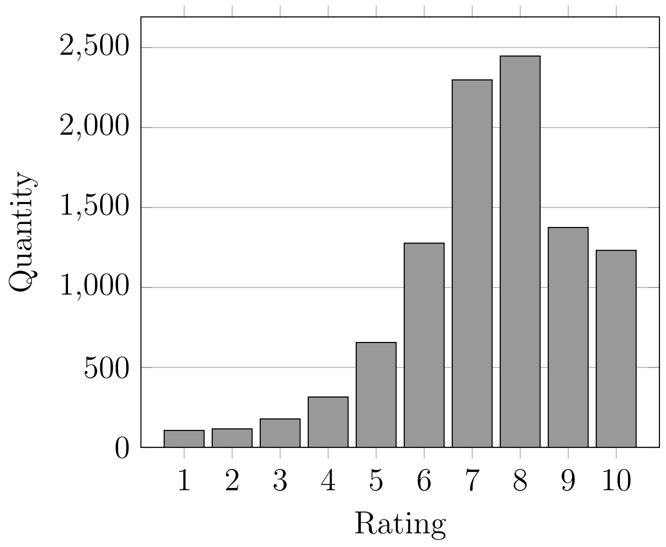


Figure 4.3: MovieTweetings 10k histogram.

R3 Yahoo! Music ratings for User Selected and Randomly Selected songs, version 1.0, available through the *Yahoo! Webscope* data sharing program[2], contains ratings collected from two different sources: normal interaction with Yahoo! Music services and randomly chosen songs for a survey conducted by Yahoo! Research. In short, this data set consists of 365,704 ratings given by 15400 users to 1000 songs and the ratings vary between 1 and 5. In figure 4.4 its possible to see R3 Yahoo! Music histogram. Note that the average rating is 2.73 and this data set follows a different distribution, indicating that users often likes or dislikes items instead of rate them

[2]http://research.yahoo.com/Academic_Relations

Table 4.1: Characteristics of the data sets.

Data sets	**Users**	**Items**	**Ratings**	**R**	**Sparsity (%)**
MovieLens 100k	943	1682	1.0×10^5	[1-5]	6.3
MovieLens 1M	6040	3900	1.0×10^6	[1-5]	4.5
MovieTweetings 10k	3794	3096	1.0×10^4	[1-10]	0.085
Yahoo! Music	15400	1000	3.7×10^5	[1-5]	2.4

with values close to the average.

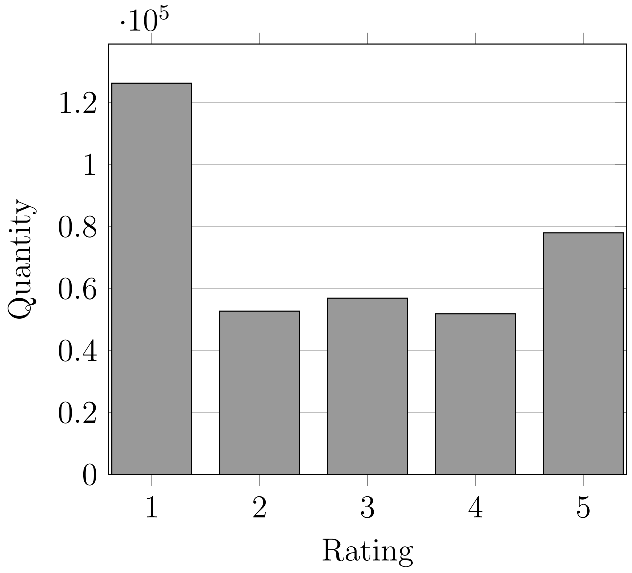


Figure 4.4: R3 Yahoo! Music histogram.

In table 4.1 we show some characteristics of the data sets used in the present work.

4.3 Methodology and Experimental Setup

For the experiments, we vary several parameters, such as the number of stacked layers, number of neurons in each stacked layer, learning rate and L2 regularization for the Autoencoder, number of trees and number of features for the Random Forest, matrix and target normalizations. In the state-of-the-art methods we vary the number of latent variables extracted while the remaining parameters, in general, remains the same of their respective works or are re-calibrated with the recommended range in its work.

In relation to ratings matrix missing values, we fill them with zero. For the existing values, we test to normalize by the average of user ratings; the average of item ratings; or no normalization at all. Previous experiments also evaluated Z-score instead of mean centering as matrix normalization method, however, the results were worse than normalizing by the average of user or item ratings. Note that for all these normalization approaches, it is necessary to adapt them to the same interval of the activation function output from an Autoencoder.

Target normalization remains the same tested on COFILS: normalization by the average user ratings or normalization by the average item rating.

For the latent variables extraction step we apply two Autoencoders: one to extract latent variables from users and another to extract latent variables from items, both with *squared error* loss function. Previously, we tested *cross entropy* as loss function, which presented worse results. Denoise method is evaluated in order to boost Autoencoder performance. Previous experiments evaluated Dropout (SRIVASTAVA *et al.*, 2014) and Sparse Autoencoders, but that techniques yields poorer results in comparison with Denoising Autoencoders.

Machine learning method chosen to be applied as supervised learning model in the last step is the one that obtained the best result in the experiments of the original methodology: Random Forest Regressor (BRAIDA *et al.*, 2015). Its parameters are reevaluated in order to achieve even better results. We also tested Support Vector Machines (SVM) and Random Forest Classifier for this task in previous experiments, however, their performance was not good enough to outperform the Random Forest Regressor. This way, all tested alternatives can be summarized as follows:

1. Pre-processing: matrix normalization by the average user rating, average item rating or no normalization at all. Target normalization by the average user rating or average item rating.

2. Latent variable extraction: Denoising Autoencoder or Ordinary Autoencoder.

3. Architecture of Autoencoder (number of latent variables used).

4. Supervised learning model: Random Forest.

The first experiment aims to define the supervised learning model parameters to be used for the remaining experiments. Parameters of the Random Forest are the overall number of trees and the overall number of features. Autoencoder parameters and normalizations are fixed in this experiment. No normalization is applied in the matrix and normalization by average item rating is applied in the target. An ordinary Autoencoder with three stacked layers and 1000, 1000 and 100 neurons in the first, two and three stacked layers respectively, with learning rate of 0.01 and L2 normalization of 0.0001 is used for both users and items.

The second experiment seeks to evaluate different normalization types that can be applied in the pre-processing step. To do that, seven distinct Autoencoder architectures with one, two or three stacked layers are chosen and applied in DL-COFILS with each normalization for a fair comparison. Parameters for this experiment are the parameters that yielded the best result in the first experiment for the Random Forest. Autoencoder parameters remains the same, except by the architecture varying.

The third experiment is intended to verify if Denoising Autoencoder outperforms a standard Autoencoder in this problem. The same distinct architectures adopted previously are applied there to validate which version of Autoencoder will be used onward. Parameters used in this experiment remains the same used in the second experiment, varying only the Autoencoder type used. Corruption level of the Denoising Autoencoder is fixed in 50% for each layer. This value was achieved on the basis of a series of previously conducted experiments varying the parameter between 10% and 70% for each stacked layer.

With the version of the Autoencoder chosen, a fourth experiment is performed to calibrate its parameters such as L2 regularization and learning rate. The same architectures used before are used again for this comparison. This time only the Denoising Autoencoder is used while the other parameters remains the same of the last experiment and learning rate and L2 regularization are varied.

The fifth experiment then is performed to choose the number of stacked layers and its respective neurons will be used. This time, various other architectures are tested in order to achieve the best possible result. Parameters are fixed in those to gave the best results for the last experiment.

Once all configurations are evaluated, a sixth experiment is performed to compare the proposed method with state-of-the-art collaborative filtering techniques from literature. Parameters of DL-COFILS is then set for those who yielded the best results on the previous experiments.

Nine algorithms, plus regular COFILS are taken for comparison. All the nine algorithms considered are model based. They are: Slope One, Regularized SVD, Improved Regularized SVD, SVD++, NMF, BPMF, RBM-CF, AutoRec and Mean Field.

In order to validate the proposal, a seventh experiment is performed. This experiment consists in running the proposed method for the other three data sets. For the MovieLens 1M, the experiment consists in running the proposed method with the best configurations chosen for MovieLens 100k. For the other two, the experiment consists in recalibrating the number of neurons, while retaining all the others configurations chosen so far, and running the proposed method for these data sets. This way, the goal is to demonstrate that the proposal can achieve similar results under different circumstances modifying few possible configurations.

For the validation experiments (I-V), hold out validation is performed to evaluate the parameters that yields the best results. Data set is split in three parts: train, validation and test. All of these experiments are performed with the train and validation partitions, while the test partition is left to validate the best parameters found in the validation process.

For the comparison experiments (VI-VII), 10-fold cross validation is performed to ensure the reliability of the obtained results.

In resume, seven experiments are carried out to evaluate the proposed method:

Experiment I

Evaluation of the Random Forest settings using no matrix normalization, target normalization by item mean, sigmoid activation function and 100 latent variables extract from a three stacked layer 1000-1000-100 Autoencoder with learning rate of 0.01 and L2 normalization of 0.0001.

Experiment II

Evaluation of Random Forest using the best configuration of the previous experiment and varying the matrix and target normalization.

Experiment III

Comparison between denoising and ordinary Autoencoders using the best configuration of the experiment two.

Experiment IV

Evaluation of Denoising Autoencoder varying L2 regularization and learning rate using the best configuration of the experiment three.

Experiment V

Evaluation varying the number of stacked layers (one or two) and the number of neurons (100 to 1000) using the best setting of the fourth experiment.

Experiment VI

Comparison with the best configuration of the experiment five with nine classical collaborative filtering approaches, plus regular COFILS: Slope One, Regularized SVD, Improved Regularized SVD, SVD++, NMF, BPMF, RBM-CF, AutoRec and Mean Field.

Experiment VII

Comparing the best configurations of the experiment six with a larger MovieLens data set. And comparing the best configurations of the experiment four, recalibrating the number of neurons (using only one layer), for the other two data sets.

4.4 Evaluation

To evaluate the results, for both validation and comparison, are chosen the most common accuracy measures for collaborative filtering: *Mean Absolute Error* (MAE) and the *Root Mean Squared Error* (RMSE) (MCLAUGHLIN & HERLOCKER, 2004). Besides that, *Normalized Mean Absolute Error* (NMAE) is chosen for comparison experiments as well (HERLOCKER *et al.*, 2004).

MAE metric is the average of the absolute differences between the predictions and the actual users ratings, as can be seen in equation 4.1, where n is the total number of ratings in the test set, p_{uv} is the algorithm prediction for the given user u evaluation to an item i and r_{uv} is the actual rating given by the same user to the same item.

$$MAE = \frac{\sum_{u,v} |p_{uv} - r_{uv}|}{n} \tag{4.1}$$

RMSE metric is highly similar to MAE, however, penalizes higher errors when in comparison with small errors. In equation 4.2 we can see it, where r is the total number of ratings in the test set, p_{uv} is the algorithm prediction for the given user u evaluation to the item i and r_{uv} is the actual rating given by the same user to the same item.

$$RMSE = \sqrt{\frac{\sum_{u,v} (p_{uv} - r_{uv})^2}{n}} \tag{4.2}$$

NMAE metric is the same metric than MAE, but normalized with respect to the range of the ratings values. In theory, this metric should permit the comparison among distinct data sets. In equation 4.3 its possible to see the metric, where n is the total number of ratings in the test set, p_{uv} is the algorithm prediction for the given user u evaluation to an item i, r_{uv} is the actual rating given by the same user to the same item, $e_{\max}$ is the maximum value that an user can rate an item and $e_{\min}$ the minimum.

$$NMAE = \frac{\frac{1}{n} \sum_{u,v} |p_{uv} - r_{uv}|}{e_{\max} - e_{\min}} \tag{4.3}$$

4.5 Results

In this subsection we present and discuss the obtained results from the evaluated experiments. In addition, we compare the proposed method with several state-of-the-art collaborative filtering methods.

4.5.1 Experiment I

In *Experiment I*, is found the out of bag error decreases highly while the number of trees increases, starting to stabilize by 200 trees, but still decreasing until 500 trees as shown in figure 4.5. Another interesting finding is that using the square root of the original number of input features, is possible to improve the results by a little margin.

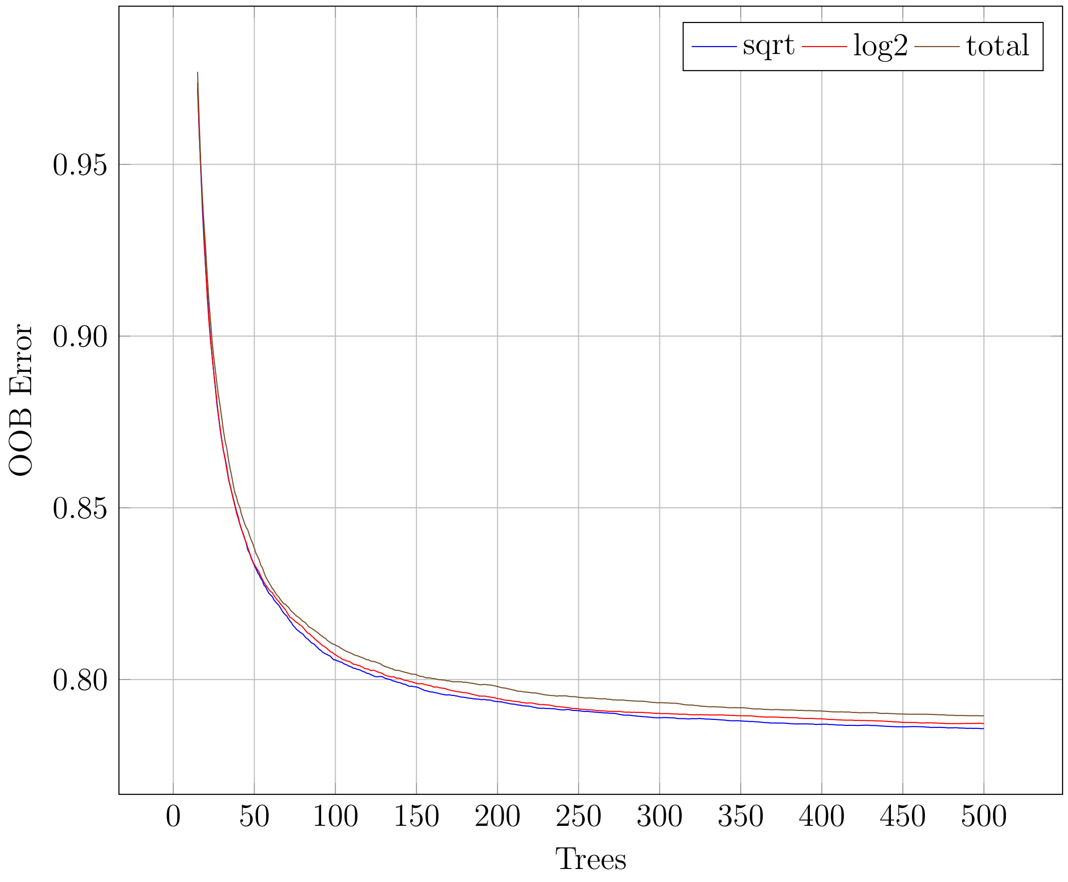


Figure 4.5: Evaluation of *Experiment I* using no matrix normalization, target normalization by item mean and an ordinary Autoencoder. Blue, red and olive lines shows, respectively the out of bag error decay when using the square root, logarithm and total of the actual features number.

4.5.2 Experiment II

In *Experiment II*, it is found the matrix normalizations used in original COFILS work does not perform very well with an Autoencoder as latent variable extraction technique. As figures 4.6 and 4.7 shows the normalization that yields the best result is just to adjust the range of ratings to the Autoencoder activation function used, with no normalization at all. Moreover, the target normalization performs equivalent results for user or item average rating when together with no matrix normalization at all.

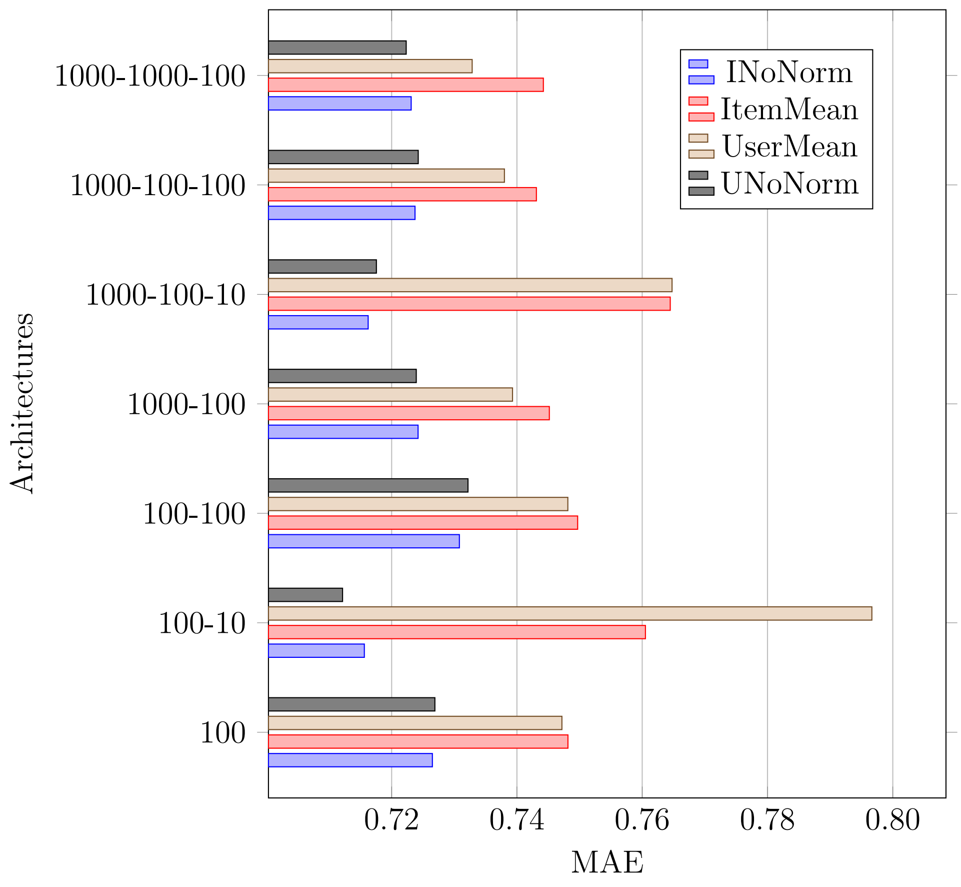


Figure 4.6: Evaluation of *Experiment II* using an ordinary Autoencoder and varying the different types of matrix normalization and target. Blue, red, beige and black bars represent, respectively, no matrix normalization with target normalization by average item rating, matrix and target normalized by average item rating, matrix and target normalized by average user rating and no matrix normalization with target normalization by average user rating.

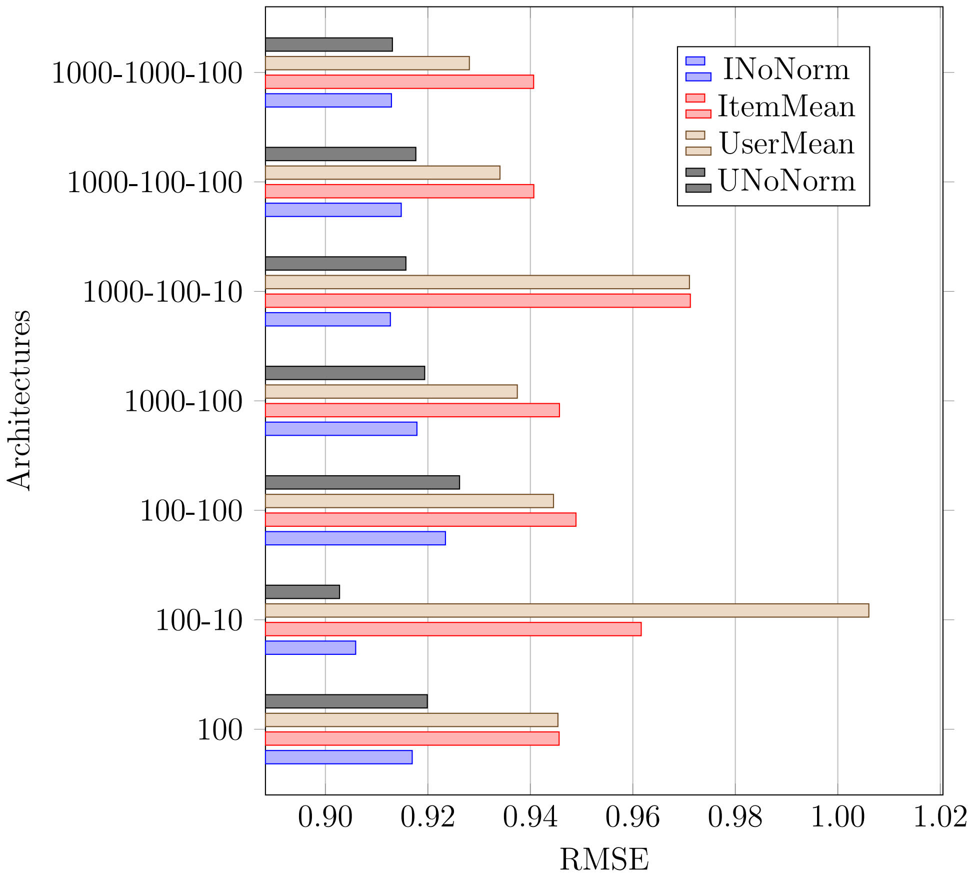


Figure 4.7: Evaluation of *Experiment II* using an ordinary Autoencoder and varying the different types of matrix normalization and target. Blue, red, beige and black bars represent, respectively, no matrix normalization with target normalization by average item rating, matrix and target normalized by average item rating, matrix and target normalized by average user rating and no matrix normalization with target normalization by average user rating.

4.5.3 Experiment III

Experiment III is responsible for evaluate the use of Denoising Autoencoder instead of a standard Autoencoder. As the results in figures 4.8 and 4.9 shown, the Denoising Autoencoder is capable of outperform the ordinary one in the most of the selected architectures, producing better latent variables.

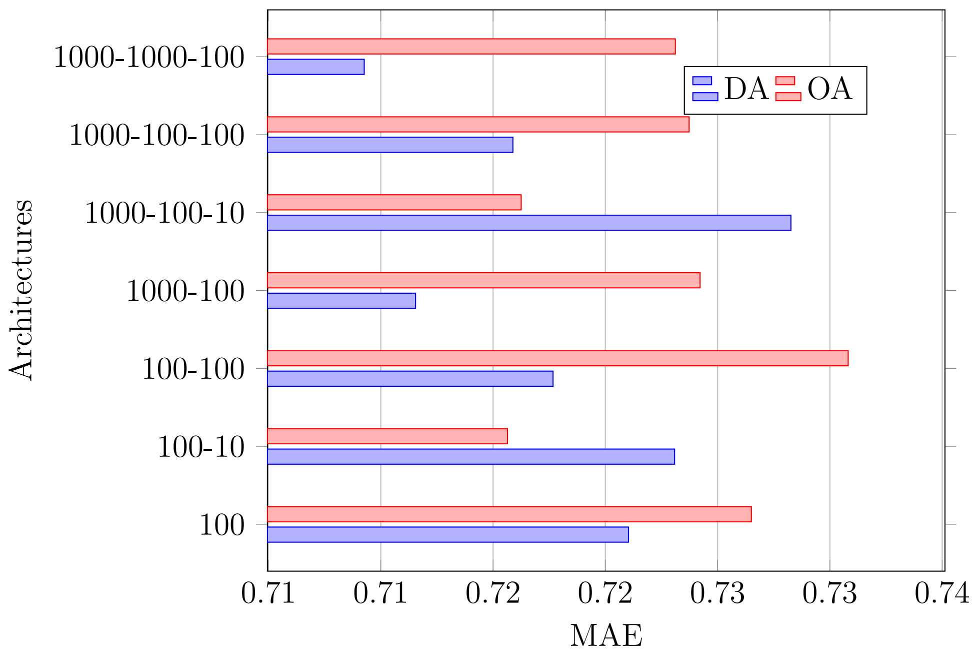


Figure 4.8: Evaluation of *Experiment III* using no matrix normalization and target normalization by item mean. The blue bar represents the Denoising Autoencoder while the red bar represents the ordinary Autoencoder.

4.5.4 Experiment IV

After the evaluation of the type of the Autoencoder, *Experiment IV* is performed in order to choose the best value for the Autoencoder's L2 regularization and learning rate. The results, showed in figures 4.10 and 4.11, indicate that 0.001 for L2 regularization and 0.01 for learning rate are the best parameters found, and will be used onwards.

4.5.5 Experiment V

With all other configurations chosen, *Experiment V* evaluates the number of stacked layers and its neurons that yields the best performance. Figure 4.12 shows five architectures that yields the best results, based on that, the chosen architecture are two stacked layers with 900 neurons in the first layer and 50 in the second.

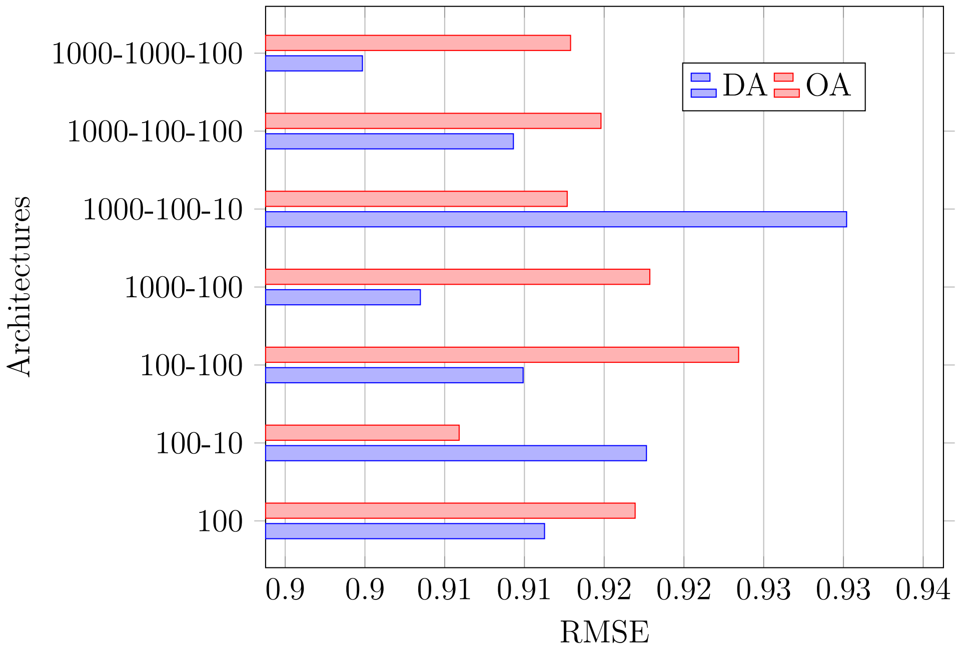


Figure 4.9: Evaluation of *Experiment II* using no matrix normalization and target normalization by item mean. The blue bar represents the Denoising Autoencoder while the red bar represents the ordinary Autoencoder.

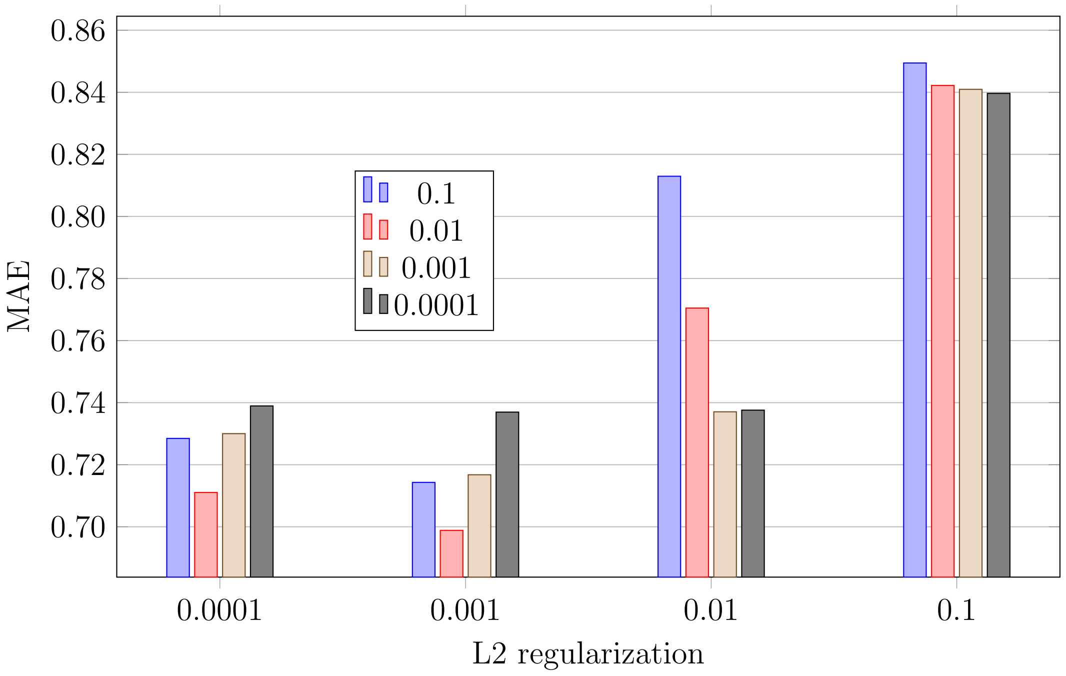


Figure 4.10: Evaluation of *Experiment IV*. The bars represent each value of learning rate tested.

4.5.6 Experiment VI

Experiment VI is performed to evaluate our method against regular COFILS and several collaborative filtering state-of-the-art techniques. The proposed approach, obtain the best overall result alongside AutoRec, compared with the other selected state-of-the-art methods, performing better than classical techniques, such as RBM-CF, SVD++ and BPMF.

Besides that, when in comparison with the baseline, DL-COFILS is able to outperform COFILS by 0.717% and 0.790% for MAE and RMSE respectively. The proposed method scores the best RMSE among all the others techniques, while AutoRec scores the best MAE. Table 4.2 shows the obtained best results for this experiment.

Table 4.2: Top results on different model-based collaborative filtering techniques in MovieLens 100k (the best ones are in bold).

Technique	**MAE**	**RMSE**	**Features**
Deep Learning COFILS	0.697 (+0.71%)	**0.885** (+0.79%)	50u, 50i
Baseline COFILS	0.702 (0.00%)	0.892 (0.00%)	10
Slope One	0.737 (-4.99%)	0.937 (-5.04%)	-
Regularized SVD	0.768 (-9.40%)	0.989 (-10.87%)	14
Improved Regularized SVD	0.757 (-7.83%)	0.954 (-6.95%)	38
SVD++	0.712 (-1.42%)	0.903 (-1.23%)	50
NMF	0.743 (-5.84%)	0.944 (-5.83%)	800
BPMF	0.705 (-0.43%)	0.901 (-1.01%)	10
RBM-CF	0.732 (-4.27%)	0.936 (-4.93%)	900
AutoRec	**0.696** (+0.85%)	0.887 (+0.56%)	500
Mean Field	0.707 (-0.71%)	0.903 (-1.23%)	2

4.5.7 Experiment VII

Once the proposed method is evaluated for one data set and is ensured that the technique is feasible, the *Experiment VII* is conducted for a similar evaluation in other three data sets.

First one is a larger version of MovieLens data sets. This data set is highly similar than the small one, but larger and more sparse. The proposed method outperforms COFILS just like in the first data set, with 0.90% and 1.18% for MAE and RMSE respectively. Note also that DL-COFILS is able to outperform AutoRec this time, by 0.15% for MAE and 0.71% for RMSE.

Regarding the others collaborative filtering techniques, DL-COFILS scores the best overall value for RMSE, outperforming all techniques, and the second best for

MAE, only being outperformed by BPMF. Table 4.3 shows the results for MovieLens 1M data set.

Table 4.3: Top results on different model-based collaborative filtering techniques in MovieLens 1M (the best ones are in bold).

Technique	**MAE**	**RMSE**	**Features**
Deep Learning COFILS	0.661 (+0.90%)	**0.838** (+1.18%)	50u, 50i
Baseline COFILS	0.667 (0.00%)	0.848 (0.00%)	13
Slope One	0.710 (-6.45%)	0.900 (-6.13%)	-
Regularized SVD	0.752 (-12.74%)	0.960 (-13.21%)	38
Improved Regularized SVD	0.719 (-7.80%)	0.907 (-6.96%)	100
SVD++	0.668 (-0.14%)	0.856 (-0.94%)	6
NMF	0.720 (-7.94%)	0.912 (-7.55%)	800
BPMF	**0.657** (+1.50%)	0.840 (+0.94%)	10
RBM-CF	0.683 (-2.40%)	0.872 (-2.83%)	800
AutoRec	0.662 (+0.74%)	0.844 (+0.47%)	500
Mean Field	0.671 (-0.60%)	0.856 (-0.94%)	2

In the third of the other three data sets, a movie one as the previous, but much more sparse and with different range of ratings. COFILS is, again, outperformed by the proposed method, this time, by a slightly smaller margin in comparison with MovieLens 1M: 0.84% and 1.02% for MAE and RMSE respectively. Another interesting result is that AutoRec is outperformed as well by 4.26% for MAE and 2.94% for RMSE.

Regarding the other state-of-the-art techniques, DL-COFILS outperforms also RBM-CF and BPMF, but is unable to overcome SVD++, which achieves the best overall result for this data set. As the table 4.4 shows, DL-COFILS is among the top 3.

Last data set is from a different domain and sightly larger than the previous, but has the same rating range than MovieLens 100k. Again, DL-COFILS indicates to be better than regular COFILS outperforming the technique with 5.59% and 2.95% for MAE and RMSE respectively. However, the proposed technique is outperformed by AutoRec by 2.23% for MAE and 0.43% for RMSE.

Note that, for this data set, the improvement over COFILS was much larger than in the other three, indicating that different characteristics in music domain data sets can favour DL-COFILS when in comparison with COFILS.

In comparison with the other methods, DL-COFILS scored the overall second best value for RMSE, and third best value for MAE, outperforming several state-of-the-art techniques such as RBM-CF and BPMF. Table 4.5 shows the proposed method performance against the baseline and the state-of-the-art techniques.

In order to demonstrate the stability of the proposed method against the com-

Table 4.4: Top results on different model-based collaborative filtering techniques in MovieTweetings 10k (the best ones are in bold).

Technique	**MAE**	**RMSE**	**Features**
Deep Learning COFILS	1.304 (+0.84%)	1.747 (+1.02%)	500u, 200i
Baseline COFILS	1.315 (0.00%)	1.765 (0.00%)	600
Slope One	1.465 (-11.41%)	1.920 (-8.78%)	-
Regularized SVD	3.927 (-198.63%)	5.191 (-194.11%)	1
Improved Regularized SVD	1.291 (+1.83%)	1.705 (+3.40%)	12
SVD++	**1.211** (+7.91%)	**1.623** (+8.05%)	10
NMF	2.480 (-88.59%)	3.620 (-105.10%)	150
BPMF	1.847 (-40.46%)	2.412 (-36.66%)	10
RBM-CF	1.436 (-9.20%)	1.845 (-4.53%)	1000
AutoRec	1.362 (-3.57%)	1.800 (-1.98%)	200
Mean Field	1.359 (-3.35%)	1.791 (-1.47%)	2

Table 4.5: Top results on different model-based collaborative filtering techniques in Yahoo Music (the best ones are in bold).

Technique	**MAE**	**RMSE**	**Features**
Deep Learning COFILS	0.895 (+5.59%)	1.152 (+2.95%)	1100u, 100i
Baseline COFILS	0.948 (0.00%)	1.187 (0.00%)	18
Slope One	0.958 (-1.05%)	1.230 (-3.62%)	-
Regularized SVD	0.950 (-0.21%)	1.213 (-2.19%)	10
Improved Regularized SVD	1.056 (-11.39%)	1.265 (-6.57%)	95
SVD++	0.883 (+6.86%)	1.173 (+1.18%)	8
NMF	0.968 (-2.11%)	1.240 (-4.46%)	850
BPMF	0.956 (-0.84%)	1.319 (-11.12%)	10
RBM-CF	0.963 (-1.58%)	1.249 (-5.22%)	800
AutoRec	**0.875** (+7.70%)	**1.147** (+3.37%)	500
Mean Field	0.898 (+5.27%)	1.162 (+2.11%)	4

pared collaborative filtering techniques, the NMAE is calculated for each technique for each data set. As shown in figure 4.13, DL-COFILS achieved the best NMAE alongside AutoRec for MovieLens 100k, the second best for MovieLens 1M and MovieTweetings and the third best for R3 Yahoo! Music data sets. These results indicate that DL-COFILS is always among the best techniques in the compared data sets, even if changing the data domain or ratings range.

The most stable technique is SVD++. However, this method has the advantage of using implicit feedback knowing in advance which movies each user rates (KOREN, 2008). Nevertheless, note that DL-COFILS still outperforms SVD++ in two data sets even without using any kind of implicit data.

While MAE and RMSE shows that the proposed method can be competitive against others state-of-the-art techniques, they are not enough to show the quality

Table 4.6: Confusion matrix of the proposed algorithm using no normalization and 1100 latent variables for the user and 100 for the item.

		Prediction				
		1	2	3	4	5
Actual	1	35.7%	44.9%	15.8%	3.3%	0.2%
	2	6.9%	45.3%	39.4%	8.0%	0.4%
	3	2.8%	27.8%	51.5%	17.1%	0.9%
	4	1.4%	15.5%	44.9%	34.8%	3.4%
	5	1.2%	9.5%	26.4%	42.0%	20.9%

of the recommendation that DL-COFILS can bring. This way, the confusion matrix of DL-COFILS (table 4.6) is computed in order to evaluate the quality of recommendation delivered by the method. Note that the predictions of the algorithm are rounded to make the confusion matrix possible to be made.

Note that the ratings 2 and 4 are often misclassified, which may be explained by the low percentage of this ratings in comparison with others. Moreover, even with 1 and 5 being more frequent than 2, 3, 4, these ratings are, respectively, the third and first most misclassified by the proposed approach within this data set. This leaves 3 as the least misclassified rating.

Although DL-COFILS is able to achieve one of the best results among the other techniques, collaborative filtering problem seen as supervised learning problem is still not able to get results close to the classic supervised learning problems. Confusion matrix shows accuracy rates are very low, demonstrating that despite the Autoencoder be able to improve the representativeness of latent variables, this is still not enough to achieve far superior results compared to classical collaborative filtering techniques. A major challenge is to improve the accuracy with either better latent variable extraction techniques or other regressors.

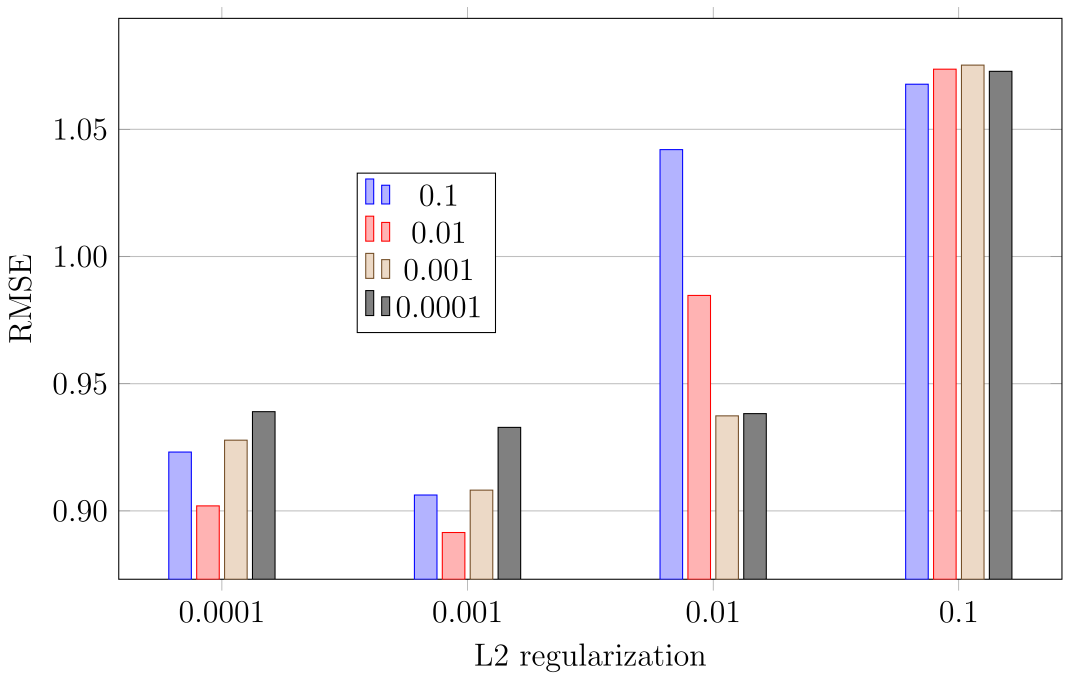


Figure 4.11: Evaluation of *Experiment IV*. The bars represent each value of learning rate tested.

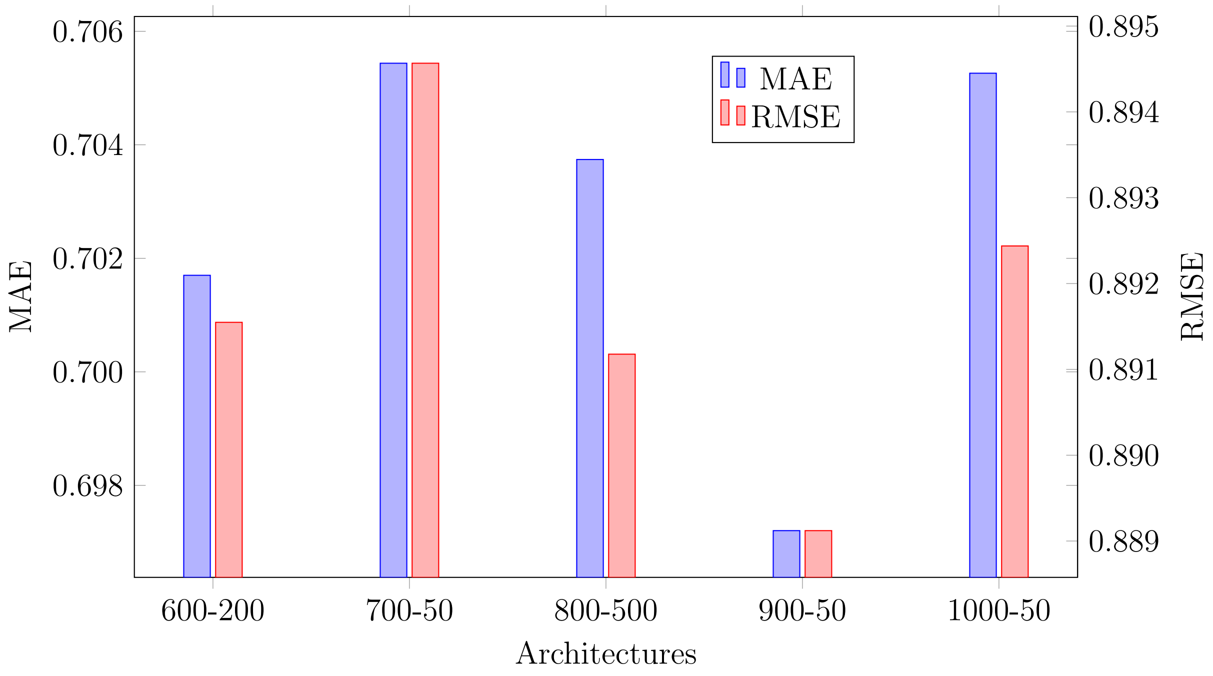


Figure 4.12: Evaluation of *Experiment V* using denosing autoencoder, no matrix normalization, target normalization by item mean.

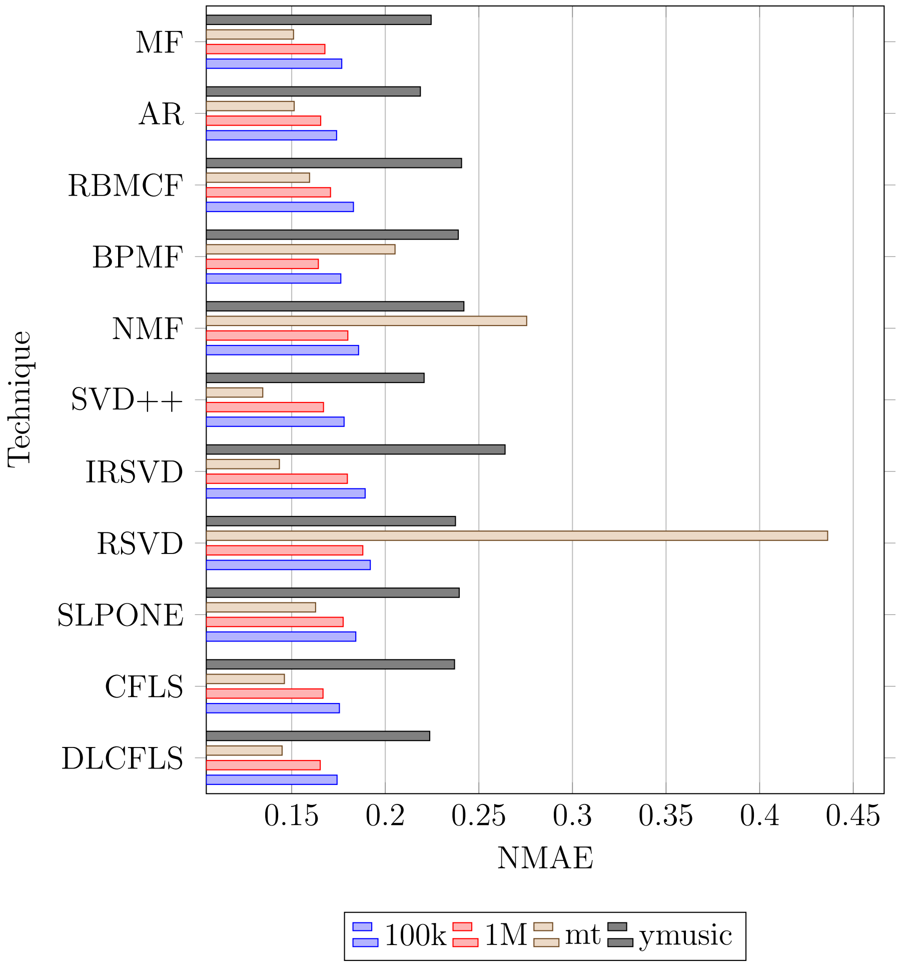


Figure 4.13: NMAE for each technique in each data set.

Chapter 5

Conclusions

This chapter discusses the final considerations of this work, the main contributions and possible future work.

5.1 Work considerations

COFILS is a methodology for transforming the collaborative filtering problem into a standard supervised learning problem. One problem of COFILS is its dependency on SVD that cannot extract non-linear representations from data. Recently, deep learning methods have arisen as useful tools for learning high level and complex representations that standard linear methods, such as SVD, can not capture.

In this work, we approach COFILS latent extraction issue. We propose switching SVD to a Stacked Denoising Autoencoder, a Deep Network with local denoising criterion. We denominate our method as DL-COFILS.

In order to evaluate our proposal, we did experiments with four well-established data sets: MovieLens 100k, MovieLens 1M, MovieTweetings and R3 Yahoo! Music. The experiments were divided into validation and comparison. In the validation phase, we conducted several experiments in MovieLens 100k with the aim of choose the model standard parameters. The parameters chosen were carried over for the comparison phase, were only few parameters were evaluated for each data set.

Our results indicate that DL-COFILS outperforms COFILS for all data sets experimented. For the MovieLens 100k, MovieLens 1M and MovieTweetings data sets, DL-COFILS slightly reduces the MAE error by 0.7%, 0.9% and 0.8%, respectively. However, COFILS was one of the best predictors for MovieLens, what justifies this reduction. Regarding the Yahoo! Music, DL-COFILS reduces the MAE error by 5.9% compared to COFILS.

5.2 Contributions

We have found that while Cross Entropy is the most used Autoencoder cost function regarding different tasks, squared error performed better for this problem, achieving superior results. No surprisingly, Denoising Autoencoder was able to outperform the ordinary one for this problem as well. Using low L2 regularization values seems to significantly improve the generated features quality.

Another issue is the performance of DL-COFILS compared to Autorec that is an autoencoder applied directly to the data. We found that, both approaches present promising results. However, DL-COFILS is not dependent on one single learning method as Autorec. Even so, both methods indicate that deep learning can overcome linear techniques even in collaborative filtering, where linear techniques are well known and well stabilized as state-of-the-art techniques. Its also important to note that, COFILS is a robust methodology and can be improved regardless of new approaches that emerge in the future.

5.3 Limitations and future work

This section describes potential areas for improvement of the proposed technique and solutions for possible limitations in the application of the technique.

Once product popularity and perception are constantly changing; and user preference either, one research direction is to explore this temporal dynamics for building more accurate models. Data sets such as MovieLens and MovieTweetings have timestamp attributes that can be exploited.

Another research direction is to divide the data into clusters and applying DL-COFILS for each cluster. This approach can scale very well for big data. Thus, it can be learned specialized features from clusters instead of general features.

Finally, since SVD++ achieved the best overall result using implicit feedback, we could improve the proposed method accuracy by incorporating this source of information into the Autoencoder cost function or during the mapping step of the methodology. This could raise the robustness of the extracted features or improve Random Forest generalization.

Bibliography

ADOMAVICIUS, G., TUZHILIN, A., 2005, “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”, *IEEE Trans. on Knowl. and Data Eng.*, v. 17, n. 6 (jun.), pp. 734–749. ISSN: 1041-4347. doi: 10.1109/TKDE.2005.99. Disponível em: <http://dx.doi.org/10.1109/TKDE.2005.99>.

ANSARI, A., ESSEGAIER, S., KOHLI, R., 2000, “Internet Recommendation Systems”, *Journal of Marketing Research*, v. 37, n. 3 (ago.), pp. 363–375. ISSN: 0022-2437. doi: 10.1509/jmkr.37.3.363.18779. Disponível em: <http://dx.doi.org/10.1509/jmkr.37.3.363.18779>.

BALABANOVIĆ, M., SHOHAM, Y., 1997, “Fab: Content-based, Collaborative Recommendation”, *Commun. ACM*, v. 40, n. 3 (mar.), pp. 66–72. ISSN: 0001-0782. doi: 10.1145/245108.245124. Disponível em: <http://doi.acm.org/10.1145/245108.245124>.

BASU, C., HIRSH, H., COHEN, W., 1998, “Recommendation As Classification: Using Social and Content-based Information in Recommendation”. In: *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI ’98/IAAI ’98, pp. 714–720, Menlo Park, CA, USA. American Association for Artificial Intelligence. ISBN: 0-262-51098-7. Disponível em: <http://dl.acm.org/citation.cfm?id=295240.295795>.

BENGIO, Y., 2009, “Learning Deep Architectures for AI”, *Found. Trends Mach. Learn.*, v. 2, n. 1 (jan.), pp. 1–127. ISSN: 1935-8237. doi: 10.1561/2200000006. Disponível em: <http://dx.doi.org/10.1561/2200000006>.

BENGIO, Y., COURVILLE, A., VINCENT, P., 2013, “Representation Learning: A Review and New Perspectives”, *IEEE Trans. Pattern Anal. Mach. Intell.*, v. 35, n. 8 (ago.), pp. 1798–1828. ISSN: 0162-8828. doi: 10.1109/TPAMI.2013.50. Disponível em: <http://dx.doi.org/10.1109/TPAMI.2013.50>.

BRAIDA, F., MELLO, C. E., PASINATO, M. B., et al., 2015, "Transforming Collaborative Filtering into Supervised Learning", *Expert Syst. Appl.*, v. 42, n. 10 (jun.), pp. 4733–4742. ISSN: 0957-4174. doi: 10.1016/j.eswa.2015.01.023. Disponível em: <`http://dx.doi.org/10.1016/j.eswa.2015.01.023`>.

BREESE, J. S., HECKERMAN, D., KADIE, C., 1998, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering". In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pp. 43–52, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. ISBN: 1-55860-555-X. Disponível em: <`http://dl.acm.org/citation.cfm?id=2074094.2074100`>.

BURKE, R., 2007, "The Adaptive Web". Springer-Verlag, cap. Hybrid Web Recommender Systems, pp. 377–408, Berlin, Heidelberg. ISBN: 978-3-540-72078-2. Disponível em: <`http://dl.acm.org/citation.cfm?id=1768197.1768211`>.

CHEN, P.-Y., WU, S.-Y., YOON, J., 2004, "The impact of online recommendations and consumer feedback on sales", *ICIS 2004 Proceedings*, p. 58.

CLAYPOOL, M., GOKHALE, A., MIRANDA, T., et al., 1999. "Combining Content-Based and Collaborative Filters in an Online Newspaper". .

CONDLIFF, M. K., LEWIS, D. D., MADIGAN, D., 1999, "Bayesian Mixed-Effects Models for Recommender Systems". In: *In ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation.*

DENG, L., YU, D., 2014, "Deep Learning: Methods and Applications", *Found. Trends Signal Process.*, v. 7, n. 3&#8211;4 (jun.), pp. 197–387. ISSN: 1932-8346. doi: 10.1561/2000000039. Disponível em: <`http://dx.doi.org/10.1561/2000000039`>.

DOOMS, S., DE PESSEMIER, T., MARTENS, L., 2013, "MovieTweetings: a Movie Rating Dataset Collected From Twitter". In: *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013.*

EKSTRAND, M. D., RIEDL, J. T., KONSTAN, J. A., 2011, "Collaborative Filtering Recommender Systems", *Found. Trends Hum.-Comput. Interact.*, v. 4, n. 2 (fev.), pp. 81–173. ISSN: 1551-3955. doi: 10.1561/1100000009. Disponível em: <`http://dx.doi.org/10.1561/1100000009`>.

FUNK, S., 2011, "Netflix update: Try this at home, 2006", *URL http://sifter.org/˜ simon/journal/20061211. html.*

GE, M., DELGADO-BATTENFELD, C., JANNACH, D., 2010, "Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity". In: *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pp. 257–260, New York, NY, USA. ACM. ISBN: 978-1-60558-906-0. doi: 10.1145/1864708.1864761. Disponível em: <`http://doi.acm.org/10.1145/1864708.1864761`>.

GEORGIEV, K., NAKOV, P., 2013, "A non-IID Framework for Collaborative Filtering with Restricted Boltzmann Machines". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, v. 28, *JMLR Workshop and Conference Proceedings*, pp. 1148–1156. JMLR.org. Disponível em: <`http://jmlr.org/proceedings/papers/v28/georgiev13.html`>.

GOLDBERG, D., NICHOLS, D., OKI, B. M., et al., 1992, "Using Collaborative Filtering to Weave an Information Tapestry", *Commun. ACM*, v. 35, n. 12 (dez.), pp. 61–70. ISSN: 0001-0782. doi: 10.1145/138859.138867. Disponível em: <`http://doi.acm.org/10.1145/138859.138867`>.

HARPER, F. M., KONSTAN, J. A., 2015, "The MovieLens Datasets: History and Context", *ACM Trans. Interact. Intell. Syst.*, v. 5, n. 4 (dez.), pp. 19:1–19:19. ISSN: 2160-6455. doi: 10.1145/2827872. Disponível em: <`http://doi.acm.org/10.1145/2827872`>.

HERLOCKER, J. L., KONSTAN, J. A., TERVEEN, L. G., et al., 2004, "Evaluating Collaborative Filtering Recommender Systems", *ACM Trans. Inf. Syst.*, v. 22, n. 1 (jan.), pp. 5–53. ISSN: 1046-8188. doi: 10.1145/963770.963772. Disponível em: <`http://doi.acm.org/10.1145/963770.963772`>.

HINTON, G. E., SALAKHUTDINOV, R. R., 2006, "Reducing the Dimensionality of Data with Neural Networks", *Science*, v. 313, n. 5786, pp. 504–507. ISSN: 0036-8075. doi: 10.1126/science.1127647. Disponível em: <`http://science.sciencemag.org/content/313/5786/504`>.

HINTON, G. E., OSINDERO, S., TEH, Y.-W., 2006, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Comput.*, v. 18, n. 7 (jul.), pp. 1527–1554. ISSN: 0899-7667. doi: 10.1162/neco.2006.18.7.1527. Disponível em: <`http://dx.doi.org/10.1162/neco.2006.18.7.1527`>.

HO, T. K., 1995, “Random Decision Forests”. In: *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR ’95, pp. 278–, Washington, DC, USA. IEEE Computer Society. ISBN: 0-8186-7128-9. Disponível em: <http://dl.acm.org/citation.cfm?id=844379.844681>.

HSIN CHEN, Y., GEORGE, E. I., 2002. “A Bayesian Model for Collaborative Filtering”. .

HSU, S. H., WEN, M.-H., LIN, H.-C., et al., 2007, “AIMED: A Personalized TV Recommendation System”. In: *Proceedings of the 5th European Conference on Interactive TV: A Shared Experience*, EuroITV’07, pp. 166–174, Berlin, Heidelberg. Springer-Verlag. ISBN: 978-3-540-72558-9. Disponível em: <http://dl.acm.org/citation.cfm?id=1763017.1763040>.

KOREN, Y., 2008, “Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model”. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’08, pp. 426–434, New York, NY, USA. ACM. ISBN: 978-1-60558-193-4. doi: 10.1145/1401890.1401944. Disponível em: <http://doi.acm.org/10.1145/1401890.1401944>.

KOREN, Y., BELL, R., VOLINSKY, C., 2009, “Matrix Factorization Techniques for Recommender Systems”, *Computer*, v. 42, n. 8 (ago.), pp. 30–37. ISSN: 0018-9162. doi: 10.1109/MC.2009.263. Disponível em: <http://dx.doi.org/10.1109/MC.2009.263>.

LANGSETH, H., NIELSEN, T. D., 2015, “Scalable learning of probabilistic latent models for collaborative filtering”, *Decision Support Systems*, v. 74, pp. 1 – 11. ISSN: 0167-9236. doi: http://dx.doi.org/10.1016/j.dss.2015.03.006. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167923615000603>.

LANGSETH, H., NIELSEN, T. D., 2012, “A latent model for collaborative filtering”, *International Journal of Approximate Reasoning*, v. 53, n. 4, pp. 447 – 466. ISSN: 0888-613X. doi: http://dx.doi.org/10.1016/j.ijar.2011.11.002. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0888613X11001654>.

LE, Q. V., RANZATO, M., MONGA, R., et al., 2012, “Building high-level features using large scale unsupervised learning.” In: *ICML*. icml.cc / Omnipress. Disponível em: <http://dblp.uni-trier.de/db/conf/icml/icml2012.html#LeRMDCCDN12>.

LECUN, Y., BENGIO, Y., 1998, "The Handbook of Brain Theory and Neural Networks". MIT Press, cap. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258, Cambridge, MA, USA. ISBN: 0-262-51102-9. Disponível em: <http://dl.acm.org/citation.cfm?id=303568.303704>.

LEE, D. D., SEUNG, H. S., 2000, "Algorithms for Non-negative Matrix Factorization". In: Leen, T. K., Dietterich, T. G., Tresp, V. (Eds.), *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pp. 556–562. MIT Press. Disponível em: <http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization>.

LEE, H., GROSSE, R., RANGANATH, R., et al., 2009a, "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations". In: *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 609–616, New York, NY, USA, a. ACM. ISBN: 978-1-60558-516-1. doi: 10.1145/1553374.1553453. Disponível em: <http://doi.acm.org/10.1145/1553374.1553453>.

LEE, H., LARGMAN, Y., PHAM, P., et al., 2009b, "Unsupervised Feature Learning for Audio Classification Using Convolutional Deep Belief Networks". In: *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, NIPS'09, pp. 1096–1104, USA, b. Curran Associates Inc. ISBN: 978-1-61567-911-9. Disponível em: <http://dl.acm.org/citation.cfm?id=2984093.2984217>.

LEMIRE, D., MACLACHLAN, A., 2005, "Slope One Predictors for Online Rating-Based Collaborative Filtering". In: Kargupta, H., Srivastava, J., Kamath, C., et al. (Eds.), *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005, Newport Beach, CA, USA, April 21-23, 2005*, pp. 471–475. SIAM. doi: 10.1137/1.9781611972757.43. Disponível em: <http://dx.doi.org/10.1137/1.9781611972757.43>.

LI, S., KAWALE, J., FU, Y., 2015, "Deep Collaborative Filtering via Marginalized Denoising Auto-encoder". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pp. 811–820, New York, NY, USA. ACM. ISBN: 978-1-4503-3794-6. doi: 10.1145/2806416.2806527. Disponível em: <http://doi.acm.org/10.1145/2806416.2806527>.

MCCULLOCH, W., PITTS, W., 1943, "A Logical Calculus of Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, v. 5, pp. 127–147.

MCLAUGHLIN, M. R., HERLOCKER, J. L., 2004, "A Collaborative Filtering Algorithm and Evaluation Metric That Accurately Model the User Experience". In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pp. 329–336, New York, NY, USA. ACM. ISBN: 1-58113-881-4. doi: 10.1145/1008992.1009050. Disponível em: <http://doi.acm.org/10.1145/1008992.1009050>.

MELVILLE, P., MOONEY, R. J., NAGARAJAN, R., 2002, "Content-boosted Collaborative Filtering for Improved Recommendations". In: *Eighteenth National Conference on Artificial Intelligence*, pp. 187–192, Menlo Park, CA, USA. American Association for Artificial Intelligence. ISBN: 0-262-51129-0. Disponível em: <http://dl.acm.org/citation.cfm?id=777092.777124>.

MITCHELL, T. M., 1997, *Machine Learning.* 1 ed. New York, NY, USA, McGraw-Hill, Inc. ISBN: 0070428077, 9780070428072.

NICHOLAS, I. S. C., NICHOLAS, C. K., 1999, "Combining Content and Collaboration in Text Filtering". In: *In Proceedings of the IJCAI'99 Workshop on Machine Learning for Information Filtering*, pp. 86–91.

O'MAHONY, M. P., SMYTH, B., 2009, "Learning to Recommend Helpful Hotel Reviews". In: *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pp. 305–308, New York, NY, USA. ACM. ISBN: 978-1-60558-435-5. doi: 10.1145/1639714.1639774. Disponível em: <http://doi.acm.org/10.1145/1639714.1639774>.

O'MAHONY, M. P., CUNNINGHAM, P., SMYTH, B., 2010, "An Assessment of Machine Learning Techniques for Review Recommendation". In: *Proceedings of the 20th Irish Conference on Artificial Intelligence and Cognitive Science*, AICS'09, pp. 241–250, Berlin, Heidelberg. Springer-Verlag. ISBN: 3-642-17079-X, 978-3-642-17079-9. Disponível em: <http://dl.acm.org/citation.cfm?id=1939047.1939075>.

OUYANG, Y., LIU, W., RONG, W., et al., 2014, "Autoencoder-Based Collaborative Filtering". In: Loo, C. K., Yap, K. S., Wong, K. W., et al. (Eds.), *Neural Information Processing - 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings,* *Part III*, v. 8836, *Lecture Notes in Computer Science*, pp. 284–291. Springer. doi: 10.1007/978-3-319-12643-2_35. Disponível em: <`http://dx.doi.org/10.1007/978-3-319-12643-2_35`>.

PATEREK, A., 2007, "Improving regularized singular value decomposition for collaborative filtering". In: *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 39–42. Disponível em: <`http://serv1.ist.psu.edu:8080/viewdoc/summary;jsessionid=CBC0A80E61E800DE518520F9469B2FD1?doi=10.1.1.96.7652`>.

PAZZANI, M. J., 1999, "A Framework for Collaborative, Content-Based and Demographic Filtering", *Artif. Intell. Rev.*, v. 13, n. 5-6 (dez.), pp. 393–408. ISSN: 0269-2821. doi: 10.1023/A:1006544522159. Disponível em: <`http://dx.doi.org/10.1023/A:1006544522159`>.

QUINLAN, J. R., 1986, "Induction of Decision Trees", *Mach. Learn.*, v. 1, n. 1 (mar.), pp. 81–106. ISSN: 0885-6125. doi: 10.1023/A:1022643204877. Disponível em: <`http://dx.doi.org/10.1023/A:1022643204877`>.

QUINLAN, J. R., 1993, *C4.5: Programs for Machine Learning.* San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. ISBN: 1-55860-238-0.

RANZATO, M. A., SZUMMER, M., 2008, "Semi-supervised Learning of Compact Document Representations with Deep Networks". In: *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 792–799, New York, NY, USA. ACM. ISBN: 978-1-60558-205-4. doi: 10.1145/1390156.1390256. Disponível em: <`http://doi.acm.org/10.1145/1390156.1390256`>.

RICCI, F., ROKACH, L., SHAPIRA, B., et al., 2010, *Recommender Systems Handbook.* 1st ed. New York, NY, USA, Springer-Verlag New York, Inc. ISBN: 0387858199, 9780387858197.

RICH, E., 1998, "Readings in Intelligent User Interfaces". Morgan Kaufmann Publishers Inc., cap. User Modeling via Stereotypes, pp. 329–342, San Francisco, CA, USA. ISBN: 1-55860-444-8. Disponível em: <`http://dl.acm.org/citation.cfm?id=286013.286035`>.

ROSENBLATT, F., 1958, "The perceptron: a probabilistic model for information storage and organization in the brain", *Psychological Review*, v. 65, n. 6 (nov.), pp. 386–408.

SALAKHUTDINOV, R., HINTON, G., 2007, "Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes". In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pp. 1249–1256, USA. Curran Associates Inc. ISBN: 978-1-60560-352-0. Disponível em: <http://dl.acm.org/citation.cfm?id=2981562.2981719>.

SALAKHUTDINOV, R., HINTON, G., 2009, "Semantic Hashing", *Int. J. Approx. Reasoning*, v. 50, n. 7 (jul.), pp. 969–978. ISSN: 0888-613X. doi: 10.1016/j.ijar.2008.11.006. Disponível em: <http://dx.doi.org/10.1016/j.ijar.2008.11.006>.

SALAKHUTDINOV, R., MNIH, A., 2008, "Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo". In: *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 880–887, New York, NY, USA. ACM. ISBN: 978-1-60558-205-4. doi: 10.1145/1390156.1390267. Disponível em: <http://doi.acm.org/10.1145/1390156.1390267>.

SALAKHUTDINOV, R., MNIH, A., 2007, "Probabilistic Matrix Factorization". In: Platt, J. C., Koller, D., Singer, Y., et al. (Eds.), *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pp. 1257–1264. Curran Associates, Inc. Disponível em: <http://papers.nips.cc/paper/3208-probabilistic-matrix-factorization>.

SALAKHUTDINOV, R., MNIH, A., HINTON, G., 2007, "Restricted Boltzmann Machines for Collaborative Filtering". In: *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 791–798, New York, NY, USA. ACM. ISBN: 978-1-59593-793-3. doi: 10.1145/1273496.1273596. Disponível em: <http://doi.acm.org/10.1145/1273496.1273596>.

SAMUEL, A. L., 1959, "Some Studies in Machine Learning Using the Game of Checkers", *IBM J. Res. Dev.*, v. 3, n. 3 (jul.), pp. 210–229. ISSN: 0018-8646. doi: 10.1147/rd.33.0210. Disponível em: <http://dx.doi.org/10.1147/rd.33.0210>.

SARWAR, B., KARYPIS, G., KONSTAN, J., et al., 2001, "Item-based Collaborative Filtering Recommendation Algorithms". In: *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pp.

285–295, New York, NY, USA. ACM. ISBN: 1-58113-348-0. doi: 10.1145/371920.372071. Disponível em: <`http://doi.acm.org/10.1145/371920.372071`>.

SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., et al., 2000, "Application of Dimensionality Reduction in Recommender System – A Case Study". In: *IN ACM WEBKDD WORKSHOP*.

SCHAFER, J. B., FRANKOWSKI, D., HERLOCKER, J., et al., 2007, "The Adaptive Web". Springer-Verlag, cap. Collaborative Filtering Recommender Systems, pp. 291–324, Berlin, Heidelberg. ISBN: 978-3-540-72078-2. Disponível em: <`http://dl.acm.org/citation.cfm?id=1768197.1768208`>.

SCHEIN, A. I., POPESCUL, A., UNGAR, L. H., et al., 2002, "Methods and Metrics for Cold-start Recommendations". In: *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pp. 253–260, New York, NY, USA. ACM. ISBN: 1-58113-561-0. doi: 10.1145/564376.564421. Disponível em: <`http://doi.acm.org/10.1145/564376.564421`>.

SEDHAIN, S., MENON, A. K., SANNER, S., et al., 2015, "AutoRec: Autoencoders Meet Collaborative Filtering". In: *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pp. 111–112, New York, NY, USA. ACM. ISBN: 978-1-4503-3473-0. doi: 10.1145/2740908.2742726. Disponível em: <`http://doi.acm.org/10.1145/2740908.2742726`>.

SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al., 2014, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *J. Mach. Learn. Res.*, v. 15, n. 1 (jan.), pp. 1929–1958. ISSN: 1532-4435. Disponível em: <`http://dl.acm.org/citation.cfm?id=2627435.2670313`>.

TAKÁCS, G., PILÁSZY, I., NÉMETH, B., et al., 2008, "Matrix Factorization and Neighbor Based Algorithms for the Netflix Prize Problem". In: *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pp. 267–274, New York, NY, USA. ACM. ISBN: 978-1-60558-093-7. doi: 10.1145/1454008.1454049. Disponível em: <`http://doi.acm.org/10.1145/1454008.1454049`>.

TAN, P.-N., STEINBACH, M., KUMAR, V., 2005, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc. ISBN: 0321321367.

TRAN, T., COHEN, R., 2000, "Hybrid recommender systems for electronic commerce". In: *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press.*

TRUYEN, T. T., PHUNG, D. Q., VENKATESH, S., 2009, "Ordinal Boltzmann Machines for Collaborative Filtering". In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pp. 548–556, Arlington, Virginia, United States. AUAI Press. ISBN: 978-0-9749039-5-8. Disponível em: <`http://dl.acm.org/citation.cfm?id=1795114.1795178`>.

UNGAR, L., FOSTER, D., ANDRE, E., et al., 1998, "Clustering Methods for Collaborative Filtering". AAAI Press.

VAPNIK, V. N., 1999. "The Nature of Statistical Learning Theory". .

VINCENT, P., LAROCHELLE, H., BENGIO, Y., et al., 2008, "Extracting and Composing Robust Features with Denoising Autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 1096–1103, New York, NY, USA. ACM. ISBN: 978-1-60558-205-4. doi: 10.1145/1390156.1390294. Disponível em: <`http://doi.acm.org/10.1145/1390156.1390294`>.

VINCENT, P., LAROCHELLE, H., LAJOIE, I., et al., 2010, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", *J. Mach. Learn. Res.*, v. 11 (dez.), pp. 3371–3408. ISSN: 1532-4435. Disponível em: <`http://dl.acm.org/citation.cfm?id=1756006.1953039`>.

VISHNUBHOTLA, S., FERNANDEZ, R., RAMABHADRAN, B., 2010, "An autoencoder neural-network based low-dimensionality approach to excitation modeling for HMM-based text-to-speech." In: *ICASSP*, pp. 4614–4617. IEEE. ISBN: 978-1-4244-4296-6. Disponível em: <`http://dblp.uni-trier.de/db/conf/icassp/icassp2010.html#VishnubhotlaFR10`>.

WANG, H., WANG, N., YEUNG, D.-Y., 2015, "Collaborative Deep Learning for Recommender Systems". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 1235–1244, New York, NY, USA. ACM. ISBN: 978-1-4503-3664-2. doi: 10.1145/2783258.2783273. Disponível em: <`http://doi.acm.org/10.1145/2783258.2783273`>.

ZHANG, S., WANG, W., FORD, J., et al., 2006, “Learning from Incomplete Ratings Using Non-negative Matrix Factorization”. In: Ghosh, J., Lambert, D., Skillicorn, D. B., et al. (Eds.), *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, pp. 549–553. SIAM. doi: 10.1137/1.9781611972764.58. Disponível em: <http://dx.doi.org/10.1137/1.9781611972764.58>.

ZHOU, R., KHEMMARAT, S., GAO, L., 2010, “The Impact of YouTube Recommendation System on Video Views”. In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC ’10, pp. 404–410, New York, NY, USA. ACM. ISBN: 978-1-4503-0483-2. doi: 10.1145/1879141.1879193. Disponível em: <http://doi.acm.org/10.1145/1879141.1879193>.