The application of Deep Learning in Collaborative Filtering

Kai Lu

School of Engineering
University of California Santa Cruz
{kailu}@soe.ucsc.edu

1 Introduction

As the fast development of Internet, users are facing severe "Information overload" problems. Recommender systems which aim to solve the information problem have been developed very in fast in the past 20 years. Recommender systems aim to provide users with a list of items which can satisfy their preferences. Among the approaches in Recommender systems, Collaborative Filtering [1] mainly utilize user's preference histories to build model and make predictions for the items which the users might be interested in.

Early approaches for collaborative filtering assume that similar users (items) have similar interests, i.e., utilize nearest neighborhood method to make recommendations. This method also named as memory-based approaches. Grouplens, for example, was an early Internet news system that used the correlation coefficient between user ratings of articles as a similarity measure [2]. However, memory-based approaches do not scale without further approximation because they require access to the ratings of the entire set community. Another great challenge for Collaborative Filter problem is ratings are severe sparse and make the memory-based approaches perform very bad. For these reasons, the model based approaches such as singular value decomposition (SVD) which are based on matrix factorization have also been proposed ([3]). 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 preferences using hidden variables over user-item-rating triplets [4]. Unlike traditional clustering models that use latent variables to indicate which cluster (i.e., group of similar users) a user belongs to Hoffman's probabilistic latent semantic analysis (pLSA) model links users and their ratings to different latent causes.

Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence. Deep Learning attracts a lot of researcher's attention as its successful application in speech, video, and image recognition. Restricted Boltzmann Machines (RBMs) is one of the approaches in Deep Learning. Most recently, Salakhutdinov et al. revised the classic RBMs into the collaborative filtering task [5]. These models were trained using an efficient learning procedure, named Contrastive Divergence (CD), which maximized an approximation to the true likelihood function. Furthermore, the authors also proposed a conditional RBMs model and report that the performance on the Neflix data set is superior to the competitive SVD model [6]. In this project, I mainly explore the RBMs' model on the collaborative filtering task and verify its effectiveness.

2 RBM Introduction

A RBM is a specific type of undirected bipartite graphical model consisting of two layers of binary variables : hidden (H) and visible (V) with no intra-layer connections shown as Figure 1.

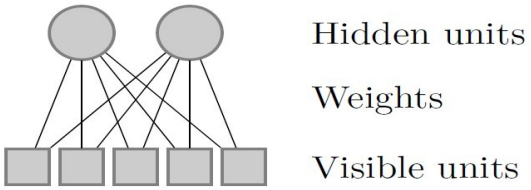

Fig. 1. The structure of an RBM with 2 hidden units and five visible units.

The joint probability 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 follows:

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

In equation (1), 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 (0/1), 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 represent the bias of the i^{th} visible nodes and b_j represent the bias of the j^{th} hidden nodes.

The parameters in the RBM include weights and biases, they can be described using a parameter vector θ, the probability of observing a single Nv-dimensional data point V is given by the marginal probability:

$$P^\theta(V) = \sum_h P^\theta(V,H) = \frac{1}{Z^\theta}\sum_h exp(-E^\theta(V,H))$$

Where Z^θ is the partition function. For a data set containing N independent and identically distributed observations, the log-likelihood can be written as:

$$l(\theta) = \frac{1}{N}\sum_{n=1}^{N} lnP^\theta(V_n) = \frac{1}{N}\sum_{n=1}^{N} ln\sum_h exp(-E^\theta(V_n,H)) - ln(Z^\theta) = l(\theta)^+ - l(\theta)^- \tag{2}$$

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

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

Use Maximum likelihood in the model need to take the gradient of $l(\theta)$. The gradient of the positive term is :

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

The quantity is just a conditional expectation with respect to data. Similarly, the gradient of the negative term is:

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

The negative term gradient is just an expectation with respect to the model. As both of the terms involve expectations, the full gradient can be written as:

$$\triangle w_{ij} = \frac{\partial log(P^\theta(V))}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

To compute the gradient, we need to evaluate $P^\theta(H|V)$ and $P^\theta(H,V)$. RBM's structure 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^θ denote the energy under this configuration. By turning node j 'on', keeping the state of all other nodes fixed, the energy changed by $\triangle E_{2-1}^\theta = -b_j - \sum_{i=1}^{Nv} w_{ij} v_i$. Let $E_2^\theta = E_1^\theta + \triangle E_{2-1}^\theta$ be the energy in this new configuration. The conditional probability $P^\theta(h_j = 1|V)$ then takes the form of a sigmoid as follows:

$$P^\theta(h_j = 1|V) = \frac{P^\theta(h_j = 1, V)}{P^\theta(h_j = 0, V) + P^\theta(h_j = 1, V)} = \frac{exp(-E_2^\theta)}{exp(-E_1^\theta) + exp(-E_2^\theta)}$$

$$= \sigma(-\triangle E_{2-1}^\theta) = \sigma\left(b_j + \sum_{i=1}^{Nb} w_{ij} v_i\right)$$

This yields a convenient form for computing $\langle \cdot \rangle_{data}$. While there is no convenient form exists for $P^\theta(H,V)$, following a similar argument $P^\theta(v_i|H) = \sigma(b_i + \sum_{i=1}^{Nh} w_{ij} h_j)$ and we can approximate $P^\theta(H,V)$ by using Gibbs sampling alternatively between $h_j^{(t)} \sim P(h_j|V^(t-1))$ and $v_t^{(t)} \sim P(v_i|H^{(t)})$. 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 $P^\theta(H,V)$.

2.1 Contrastive Divergence

Obtaining unbiased estimates of log-likelihood gradient using MCMC methods typically requires many sampling steps, so it will take a long time to generate samples. Recently it was shown that estimated obtained after running the chain for just a few steps can be sufficient for model training [7]. This leads to contrastive divergence (CD) learning, which has become a standard way to train RBMs.

The idea of T-step contrastive divergence learning (CD-T) is quite simple: Instead of approximating the second term in the log-likelihood gradient by a sample from the RBM-distribution (which would require to run a Marko chain until the stationary distribution is reached), a Gibbs chain is run for only T steps (and usually $T = 1$). The Gibbs chain is initialized with a training example $v^{(0)}$ of the training set and yields the sample v^T after T steps. Each step t consists of sampling $h^{(t)}$ from $p(h|v^{(t)})$ and sampling $v^{(t+1)}$ from $p(v|h^{(t)})$ subsequently. The gradient (Equation (2)) w.r.t. θ of the log-likelihood for one training pattern $v^{(0)}$ is then approximate by:

$$CD_k(\theta, v^{(0)}) = -\sum_h p(h|v^{(0)}) \frac{\partial E(v^{(0)}, h)}{\partial \theta} + \sum_h p(h|v^{(T)}) \frac{\partial E(v^{(T)}, j)}{\partial \theta}$$

The derivatives in direction of the single parameters are obtained by "estimating" the expectation over $p(v)$ and the single sample $v^{(T)}$.

3 RBM in Collaborative Filtering

3.1 Basic RBMs in Collaborative Filtering

In the collaborative filtering (CF) problem, we are presented with an $N \times M$ ratings matrix, where N is the number of customers, M is the number of items and each item rated using

integer values from 1 to K. If $K = 2$, the form of data presents the problem of the 'standard' RBM model described in Section 2. While ratings are often non-binary. In the Netflix data set, $K \in [1, 5]$. Moreover, the ratings matrix is severe sparse. So the binary RBM model can not used to deal with the problem.

The CF RBMs treat each user as a single training case for an RBM which had M "softmax" visible units symmetrically connected to a set of binary hidden units. Every RBM has the same number of hidden units but an RBM only has visible softmax units for the movies rated by that user. Each RBM only has a single training case, but all of the corresponding weights and biases are tied together, so if two users have rated the same movie, their two RBM's must use the same weights between the softmax visible unit for that movie and the hidden units. The comparison between the standard RBMs and RBMs in Collaborative Filtering is shown in Figure 2.

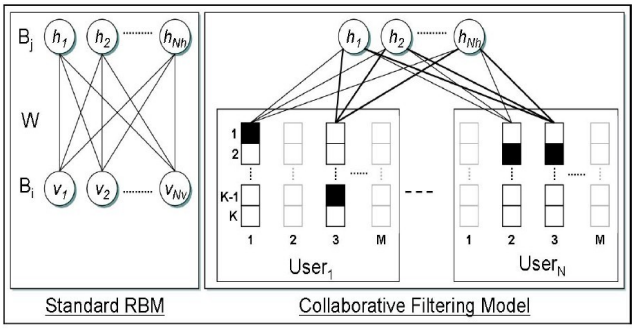

Fig. 2. Comparison between binary RBM and RBM in Collaborative Filtering

Suppose a user rated m movies. Let V be a $K \times m$ observed binary indicator matrix with $v_i^k = 1$ if the user rated movie i as k and 0 otherwise. We also let h_j, $j = 1, ..., F$, be the binary values of hidden (latent) variables, that can be thought of as representing stochastic binary features that have different values for different users.

The new conditional probability $P^\theta(vi|H)$ will take softmax function instead of sigmoid function:

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

$$P^\theta(h_j = 1|V) = \sigma(b_j + \sum_{i=1}^{m} \sum_{k=1}^{K} v_i^k W_{ij}^k) \tag{4}$$

In the CF model, the energy function for each RBM is:

$$E^\theta(V, H) = -\sum_{i=1}^{m} \sum_{j=1}^{F} \sum_{k=1}^{K} W_{ij} h_j v_i^k + \sum_{i=1}^{m} logZ_i - \sum_{i=1}^{m} \sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{F} h_j b_j \tag{5}$$

Where $Z_i = \sum_{l=1}^K exp(b_i^l + \sum_j h_j W_{ij}^l)$ is the normalization term that ensure that $\sum_{l=1}^K p(v_i^l = 1|h) = 1$. The items with missing ratings do not make any contribution to the energy function.

The learning for the parameters are similar with that of the binary RBMs, and we utilize "Contrastive Divergence" (CD) to learn the weights and bias, the gradient are as follows:

$$\triangle w_{ij}^k = \frac{\partial log(P^\theta(V))}{\partial w_{ij}^k} = \xi \cdot (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_T)$$

$$\triangle b_i^k = \xi \cdot (\langle v_i^k \rangle_{data} - \langle v_i^k \rangle_T) \qquad \triangle b_j = \xi \cdot (\langle h_j \rangle_{data} - \langle h_j \rangle_T)$$

Given the observed ratings V, we can predict a rating for a new query movie q as follows:

$$p(v_q^k = 1|V) \propto \sum_{h_1,\ldots,h_p} exp(-E(v_q^k, v, h)) \propto exp(v_q^k b_q^k) \prod_{j=1}^{F} (1 + exp(\sum_{ij} v_q^k W_{qj}^k + b_j))$$

Once we obtain un-normalized scores, we can perform normalization over K values to get probabilities $p(v_q = k|V)$ and take the expectation $E[v_q]$ as out prediction.

3.2 Conditional RBMs in Collaborative Filtering

In the basic CF RBMs model, it ignores an important factor, i.e., there are some items users have rated or watched while we don't have ratings. This implicit information also provides additional insight into a user's preferences. E.g., if we already know that a user has "Rocky 5", we will have a good bet about the kinds of movies he likes.

The conditional RBM model takes this extra information into account. Define $r \in \{0,1\}^M$ is a binary vector of length M (total number of movies), indicating all the movies the user rated. The idea is to define a joint distribution over (V, h) conditional on r. In Conditional RBM, r will affect the states of the hidden units shown in Figure 3.

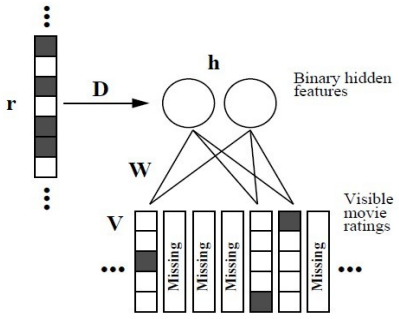

Fig. 3. Conditional RBM in Collaborative Filtering

The model is shown as :

$$p(v_i^k = 1|h) = \frac{exp(b_i^k + \sum_{j=1}^{F} h_j w_{ij}^k)}{\sum_{l=1}^{K} exp(b_i^l + \sum_{j=1} Fh_)}$$

$$p(h_j = 1|V, r) = \sigma\left(b_j + \sum_{i=1}^{m}\sum_{k=1}^{K} v_i^k W_{ij}^k + \sum_{i=1}^{M} r_i D_{ij}\right)$$

Here D_{ij} is an element of an $M \times F$ that models the effect of the implicit information on the hidden nodes. Learning D using CD is similar to learning biases and takes the form:

$$\triangle D_{ij} = \xi\left(\langle h_j \rangle_{data} - \langle h_j \rangle_T\right) r_i$$

4 SVD Model

SVD seeks a low-rank matrix $X = UV^{'}$, where U is $N \times K$ and V is $M \times C$ matrix, this factorization gives a low dimensional numerical representation of both users and items. In the case of the given problem, most of the entries in the ratings Y will be missing and the unknown ratings cannot be represented by zero. So the sum-squared distance is minimized with respect to the partially observed entries of the target matrix Y. Unobserved entries of Y are predicted by using the inner product of user factor vector and item factor vector (i.e., $\hat{r}_{ui} = \sum_{k=1}^{C} u_{ik} v_{jk}$).

The optimization objective function is as follows:

$$f = \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij}(u_i v_j^{'} - Y_{ij})^2 + \lambda \sum_{ij} I_{ij}(\parallel u_i \parallel^2 + \parallel v_j \parallel^2)$$

$$(U^*, V^*) = argmin(f)$$

Here I_{ij} the indicator function, $I_{ij} = 1$ if user i rated movie j, otherwise $I_{ij} = 0$. The gradient function for SVD is :

$$\frac{\partial}{\partial u_{ik}} = -e_{ij} v_{kj} + \lambda u_{ik} \qquad \frac{\partial}{\partial v_{jk}} = -e_{ij} u_{ik} + \lambda v_{kj}$$

Here $e_i j$ is the prediction error. We initialize the matrices U and V with small random values sampled from a zero-mean normal distribution with standard deviation 0.01, and we update the weights in the direction opposite to the gradient:

$$u'_{ik} = u_{ik} + \eta(e_{ij} v_{kj} - \lambda u_{ik}) \qquad v'_{kj} = v_{kj} + \eta(e_{ij} u_{ik} - \lambda v_{kj})$$

5 Experiments

5.1 Data set

We use Netflix movies rating data to compare the related model's performance. The data were collected between Oct. 1998 and Dec. 2005. The whole data set are split into training data, validation data and test data. As the test data is not public, so we just compare the results on the validation data. The information about the ratings are shown in Table 1:

Table 1. Netflix rating information

#User	#Item	rating value	#Train	#Validation
480,189	17,770	[1,5]	99, 072, 112	1,408,395

As a baseline, Netflix provided the score of its own system trained on the same data, which is *0.9514*.

5.2 Evaluation Metric

We use root mean squared error (RMSE) to evaluate the referred models. Compared with MAE, RMSE gives more weights for predictions with bigger errors and is used more in recent years. The evaluation rule is in the following form:

$$RMSE = \sqrt{\frac{\sum_{(u,i) \in T} (r_{ui} - \hat{r}_{ui})^2}{\parallel T \parallel}}$$

5.3 Experimental Results

To compare the models, we train RBMs with $F = 100$, and $C = 40$ firstly. Then we compared the approaches RBM, conditional RBM, and SVD under the same conditions. The experimental results are shown in Figure 4:

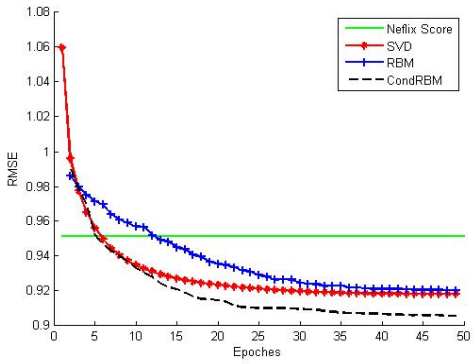

Fig. 4. Performance of various models on the validation data

Figure 4 shows the performance of referred models on the validation data set. The x-axis shows the epoches, i.e, the iterations number, and the y-axis shows the RMSE, the lower value, the model is better. Compared with the baseline model, we can see that all of RBMs, CondRBMs and SVD perform significantly better. Compared between RBM and SVD, we can see that RBM has shown very good performance on the first 5 epoches, it firstly outperforms on the first two iterations, while RBM converges very slow and needs more epoches to converge. The whole performance of RBMs is a little worse than that of SVD and it achieves closer to SVD as the epoches grows. When comparing Conditional RBM and RBMs, we can see that Conditional RBM performs greatly better than RBM, it justifies the effectiveness of utilizing the items which we have known users have rated while the rating values are unknown. When compared CondRBM and SVD, we can see that as the epoches goes, the CondRBM outperforms SVD after 10 epoches. So we verify the superiority of Conditional RBM based on $C = 40$.

To see the effectiveness of RBMs, we further conduct experiments with varying factor size C. The results are shown in Figure 5.

From Figure 5, we can clearly see that through increasing the factor size, all models achieve better performance, and the RMSE decreases very fast before $C = 50$. While when $C > 50$, although the models we further improve the performance, but the improve become less as the factor size grows. When comparing the results of all the three models, we see that the gap between between SVD and RBM become smaller as the factor size grows, and CondRBM outperforms SVD more as the factor size grows, so we can conclude the effectiveness of RBMs in collaborative filtering.

6 Conclusions

In this project, I mainly explore the performance of RBMs model in the collaborative filtering, and compare the performance of RBMs, Conditional RBMs and SVD model. Through conducting experiments on the Netflix ratings, I verify the effectiveness of RBM model.

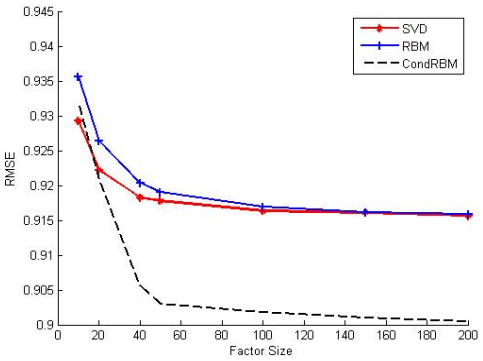

Fig. 5. Performance of RBMs, CondRBM, SVD on the validation data with varying factor size

In my experiment, one of the disadvantages of the RBM model is that it takes a long time to run for each iteration, and also it needs many epoches to converges. Therefore, exploring more efficient learning method attracts me a lot. Although RBMs costs a lot time to train, but it can greatly decrease the prediction errors through ensembling with other models. So it is meaningful to do research on it.

References

1. Adomavicius, G., Tuzhilin, A.: Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE TKDE 17(6) (2005) 734–749
2. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on Computer supported cooperative work, ACM (1994) 175–186
3. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval **4**(2) (2001) 133–151
4. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Transactions on Information Systems (TOIS) **22**(1) (2004) 89–115
5. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th international conference on Machine learning, ACM (2007) 791–798
6. Funk, S.: Netflix update:. http://sifter.org/simon/journal/20061211.html (2006)
7. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural computation **14**(8) (2002) 1771–1800