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 10, 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 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. Three years later on September 18, 2009 Netflix announced the winner of the Grand Prize which has beaten the current algorithm “Cinematch” Netflix already used by over 10% of error rate. The solution of the winner-team “BellKor’s Pragmatic Chaos” was based on [11] and therefore used a special implementation of restricted Boltzmann machines (RBM). The competition has shown that RBMs build a state-of-the-art tool for collaborative filtering.

RBMs have also found their application in other machine learning topics as well. For example, in dimensionality reduction, classification and topic modelling. Due to their basic structure, good properties and fast learning algorithms they are used as basic building blocks in deep neural networks, for example in deep belief networks.

2 The Problem

Collaborative filtering as seen in a modern narrow sense basically can be described as a method to make automatic predictions about interests of users by collecting the preferences or tastes of many users. The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B’s opinion on a different issue than that of a randomly chosen person.

To understand the statement of the problem we will use the prediction of movie ratings as done for the “Netflix Prize” competition. In table 1 one can see some examples for these ratings. The entries with 0 and 1 are already known and shall be used to predict the unknown values with $\times$. 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 the user “James T. Kirk” likes the movie “Van Helsing” and that the user “Thorin Oakenshield” does not like the movie “Harry Potter”.

The abstract goals to achieve this can be described as follows.

- Approximately represent probability distributions over ratings
- Learn a probability distribution based on some given ratings.
- Make predictions for unrated movies for learned parameters.

3 Background

Before we go into the details of the model of an RBM

Table 1: The table shows examples for ratings of movies made by some users. Every row represents a user and every column a movie. 0 stands for “does not like” and 1 for “likes”. $\times$ 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	$\times$	0	$\times$
Trinity	$\times$	1	0	1	1
Anna Valerious	$\times$	$\times$	1	$\times$	0
Severus Snape	0	1	0	1	0
Thorin Oakenshield	1	1	1	$\times$	0

let us consider some fundamentals in stochastics and statistics to better understand the approach taken by the RBM.

4 The Model

To understand the model of an RBM let us first consider the general 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. For an RBM every edge is undirected. Therefore the influence of one neuron to another cannot be computed directly as one maybe used to be in the FFNN.

Apart from this subtle difference there are two obvious properties which are defining the RBM. First, the neurons in the RBM are separated into two subsets, the hidden units and the visible units. And second, connections between neurons via edges are only allowed between those two subsets.

Based on these properties there is no real obvious interpretation for values of these units. But we have to remember that we want to model a probability distribution. So we will interpret every unit as input value for our RBM. Looking at the right part of figure 1 it is shown applied on the movie ratings from a 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.

Because we want to model the probability distribution with the given undirected bipartite graph, we first have to define some parameters. Here we can choose the standard approach of introducing biases for every neuron and weights for every edge. Therefore we get two bias vectors for the hidden and visible values and one weight matrix. Figure 2 shows schematically.

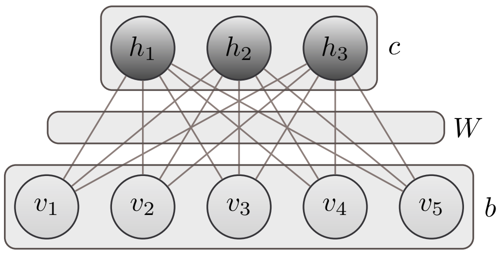

Figure 2: The figure shows the basic scheme of an RBM with the weight matrix W and bias vectors b and c as parameters describing the probability distribution modelled by the RBM.

To be precise, we will first define the sets for the input parameters.

$$v \in V := \{0,1\}^n$$

$$h \in H := \{0,1\}^m$$

The parameters describing the RBM are taken to be real. We will abbreviate the weight matrix and the two bias vectors as one parameter. This will make the following formulas much more readable.

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

Now we will define the probability distribution the RBM is modelling with respect to its parameters. This

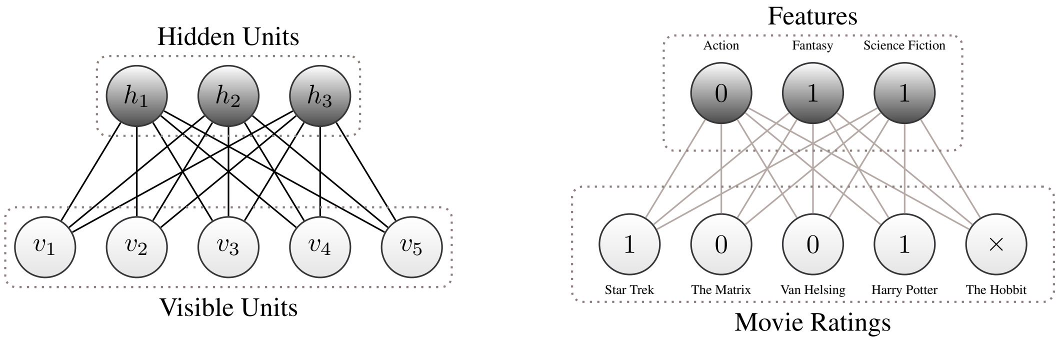

Figure 1: Figure for scheme and example!

is a definition and not a corollary. Of course, we could choose another approach. But this would not be a RBM.

$$p[\vartheta]\colon V \times H \to [0,1]$$
$$p[\vartheta](v,h) \coloneqq \frac{e^{-E[\vartheta](v,h)}}{Z(\vartheta)}$$

Here we define $E[\vartheta]$ to be the so called energy function of the RBM. We have already introduced non-linearity by the exponential function. Therefore one chooses $E[\vartheta]$ as simple as possible.

$$E[\vartheta]\colon V \times H \to \mathbb{R}$$
$$E[\vartheta](v,h) \coloneqq -v^{\mathrm{T}}Wh - v^{\mathrm{T}}b - h^{\mathrm{T}}c$$

For completeness the normalization will be supplied. The definition for $Z(\vartheta)$ is straightforward and not important for the learning or inference processes.

$$Z(\vartheta) \coloneqq \sum_{v\in V}\sum_{h\in H} e^{-E[\vartheta](v,h)}$$

At this point one could think, that major problem in our model. If we cannot observe hidden values then how should we able to model a probability distribution over them. We can omit this problem by defining the probability distribution for the visible values.

$$p[\vartheta]\colon V \to [0,1]$$
$$p[\vartheta](v) \coloneqq \sum_{h\in H} p[\vartheta](v,h)$$

This is the complete description of our model. But because of the two main properties of an RBM we can derive an important equation which will enable us to learn an RBM and to do some inference.

$$p[\vartheta](h|v) = \prod_{j=1}^{m} p[\vartheta]\,(h_j = 1|v)$$

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.

$$\mathcal{L}[\mathcal{S}]\colon \mathbb{R}^{n\times m+n+m} \to \mathbb{R}$$
$$\mathcal{L}[\mathcal{S}](\vartheta) \coloneqq \frac{1}{s}\sum_{k=1}^{s} \ln p[\vartheta]\,(\mathcal{S}_k)$$

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

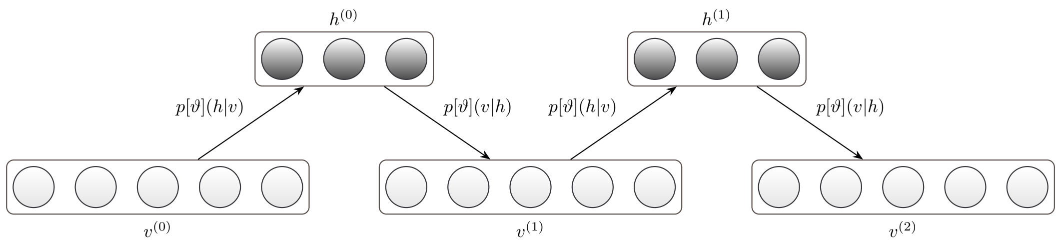

Figure 3: The figure shows the basic scheme of Gibbs sampling.

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.

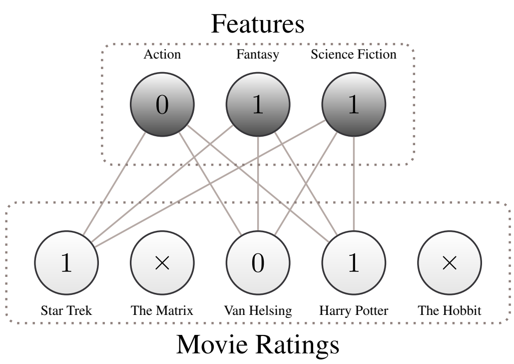

Figure 4: The figure shows the application of the learning algorithm to the movie ratings for some user.

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 completed the topic of RBMs. They seem to be promising in explaining the connection of quantum theory and

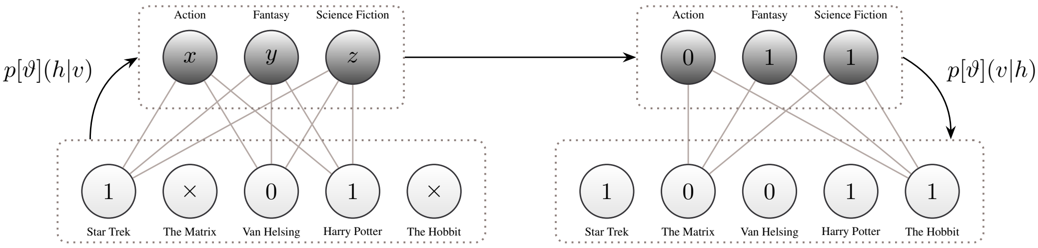

Figure 5: The figure shows the application of inference to the prediction of movie ratings for users.

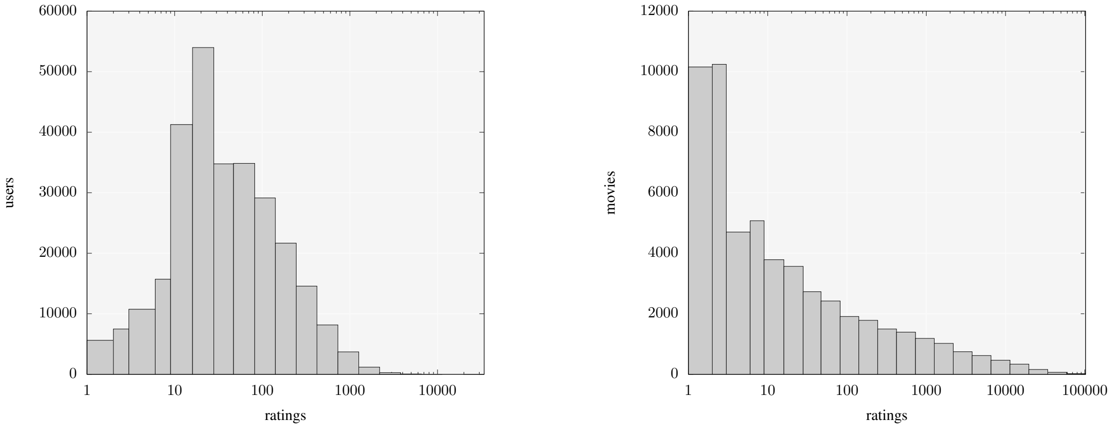

Figure 6: The figure shows the two histograms for rating counts over users and movies.

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

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

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

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

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

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

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

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

[11] 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.