Final Project Proposal

Predict decisions on Weibo with Factorization Machine

Kyuhyun Cho(20133677) Seunghwan Baek(20133331)

Joongyoun Lee(20115494) Hyeonjun Jo(20133693)

**1. PROBLEM**

How can we get to know if a person chooses something or not? Basically we can ask them directly. However, how about guessing the decision without explicit information? It is a different problem. We may need to gather available data which seems to be related on the decision. At this point, we think that perhaps A.Is could do the ‘guessing’ task. They could collect related information, look into it, and compile statistics on it. Then, they could find some rules and finally make a decision. This is what we are doing here. There are amazingly countless data on webs especially Weibo. We are going to create an A.I using that data, and train it. Finally the A.I is going to make a decision if a user will choose to follow an item (a person, a group, a product anything) recommended.

The predictor may perform a regression based on training data, but there could be a lot of noise in the raw data. Thus, we have to choose data to train A.I. And we believe it should be the most challenging problem. The second problem we have to consider is the “overfitting”. If our predictor is too optimized on training data, a precision on the test data could be very low. The third-problem is the “cold-start” problem.

**2. DATA**

We are going to look into through all data, and pick some data only we need. A preprocessing will be performed on selected dataset to remove noise. We could remove some dummy users or duplicate data. We are also concerning the dimensionality reduction technique such as PCA not only to enhance precision, but also to reduce prediction and training time.

**3. METHOD**

We are going to adopt Factorization Machines which are a new model that combines the advantages of Support Vector Machines with factorization models. FMs have following advantages [1].

1. FMs allow parameter estimation under very spare data
2. FMs have linear complexity and can scale to large datasets with 100M of training instances.
3. FMs can mimic state-of-the-art models like biased MF, SVD++, PITF or FPMC because FMs are a general predictor that can work with any real valued feature vector.

Before mentioning about how to do, there is the list of what we are going to focus on. The first is temporal dynamics. This means trends of items. If an item is recommend to users when the item is very trendy, the users’ following the item has relatively high probability. So we are going to deal with this temporal dynamics. The second is a users’ active period. Supposed a user is really actively doing Weibo at the moment. And there is a recommendation one item to that user. There would be a stronger chance to accept the item compared to when the user is not so active. The next is the implicit feedback. This is not just the target user’s interest. We are going to focus more on the users who are similar to the target user. If there are some common items among the users, we could guess that the target user also could be interested in the items. Lastly, the sequential information is also really important. This can be called pattern. We had better know this information to guess. This has varied data. One of them is the time gap between one item recommended and another item recommended. With this gap, we can get to know how long users spend their time some items.

And the following issue is how to do these FMs. The answer is the libFM. This is an implementation of FM. For libFM we need a binary data format though, there is a converter form the standard text format to the libFM data format [2]. We will see through all about this soon.

Lastly, we have been looking for other useful and creative things such as methods, logics, or features which can be help for this problem. We could not help using FMs because of its performance. However we reckon it would be worth to implement something our own.

**4. IMPLEMENT**

We mainly use C++, because C++ is easy to implement libFM. And we probably use Python partially.

**5. REFERENCES**

[1] Steffen Rendle, “Factorization Machines,” in the Institute of Scientific and Industrial Research Osaka University, Japan

[2] Steeffen Rendle, “Factorization Machines with libFM,” in University of Konstanz