# A Comparison of Algorithms for Collaborative Filtering on RBMs

Andrew Gelfand

CS277 Final Report: Due 3/18/2010

## 1 Introduction

Almost all web retailers employ some form of recommender system to tailor the products and services offered to their customers. A common approach to recommendation tasks is collaborative filtering, which uses a database of the preferences of different users - i.e. collaboration - to predict user preferences - i.e. filtering [3]. Collaborative filtering embodies a nearest-neighbor philosophy in that predictions are largely based upon the preferences of other 'like-minded' users. For example, if students A and B both positively rated the text 'Principles of Data Mining' and student A also favorably rated the text 'Pattern Recognition and Machine Learning', then it is reasonable to think that student B would also enjoy this text. In principle, collaborative filtering can be used to make recommendations to any user on any item. In practice, though, the database of preferences will be very sparse as only a few (if any) customers may have rated each item. In such situations it is not immediately clear how to identify similar users, much less make predictions that lead to purchases.

Early approaches to collaborative filtering relied on simple metrics to find similar users. GroupLens, for example, was an early Internet news system that used the correlation coefficient between user ratings of articles as a similarity measure [8]. However, memory-based approaches like GroupLens do not scale without further approximation because they require access to the ratings of the entire user community. For this reason, approaches such as singular value decomposition (SVD) that are based on matrix factorization have also been proposed (see e.g. [4, 7]). However, application of matrix factorization methods to sparse ratings matrices can be a non-trivial challenge. As such, Hoffman proposed a formal statistical model of user preference using hidden variables over user-item-rating triplets [6]. Unlike traditional clustering models that use latent variables to indicate which cluster (i.e. group of like minded people) a user belongs to, Hoffman's probabilistic latent semantic analysis (pLSA) model links users and their ratings to different latent causes.

More recently, Salakhutdinov et al. proposed an approach to collaborative filtering using Restricted Boltzmann Machines (RBMs)[9]. These models were trained using an efficient learning procedure, known as Contrastive Divergence (CD), that maximizes an approximation to the true likelihood function[5]. Furthermore, the authors report performance on the Netflix data set that is superior to the competitive SVD model posed in [7]. In this report, we investigate the use of an alternative learning procedure known as Persistent CD[10]. In addition, we demonstrate that outperforming SVD models using the RBM models is a non-trivial matter, requiring careful parameter tuning, performance-enhancing tricks and lots of CPU time for training.

The remainder of this report is structured as follows. In the next section I present some necessary background on RBMs and also discuss the learning CD and PCD learning procedure. In Section 3, I present Salakhutdinov et al.'s models for collaborative filtering using RBMs. Section 4 contains experimental results and Section 5 contains some concluding remarks.

## 2 Background

### 2.1 Restricted Boltzmann Machines

RBMs are a specific type of undirected graphical model consisting of two layers of binary variables - hidden ($H$) and visible ($V$) - with no intra-layer connections (see Figure 2)[1]. The joint over the visible and hidden variables is given by the Gibbs Distribution $P(V,H) = \frac{1}{Z}\exp(-E(V,H))$. In this expression $E(V,H)$ is an energy function

defined as

$$E(V,H) \quad = \quad -\sum_{i=1}^{Nv}\sum_{j=1}^{Nh} v_i h_j w_{ij} - \sum_{i=1}^{Nv} v_i b_i - \sum_{j=1}^{Nh} h_j b_j \tag{1}$$

where $i$ is used to index the $Nv$ visible nodes, $j$ is used to index the $Nh$ hidden nodes, $v_i$ is the state of the $i^{th}$ visible node (either 0 or 1), $h_j$ is the state of the $j^{th}$ hidden node, $w_{ij}$ is the strength of connection (i.e. weight) between the $i^{th}$ visible node and the $j^{th}$ hidden node and $b_i$ and $b_j$ represent the bias of the $i^{th}$ visible and $j^{th}$ hidden nodes, respectively. The parameters in the RBM - i.e. the weights and biases - can be described using a parameter vector $\theta$. Throughout this report we will make the dependence on $\theta$ explicit by writing $E^\theta(V,H)$ rather than $E(V,H)$.

The probability of observing a single $Nv$-dimensional data point $V$ is given by the marginal $P^\theta(V) = \sum_h P^\theta(V,H) = \frac{1}{Z^\theta}\sum_h \exp(-E^\theta(V,H))$, where $Z^\theta$ is the partition function. For a data set consisting of $N$ independent and identically distributed observations, the log-likelihood can be written as

$$l(\theta) \quad = \quad \frac{1}{N}\sum_{n=1}^{N} \ln P^\theta(V_n) = l(\theta)^+ - l(\theta)^- \tag{2}$$

$$l(\theta)^+ = \frac{1}{N}\sum_{n=1}^{N} \ln \sum_h \exp(-E^\theta(V_n,H)) \qquad l(\theta)^- = \ln(Z^\theta) = \ln \sum_{h,v} \exp(-E^\theta(V,H))$$

where the expression for $l(\theta)$ has been broken into two parts: 1) a positive term, corresponding to the evaluation of each training point under $\theta$; and 2) a negative term, corresponding to the log partition function.

Performing Maximum Likelihood (ML) learning in this model requires taking the gradient of $l(\theta)$. The gradient of the positive term is:

$$\frac{\partial l(\theta)^+}{\partial \theta} \quad = \quad \frac{1}{N}\sum_{n=1}^{N} \frac{\partial}{\partial\theta} \ln \sum_h \exp(-E^\theta(V_n,H)) = \frac{1}{N}\sum_{n=1}^{N}\sum_h P^\theta(H|V_n) \frac{\partial\left(-E^\theta(V_n,H)\right)}{\partial\theta}$$

This quantity is just a conditional expectation with respect to the data, which for weight $w_{ij}$ takes the form $\frac{\partial l(\theta)^+}{\partial w_{ij}} = \frac{1}{N}\sum_{n=1}^{N}\sum_h P^\theta(h_j|v_i^n)\cdot v_i^n$, where $v_i^n$ is the state of the $i^{th}$ visible unit in the $n^{th}$ data point. Similarly, the gradient of the negative term is:

$$\frac{\partial l(\theta)^-}{\partial \theta} \quad = \quad \frac{\partial}{\partial\theta} \ln \sum_{h,v} \exp(-E^\theta(V,H)) = \frac{\sum_{h,v}\frac{\partial}{\partial\theta}\exp(-E^\theta(V,H))}{\sum_{h,v}\exp(-E^\theta(V,H))} = \sum_{h,v} P^\theta(V,H)\frac{\partial(-E^\theta(V,H))}{\partial\theta}$$

which is just an expectation with respect to the model. Since both the terms involve expectations, the full gradient is often written as $\Delta w_{ij} = \frac{\partial \log P^\theta(V)}{\partial w_{ij}} = \langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model}$.

Computing the gradient requires a means to evaluate $P^\theta(H|V)$ and $P^\theta(H,V)$. The structure of the RBM affords a convenient representation of $P^\theta(h_j|V)$ (and $P^\theta(v_i|H)$). Consider a configuration of all nodes in which some hidden node $j$ is in state 0. Let $E_1^\theta$ denote the energy under this configuration. By turning node $j$ 'on', keeping the state of all other nodes fixed, the energy changes by $\Delta E_{2-1}^\theta = -b_j - \sum_{i=1}^{Nv} w_{ij}v_i$. Let $E_2^\theta = E_1^\theta + \Delta E_{2-1}^\theta$ be the energy in this new configuration. The conditional probability $P^\theta(h_j = 1|V)$ then takes the form of a sigmoid as

$$P^\theta(h_j=1|V) \quad = \quad \frac{P^\theta(h_j=1,V)}{P^\theta(h_j=0,V)+P^\theta(h_j=1,V)} = \frac{\exp(-E_2^\theta)}{\exp(-E_1^\theta)+\exp(-E_2^\theta)} = \sigma(-\Delta E_{2-1}^\theta) = \sigma\left(b_j + \sum_{i=1}^{Nv} w_{ij}v_i\right)$$

This yields a convenient form for computing $\langle\cdot\rangle_{data}$. While no convenient form exists for $P^\theta(H,V)$, following a similar argument $P^\theta(v_i|H) = \sigma\left(b_i + \sum_{i=1}^{Nh} w_{ij}h_j\right)$ and we can approximate $P^\theta(H,V)$ by generating samples from a Gibbs sequence alternating between $h_j^{(t)} \sim P(h_j|V^{(t-1)})$ and $v_i^{(t)} \sim P(v_i|H^{(t)})$. MCMC theory states that under mild conditions after a suitable length of time $v^{(t)}$ and $h^{(t)}$ will be samples from the stationary distribution $P^\theta(H,V)$. In this case, the negative term of the gradient is simply $\Delta w_{ij}^- = P^\theta(h_j=1, v_i=1)$.

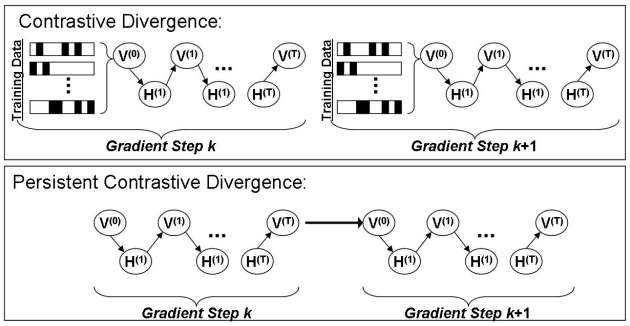



Figure 1: Illustration of CD and PCD. In CD the chain is started at each step in the ascent from training data. In PCD, the chain is 'persisted' and started from the previous model.

### 2.2 Efficient Learning in RBMs

In most applications of MCMC sampling, the goal is to approximate some (posterior) distribution $P^\theta(H, V)$. Therefore, it is often acceptable (from a computational standpoint) to run several long chains to get a more accurate estimate. In the training of RBMs, we are sampling to estimate a gradient and since this estimate is required at every step during ascent on the ML objective, it would be very time consuming to 'nail down' the gradient estimates. To make learning in RBMs tractable, one typically runs the Gibbs sequence for only a few iterations. While doing so introduces bias, we can tolerate such inaccuracy as long as the gradient estimate is in the right direction. Contrastive Divergence (CD) and Persistent CD (PCD) are two learning procedures that employ this idea [5, 10].

CD uses precisely the approximation just described - namely the sequence is run for only $T$ steps, where a step is an update of both $H^{(t)}$ and $V^{(t)}$. The initial state, $V^{(0)}$, of each truncated Gibbs sequence is selected from points in the training set. This procedure is referred to as CD-$T$ and gradient updates under this approximation are written $\Delta w_{ij} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_T$. The justification of CD lies in the following two assumptions (see [2] for more detail). If we regard the training data as samples from the true distribution, then since our model is trying to mimic the empirical distribution, it is reasonable to start each chain from points within the training set. Second, if we start the sampler from each of the $N$ training points, compute $N$ different gradient estimates and then average over the $N$ estimates, the resulting estimate should at least be in the direction of the true gradient.

PCD approximates the gradient in a slightly different manner by observing that the effectiveness of CD-$T$ hinges upon the mixing rate of the Markov chain. If the mixing rate is slow, the gradient estimates produced by CD-$T$ will poorly reflect the gradient of the actual likelihood function[10]. In such situations, we could run CD with a larger $T$, but doing so would defeat the efficiencies afforded by CD. The PCD procedure produces longer chains by noting that in any step of gradient ascent, the value of $\theta$ will change only slightly (especially with small learning rates). Therefore, we can generate a long chain by 'persisting' the chain and starting the Gibbs sequence from the end state of the previous step. The difference between CD and PCD is illustrated in Figure 1.

## 3 RBMs for Collaborative Filtering

### 3.1 The Collaborative Filtering (CF) Model

In the collaborative filtering problem we are presented with an $N \times M$ ratings matrix, where $N$ is the number of customers, $M$ is the number of items (e.g. movies) and each item may be rated using integer values from 1 to $K$ . This form of data presents two problems for the 'standard' RBM model described in Section 2. First, ratings are often non-binary ($K \neq 2$). In the Netflix data set, for example, ratings are integer values from 1 to 5. Second, the ratings matrix is typically very sparse. If all $N$ customers rated the same set of $Nv$ items, the likelihood function in 2 could be applied directly, by treating each customer's ratings as a data point. However, since different items will be rated by different users an alternative model is needed.

For $K > 2$ rating values, the conditional $P^\theta(v_i|H)$ will no longer be a sigmoid function; instead, it will take the

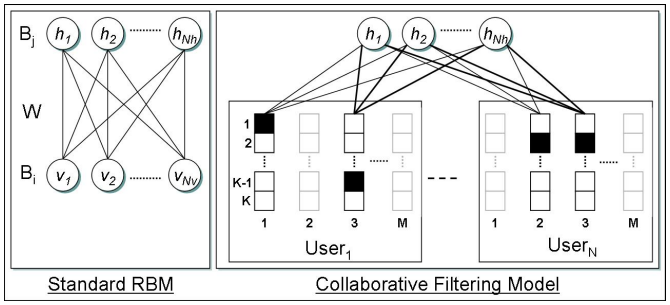



Figure 2: Standard RBM (left), comprised of $Nh$ hidden nodes and $Nv$ visible nodes. The Collaborative Filtering (CF) model (right) uses a different RBM for each user, but shares weights between users on commonly rated items. Notice that even though the set of items rated by each user are different, since both User 1 and User $N$ rated item 3 they share the weights bet-wen this item and the hidden units (as illustrated by the emboldened connections). In addition, each visible node in the CF model is K-ary rather than binary.

form of a normalized exponential or 'softmax' function

$$P^{\theta}(v_i^k = 1|H) \quad = \quad \frac{\exp\left(b_i^k + \sum_{j=1}^{Nh} w_{ij}^k h_j\right)}{\sum_{l=1}^{K} \exp\left(b_i^l + \sum_{j=1}^{Nh} w_{ij}^l h_j\right)} \tag{3}$$

where $v_i^k = 1$ indicates that $v_i = k$ and there are now a unique set of biases, $b_i^k$, and weights, $w_{ij}^k$, for each of the $K$ possible ratings.

Since users rate only a subset of the the inventory and the items rated by different users will most likely be different, it is not immediately clear how to define a joint model over all users and items. Salakhutdinov et al. address this issue by assuming a different RBM for each user and 'sharing' weights on items rated by more than one user[9]. In other words, rather than having a single distribution with $N$ i.i.d training points, there are a total of $N$ distributions - one per user - with a single observation from each distribution. These *individual* models are tied together by requiring that all users who rate the same item share the weights between the softmax for that item and all hidden units - i.e. for some item $q$, $w_{qj}^k$ and $b_q^k$ are shared by all users. This further requires that each of the $N$ RBMs have the same number of hidden nodes. The collaborative filtering model is shown in Figure 2.

Due to the $K$-ary ratings, in the CF model the energy function for each RBM is

$$E^{\theta}(V, H) \quad = \quad -\sum_{i=1}^{m}\sum_{j=1}^{Nh}\sum_{k=1}^{K} v_i^k h_j w_{ij}^k + \sum_{i=1}^{m} \ln Q_i - \sum_{i=1}^{m}\sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{Nh} h_j b_j \tag{4}$$

where $m$ denotes the subset of items rated by the current user and $Q_i = \sum_{k=1}^{K} \exp(b_i^k + \sum_{j=1}^{Nh} b_j w_{ij}^k)$ is a normalization term. The gradients of the log-likelihood are

$$\Delta w_{ij}^k = \left\langle v_i^k h_j \right\rangle_{data} - \left\langle v_i^k h_j \right\rangle_{model} \quad \Delta b_i^k = \left\langle v_i^k \right\rangle_{data} - \left\langle v_i^k \right\rangle_{model} \quad \Delta b_j = \left\langle h_j \right\rangle_{data} - \left\langle h_j \right\rangle_{model}$$

Note that these gradients are for updating the parameters of a user-specific model. The parameters of the joint, 'shared weights,' model are updated by averaging across all $N$ users.

## 3.2 The Conditional Collaborative Filtering (CCF) Model

The model in the previous section ignores an important piece of information - namely, there are some items a user has purchased (or watched) but we don't have ratings for. This implicit information may provide additional insight into a user's preferences. For example, consider a situation in which a user has watched a total of five movies, but rated none of them. If all five movies were documentaries, this suggests that the user enjoys documentaries even without explicit ratings. To leverage this implicit information, Salakhutdinov et al. consider a *conditional* model in which the joint distribution $P^{\theta}(H, V)$ is conditioned on the set of items (movies) purchased (watched) by a user. In particular, they introduce another set of binary nodes, $R$, to describe the items purchased by a user and model

the interaction between $R$ and the hidden nodes $H$. Letting $r_i = 1$ if item $i$ was purchased and 0 otherwise, the conditional distribution takes the form

$$P^{\theta}(h_j|V,R) \quad = \quad \sigma\left(b_j + \sum_{i=1}^{m}\sum_{k=1}^{K} w_{ij}^k v_i^k + \sum_{i=1}^{M} r_i d_{ij}\right)$$

where $d_{ij}$ is an element of an $M \times Nh$ matrix $D$ that models the effect of the implicit information on the hidden nodes. In this model, the rating-conditional distribution $P^{\theta}(v_i^k|H)$ is still the softmax defined in 3, but the introduction of the parameters in $D$ requires specifying the gradient $\Delta d_{ij} = \frac{\partial \log P^{\theta}(V)}{\partial d_{ij}} = \left(\langle h_j \rangle_{data} - \langle h_j \rangle_{model}\right) r_i$.

### 3.3 Learning in the Collaborative Filtering Models

Applying the CD learning procedure described in Section 2.3 to the collaborative filtering model is rather straightforward. Since there is only one training point for each of the $N$ user specific models, a single chain of length $T$ is run for each user model, starting from the lone training point. The parameters of the joint model are then updated by averaging across all $N$ users

$$\Delta w_{ij}^k \quad = \quad \frac{1}{N}\sum_{n=1}^{N} \langle v_i^k h_j \rangle_{data}^n - \frac{1}{N}\sum_{n=1}^{N} \langle v_i^k h_j \rangle_T^n \tag{5}$$

Application of the PCD learning procedure is not as straightforward. Since each user may rate a different set of items, we cannot persist a chain across different users. This means we must retain the end state of each user's chain after every iteration of the ascent.

### 3.4 Predicting User Ratings

In learning a collaborative filtering model, our goal is to be able to accurately predict ratings for items not yet purchased by a user. More formally, we want to predict the (expected) rating $E[v_s]$ of some unrated item $s \in \{1, ..., M\}$ for user $n$. The expected rating is $E[v_s] = \sum_{k=1}^{K} k \cdot P^{\theta}(v_s^k|V_n)$, where $V_n$ is an $m_n \times K$ matrix containing the known ratings of user $n$ and where the predictive distribution is

$$P^{\theta}(v_s^k|V_n) \quad \propto \quad \sum_{h_1,..,h_{Nh}} \exp\left(-E^{\theta}(v_s^k, V_n H)\right) \propto \exp(v_s^k b_s^k) \prod_{j=1}^{Nh} \left\{1 + \exp\left(\sum_{i=1}^{m_n}\sum_{l=1}^{K} w_{ij}^l v_i^l + v_s^k w_{sj}^k + b_j\right)\right\}$$

The predictive distribution can be written in this form because the summations over $m$, $K$ and $Nh$ in 4 are interchangeable and because the factorization of the RBM model allows us to interchange the sum and product when marginalizing out over all $H$. The expectation is then computed by normalizing over all $K$ possible ratings.

## 4 Experimental Evaluation

Experimental evaluation of the CD and PCD learning procedures were performed using the Netflix data set. In particular, these procedures were used to train both the collaborative filtering (CF) model presented in Section 3.1 and the conditional (CCF) model presented in Section 3.2. As with the Netflix prize challenge, performance was evaluated using the root mean squared error (RMSE) computed on the test/validation set. In addition to the RMSE, the time performance of each learning procedure was assessed by recording the difference between the start and end time of each learning epoch, where a learning epoch is defined to be the amount of time needed to complete a pass through the entire data set (i.e. all $N$ users).

### 4.1 Netflix Data Set

The Netflix prize data set contains ratings from over 480,000 Netflix users on nearly 18,000 movies.The Netflix data set comes as a set of text files (one for each movie) of the ratings made by each customer. The information in these ratings files is organized one customer per line, with each line containing a customer ID, rating, and date of the

| | CF Model | | | | | | CCF Model | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\eta$ | $\lambda$ | $\epsilon_{init}$ | $\epsilon_{final}$ | $T_{init}$ | $T_{final}$ | $\eta$ | $\lambda$ | $\epsilon_{init}$ | $\epsilon_{final}$ | $T_{init}$ | $T_{final}$ |
| CD | 0.05 | 1e-4 | 0.1 | 0.01 | 1 | 7 | 0.1 | 1e-4 | 0.1 | 0.1 | 1 | 7 |
| PCD | 0.05 | 1e-4 | 0.1 | 0.01 | 1 | 3 | 0.05 | 1e-4 | 0.3 | 0.2 | 1 | 3 |

Table 1: Experimental Parameters

rating. To get this data into a more manageable format (and one usable by Matlab) some Java code was written to randomly select a subset of the ratings of $N$ unique users on $M$ of the movies. Throughout these experiments, $N = 50,000$, $M = 1000$ and further pre-screening was performed to ensure that each of the $N$ users rated at least 10 movies. Note that this was done to allow us to establish a baseline prediction as the average across the ratings made by a user (see following section for details). The resulting $N \times M$ ratings matrix was further decomposed into 90% training, 7.5% validation and 2.5% test sets.

## 4.2 Baseline Models

To understand the performance of the CD and PCD trained models in more absolute terms, we implemented three additional baseline models. The first baseline is a Movie-Average model, where the predicted rating for user $n$ on movie $s$ is simply the average over all known ratings for movie $s$. The second baseline is a User-Average model, where the predicted rating for user $n$ on movie $s$ is simply the average over all ratings made by user $n$. The last baseline is the basic matrix factorization model presented in [7]. In this model, predictions for user $n$ on movie $m$ are made as $\hat{y}_{nm} = u_n^T v_m$, where $u_n$ and $v_m$ are $C$-dimensional vectors ($C \ll N$ and $C \ll M$). The objective to be minimized is $f = \sum_{n=1}^{N} \sum_{m=1}^{M} I_{n,m}(\hat{y}_{nm} - y_{nm}) + \lambda I_{n,m}(\|u_n\|^2 + \|v_m\|^2)$, where $I_{n,m}$ is an indicator function that is 1 if user $n$ rated movie $m$ and 0 otherwise and where $y_{nm}$ is the actual rating if it was made. This objective is minimized using stochastic gradient descent with a learning rate of 0.005 and regularization parameter $\lambda = 0.01$.

## 4.3 Algorithmic Details

The CF and CCF models were trained with either $Nh = 100$ or 200 hidden nodes. The $w_{ij}^k$'s, $b_j$'s and $d_{ij}$'s were initialized by sampling from $\mathcal{N}(0, 0.01)$. The $b_i^k$'s were initialized by taking the log of the base rate over all users - i.e. $b_i^k = \ln(\#\text{ times movie } i \text{ given rating } k)/(\#\text{ times movie } i \text{ rated by all users})$. Both the CF and CCF models were trained using the CD and PCD procedures for $t_{Total} = 50$ epochs. The joint gradient was updated in batches of 1000 users, according to the following rule:

$$\begin{aligned} w_{ij}^k &\leftarrow w_{ij}^k + inc(w_{ij}^k, t) \\ inc(w_{ij}^k, t) &= \eta \cdot inc(w_{ij}^k, t-1) + \epsilon_t(\Delta w_{ij}^k - \lambda w_{ij}^k) \end{aligned}$$

where $t$ denotes the current epoch, $\Delta w_{ij}^k$ is the gradient specified in 5, $\eta$ is a momentum parameter, $\epsilon_t$ is the learning rate during epoch $t$, $\lambda$ is a weight decay parameter and $inc(w_{ij}^k, t)$ denotes the increment to parameter $w_{ij}^k$ at epoch $t$. In our experiments, $\epsilon$ was changed according to a geometric 'cooling' schedule of the form $\epsilon_t = \epsilon_{init}(\epsilon_{final}/\epsilon_{init})^{(t/t_{Total})}$ and $T$ was increased linearly from $T_{init}$ to $T_{final}$. Table 1 contains the parameter values used in all experiments.

## 4.4 Results

Figure 3 contains a plot of the performance of the CF and CCF models trained using both CD and PCD. In particular, the plot shows the RMSE performance on the validation set over consecutive learning epochs. Despite some noise in the first few epochs, the PCD procedure slightly outperforms the CD procedure on both the CF and CCF models. In the CF models, PCD appears to converge faster than CD and in the CCF model, PCD approaches a smaller RMSE. This suggests that the mixing rate of the MCMC approximation to the true likelihood function might be slow. Despite slower convergence, the CCF model also slightly outperforms the CF models, indicating that there is a benefit in using implicit information when making predictions. Slower convergence in the CCF model

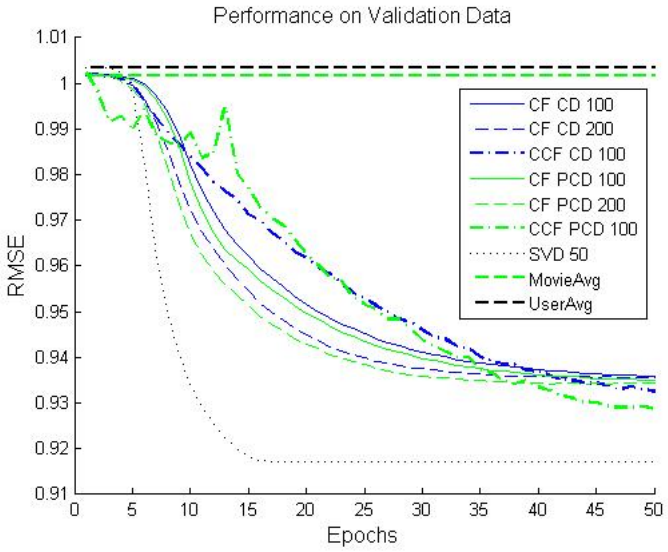



Figure 3: Performance of CD and PCD on the CF and CCF models on the validation data. Notice how CD and PCD converge much faster on the CF model than on the CCF model. Despite some noise in the first few epochs, PCD seems to outperform CD.

is contradictory to the results reported in [9]. However, in all cases the RMSE performance of the SVD model with $C = 50$ components is far superior.

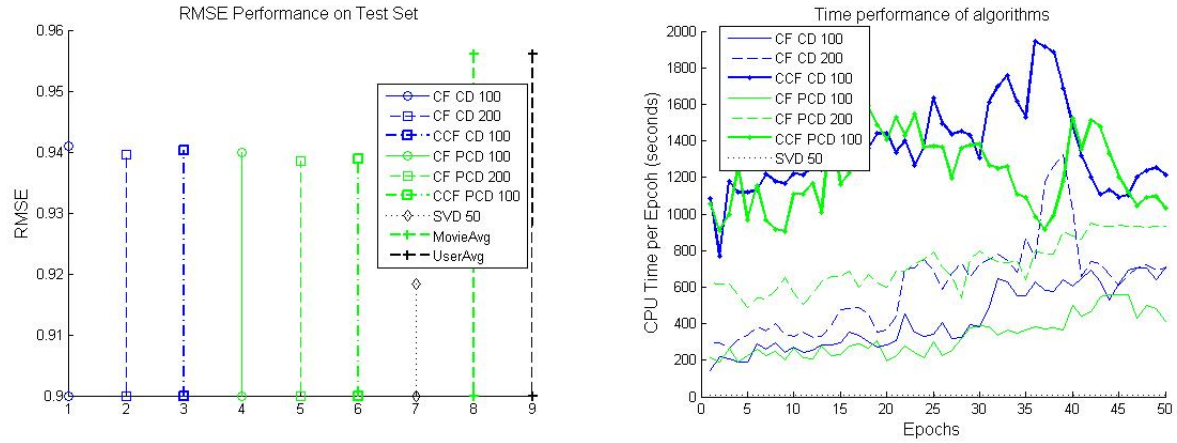
Figure 4a contains a plot of the performance of the CF and CCF models trained using CD and PCD on the test set. Once again, the SVD model yields by far the most accurate predictions. While very slight, the PCD trained models do appear to outperform the CD trained models. It is also worth noting that the CCF models with only 100 hidden nodes perform comparably with the CF models with 200 hidden nodes.

Figure 4b contains a plot of the time per training epoch for each of the different models and learning procedures. This plot is somewhat noisy because all training was performed on a shared server. Nonetheless, we can still see the linear increase in training time corresponding to when the sequence lengths are incremented. It is also clear that PCD is more efficient than CD because it requires fewer Gibbs updates near the end of training. It is also apparent that training the CCF models is more computationally intensive. This is not surprising because the gradient $\Delta d_{ij}$ must be updated prior to updating those of the weights and biases. It also clear that while PCD is more computationally efficient than CD, training SVD is by far the most efficient.

# 5 Conclusions

In this report, we investigated the use of an efficient procedure, known as PCD, for learning in two types of collaborative filtering RBM models. On these type of models, PCD-based training outperformed the CD learning procedure presented in [9] in terms of both the RMSE and total training time. However, no matter how much we tuned the parameters of the CD and PCD learning procedures, we were never able to outperform an SVD model as was reported in [9]. In addition, our tests indicate that while the RBM models can be tuned to approach the performance of the SVD model on the validation set, the SVD model outperforms the RBM models by a wide margin on the test set. Interestingly, the authors in [9] did not report RMSE performance on their test sets.

While the RBM models in [9] are an exciting and (fairly) new approach to collaborative filtering, clear computational gains are needed in order for them to be usable in real world systems. For this reason, it would be interesting to investigate the use of Welling's herding algorithm to the RBM collaborative filtering models [11].




(a) Performance of models and algorithms on the test set.

(b) Time performance of models and algorithms.

Figure 4: Performance of CD and PCD on CF and CCF models.

# References


[1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. pages 522–533, 1987.

[2] Yoshua Bengio and Olivier Delalleau. Justifying and generalizing contrastive divergence. *Neural Comput.*, 21(6):1601–1621, 2009.

[3] Alexander Felfernig, Gerhard Friedrich, and Lars Schmidt-Thieme. Recommender systems. *IEEE INTELLIGENT SYSTEMS*, 22(03).

[4] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm, 2000.

[5] Geoffrey Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:2002, 2000.

[6] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[7] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[8] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. pages 175–186. ACM Press, 1994.

[9] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 791–798, New York, NY, USA, 2007. ACM.

[10] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, New York, NY, USA, 2008. ACM.

[11] Max Welling. Herding dynamical weights to learn. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128, New York, NY, USA, 2009. ACM.