Comparison of Underlying Algorithms in Recommendation Systems

Shuyi Wang[1], Mufeng Yang[2*], Ziyang Liu[3], and Linyan Li[4]

[1]*Department of Mathematics, University of North Carolina at Chapel Hill, Chapel Hill, USA*
[2]*Department of Computer Science, Fu Jen Catholic University, Taiwan, China*
[3]*Department of Mathematics, University of California, Santa Barbara, Santa Barbara, USA*
[4]*Department of Mathematics, Henan University of Economics and Law, Henan, China*
[*]*Corresponding author: yangmufeng233@gmail.com*

Abstract

Recommendation systems have been one of the most popular fields in machine learning and received significant attentions due to its applicability to everyday lives. This paper reviews the developing history of recommendation systems from CF to more recent LS-PLM. In this work, we first introduce representative algorithm in each time period and shed light on how they evolved with various demands of mobile applications and scientists' increasing understanding of general machine learning. To concretely illustrate each algorithm, we present the basic framework, underlying mechanics, application methods, and potential weaknesses for comparison. We also run numerical experiments on four of the introduced algorithms, FM, FFM, GBDT+LR, and LS-PLM to test their performance on a particular dataset, KKBox's Music Recommendation Challenge. Finally, we conclude with higher accuracy of FFM and attribute its advantage to better dealing with sparser features. This work presents meaningful viewpoints on comparison of recommendation algorithms through descriptions and experiments and further facilitates the emergence of new state-of-the-art algorithms.

Keywords

Factorization Machine, Field-aware Factorization Machine, Gradient Boost Decision Tree+Linear Regression, Large Scale Piece-wise Linear Model

1. Introduction

Since the concept of online shopping first appeared in the late 20 century, it has played an essential role in daily business. For example, Amazon has over 300 million active customer accounts worldwide with 1.6 million packages and 385 dollar sales reached per day! As more and more products are selling online, one problem came up: how to recommend those products to their potential customers who need them? Amazon used a recommendation system that increased about 35% of over sales. In fact, not only goods companies like Amazon but also companies like Instagram, which recommend a person from social media to you, used a recommendation system in their operation.

One of the fundamentals of the recommendation system is embedding. Embedding is a relatively low-dimension space into which you can translate high-dimension vectors. Objects are distributed in such space through their feature – one common feature could be one dimension. Such as one dimension shows the size of objects, another dimension shows what age group uses more. Some companies use embedding to process the similarities between objects and display their similarities by distance in the space between them. For example, both burger and sandwich are food, so the distance between burger and sandwich should be smaller than the distance between burger and fossil. Also, the distance between objects should have some meaning in Embedding – the distance between women and men should approximately equal the distance between males and females. After having the relation, or distance, between different objects, the recommendation system would recommend similar goods to customers based on their preference or history purchase record.

2. Pre- Deep Learning Era—Recommender System

This is the evolution diagram of the recommender system model.

Figure 1: Algorithms of recommender system

Figure 1 shows the traditional recommendation model consists of the following four components: CF, LR, FFM and the combined model (GBDT+LR).

2.1. Collaborative Filtering

CF is a collaborative process of filtering a large amount of information through feedback, reviews and opinions, and filtering the information that may be of interest to the target user. About the development of CF: Tapestry was the first design to apply CF, solving the information load problem of the research center mail system with no way to filter the classification; [1] GroupLens as a milestone, the system was applied to news filtering, helping readers filter their interested news content through the record of readers' rating of the content. Similar web recommender platforms are still prevalent today, like YouTube; it has also achieved outstanding success in the field of e-commerce.

CF is mainly divided into User-based, Item-based, Memory-based and Model-based.

User-based: First, collect information that represents user interests, usually user ratings or evaluations, which are called "active ratings", and replace user evaluations by the system through user behavior, which is called "passive ratings". The starting point of UserCF is to calculate the similarity between two users. For example to get user A is interested in whether the commodity X, based on the historical evaluation data (user A historical evaluation data for other commodities, and other users of the historical evaluation data) of these commodities to analysis with the user A to find most similar n users, and then the synthetic evaluation of similar users for the commodities, the weighted average of user similarity and similar users' evaluation is usually used to obtain the evaluation prediction of target users. Through the operation of this formula, the final recommendation list can be sorted according to the predicted score.

$$R_{u,s} = \frac{\sum_{s \in S}\left(w_{u,s} \cdot R_{s,p}\right)}{\sum_{s \in S} w_{u,s}} \tag{1}$$

The weight $w_{u,s}$ represents the similarity between user u and user s, $R_{s,p}$ represents user s's rating of item p.

If the rating is negative, the recommender system will not recommend X to A. Otherwise, the commodity X is recommended to the user A.

Item-based: Similar to user-based, the historical data information should be collected first, and then calculate the similarity between the evaluated commodities and the commodities with prediction. Find out k commodities that are most similar to the commodities to be predicted, and the similarity should be used as the weight to rank the scores of the evaluated commodities by weighting them, and finally the predicted value of the commodities to be predicted can be obtained.

How to calculate the similarity?

- Cosine Similarity measures the vector angle between user vector i and user vector j :

$$\text{sim}(i,j) = \cos(i,j) = \frac{i \cdot j}{\|i\| \cdot \|j\|} \tag{2}$$

- Pearson's correlation coefficient is modified by using average user scores:

$$\text{sim}(i,j) = \frac{\sum_{p \in P}\left(R_{i,p} - \overline{R_i}\right)\left(R_{j,p} - \overline{R_j}\right)}{\sqrt{\sum_{p \in P}\left(R_{i,p} - \overline{R_i}\right)^2}\sqrt{\sum_{p \in P}\left(R_{i,p} - \overline{R_j}\right)^2}} \tag{3}$$

$R_{i,p}$ represents user I's rating of item P,$\overline{R_i}$ represents user I's average rating of all items, P represents the collection of all objects.

- Based on the idea of Pearson coefficient, the article average score is introduced:

$$\text{sim}(i,j) = \frac{\sum_{p\in P}\left(R_{i,p} - \overline{R_p}\right)\left(R_{j,p} - \overline{R_p}\right)}{\sqrt{\sum_{p\in P}\left(R_{i,p} - \overline{R_p}\right)^2}\sqrt{\sum_{p\in P}\left(R_{i,p} - \overline{R_p}\right)^2}} \quad (4)$$

$\overline{R_p}$ represents the average score of all the scores for item P.

In traditional collaborative filtering systems, the amount of work increases with the number of participants in the system. [2] One of the disadvantages of UserCF is that the number of users is often greater than the number of items, resulting in the storage overhead of user similarity matrix is very large, and its spatial complexity grows rapidly at the speed of N square times, which is an unbearable expansion speed. And because the data vector is usually very sparse, UserCF is not suitable for applications where positive feedback is difficult to obtain. In combination with the above two types of collaborative filtering, UserCF has stronger social features, which can make a point of interest that is not in the range of their interests before, or quickly update their recommendation list through the dynamic of "friends". It is applicable to hotspot discovery and hotspot trend tracking, and news recommendation scenarios. [2] ItemCF is suitable for applications with stable interest changes, so ItemCF is better than UserCF to recommend videos with similar styles and types.

In general, CF is a very intuitive and explicable model. It can filter the information that is difficult for the machine to analyze automatically, share the experience of others, and have the ability to recommend new information. The recommendation is personalized and highly automated. The main problems are as follows: the recommendation quality is poor at the beginning of the system, and its quality depends on the historical data. The head effect of recommendation results is obvious, and the ability to process sparse vectors is weak, and the system is not strong in extensibility, it does not have strong generalization ability.

In order to solve the above problems and increase the generalization ability of the model, matrix factorization technique is proposed.

2.2. Matrix Factorization

On the basis of co-occurrence matrix of collaborative filtering, the concept of hidden vector is added to strengthen the ability of the model to deal with sparse matrix and solve the main problems of collaborative filtering specifically. As a tool of matrix mathematics, matrix factorization is good at finding information hidden under data. It is mainly used for scoring prediction. In general, we have many users and commodities and express them in the form of a two-dimensional matrix, in which each element represents the rating of the corresponding user on the corresponding commodity. Obviously, such matrix is sparse in real life, because the tastes of the public are different, and each user may not have all the corresponding commodities and score them. Scoring prediction can be regarded as a process of completing the missing elements in the matrix, that is, inferring the missing data according to the existing data in the matrix. [3]

Matrix factorization is to complete the matrix better. In essence, it is to decompose a matrix so that two or more small matrices can be found and multiplied back to the original matrix. In terms of application, matrix decomposition can find the implicit characteristics of interaction between different users. Such information is obtained by analyzing users' behaviors instead of being given directly, which is its advantage. Thus, by discovering these underlying characteristics, we can predict the rating of a given user and a given item because the relevant characteristics match. Therefore, we can predict whether users will like something they have never touched before. When using this method, we assume that the number of hidden features is smaller than the number of users and items. If each user is independently associated with a feature, that is, people and objects have nothing in common, then recommendation will have no meaning. Now we have a set U of users, and a set D of items, create a matrix M with dimensionality of (m, n) this matrix can be viewed as a dot product between two matrix U, V with each matrices having dimensions of (m, k) and (k, n).

$$\hat{R}_{ui} = q_i^T p_u \quad (5)$$

We classify matrix factorization into Eigen Decomposition, Singular value Decomposition(SVD) and FunkSVD (Gradient Descent).

Since Eigen Decomposition can only be applied to square matrices, it is generally not applicable to decompose matrices between users and items.

Singular Value Decomposition factorizes the matrix into:

$$M \approx U_{m\times k}\Sigma_{k\times k}V^{T}_{k\times n} \tag{6}$$

Σ is a diagonal matrix, in which the larger k elements are taken as hidden features, and other dimensions in Σ as well as dimensions in U and V are deleted, the matrix decomposition of k-dimensional hidden vectors is completed.

However, the premise of applying SVD requires that the original matrix must be dense. In general, the co-occurrence matrix in the Internet scene is very sparse. If this method is used, missing data must be filled, so this method is limited. In addition, the computational complexity of traditional decomposition is as high as $0(mn^2)$, which is unacceptable.

To solve these problems, Simon Funk proposed an improved method of FunkSVD, [4] which decomposed the original matrix into three matrices into two low-rank matrices while reducing the complexity:

$$\min_{q^*,p^*}\sum_{(u,i)\in K}\left(r_{ui}-q_i^T p_u\right)^2 \tag{7}$$

In this way, the difference between the original score R and the product of user and item vectors can be minimized, and the original data of the co-occurrence matrix can be retained to the maximum extent, where K is the sample set of all user scores. In order to avoid the over-fitting phenomenon, the objective function of FunkSVD after adding the regularization term of L2 was proposed:

$$\min_{q^*,p^*}\sum_{(u,i)\in K}\left(r_{ui}-q_i^T p_u\right)^2+\lambda(\|q_i\|+\|p_u\|)^2 \tag{8}$$

λ is called the regularization coefficient, and the larger λ is, the stronger the regularization constraint. Regularization, that is, in order for the model to be more regular and stable, it needs to put some constraints on the model to avoid predicting unpredictable results. Generally, the objective function is solved by gradient descent method. We take partial derivatives of the objective functions with respect to q_i and p_u respectively, and get the results:

$$2\left(r_{ui}-q_i^T p_u\right)p_u-2\,\lambda q_i \tag{9}$$

$$2\left(r_{ui}-q_i^T p_u\right)q_i-2\,\lambda p_u \tag{10}$$

According to the results, the parameters are reversely updated along the gradient:

$$q_i \leftarrow q_i-\gamma\left(\left(r_{ui}-q_i^T p_u\right)p_u-\lambda q_i\right) \tag{11}$$

$$p_u \leftarrow p_u-\gamma\left(\left(r_{ui}-q_i^T p_u\right)q_i-\lambda p_u\right) \tag{12}$$

λ is the learning rate.

At the same time, in order to eliminate the user and item scoring bias, deviation vectors of users and items are often added in matrix factorization:

$$r_{ui}=\mu+b_i+b_u+q_i^T p_u \tag{13}$$

μ is the global deviation constant. 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 item's characteristics. [3] The objective function of matrix factorization will also become:

$$\min_{q^*,p^*,b^*}\sum_{(u,i)\in K}(r_{ui}-\mu-b_i-b_u-p_u^T q_i)^2+$$

$$\lambda\left(\|q_i\|+\|p_u\|+b_u^2+b_i^2\right)^2 \tag{14}$$

FunkSVD takes matrix factorization to a new height and is widely used in daily life. For matrix factorization, it is simple to operate, easy to program, low spatial complexity, has good generalization ability, scalability and flexibility, is a good choice to deal with small recommendation system, large recommendation system will not have obvious advantages. At the same time, due to the inconvenience of adding user, item and other related features, matrix factorization loses the opportunity to use a lot of effective information; unable to make effective recommendations in the absence of user behavior.

3. Logistic Regression

To make up for the defect of matrix factorization, logistic regression model is developed, which can integrate a variety of different features and generate relatively comprehensive recommendation results.

Logistic regression is actually a classification model, mainly used to solve dichotomous problems, used to estimate the probability of something. Its essence is to use maximum likelihood estimation to estimate parameters after assuming data obey a certain distribution. As a generalized linear model, logistic regression can be said to be based on linear regression and introduce nonlinear factors through Sigmoid function, which can easily solve the problem of binary classification.

Logistic regression models transform the recommendation problem into a CTR estimation problem. The recommendation process is roughly as follows: Firstly, each feature is converted into a numerical feature vector to determine the optimization objective (taking the optimization of "click rate" as an example). By using the existing sample data to train the model, the internal parameters of the logistic regression model are determined. In the model service stage, feature vectors are input, and the probability of users "clicking" items is obtained through the judgment of the model. Finally, a list of recommendations is made by ranking all items using the probability of "clicks".

The emphasis is on model training and online inference using feature vectors of samples.

First, about the sigmoid function(also called logistic function):

$$f(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(w\cdot x+b)}} \tag{15}$$

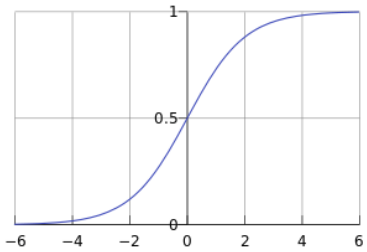

Figure 2: Example for the sigmoid value

Figure 2 shows the graph of the function. Then input feature vector X, and then assign corresponding weight w to each feature to represent the difference in importance. Sum up each feature by weight to get $x^T w$. Input sigmoid function of $x^T w$ to map it to the range of 0~1 to get the final "click rate".

And Gradient descent is a common training method for logistic regression models. It is a first-order optimization algorithm, also known as the fastest descent method, whose purpose is to find the local minimum of a function. So we just take the derivative of the objective function to get the direction of the gradient, and then we go down in the opposite direction of the gradient, and we iterate until we find the local minimum. Input sample x, and the probability of positive sample and negative sample is:

$$\begin{cases} \mathrm{P(y=1|x;w)} = \mathrm{f_w(x)} \\ \mathrm{P(y=0|x;w)} = 1 - \mathrm{f_w(x)} \end{cases} \tag{16}$$

merge:$\mathrm{P(y|x;w)} = \left(\mathrm{f_w(x)}\right)^{\mathrm{y}}\left(1-\mathrm{f_w(x)}\right)^{1-\mathrm{y}}$

The objective function is obtained according to the maximum likelihood estimation principle:

$$L(w) = \prod_{i=1}^{m} P(y|x;w) \tag{17}$$

By taking the partial derivative of each parameter after deformation:

$$\frac{\partial}{\partial w_j} J(w) = \frac{1}{m}\sum_{i=1}^{m}\left(f_w\left(x^i\right) - y^i\right)x_j^i \tag{18}$$

$$w_j \leftarrow w_j - \gamma\frac{1}{m}\sum_{i=1}^{m}\left(f_w\left(x^i\right) - y^i\right)x_j^i \tag{19}$$

Finally, let's analyze the advantages of logistic regression model. It is suitable to integrate different features in form to form a comprehensive recommendation result. Secondly, as a kind of generalized linear model, it has the support of numerical meaning. Moreover, it makes the model highly interpretable and can effectively reduce the communication cost. It is also the need of engineering: because it is easy to parallelize, simple model, low training cost and so on, this model occupies the mainstream of engineering field.

As a basic model, logistic regression is simple, intuitive and easy to operate, but its limitations are also very obvious. Its expression ability is not strong, unable to carry out a series of relatively advanced operations such as feature crossing and feature screening, inevitably resulting in the loss of information.

In order to solve these problems, recommendation systems continue to develop towards the direction of complexity, derived from the Field-aware Factorization Machine (FFM) and other high-dimensional complex models.

4. pOLY2, fm, and ffm

4.1. Poly 2

By using Poly2 for mapping can effectively capture the information of feature conjunctions. Also, Poly 2 is a feature Cross that use for the recommendation system. Here is the mathematical equation:

$$\text{ØPoly2}(w,x) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} w_h(j_1,j_2) x_{j_1} x_{j_2} \tag{20}$$

Where w_h is the weight, and x_{j_1} x_{j_2} are features. However, Poly2 has some disadvantages. First, when people deal with data from the internet, they will use one-hot coding for the most time. However, one-hot coding would cause the feature vector to be very sparse. One-hot coding is a way to transfer a feature into a vector. For example, Wednesday may be represented by [0 0 1 0 0 0 0]. (The first place represents Monday, the second represent Tuesday...) Thus, if we have lots of features, most of the dimensions would be zero except for one dimension.

2, Using Poly2, the number of weights we have complexity of $O(n^2)$ because there are $n \times k$ variables. Since it will consider every possible pairs, makes Poly2 impractical when n is very large. Thus, FM could solve this problem and become more practical.

4.2. FM

To solve the problem of Poly2, FM was suggested in 2010 by Rendle. [5]

The formula is:

$$\text{ØFM}(w,x) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} \left(w_{j_1} \cdot w_{j_2}\right) x_{j_1} x_{j_2} \tag{21}$$

FM studied a latent vector, k, for every feature, very similar to matrix decomposition. FM proposed a latent vector for each feature. There exists k feature factors in each feature vector, where k is a user specified parameter.

FM reduced the complexity, which is n^2 in Poly2, to nk $(n >> k)$, so it is more practical in solving sparse feature vectors.

FFM is an updated version of FM. Instead of using one latent vector, FFM uses a group of latent vectors for each feature. Also, FFM uses a concept, field-aware, to make improvements to this model.

4.3. FFM

The idea of FFM originated from PITF, which is a recommendation system for personalized labels. Three variables were assumed in PITF.

In FFMs, each feature has several potential vectors. Depending on the other features, one of them is used to do the inner product, which is shown in the equation.

$$\text{ØFFM}(w,x) = \sum_{j_1=1}^{n} \sum_{j_2=j_1+1}^{n} \left(w_{j_1 f_2} \cdot w_{j_2 f_1}\right) x_{j_1} x_{j_2} \tag{22}$$

f_1 and f_2 are respectively the field the j_1 and j_2. As shown in the equation, there are three variables: nkf. Usually, $k_{FMM} \ll k_{FM}$, and the complexity to compute is $O(n^2 k)$.

4.4. Experiments

We choose FM as first algorithm to try out. The reason why we choose Factorization Machines (FM) is mainly because it serves as a approach that at the middle part of the whole recommendation system development. Before FM, there are methods such as Logistic Regression (LR) and Support Vector Machines (SVM). After FM, improved FM with field awareness which called Field-aware Factorization Machines (FFM), DeepFM which combines FM and deep learning algorithm and Gradient Boosting Decision Tree (GBDT) with LR bring recommendation quality to another level. As far as we concerned, FM can be a good start point for us to dive into recommendation system. The similarity between FM and embedding is another reason that leads us to here.

Since FM is designed and well known for its prediction on CTR (click-through rate).

Firstly, we try to find an dataset and ends up finding a dataset of anime [6] which includes anime's attribute such as name, genre, episode, member, type and average rating and users' rating for anime. Here we use only anime's genre as feature and average rating as its FM's label in order to simulate the situation that CTR have, the situation which only item's features are given. Moreover, we convert the rating to binary value where values over 7.0 are set to 1 and less than 7.0 are set to 0. Secondly, we clean the data by one-hot encoding the genre field. Due to the fact that every anime can have more than one genre, we have to extract the genres name and mapping genre of every anime to 1 in the specific genre name field. After data munging is done, we divide data into training and testing set by KFold with shuffle. When it comes to training, we use an efficient package called xLearn that had already implemented FM.

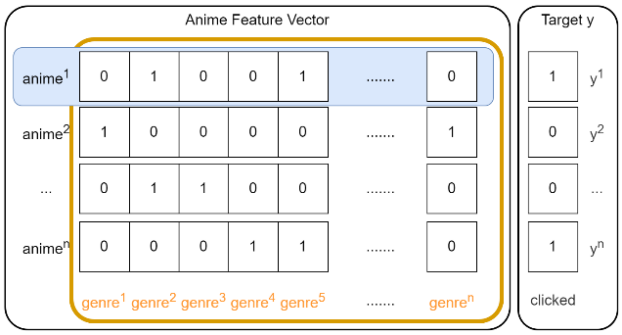

Figure 3: Example for the real valued anime feature vector from the dataset. Every row represents a feature vector of a specific anime with its corresponding target y^n - here is been clicked or not. The first row (blue) represents a anime and its features (genres); the columns (orange) represent indicator variables for each genre. The rightmost column is the target - here the clicked.

Figure 3 shows feature vector pattern from dataset [6] The result of training is not very satisfying. While we play around with the hyper parameter, the accuracy or any other valuations did not change a little. We realized that the features in this dataset are not enough and too simple for model to improve itself, thus the training usually face early-stop at early training epoch. Instead of apply FFM to this dataset, we searched for another dataset, which is from KKBox [7], to apply FFM and FM to. We user xLearn [8] package which is focus on fast FM, FFM and LR training. During this time, we found that FFM is sensitive to epoch and number of features and k, which is the number of latent vectors. The result is impressive when k is equal to 4 with 70,000 features and 14 fields.

The whole training process can be divided to 2 independent phases data cleaning and train, first we clean up the data, convert features, fields data into int, thus serve as values of latent vector. Before training the data styles that FM model and FFM model receive are libsvm [9,10] and libffm [11]. After cleaning the data, we use xLearn to train and try out as much hyper parameters as we can to find out the best for our dataset.

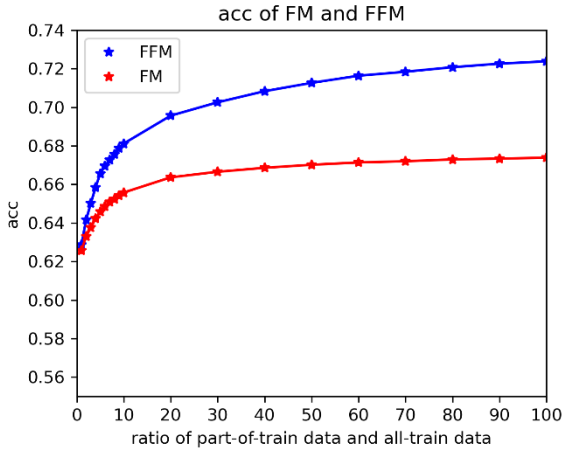

Figure 4: FM's and FFM's Accuracy of test data

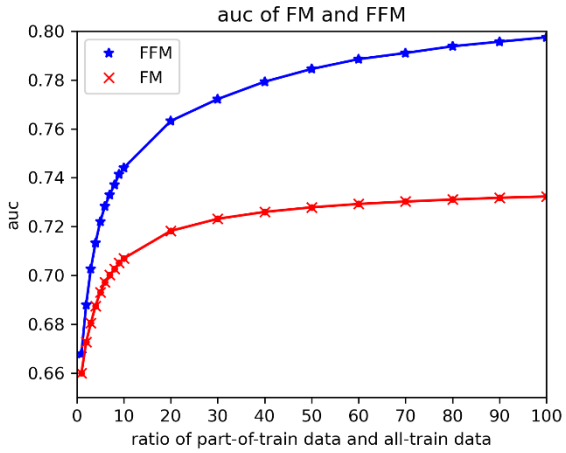

Figure 5: FM's and FFM's AUC(Area Under the Curve) of test data

Figure 4, 5 show that FFM has better ACC and AUC than FM and the accuracy grow faster than FM when training data ratio is around 1~20%.

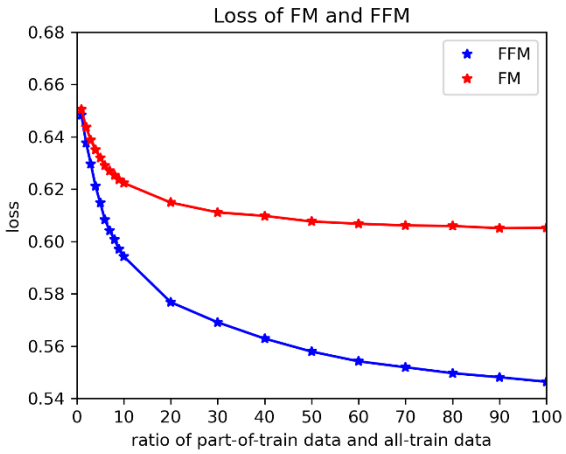

Figure 6: FM's and FFM's AUC(Area Under the Curve) of test data

Figure 6 shows that loss of FFM is smaller and decreases faster than FM.

5. Extension of Logistic Regression

5.1. GBDT+LR

FM and FFM increase their performance in model's accuracy by multiplying pairs of features with latent vectors. However, combination of multiple features in reality sometimes determines the output. FM and FFM fall short of this aspect because, once they take into account multiplication of triple combinations, space complexities increase to an intolerable extent. To solve this problem, Facebook in 2014 came up with a method of GBDT+LR to predict Click-Through Rate (CTR), that fits certain combinations of features into models at the same time of avoiding combinatorial explosion [12].

They briefly divided the procedure into 2 independent phases. In Phase I, training data are fed into the Gradient Boosting Descent Tree (GBDT). Here, the divide-and-conquer algorithm generates several trees and sets each tree to fit the loss between former trees and real outputs. However, after training the model of GBDT, the results of predictions are unimportant while the trees themselves play a significant role. Then, the training data are again fed into the trained trees and generate a feature by one-hot encoding based on which leaf they finally fall into. For example, if each tree has 64 leaves, when a piece of data is put into the tree model and falls into the 43rd position, it generates one-hot-encoding with 1 on 43rd position and 0s on every other one. This 64-dimensional feature, together with features generated from other trees, becomes the transformed feature of this piece of data. In Phase II, all transformed features of data are fed into a Logistic Regression algorithm, that finally trains the model and yields predictions.

This GBDT+LR works well because of its efficiency in combining features, which is mainly the benefit of using GBDT for transforming feature vectors. Each transformed feature represents not only one feature, but a set of determinant features that come from each node of a tree. It's because when building the trees in GBDT, they automatically select features that work well together. Thus, while putting these multiple features into Logistic Regression, they're treated linearly but indeed are meaningful for several features. Figure 7 shows the AUC and F1 curve of GBDT+LR model on test samples with respect to the percentage of training samples. This graph indicates the performance of this model on the dataset stabilizes around 0.65 (F1) and 0.67 (AUC) after training enough data which avoid underfitting.

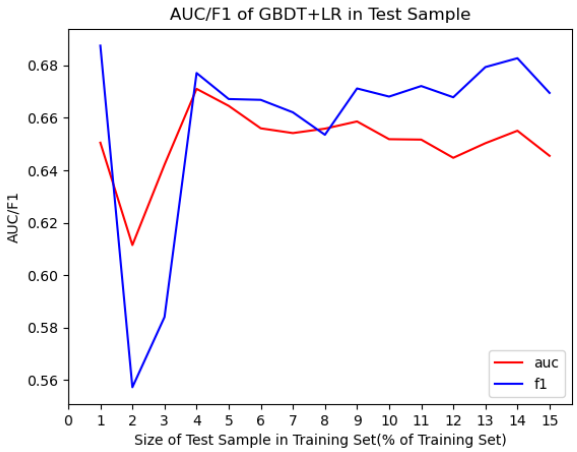

Figure 7: GBDT+LR's AUC(Area Under the Curve) and F1 of different size test sample

5.2. LS-PLM

Large Scale Piece-wise Linear Model (LS-PLM), or Mixed Logistic Regression (MLR), is another effective model that Alibaba once used for its recommendation system. Though the model is still linear, it separates different clusters of training data and applies Logistic Regression to each cluster [13].

LS-PLM consists of two steps. In the first one, a function σ divides the feature space into m parts. Then, within each part, normal Logistic Regression η is applied to features to produce a prediction. The overall output is the sum of products of weights and predictions in each subspace.

$$p(y = 1|x) = g\left(\sum_{j=1}^{m} \sigma(u_j^T x)\eta(w_j^T x)\right) \tag{23}$$

LS-PLM is effective because when training data have obvious group divisions, normal Logistic Regression will treat them in the same way and lose useful information. MLR, on the other hand, fits training data in each group separately and uses different parameters to build the model. The hyperparameter m is arbitrary, where the optimal choice is based on experiments. Alibaba determined their optimal m to be 12, while in the dataset of this paper, the optimal m is approximately 10. Figure 8 shows the AUC curve of LS-PLM model on test samples with respect to the percentage of training samples. This graph presents that the performance of this model is approximately 0.73 (AUC) after sufficient training data are fed, which is slightly better than GBDT+LR.

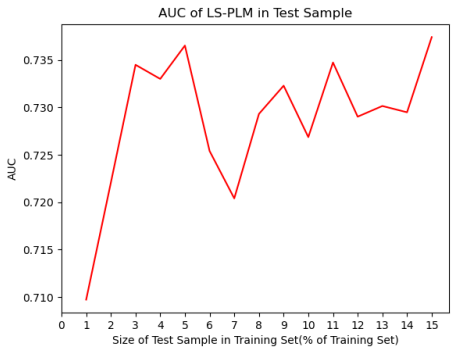

Figure 8: LS-PLM's AUC(Area Under the Curve) of different size test sample

5.3. Kernel LR

Sometimes, the subspaces between different clusters of training data are hard to distinguish using LS-PLM. To solve this problem, a combination of Kernel Method and Logistic Regression can make boundaries easier to divide.

In LS-PLM, the space division function σ is usually linear. However, in the reality, the training data may not always have plane boundaries described in terms of linear variables. For example, a dataset of 2-dimention features may have a hyperbola as their boundary. And the equation for hyperbola is caught by normal methods of LS-PLM. Thus, kernel functions are introduced to solve this problem. Kernel functions usually raises the dimension from n (the number of features) to m. The additional dimensions are the function of original dimensions and allow the boundary to be written in linear combination of them. In the same example, one of the additional dimensions may be $\frac{x^2}{a^2} - \frac{y^2}{b^2}$, so that the boundary between clusters of data is again a plane in higher dimensions.

6. Challenges IN recommendation system

In the application process of recommendation system, user behavior is mostly observational, not experimental. This introduces a lot of bias, if these deviations are not considered, blind data fitting, will lead to a lot of serious problems.

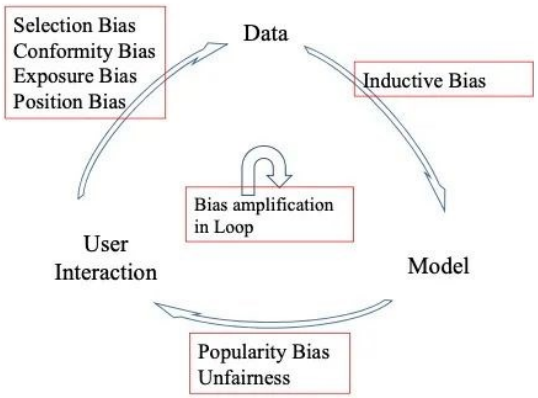

Figure 9: Bias type caused by different processes

The Figure 9 shows the feedback mechanism of the recommendation system, which can be divided into the following processes. Information about the interaction between users and commodities is collected from users. The collected interaction is expressed as feedback matrix between users and commodities. Generally, there are two kinds of feedback: implicit (if there is interaction, it is 1; otherwise, it is 0) and explicit (users' score on commodities). Based on the collected data to learn the recommendation model, according to the historical data to predict the relative probability; Feedback the results of the recommendation to the user, this process will influence the user's behavior and decision.

One of the challenges that exist in the Recommendation system is bias. There are several factors about bias: First, User behavior is observational rather than experimental – user generates behaviors based on exposed items, so the observational data is confounded by the way the recommendation system exposes the data. Second, some items are more popular than others, which would cause the recommendation system to receive more feedback from popular items than others. As a result, those popular items would have a more significant impact on the modal training. Third, since the recommendation system has a feedback loop – it will intensify the bias, cause "the rich get richer."

Usually, users only rate items with the most interest – positive (like) or negative (dislike), and users tend to rate particularly good or bad items. Thus, cause selection bias. Selection bias exists when users decide which items they will rate by their willingness, missing randomness.

Conformity bias also comes with rating items. People tend to rate similarly to others in the same group, which will even come against their ideas and judgments. An item with a higher score will receive more high rating comments than it deserves, which will make the better one better, and the worse one worse. Thus, it will not show the true preference of people.

Position bias describes the tendency that users to note and interact with certain positions in the recommendation list with higher possibility, especially there are lots of items on the list. Moreover, users tend to trust more on first a few items on the list. Therefore, it fails to show the real preference of users.

Those are biases caused by users. However, there are also bias exists in the recommendation system itself.

Popularity bias causes popular products to be recommended even more often than they are. In most cases, a small fraction of the whole population of items accounts for the most interaction with users, which is the long-tail phenomenon.

7. Conclusions and future works

In this paper, we mainly compare the efficiency between 4 traditional methods used in recommender system. It displays that for certain kinds of data sets, FFMs outperform the GBDT+LR, LS-PLM and FM in terms of loss and accuracy.

For the future work, the performance of GBDT+LR in our experiments is worse than that of other models. The reason is probably that while using one-hot encoding on leave nodes (features set), which GBDT generate, the space complexities grow rapidly (fields expands to 10,000) and ends up cost too much time for LR to train. In addition, the huge sparsity caused by one-hot encoding might lead to low

accuracy too. When it comes to FM and FFM, it only takes 2.5 minutes for FM and FFM to train 1.8M entries, each epoch takes 1.6 seconds in average. However, the performance of FFM is better than the other two methods (GBDT+LR and LS-PLM)

8. Acknowledgement

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor David Woodruff, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would like to acknowledge my teacher assistant Hudson Li from my Algorithm for Big Data course. I want to thank you for your patient support and for all of the opportunities and advice I was given to further my research.

I hereby express gratitude to my dear partner Shuyi Wang, without his effort, this thesis cannot be accomplished. In the process of compilation, he made great contribution on data collecting and algorithm implementation. Besides, he completed the Chapter 4 by himself. This thesis is rather a common achievement than a private possession. Also, I would like to acknowledge my teammates also classmates Ziyang Liu and Linyan Li from Algorithm for Big Data course for their wonderful collaboration.

9. References

[1] Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.

[2] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295).

[3] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

[4] Netflix update: Try this at home. (n.d.). Retrieved February 28, 2022, from https://sifter.org/simon/journal/20061211.html

[5] Rendle, S. (2010, December). Factorization machines. In 2010 IEEE International conference on data mining (pp. 995-1000). IEEE.

[6] CooperUnion. (2016, December 21). *Anime recommendations database*. Kaggle. Retrieved December 7, 2021, from https://www.kaggle.com/CooperUnion/anime-recommendations-database.

[7] KKBOX. (n.d.). *WSDM - KKBOX's Music Recommendation Challenge*. Kaggle. Retrieved December 7, 2021, from https://www.kaggle.com/c/kkbox-music-recommendation-challenge/data.

[8] Aksnzhy. (n.d.). Aksnzhy/xlearn: High performance, easy-to-use, and Scalable Machine Learning (ML) package, including linear model (LR), factorization machines (FM), and field-aware factorization machines (FFM) for Python and CLI interface. GitHub. Retrieved December 7, 2021, from https://github.com/aksnzhy/xlearn.

[9] Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST), 2(3), 1-27.

[10] Rendle, S. (2010, December). Factorization machines. In 2010 IEEE International conference on data mining (pp. 995-1000). IEEE.

[11] Juan, Y., Zhuang, Y., Chin, W. S., & Lin, C. J. (2016, September). Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 43-50).

[12] He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., ... & Candela, J. Q. (2014, August). Practical lessons from predicting clicks on ads at facebook. In Proceedings of the Eighth International Workshop on Data Mining for Online Advertising (pp. 1-9).

[13] Gai, K., Zhu, X., Li, H., Liu, K., & Wang, Z. (2017). Learning piece-wise linear models from large scale data for ad click prediction. arXiv preprint arXiv:1704.05194.