# What Do You Know?

**Hubert Soyer** #3106011, SOYER@IN.TUM.DE
**Thomas Möllenhoff** #3106628, MOELLENH@IN.TUM.DE
Technical University Munich

## Abstract

We participated in the machine learning challenge "What Do You Know?", which was about automated student performance prediction. By interpreting the problem as a collaborative filtering task we were able to apply a Restricted Boltzmann Machine (RBM) model to the problem. We give a short introduction to RBMs in general, how they can be applied to the problem at hand and how well they perform. Linearly combining the benchmark provided by the challenge creator with our model resulted in a final rank 13 out of 257 at the end of the competition.

## 1. Introduction

The online social learning platform **Grockit**[1] allows students to practice for the standardized college admission tests (e.g. SAT, GMAT, ACT). For an efficient learning experience it is useful to know how good a student is prepared already and what topics he still has to study more for. In order to improve the state of the art in automated student assessment, Grockit hosted a competition on the machine learning contest platform **kaggle**[2], providing $5000 prize money. To succeed in the contest, the participating teams had to predict as accurately as possible how probable it is that students answer test questions correctly. Our goal for the final project was to participate in this challenge and explore how well Restricted Boltzmann Machines can be utilized for this kind of task.

[1] http://www.grockit.com
[2] http://www.kaggle.com/c/WhatDoYouKnow

### 1.1. The Grockit Dataset

In the training set, there is information about the outcome of around 5 million answers to 6046 unique questions. The questions were answered by 131072 students. The dataset distinguishes between four different outcomes (correct, incorrect, skipped and timeout) indicating the respective event. Additionally, for each given answer there is some meta-data describing the questions (e.g. is this a geography or math question, what kind of test does this question belong to) and also some meta-data describing the way the student answered the question (time he took to answer, date of answer). After a model has been learned, the goal is to predict the probability of a student answering a question correctly. The test set therefore consists of tuples containing a student's ID, a question's ID and the same meta data that is available in the training set as well. Every student that appears in the test set is granted to be also present with at least one entry in the training set. The dataset includes a validation set for model validation and a training and test data set that is used for contest submission.

### 1.2. Approach

We approached the problem of student performance prediction as a collaborative filtering problem. Our model was first described by Salakhutdinov et al. (2007), who showed that Restricted Boltzmann Machines (RBMs) can be successfully employed for collaborative filtering tasks. In order to explain our approach we will first describe Restricted Boltzmann Machines in general. Next, we will explain how we used RBMs as a model for the Grockit challenge, picking up some of the work by Salakhutdinov et al. (2007). We also provide a detailed mathematical derivation of some steps in the Appendix. In the last section we present and discuss the results of our implementation and application to the Grockit dataset.

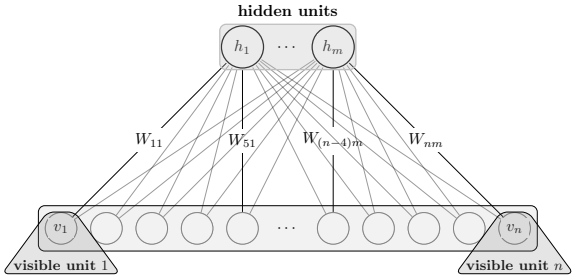



*Figure 1.* Illustration of an RBM

## 2. Restricted Boltzmann Machines (RBMs)

A Restricted Boltzmann Machine (RBM) is a stochastic recurrent neural network with one hidden and one visible layer of neurons. Each unit of the visible layer is bidirectionally connected to every unit of the hidden layer. Every connection between two neurons has a specific weight, every neuron has a specific bias value. In contrast to a Boltzmann Machine, in an RBM the visible as well as the hidden units are not connected amongst each other. The RBM models a probability distribution assigning a probability to each possible configuration of hidden and visible states. Training the RBM means increasing the probability of the training data by adjusting the weights of the connections between hidden and visible units and decreasing the probability of every other possible configuration. A graphical representation of an RBM is depicted in figure 1. In the following we present the model of the RBM similarly to Hinton (2010). We will denote the visible units as $\mathbf{v} = (v_1, ..., v_n)^\mathsf{T}$ and the hidden units as $\mathbf{h} = (h_1, ..., h_m)^\mathsf{T}$. In the most basic case of an RBM both types of units (hidden and visible) can only take on binary values. The weight between visible unit $v_i$ and hidden unit $h_j$ will be referred to as $W_{ij}$ ($\mathbf{W} \in \mathbb{R}^{n \times m}$). We will call the visible biases $\mathbf{a} = (a_1, ..., a_n)^\mathsf{T}$ and the hidden biases $\mathbf{b} = (b_1, ..., b_m)^\mathsf{T}$. Every configuration of hidden and visible units has an energy associated with it. This energy is described by

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^\mathsf{T}\mathbf{v} - \mathbf{b}^\mathsf{T}\mathbf{h} - \mathbf{v}^\mathsf{T}\mathbf{W}\mathbf{h} \tag{1}$$

The joint probability distribution over the hidden and visible layer is defined by the energy in the following way:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v}, \mathbf{h})} \tag{2}$$

$Z$ is a normalizing constant summing over all possible visible and hidden unit configurations

$$Z = \sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} \tag{3}$$

Marginalizing over the hidden units yields the probability density of the visible units $\mathbf{v}$

$$p(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} \tag{4}$$

Training the RBM is done by maximizing the probability of the training data using gradient ascent on $p(\mathbf{v})$ with respect to the weights and the biases. The derivative of $-\log p(\mathbf{v})$ computes as

$$-\frac{\partial \log p(\mathbf{v})}{\partial \theta} = \tag{5}$$
$$\mathbb{E}_{data}\left[\frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta}\right] - \mathbb{E}_{model}\left[\frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta}\right]$$

For a detailed derivation of equation 6 see Appendix A. For the specific parameters the gradient computes as

$$\frac{\partial \log p(\mathbf{v})}{\partial W_{ij}} = \mathbb{E}_{data}\left[v_i h_j\right] - \mathbb{E}_{model}\left[v_i h_j\right] \tag{6}$$

$$\frac{\partial \log p(\mathbf{v})}{\partial a_i} = \mathbb{E}_{data}\left[v_i\right] - \mathbb{E}_{model}\left[v_i\right] \tag{7}$$

$$\frac{\partial \log p(\mathbf{v})}{\partial b_j} = \mathbb{E}_{data}\left[h_j\right] - \mathbb{E}_{model}\left[h_j\right] \tag{8}$$

Evaluating the expectations directly is not feasible in practice. Therefore the expectations are approximated by sampling. Thus we set $\mathbb{E}[v_i h_j] \approx v_i h_j$. The $v_i$ in the approximation is simply taken from a training sample, the $h_j$ are also computed based on a training sample applying the following formula

$$p(h_j = 1 | \mathbf{v}) = \sigma(b_j + \sum_i v_i W_{ij}) \tag{9}$$

The probability of the visible units given the hidden units can be computed analogously:

$$p(v_i = 1 | \mathbf{h}) = \sigma(a_i + \sum_j h_j W_{ij}) \tag{10}$$

Note that this implicitly exploits the constraint, that in **Restricted** Boltzmann Machines the units of each layer are not interconnected amongst each other and are therefore independent of each other. A mathematical derivation of equation 9 and 10 is presented in Appendix A. For the model part of our learning rule, an approximation of $\mathbb{E}_{model}[v_i h_j]$ would be desirable as well. It turns out that acquiring an unbiased sample of $p(\mathbf{v}, \mathbf{h})$ is more difficult. A way to do it is to start at any random state for the visible units and perform alternating Gibbs sampling for an infinite, or at least very long time. Every iteration of alternating Gibbs sampling consists of two steps:

1. compute the probability for each hidden unit by employing equation 9 and then sample them.

2. compute the probability for each visible unit with equation 10.

Iterating those steps for an infinite or very long time is not doable in practice. Hinton proposed in 2002 a much faster way of training an RBM based on minimizing contrastive divergence (Hinton, 2002). Initially, we set all visible units of the RBM to a configuration given by a training sample. Now, one iteration of Gibbs sampling is performed which yields one probability value for each visible unit. In the next step, the visible units are "sampled", setting the state of each unit to 1 with the probability corresponding to the unit and to zero with its inverse probability. The new values for the visible units can now be used to start the whole procedure again. The resulting process is often referred to as "CD-**n**" with **n** being the number of repetitions. Using the visible and hidden states acquired by this procedure, we can compute an approximation of $\mathbb{E}_{model}[v_i h_j] \approx \mathbb{E}_{CD_n}[v_i h_j]$ as before for $\mathbb{E}_{data}[v_i h_j]$ but with different hidden and visible states. Letting **n** in CD-**n** go to infinity results in an unbiased sample from the model. For training the RBM, we now compute a weight update value

$$\Delta W_{ij} = \epsilon \left(\mathbb{E}_{data}[v_i h_j] - \mathbb{E}_{CD_n}[v_i h_j]\right) \tag{11}$$

Further, we adapt the biases in a similar fashion by using the difference between the approximated data expectation and the approximated model expectation that we acquired using the results from the contrastive divergence step. Formally, the bias updates write as

$$\Delta a_i = \epsilon \left(\mathbb{E}_{data}[v_i] - \mathbb{E}_{CD_n}[h_j]\right) \tag{12}$$

$$\Delta b_j = \epsilon \left(\mathbb{E}_{data}[h_j] - \mathbb{E}_{CD_n}[h_j]\right) \tag{13}$$

The update steps that were just described are preformed epoch wise sequentially for every training sample. Grouping the training samples to "mini-batches" and updating the weights and biases at the end of each batch allows for faster learning.

### 2.1. RBM for student performance prediction

The idea behind our RBM based approach on student performance prediction exploits an RBM as a generative model. In a way, we can see the RBM as a model that has some implicit understanding of our data. We assume if we give a new sample to the visible layer of our RBM and perform one or more iterations of Gibbs sampling, it will return an altered version of the sample with some touch of how the RBM interprets it in the sense of the underlying model. More specific on the Grockit dataset, each student represents one input sample. Our RBM features as many visible units, as there are questions. For each question that a student answered correctly, we set the corresponding visible unit to 1, for each question he answered incorrectly, we set it 0. Assuming that we have an RBM representing some kind of inherent model of our data, we could give it the sample vector for a student that we want to predict for, use equation 9 and 10 and get a probability for each question to be answered correctly by this student by looking at the corresponding visible unit. Unfortunately, in reality not every student has answered every question which makes building a standard RBM impossible. To overcome this obstacle of massively missing data, during training we only update the weights that are connected to visible units which correspond to questions the student has actually answered. This principle can be interpreted as training a tiny RBM for every student with as many visible units as the student has answered questions. All these RBMs operate on parts of the same weight matrix, sharing the weights they have in common. During learning, each RBM tries to obtain an optimal weight matrix, but since the weight matrix is shared among all RBMs they have to "compromise" on a common weight matrix. Hinton (2010) describes this as a family of models rather than a single model. Still, the proposed model has some flaws. Looking at equation 9 that is used to compute the hidden unit probabilities, the "0" state for the visible units that we set in case of incorrectly answered questions does not have any influence at all. In this version of the model, answering a question incorrectly is equal to not answering a question at all. We throw information away.

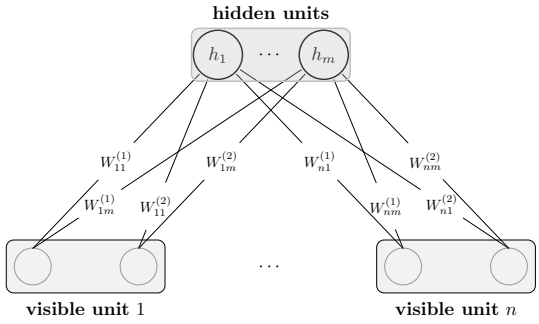



*Figure 2.* Model of an RBM with softmax visible units

### 2.2. Softmax visible units

In order to resolve the problem mentioned at the end of the previous section, we used softmax visible units instead of binary units. Softmax units are basically a set of binary units which are mutually constrained,

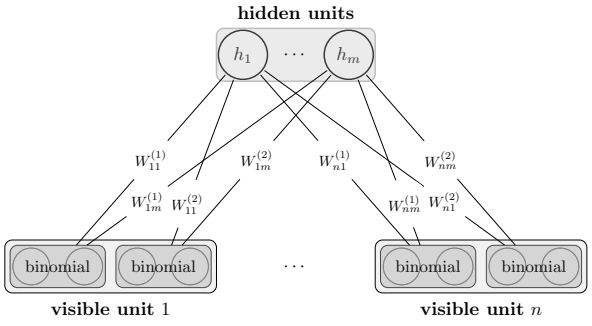



*Figure 3.* Model of a binomial softmax RBM

meaning that if one unit has high probability the others have lower probabilities. As depicted in figure 2, every visible unit consists of two binary units $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$ in our case, corresponding to the two different question outcomes (correct, incorrect) that we model. For a training sample, we set $v_i^{(1)}$ to 1 and $v_i^{(2)}$ to 0 if the student answered question $i$ correctly, and the other way around if he answered the question incorrectly. Each visible layer also has an own weight matrix $\mathbf{W}^{(1)}, \mathbf{W}^{(2)} \in \mathbb{R}^{n \times m}$ and bias. The learning rule stays the same except that we have to learn two weight matrices now. The probabilities of the hidden and visible units required in the learning and prediction process change slightly:

$$p(v_i^{(k)} = 1|\mathbf{h}) = \frac{\exp(a_i^{(k)} + \sum_{j=1}^{m} h_j W_{ij}^{(k)})}{\sum_{l=1}^{2} \exp(a_i^{(l)} + \sum_{j=1}^{m} h_j W_{ij}^{(l)})} \quad (14)$$

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_{i=1}^{n} \sum_{k=1}^{2} v_i^{(k)} W_{ij}^{(k)}) \quad (15)$$

Using softmax visible units improved the performance of our model significantly.

### 2.3. Binomial softmax visible units

We further extended our model by replacing the binary visible unit $v_i^{(k)}$ inside the softmax unit by two binary units $v_{i,1}^{(k)}, v_{i,2}^{(k)}$ which share the same weight matrix $\mathbf{W}^{(k)}$ and the same biases. Hinton (2010) refers to a set of binary units which share the same weight as a binomial unit. The overall structure is displayed in figure 3. Again, the learning rule does not change but the probability of the hidden units is now calculated differently.

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_{i=1}^{n} \sum_{k=1}^{2} (v_{i,1}^{(k)} + v_{i,2}^{(k)}) W_{ij}^{(k)}) \quad (16)$$

Using binomial softmax visible units further improved the performance of our model.

### 2.4. Factorizing the weight matrix

Following Salakhutdinov et al. (2007) we also implemented a so called "Factored RBM". In that model the weight matrices $\mathbf{W}^{(1)}, \mathbf{W}^{(2)} \in \mathbb{R}^{n \times m}$ are factorized into low rank matrices $\mathbf{A}^{(1)}, \mathbf{A}^{(2)} \in \mathbb{R}^{n \times C}$ and $\mathbf{B} \in \mathbb{R}^{C \times m}$.

$$\mathbf{W}^{(k)} = \mathbf{A}^{(k)} \mathbf{B} \quad (17)$$

In order to reduce the number of free parameters in the model, one usually chooses $C \ll n$ and $C \ll m$. Instead of learning the weight matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$, we now learn $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{B}$. The derivatives of the energy function with respect to $\mathbf{A^{(k)}}$ and $\mathbf{B}$ needed for contrastive divergence learning are calculated using basic matrix calculus:

$$\frac{\partial E(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{h})}{\partial \mathbf{A}^{(k)}} = -\mathbf{v}^{(k)} (\mathbf{B}\mathbf{h})^{\mathsf{T}} = -\mathbf{v}^{(k)} \mathbf{h}^{\mathsf{T}} \mathbf{B}^{\mathsf{T}} \quad (18)$$

$$\frac{\partial E(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{h})}{\partial \mathbf{B}} = -\mathbf{v^{(1)}}^{\mathsf{T}} \mathbf{A}^{(1)} \mathbf{h} - \mathbf{v^{(2)}}^{\mathsf{T}} \mathbf{A}^{(2)} \mathbf{h} \quad (19)$$

We initialized $\mathbf{A}^{(k)}$ in the same way as $\mathbf{W}^{(k)}$, and $\mathbf{B}$ to the (non-square) identity matrix. Unlike to Salakhutdinov et al. (2007), in our experiments the "Factored RBM" sadly performed slightly worse than the non-factored one.

### 2.5. Conditional RBMs

Right now the only information from the dataset that is being utilized is just whether a student answered a question correctly or incorrectly. However, in the dataset there are many instances where the student skips a question or he can't answer the question in the given time limit. In order to make use of this valuable source of information (a student has *seen* a question, but the outcome is unknown – this also applies to the data in the test set), we implemented a "Conditional RBM" the way as described by Salakhutdinov et al. (2007). Let $\mathbf{r} = (r_1, ..., r_n)^{\mathsf{T}} \in \{0, 1\}^n$ indicate whether a student has *seen* a question. Identically to Salakhutdinov et al. (2007), we let $r$ influence the probabilities of the hidden units in the following way:

$$p(h_j = 1|\mathbf{v}, \mathbf{r}) = \sigma(b_j + \sum_{i=1}^{n} \sum_{k=1}^{2} v_i^{(k)} W_{ij}^{(k)} + \sum_{i=1}^{n} r_i D_{ij}) \quad (20)$$

In equation 20, $D \in \mathbb{R}^{n \times m}$ is a matrix describing how $\mathbf{r}$ influences $\mathbf{h}$. The learning rule for $\mathbf{D}$ can be computed by calculating the derivative of the new energy function with respect to $\mathbf{D}$.

$$\frac{\partial}{\partial \mathbf{D}} E(v, h, r) = \mathbf{r}\mathbf{h}^{\mathsf{T}} \quad (21)$$

Thus the update rule for learing with CD is like this:

$$\Delta D_{ij} = \epsilon \left( \mathbb{E}_{data}[h_j] - \mathbb{E}_{model}[h_j] \right) r_i \qquad (22)$$

Where $r_i$ can be pulled out of the expectation since it is constant. Unfortunately, in our case the "Conditional RBM" did not improve the performance of our model at all. This might be a topic of further investigation.

## 3. Experimental Results

We implemented the presented models in python employing numpy to speed up matrix-vector calculations. The source code is included as supplementary material in our submission. Due to the sparse nature and big size of our family of RBMs we found it difficult to use the usual matrix-matrix products or to utilize a GPU to speed up learning. Training one epoch on the Grockit dataset however, only takes up about 3 to 10 minutes (depending on learning parameters) on a 2.26GHz dual core notebook CPU with parallel BLAS[1] installed. The set of parameters that lead to the best outputs was mainly the same for all the different models. The training data set was portioned into mini-batches consisting of 1000 students each. We had the best results with $m = 15$ hidden units for the non-factored RBMs and $m = 40$ hidden units with $C = 10$ factors for the factored RBM. We used a learning rate for the weights and biases of 0.001 for the RBMs with binary visible units and a learning rate of 0.0002 for the RBMs with binomial visible units. Further the momentum was set to 0.9 and to prevent overfitting the weight decay was set to 0.1. The weights were initialized by sampling from a gaussian distribution with mean 0 and standard deviation 0.001. The biases were initialized to 0. There are several ways to get a basic understanding on whether the RBMs learn correctly. The images presented in figure 5 show the probability of each hidden unit on some training cases. The rows in every picture correspond to a student, the 15 columns represent the 15 hidden units. Figures 5 a) - e) were captured at different states of the learning procedure. It can be seen that after initialization the hidden units all have a probability of 0.5, but after a few epochs the hidden units get more and more certain. In addition to exploiting the RBM as a generative model, its hidden units can be used as a condensed input for discriminative models. We found it interesting on the one hand to see how strongly the hidden unit probabilities differ for different questions and on the other hand if there is a sensible interpretation behind the states the hidden units take on. Figure 6 was generated by first training an RBM with only 2 hidden units on the training set and then plotting the

[1] http://www.netlib.org/blas/

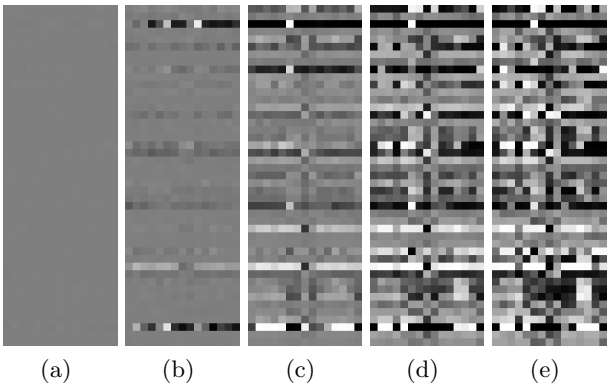



*Figure 5.* Probabilities of the hidden units after setting the visible units to different students from the training set. (a) after initialization, (b) 1 epoch, (c) 3 epochs, (d) 10 epochs and (d) 40 epochs

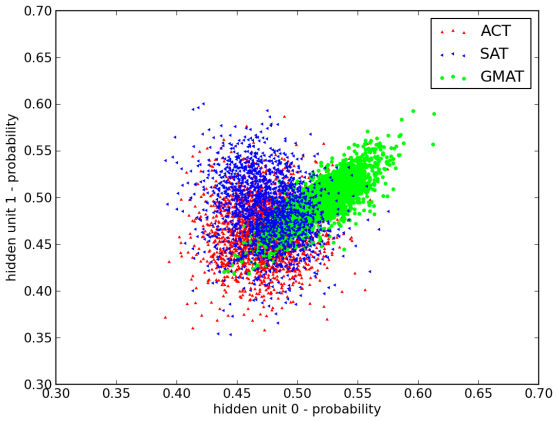
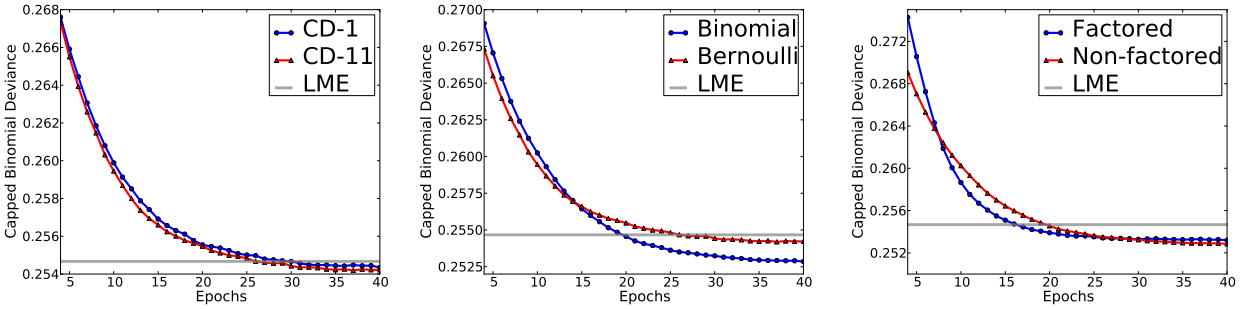



*Figure 6.* Hidden unit probabilities of RBM with 2 hidden units - each point corresponds to an input with only one visible unit set to 1

probabilities of the two hidden units when only one visible unit is active at a time. Therefore a point in figure 6 corresponds to the hidden representation of a single question. It turned out that the plots can differ strongly if they are repeatedly created the same way. Since the error on the test set is similar every time, we assume that the RBM models the data each time with a different but equally expressive model. We used a different color for the 3 test groups present in Grockit (GMAT,SAT,ACT) in the plot shown in 6. The questions from GMAT seem to cluster slightly but no clear structure can be observed.

Figure 4 shows the test error over time of the different RBM models. The y-axis of each plot denotes




(a) CD-1 vs CD-11 (b) Bernoulli vs Binomial (c) Factored vs. Binomial

*Figure 4.* Learning progress

the Capped Binomial Deviance[2] while the x-axis displays the number of training epochs. Each plot also includes a horizontal line marking the result that can be achieved with the benchmark method provided by Grockit.It is a Linear Mixed Effects (LME) method. Figure 4 a) compares the Bernoulli Softmax RBM trained with CD-1 and CD-11. It seems like learning with CD-11 slightly outperforms CD-1 but the difference is negligible. This is an unexpected behavior, since increasing the number of CD steps resulted in better learning for Salakhutdinov et al. (2007). From figure 4 b) can be seen that binomial visible units outperform bernoulli visible units. Figure 4 c) shows the error curve of the binomial softmax RBM next to its factored sibling. While the error of the factored version decreases a little bit faster at start, the non-factored version of the binomial softmax RBM catches up at around epoch 25 and performs slightly better eventually.

## 4. Summary

We presented different RBM models and tested them on the Grockit dataset. We ranked 13th out of 257 with a linear combination of LME and an RBM model with binomial softmax visible units. Table 1 gives an overview of the final standings on the kaggle challenge. Linearly combining our RBM model with LME improved our score significantly, which indicates that the RBM might be a good addition to an ensemble of different methods. At the end of the day, the binomial non-factored RBM scored best on the final kaggle submission test set. The other variations of our RBM, however, showed results that would still have ranked in the top 50 submissions.

[2] The Capped Binomial Deviance (CBD) of a model $M$ on a test dataset $X = \{x_1, ..., x_N\}$ with targets $Z = \{z_1, ..., z_N\}$ is measured the following way:

$$c = \frac{1}{N} \sum_{n=1}^{N} -(z_n \log_{10} M(x_n) + (1 - z_n) \log_{10}(1 - M(x_n)))$$

Where the outputs of $M$ get capped to $[0.01, 0.99]$. A model $M$ which always outputs 0.5 thus has a CBD of $\log_{10} 0.5 \approx 0.301030$.

| Teamname | Method | Rank | CBD |
|---|---|---|---|
| Steffen | Factorization Machine[a] | 1 | 0.24598 |
| camel_case | 0.55·RBM+0.45·LME | 13 | 0.25354 |
| camel_case | RBM | (25) | 0.25469 |
| Grockit | LME | 76 | 0.25766 |

[a] http://www.libfm.org

*Table 1.* Overview of how our method ranked in the "What Do You Know?" challenge. We participated in the challenge as team "camel_case". See http://www.kaggle.com/c/WhatDoYouKnow/leaderboard for the final ranking.

### 4.1. Future Work

We briefly investigated other types of hidden units, such as gaussian hidden units with fixed variance and rectified linear units. During our short evaluation those types of hidden units performed a lot worse than binary ones, but to come to a definite conclusion more testing is required. It might be worth a try to train the RBM using Persistent Contrastive Divergence (Tieleman, 2008) which is presumably similar to using CD-$n$ during training with a high $n$. However, since in our experiments the difference between training with CD-1 and CD-11 was so small, we didn't explore this option further. Our collaborative filtering approach only makes use of a small subset of the data available in the Grockit dataset. In order to include the remaining data for prediction, a discriminative approach that builds on top of the hidden units of our RBMs as well as on the unused data could help increase the quality of the outputs.

## References


Hinton, Geoffrey. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL http://dx.doi.org/10.1162/089976602760128018.

Hinton, Geoffrey. A Practical Guide to Training Restricted Boltzmann Machines. Technical report, 2010. URL http://www.cs.toronto.edu/~hinton/absps/guideTR.pdf.

Salakhutdinov, Ruslan, Mnih, Andriy, and Hinton, Geoffrey. 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, 2007. ACM. ISBN 978-1-59593-793-3.

Tieleman, Tijmen. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pp. 1064–1071, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390290. URL http://doi.acm.org/10.1145/1390156.1390290.


## Appendix A

Derivation of the log probability derivative:

$$
\begin{aligned}
&-\tfrac{\partial}{\partial\theta}\log p(\mathbf{v}) = -\tfrac{\partial}{\partial\theta}(\log\sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h})) = \\
&= -\tfrac{\partial}{\partial\theta}(\log\sum_{\mathbf{h}} \tfrac{e^{-E(\mathbf{v},\mathbf{h})}}{Z}) = \\
&= -\tfrac{Z}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}(\sum_{\mathbf{h}} \tfrac{\partial}{\partial\theta}\tfrac{e^{-E(\mathbf{v},\mathbf{h})}}{Z}) = \\
&= -\tfrac{Z}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}(\sum_{\mathbf{h}} \tfrac{\frac{\partial}{\partial\theta}e^{-E(\mathbf{v},\mathbf{h})}Z - e^{-E(\mathbf{v},\mathbf{h})}\frac{\partial}{\partial\theta}Z}{Z^2}) = \\
&= -\tfrac{Z}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}(\sum_{\mathbf{h}} \tfrac{\frac{\partial}{\partial\theta}e^{-E(\mathbf{v},\mathbf{h})}}{Z} - \sum_{\mathbf{h}} \tfrac{e^{-E(\mathbf{v},\mathbf{h})}}{Z^2}\tfrac{\partial}{\partial\theta}Z) = \\
&= -\tfrac{1}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}(\sum_{\mathbf{h}} \tfrac{\partial}{\partial\theta}e^{-E(\mathbf{v},\mathbf{h})} - \sum_{\mathbf{h}} \tfrac{e^{-E(\mathbf{v},\mathbf{h})}}{Z}\tfrac{\partial}{\partial\theta}Z) = \\
&= -\sum_{\mathbf{h}} \tfrac{\frac{\partial}{\partial\theta}e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\tilde{\mathbf{h}}} e^{-E(\mathbf{v},\tilde{\mathbf{h}})}} + \tfrac{1}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\tfrac{1}{Z}\tfrac{\partial}{\partial\theta}Z = \\
&= (\sum_{\mathbf{h}} \tfrac{e^{-E(\mathbf{v},\mathbf{h})}\frac{\partial}{\partial\theta}E(\mathbf{v},\mathbf{h})}{\sum_{\tilde{\mathbf{h}}} e^{-E(\mathbf{v},\tilde{\mathbf{h}})}}) + \tfrac{1}{Z}\tfrac{\partial}{\partial\theta}Z = \\
&= \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v})\tfrac{\partial}{\partial\theta}E(\mathbf{v},\mathbf{h}) + \tfrac{1}{Z}\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}(-\tfrac{\partial}{\partial\theta}E(\mathbf{v},\mathbf{h})) = \\
&= \sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v})\tfrac{\partial}{\partial\theta}E(\mathbf{v},\mathbf{h}) - (\sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h})\tfrac{\partial}{\partial\theta}E(\mathbf{v},\mathbf{h})) = \\
&= \mathbb{E}_{data}\left[\tfrac{\partial}{\partial\theta}E(\mathbf{v},\mathbf{h})\right] - \mathbb{E}_{model}\left[\tfrac{\partial}{\partial\theta}E(\mathbf{v},\mathbf{h})\right].
\end{aligned}
$$

Derivation of the hidden unit probabilities (using the independence of the hidden units):

$$
\begin{aligned}
&p(\mathbf{h}|\mathbf{v}) = \tfrac{p(\mathbf{v},\mathbf{h})}{p(\mathbf{h})} = \tfrac{\exp(\mathbf{a}^\top\mathbf{v}+\mathbf{b}^\top\mathbf{h}+\mathbf{v}^\top\mathbf{W}\mathbf{h})}{\sum_{\tilde{\mathbf{h}}}\exp(\mathbf{a}^\top\mathbf{v}+\mathbf{b}^\top\tilde{\mathbf{h}}+\mathbf{v}^\top\mathbf{W}\tilde{\mathbf{h}})} = \\
&= \tfrac{\prod_{i=1}^{m} p(\mathbf{v},h_i)}{\prod_{i=1}^{m} p(h_i)} = \tfrac{\prod_{i=1}^{m}\exp(\mathbf{a}^\top v)\exp(h_i b_i + h_i W_{\_,i}\mathbf{v})}{\prod_{i=1}^{m}\exp(\mathbf{a}^\top v)\sum_{\tilde{h}_i}\exp(\tilde{h}_i b_i + \tilde{h}_i W_{\_,i}\mathbf{v})} = \\
&= \prod_{i=1}^{m}\tfrac{\exp(h_i b_i + h_i W_{\_,i}\mathbf{v})}{\sum_{\tilde{h}_i}\exp(\tilde{h}_i b_i + \tilde{h}_i W_{\_,i}\mathbf{v})} = \prod_{i=1}^{m} p(h_i|\mathbf{v}).
\end{aligned}
$$

$W_{\_,i}$ denotes the $i$-th column of $\mathbf{W} \in \mathbb{R}^{n\times m}$. We can follow:

$$
\begin{aligned}
&p(h_i = 1|\mathbf{v}) = \tfrac{\exp(1\cdot b_i + 1\cdot W_{\_,i}\mathbf{v})}{\exp(0\cdot b_i + 0\cdot W_{\_,i}\mathbf{v}) + \exp(1\cdot b_i + 1\cdot W_{\_,i}\mathbf{v})} = \\
&= \tfrac{\exp(b_i + W_{\_,i}\mathbf{v})}{1+\exp(b_i + W_{\_,i}\mathbf{v})} = \sigma(b_i + W_{\_,i}\mathbf{v}).
\end{aligned}
$$

The derivation of $p(v_i = 1|\mathbf{h})$ can be obtained similarly.