Friedrich-Schiller-Universität Jena
Physikalisch-Astronomische Fakultät

Restricted Boltzmann Machines for Collaborative Filtering

REPORT

for the lecture "Computational Physics III - Machine Learning"

submitted by Markus Pawellek

Student Number:	144645
E-Mail Address:	markuspawellek@gmail.com

Jena, February 11, 2019

Contents

RESTRICTED BOLTZMANN MACHINES FOR COLLABORATIVE FILTERING

Markus Pawellek
markuspawellek@gmail.com

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1 Introduction

On October 2, 2006 Netflix announced the start of the so-called "Netflix Prize" competition. It was an open competition aimed to find the best algorithm for collaborative filtering to predict user ratings for movies based on previous ratings without any other information about the users or the movies. Netflix's current algorithm "Cinematch" introduced the threshold that had to be bested. For this Netflix provided a training dataset with over 100,000,000 ratings that more than 480,000 users gave about 17,000 movies. The complete competition lasted over three years and included two "Progress Prizes" in the years 2007 and 2008. Finally on September 18, 2009 Netflix announced the winner-team of the $\$\,1,000,000$ "Grand Prize" with its last submission 24 minutes before the conclusion of the contest. The solution of the winner-team "BellKor's Pragmatic Chaos" was based on the work of [13] which used restricted Boltzmann machines (RBMs) to efficiently predict the user ratings. With this algorithm the winner-team was able to achieve an improvement over Netflix's current algorithm by $10.05\,\%$. [9, 10, 13]

In [13] different RBMs were applied to the task of collaborative filtering for the first time. Even with the basic ideas presented in this paper one could achieve an error rate that was well over $6\,\%$ better than the score of Netflix's own system. Additionally, in comparison to other proposed solutions RBMs were able to deal much more efficiently with big datasets, like the ones given by Netflix for the competition. [13]

RBMs have found their application in other machine learning topics as well. In [8] an RBM is introduced in topic modelling as a more precise alternative to the common latent Dirichlet allocation. In [6] discriminative RBMs are used for classification in a self-contained framework and in [4] they are even used as basic building blocks in much bigger deep neural networks (DNNs) to efficiently reduce the dimensionality of some given data. According to [7] RBMs play an important role in the mathematical theory of machine learning and are not fully investigated, yet.

RBMs in general consists of a basic and simple structure with good mathematical properties for which one can find efficient learning algorithms. The successful application of RBMs in these different topics of machine learning make them an interesting subject to study and understand. In the next sections we will talk about the details of RBMs used for the topic of collaborative filtering and especially about how to predict user ratings for movies as a direct application to a real world problem.

2 The Problem

Collaborative filtering as seen in a modern narrow sense basically can be described as a technique to make automatic predictions about interests of users by collecting

Table 1: The table shows examples of binary ratings for movies made by some imaginary users. Every row represents a user and every column a movie. The number 0 is used to point out that the user does not like the movie and 1 for the opposite. The symbol × is used if there was no rating for the movie by this user.

	Star Trek	The Matrix	Van Helsing	Harry Potter	The Hobbit
James T. Kirk	1	1	×	0	×
Trinity	×	1	0	1	1
Anna Valerious	×	×	1	×	0
Severus Snape	0	1	0	1	0
Thorin Oakenshield	1	1	1	×	0

the preferences or tastes of many users. One often refers to predicting as "filtering" and to collecting as "collaborating". The underlying assumption of the collaborative filtering approach is that if two persons have the same opinion on one issue then it is likely that they will also have the same opinion on another issue. [14]

For convenience table 1 shows examples for movie ratings made by some imaginary users. The entries with 0 and 1 determine if a user likes a movie or not and are already known to us. They shall be used to predict the unknown values marked with ×. In the example the users "James T. Kirk" and "Thorin Oakenshield" both like the movies "Star Trek" and "The Matrix". So one could assume that both users have similar preferences and therefore the user "James T. Kirk" would also like the movie "Van Helsing" because the same is already true for the user "Thorin Oakenshield". Vice versa one may say that the user "Thorin Oakenshield" does not like the movie "Harry Potter" since "James T. Kirk" provided a negative rating for that movie.

Here the collaborative filtering problem of predicting user ratings for movies shall be solved by using RBMs as it was done in [13]. For that we have to achieve three main goals that together are forming a basic outline of understanding RBMs and their application to collaborative filtering.

Model: Approximately represent probability distributions over different user-movie-ratings.

Learn: Learn an optimal probability distribution in this representation based on some given samples.

Infer: Make predictions for unrated movies.

3 Background

Before we go into the details of the model of an RBM let us consider some fundamentals in stochastics and statistics to better understand the approach taken by the RBM.

4 The Model

4.1 Basic Idea

To understand the model of an RBM let us first consider the basic idea by looking at the left part of figure 1. It shows a simple schematic example of an RBM. At first sight there seems to be no real difference to a standard feed-forward neural network (FFNN) with two layers. But the subtle difference lies in the fact that every edge connecting different units in an RBM has to be undirected and not directed as in a typical FFNN. Therefore the influence of one unit to another cannot be computed directly through a feed-forward pass. [8]

Apart from this there are two obvious properties which are defining the RBM. First, the units in the RBM are separated into two subsets, the hidden units and the visible units. Second, connections between units are only allowed between those two subsets. This makes it possible to equivalently define an RBM to be an undirected bipartite graph. [7, 8]

These properties give us no real obvious interpretation for the values of these units. But we have to remember that we want to model a probability distribution over ratings. Based on this mathematically we will try to interpret every unit as a binary random variable. Later, this will enable us to sample from the probability distribution represented by the RBM and predicting the

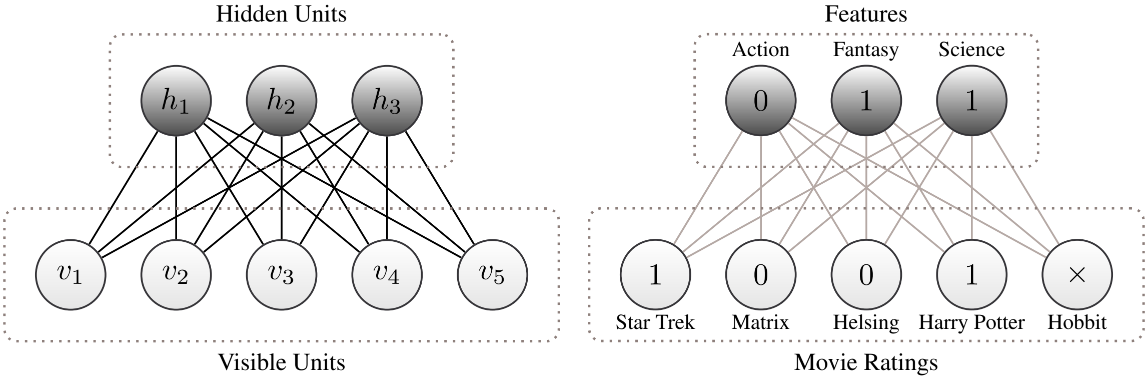

Figure 1: The left part of the figure shows a schematic example of an RBM. The nodes are separated into visible and hidden units and connections are only allowed between those two subsets. The right part shows an example application to the collaborative filtering problem of predicting movie ratings. For one user the visible values are the movie ratings and the hidden values are some features, like movie genres, implicitly learned by the RBM.

outcome. Interpreting user ratings to be realizations of the visible random variables we can then see that every visible value in an RBM stands for a movie rating made by some user. The realizations of the hidden random variables can be thought of as non-observable features the RBM has learned implicitly. [7, 8, 13]

Looking at the right part of figure 1 one sees an example of these ideas for an imaginary user. Every rating from one user can be seen as a visible value. Based on these visible values an RBM can assign some hidden values based on some learned features, like movie genres "Action" or "Fantasy". These hidden values for example would describe if the user likes the movie genre. [12]

Until now we have described how to model the collaborative filtering problem by an RBM but not how to model the RBM itself. Hence, we first have to define some parameters to be able to represent a set of probability distributions which can be learned. The procedure for an RBM is straightforward. We choose the standard approach of introducing biases for every unit and weights for every edge. Therefore we get two bias vectors for the hidden and visible units and one weight matrix for all connections. Figure 2 shows this schematically. [5, 7, 8, 13]

4.2 Mathematical Details

For the sake of simplicity, we will only assume so-called binary RBMs with binary hidden and visible values $\mathcal{B} := \{0, 1\}$ and will further on describe them only as RBMs. These RBMs are fundamental to every generalization and are not that much of a constraint. According to [13] and [8] binary RBMs can be easily extended to categorical RBMs or Gaussian RBMs. But it turns out that it is important for the hidden values to stay binary to introduce a sort of information bottleneck which strongly regularizes the mathematical properties of an RBM. [5, 7, 8, 13]

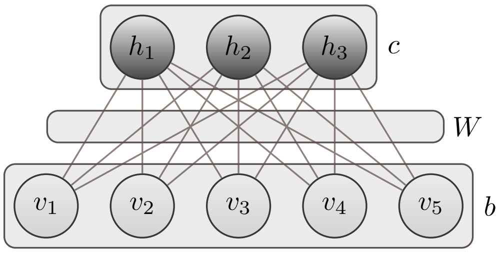

Figure 2: The figure shows the basic scheme of an RBM with the weight matrix W and bias vectors b and c as parameters describing the probability distribution modelled by the RBM.

DEFINITION: (RBM)

For $n, m \in \mathbb{N}$ an RBM ϑ is given by a tuple (W, b, c) consisting of its weight matrix W, visible bias vector b and hidden bias vector c.

$$\vartheta := (W, b, c) \in \mathbb{R}^{(n \times m)+n+m}$$

We call n the number of visible units and m the number of hidden units of ϑ.

As seen in figure 2 the mathematical definition of an RBM as data structure is straightforward. We want to keep such an approach for the parameterization of the probability distribution as well. Therefore we will first define the influence of every unit with respect to its connections by introducing an energy term. In this energy term every weight and bias linearly connects visible and hidden values. To model non-trivial probability distributions we need some simple non-linear function with good properties to be applied to this energy term. Afterwards, we have to make sure the resulting function is normalized such that it satisfies the definition of a probability distribution. [5, 8, 13]

DEFINITION: (RBM Probability Distribution)

Let $n, m \in \mathbb{N}$ and ϑ be an RBM with n visible and m hidden units. The probability distribution of ϑ is then given by $p[\vartheta]$.

$$p[\vartheta]\colon \mathcal{B}^n \times \mathcal{B}^m \to [0,1]$$

$$p[\vartheta](v,h) := \frac{1}{Z(\vartheta)} e^{-E[\vartheta](v,h)}$$

Here, the energy term $E[\vartheta](v,h)$ is given by the following expression.

$$E[\vartheta](v,h) := -v^{\mathrm{T}}Wh - v^{\mathrm{T}}b - h^{\mathrm{T}}c$$

$Z(\vartheta)$ is the normalization factor of $p[\vartheta]$.

$$Z(\vartheta) := \sum_{v\in\mathcal{B}^n} \sum_{h\in\mathcal{B}^m} e^{-E[\vartheta](v,h)}$$

At this point one could think, that we have introduced a major design problem in our RBM model. If we cannot observe hidden values then how should we be able to use a probability distribution with hidden values as its arguments. We can omit this problem by inserting a layer of indirection. We will simply use only the probability distribution for the visible values. For convenience here we use function overloading.

$$p[\vartheta]\colon \mathcal{B}^n \to [0,1] , \qquad p[\vartheta](v) := \sum_{h\in\mathcal{B}^m} p[\vartheta](v,h)$$

This finishes the complete mathematical description of the RBM model. But because of the main properties of an RBM it seems appropriate to deduce a proposition for the posterior probability to show one big advantage of an RBM. Due to its property that connections are only allowed between hidden and visible values we know that under a given visible value vector the coordinates of the hidden value vector have to be independent and vice versa. So for all $v \in \mathcal{B}^n$ and $h \in \mathcal{B}^m$ one obtains the following. [5, 8]

$$\begin{aligned} p[\vartheta](h|v) &= \prod_{j=1}^{m} p[\vartheta]\left(h_j = 1|v\right) \\ &= \prod_{j=1}^{m} \operatorname{sigm}\left(c_j + \sum_{i=1}^{n} v_i W_{ij}\right) \end{aligned}$$

$$\begin{aligned} p[\vartheta](v|h) &= \prod_{i=1}^{n} p[\vartheta]\left(v_i = 1|h\right) \\ &= \prod_{i=1}^{n} \operatorname{sigm}\left(b_i + \sum_{j=1}^{m} W_{ij} h_j\right) \end{aligned}$$

The logistic sigmoid function sigm is given by the typical definition.

$$\operatorname{sigm}\colon \mathbb{R} \to (0,1) , \qquad \operatorname{sigm}(x) := \frac{1}{1+e^{-x}}$$

5 Learning

The learning procedure for an RBM is rather easy due to its basic structure and good properties. First, we need some scalar potential to optimize. Learning is always about optimizing some sort of function. Because we want to learn a probability distribution on the data set we will use the maximum-likelihood estimation and will try to find a maximum.

$$\mathcal{S} \in V^s$$

As always we will not use the maximum-likelihood function but the log-likelihood function which simplifies the process of computing derivatives and gives us an equivalent optimization condition.

$$\mathcal{L}[\mathcal{S}]\colon \mathbb{R}^{n\times m+n+m} \to \mathbb{R}$$

$$\mathcal{L}[\mathcal{S}](\vartheta) := \frac{1}{s} \sum_{k=1}^{s} \ln p[\vartheta]\left(\mathcal{S}_k\right)$$

We will take one of the simples algorithms to maximize this function. "Gradient Ascent" works exactly like

"Gradient Descent" but finds the maximum instead of the minimum. For this we need the gradients of the log-likelihood function with respect to the weight matrix and the bias vectors.

$$\nabla_W \mathcal{L}[\mathcal{S}](\vartheta) = \frac{1}{s} \sum_{k=1}^{s} \mathbb{E}_\vartheta \left[\mathcal{V}\mathcal{H}^{\mathrm{T}} \middle| \mathcal{S}_k \right] - \mathbb{E}_\vartheta \left[\mathcal{V}\mathcal{H}^{\mathrm{T}} \right]$$

At first sight this formula seems to be complicated. But the left part can be easily computed. The right part is much more difficult. The typical method of finding the expectation of the model itself one has to do Gibbs sampling. Figure 3 demonstrates this method schematically.

Using Gibbs sampling for the right part of the gradient is mathematically ideal but fails when applied to reality because the algorithm is slow. The typical procedure to make things good again is to abort the series after some given integer. Then one can approximate the expectation as follows. This is called "Contrastive Divergence".

$$\mathbb{E}_\vartheta \left[\mathcal{V}\mathcal{H}^{\mathrm{T}} \right] \approx v^{(k)} {h^{(k)}}^{\mathrm{T}}$$

The algorithm is shown in the following example listing.

Now one has to talk about the application of the algorithm to collaborative filtering. For this every user will get its own RBM which learns only based on the rated and not the unrated movies. To not have a set of independent RBMs trained with only one sample one connects the weights and biases of each RBM. This means that if two users have rated the same movie then for this movie the same weights and biases will be used. Figure 4 shows the application of this algorithm to the movie ratings by a user again.

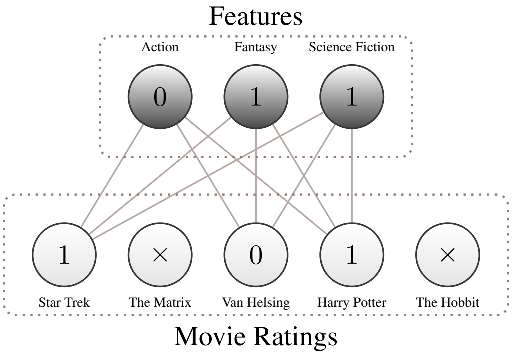

Figure 4: The figure shows the application of the learning algorithm to the movie ratings for some user.

6 Inference

The general inference for an RBM can be done by using the Gibbs sampling method explained in the last chapter. Here we will explicitly talk about the inference for the collaborative filtering problem. Figure 5 shows such an application in a schematic example.

First, we get the vector of rated and unrated movies from a given user. We then have to sample the hidden values by using only the values for the rated movies via the a posterior probability. After this we are now able to sample values for the unrated movies again by using the a posterior probability. The values sampled are then the predictions of the user ratings.

7 Implementation

8 Conclusion

RBMs have a simple structure and can be trained with CD which is an efficient algorithm. Based on SOURCE they seem to be one of the best known methods for collaborative filtering. As said in the introduction this is not the only application. RBMs should be used as basic building blocks. They are a powerful tool. Use them if there is a simple connection to hidden features in your data and if you want to predict something. It may be a good idea to insert these into your DNNs as dimensionality reduction.

Of course one should consider to tweak the explained ideas and algorithms. One can use momentum, weight decay and different types of units. There are some variants of the contrastive divergence as well. According the PAPER even mathematics has not com-

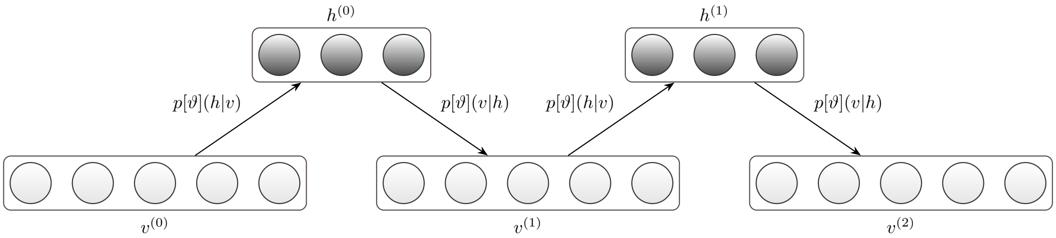

Figure 3: The figure shows the basic scheme of Gibbs sampling.

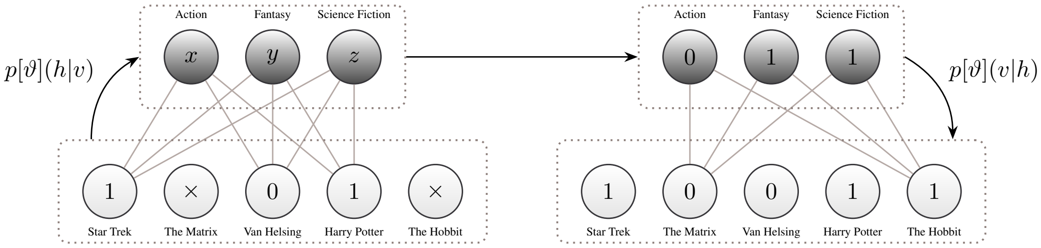

Figure 5: The figure shows the application of inference to the prediction of movie ratings for users.

pleted the topic of RBMs. They seem to be promising in explaining the connection of quantum theory and deep neural networks.

References

[1] Fischer, Asja and Christian Igel: *An introduction to restricted boltzmann machines*. LNCS, 7441:14–36, 2012.

[2] GroupLens: *Movielens dataset*, 2018. https://grouplens.org/datasets/movielens/latest/, visited on 2019-01-21.

[3] Harper, F. Maxwell and Joseph A. Konstan: *The movielens datasets: History and context*. ACM Trans. Interact. Intell. Syst., 5(4):19:1–19:19, December 2015, ISSN 2160-6455. http://doi.acm.org/10.1145/2827872.

[4] Hinton, G. E. and R. R. Salakhutdinov: *Reducing the dimensionality of data with neural networks*. SCIENCE, pages 504–507, 2006.

[5] Hinton, Geoffrey: *A practical guide to training restricted boltzmann machines: Version 1*. 2010. https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf.

[6] Larochelle, Hugo and Yoshua Bengio: *Classification using discriminative restricted boltzmann machines*. Proceedings of the 25th International Conference on Machine Learning, 2008.

[7] Montúfar, Guido: *Restricted boltzmann machines: Introduction and review*. CoRR, abs/1806.07066, 2018. http://arxiv.org/abs/1806.07066.

[8] Murphy, Kevin P.: *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012, ISBN 978-0-262-01802-9.

[9] Netflix: *Netflix prize*, 2009. https://www.netflixprize.com/index.html, visited on 2019-01-21.

[10] Netflix: *Netflix prize dataset*, 2009. https://archive.org/details/nf_prize_dataset.tar, visited on 2019-01-21.

[11] Netflix: *Netflix logo*, 2018. https://mms.businesswire.com/media/20150827005946/en/482959/5/etflix-Logo.jpg?download=1, visited on 2019-01-21.

[12] Oppermann, Artem: *Deep learning meets physics: Restricted boltzmann machines part i*, 2018. https://towardsdatascience.com/deep-learning-

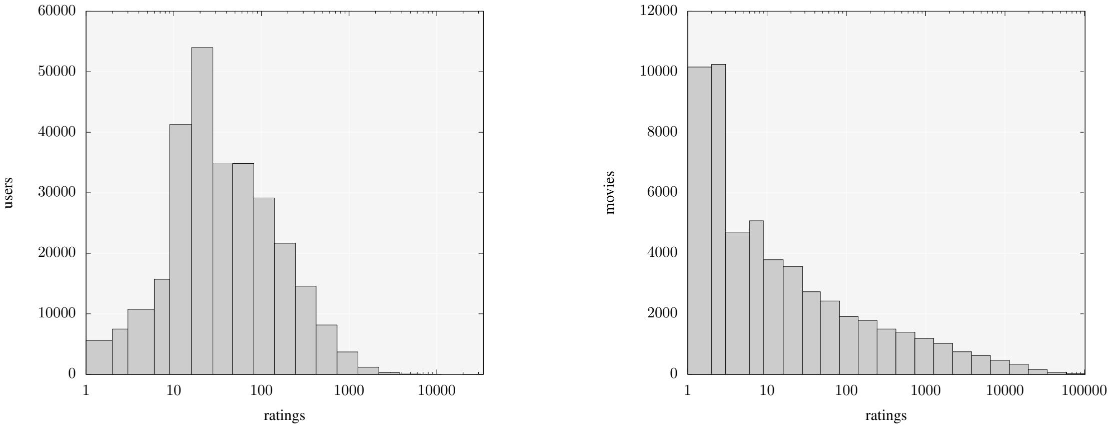

Figure 6: The figure shows the two histograms for rating counts over users and movies.

meets-physics-restricted-boltzmann-machines-part-i-6df5c4918c15, visited on 2019-01-22.

[13] Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton: *Restricted boltzmann machines for collaborative filtering*. Proceedings of the 24th international conference on Machine learning, pages 791–798, 2007.

[14] Wikipedia: *Collaborative filtering*, 2019. https://en.wikipedia.org/wiki/Collaborative_filtering, visited on 2019-02-11.