A Brief Survey on RBM and Matrix Factorization for CF

Dingquan Wang

F0703028
5070309650

1 Introduction

In this semester, I mainly focus my laboratory on Restrict Boltzmann Machine and finished a Collaborative Filtering predictor using C++ with the help of Chen Tianqi. Then we try this program on the data set of movielens which is a recommender system and virtual community website that recommends films for its users to watch , based on their film preferences. In the end of April, I participated the KDD Cup 2010 year's Challenge with Zhang Weinan and Zheng Zhao. This year's challenge is to predict student performance on mathematical problems from logs of student interaction with Intelligent Tutoring Systems. This task presents interesting technical challenges, has practical importance, and is scientifically interesting. We reduced this problem to a Collaborative Filtering task, and used the Matrix Factorization approach to solve the problem. This survey is mainly expressing the two topics above I had faced this semester.

2 Restrict Boltzmann Machine

2.1 Boltzmann Machine

A Boltzmann machine is the name given to a type of stochastic neural network, who has an global "Energy" with the units biases and the relative feed in between each units. The "Energy" talked above is defined as follows:

$$E = -\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i \tag{1}$$

In the Boltzmann Machine model, the likelihood of this model can be described as:

$$P(x) \propto exp(\frac{E}{T}) \tag{2}$$

where T is considered to the temperature of this model, indicating the stochastic factor of the model.

In most utilization of the Boltzmann Machine, units of the network are always separated into visible unit v and hidden unit h, the visible unit is always used to

fit the observable facts, and the hidden unit is the higher level feature indicated by the model.Then the "Energy" can be written in the matrix multiplication form as:

$$E(v,h) = -b'v - c'h - h'Wv - v'Uv - h'Vh \tag{3}$$

Since the likelihood is the function of all observed data, then it is the function of the parameter given only by the visible unit. Then we can get the likelihood as follows:

$$P(v) = \sum_h P(v,h) \tag{4}$$

such that:

$$P(v,h) = \frac{exp(-E(v,h))}{Z} \tag{5}$$

For the additional talk, we rewrite the likelihood into log-likelihood:

$$\log P(v,h) = \log \sum_h exp(-E(v,h)) - \log \sum_{v,h} exp(-E(v,h)) \tag{6}$$

We use the gradient ascent to justify the parameters fitting the model, the gradient of single parameter θ can be written as follows:

$$\frac{\partial \log P(v)}{\partial \theta_j} = -\frac{\sum_h exp(-E(v,h))\frac{\partial E(v,h)}{\partial \theta_j}}{\sum_h exp(-E(v,h))} + \frac{\sum_{v,h} exp(-E(v,h))\frac{\partial E(v,h)}{\partial \theta_j}}{\sum_{v,h} exp(-E(v,h))} \tag{7}$$

$$= -\sum_h P(h|v)\frac{\partial E(v,h)}{\partial \theta_j} + \sum_{v,h} P(v,h)\frac{\partial E(v,h)}{\partial \theta_j} \tag{8}$$

$$= - < \frac{\partial E(v,h)}{\partial \theta_j} >_{h|v_i} + < \frac{\partial E(v,h)}{\partial \theta_j} >_{v_i,h} \tag{9}$$

Therefore,from the equation, if sampling from the model was possible, we can obtain a stochastic gradient for use in training the model, as follows. Two samplers are necessary: h given x for the first term, which is called the positive phase, and an (x,h) pair from $P(x,h)$ in what is called the negative phase. A general sampling method is called the Gibbs sampling, which is a special case of MCMC to generate random values from given distribution.

In former Boltzmann Machine, the linkings between different units has no restricted. It can be inferred that the sampling process is of highly cost to making the effective training and predicting process.

In this case the Restrict Boltzmann Machine is proposed by largely reducing the complexity of the former Restrict Boltzmann Machine model. Which has achieve a considerable trade-off between the effectiveness and the accurate of the model.

2.2 Model Description

In the Restrict Boltzmann Machine model, the linkings between the same type of units are extracted, only leaves the edges between visible unit and hidden unit, i.e. the neural network is simplified into a bipartite graph whose visible unit is on one side while the hidden unit on the other. Thus the Energy function can be simplified into

$$E(v,h) = -b'v - c'h - h'Wv \tag{10}$$

where Z is the normalize where $Z = \sum_{x,h} exp(-E(v,h))$ is a normalization term. Where b is the bias of the visible unit, and c is the bias of hidden unit. W is the interaction terms weighting the links between hidden unit and visible unit.

Benefited from this simplification we can keep on refine the equation above to:

$$\frac{\partial \log P(x)}{\partial \theta_j} = -\sum_h P(h|v)\frac{\partial E(v,h)}{\partial \theta_j} + \sum_{v,h} P(v,h)\frac{\partial E(v,h)}{\partial \theta_j} \tag{11}$$

$$= -P(H_i|v)\cdot v_j + E_X[P(H_i = 1|V)\cdot V_j] \tag{12}$$

and by the property of RBM, we can find that:

$$E[H_i|v] = P(H_i = 1|V = v) = \frac{1}{1 + exp(-c_i - W_i x)} = sigm(c_i + W_i x) \tag{13}$$

We can find that we can calculate the expectation precisely and effectively without any sampling process, which can largely speed up the training operation.

For the negative phase we can using an approximation approach called Contrastive Divergence.The $k - step$ Contrastive Divergence involves a second approximation besides the use of $MCMC$ to sample from P. This additional approximation introduces some bias in the initial gradient by starting from the observed data, by this means, the $MCMC$ process can only run a few of k steps.

3 Matrix Factorization for Collaborative Filtering

3.1 Collaborative Filtering

Collaborative Filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets. Collaborative filtering methods have been applied to many different kinds of data including sensing and monitoring data - such as in mineral exploration, environmental sensing over large areas or multiple sensors; financial data - such as financial service institutions that integrate many financial sources; or in electronic commerce and web 2.0 applications where the focus is on user data, etc.

A most simple Collaborative Filtering task is giving a user-item matrix, the predictor try to fill the missing values into the matrix by observing the existing entries. Collaborative filtering systems usually take two steps:

- Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
- Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user

Currently there are mainly two types of approach for Collaborative Filtering:

Memory-Based This mechanism uses user rating data to compute similarity between users or items. This is used for making recommendations. This was the earlier mechanism and is used in many commercial systems. It is easy to implement and is effective. Typical examples of this mechanism are neighborhood based CF and item-based/user-based top-K recommendations.

The advantages with this approach is the explainability of the results, which is an important aspect of recommendation systems. It is easy to create and use. New data can be added easily and incrementally. It need not consider the content of the items being recommended. The mechanism scales well with co-rated items.

Model-Based Models are developed using data mining, machine learning algorithms to find patterns based on training data. These are used to make predictions for real data. There are many model based CF algorithms. These include Bayesian Networks, clustering models, latent semantic models such as singular value decomposition, probabilistic latent semantic analysis, Multiple Multiplicative Factor, Latent Dirichlet allocation, markov decision process based models.

There are several advantages with this paradigm. It handles the sparsity better than memory based ones. This helps with scalability with large data sets. It improves the prediction performance. It gives an intuitive rationale for the recommendations.

3.2 Matrix Factorization

The Matrix Factorization algorithm is a model based approach for Collaborative Filtering, which is focus on finding the latent factor linked both user and item, and making prediction by product the latent factor.

Consider the input is a user-item matrixr_{ui}, we propose to find a joint latent factor space, such that user-item interactions are modeled as inner products in this space. Accordingly, each item i is associated with a vector $q_i \in \mathbb{R}$, and each user u is associated with a vector $p_u \in \mathbb{R}$. For a given item i, the elements off q_i measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product $q_i^T p_u$ captures the interaction between user u and item i the user's overall interest in the items characteristics. This approximates user $u's$ rating of item i, which is denoted by r, leading to the estimate:

$$r_{ui} = q_i^T p_u \tag{14}$$

If we decided the q and p, the ratings can be effectively predicted by above procedure. Now, the major challenge is computing the mapping of each item and user to factor vectors, i.e. to decide q and p, the same method as Restrict Boltzmann Machine, we use the gradient descent method by minimal the loss function as follows:

$$\sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| q_u \|^2) \tag{15}$$

Here λ is to avoid the model parameter overfitting the training data. Now define:

$$e_{ui} = r_{ui} - q_i^T p_u \tag{16}$$

Then the gradient is given as:

$$\nabla q_i = e_{ui} \cdot p_u - \lambda \cdot q_i \tag{17}$$

$$\nabla q_u = e_{ui} \cdot q_i - \lambda \cdot p_u \tag{18}$$

Then the updating law:

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i) \tag{19}$$

$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u) \tag{20}$$

Thus, after several iteration, the loss function may fall into a local minimal area, and the training step is over . The benefit of the Matrix Factorization model is that it is very easy to add features into the loss function and training different models fitting different data.

In this year's KDD Cup with the help of Chen Tianqi, we had implemented the Matrix Factorization algorithm on large scale data set, and try a lot of features such as the problem count, problem id, user history and user profile. And the best result of our result under RMSE is 29.98% (the first place is under RMSE of 27.3%), indicating that the MF of Collaborative Filtering really works and we still have spaces to improve.