AFFM: Auto feature engineering in field-aware factorization machines for predictive analytics

Conference Paper · November 2015

DOI: 10.1109/ICDMW.2015.245

4 authors, including:

Bikash Agrawal
University of Stavanger (UiS)
12 PUBLICATIONS **25** CITATIONS

SEE PROFILE

AFFM: Auto Feature Engineering in Field-aware Factorization Machines for Predictive Analytics

Lars Ropeid Selsaas†, Bikash Agrawal*, Chunming Rong*,Tomasz Wiktorski*
* Department of Computer and Electrical Engineering
University of Stavanger
Stavanger, Norway
{bikash.agrawal, chunming.rong, tomasz.wiktorski}@uis.no
† Gamut
lselsaas@gamut.media

***Abstract*—User identification and prediction is one typical problem with the cross-device connection. User identification is useful for the recommendation engine, online advertising, and user experiences. Extreme sparse and large-scale data make user identification a challenging problem. To achieve better performance and accuracy for identification a better model with short turnaround time, and able to handle extremely sparse and large-scale data is the key. In this paper, we proposed a novel efficient machine learning approach to deal with such problem. We have adapted Field-aware Factorization Machine's approach using auto feature engineering techniques. Our model has the capacity to handle multiple features within the same field. The model provides an efficient way to handle the fields in the matrix. It counts the unique fields in the matrix and divides both the matrix with that value, which provide an efficient and scalable technique in term of time complexity. The accuracy of the model is 0.864845, when tested with Drawbridge datasets released in the context of the ICDM 2015 Cross-Device Connections Challenge.**

Keywords*—Predictive analytics, FFM, Factorization machines, cross-device connections

I. Introduction

Predictive analytics is an important technique with applications in many fields ranging from business to science. Applications such as online adverting, e-commerce website, personalized search, social networks, and etc., make use of predictive analytics and data mining to mine large-scale data to identify users. How to provide effective personalized recommendations while moving users across cross-devices, becomes one of the most important research topics in the past few decades. The accurate predictive modeling for identifying users on cross-devices are required. This identification will help in recommendation engine, online advertising and will improve user experiences.

The task of ICDM 2015: Drawbridge cross-device connections competition is to identify the users using different devices. Factorization machines (FM) is used as the base model to predict the users. FM is a generic approach that allows to mimic most factorization models by feature engineering. The factorization machines combine the generality of feature engineering with the superiority of factorization models in estimating interactions between categorical variables of a large domain.

Fig. 1: User connected to multiple cross-devices.

Problem Statement: Now a days consumers are using many devices to complete the online task or browse the internet. Consider the case, where a user wants to plan the holiday trip: he reads a travel blog on his smartphone on the way to work, search for hotels on his laptop after work, search for flight tickets on his tablet during dinner, search for trip advisor rated best restaurant in the area on his PC, and also download a travel book on his Kindle. As the users move across the devices to complete their online tasks, their identity becomes fragmented. The ads, recommendations, and messages are not always able to separate whether the activity on the different devices is tied to one user or many users. The model is required that can predict the users as they switch between devices (websites/mobile apps) as shown in figure 1.

A. Our Contribution

We propose an automated and scalable model for identifying the users across the cross-device connection. Our approach provides a novel way of feature engineering in FFM model. The learning approach used in our model provides an efficient learning rate compared to classical FFM. During the learning phase, the learning rate is divided by the number of features in the opposite field to balance out how much each weight is updated.

B. Paper Structure

Section II gives an overview of the background. Section III introduces the design and approach of our model. Section IV evaluates our algorithm and presents the results. Section V concludes the paper.

C. Related Work

In 2010, Rendle [1] introduce Factorization Machines (FM), which combines the advantages of Support Vector Machines (SVM) [2] with factorization models. However, with raw data the performance of FM is culpable. Feature engineering is required beforehand to use FM. Field-aware factorization machines (FFM) [3] have been used to win two Kaggle clickthrough rate prediction competitions hosted by Criteo[1] and Avazu[2]. The performance of the model degrades with large matrices. Moreover, FFM requires pre-feature engineering on the raw data before predicting. Factorization Machines are available in the large scale machine learning libraries GraphLab [4], Spark-libFM[3], and Bidmach [5]. The reason we adopted FFM because it provides an efficient way to solve the larger problems with better accuracy.

The challenges of track 2 of the KDD cup 2012 competition was to predict the click-through rate (CTR) of advertisements [6]. They used featured engineering with several models ANN, probability models and collaborative filters. They used an interesting method of featured engineering to design their factor model. Another research on predicting clicks on ads on Facebook provides an interesting approach for featured engineering [7]. In the paper, it shows how the number of parameters has an impact on the overall system performance. However, it uses decision tree for prediction over featured engineering.

The approach mentioned in this paper provides an efficient way for the users to perform machine learning without caring much about feature engineering.

II. BACKGROUND

A. FM (Factorization Machines):

Factorization machines are a new model class that combines the advantages of Support Vector Machines (SVM). Steffen Rendle [1] introduced this model in 2010. This model exhibits similar properties of SVM. But FM is used as general prediction tasks. Like other models, nonlinear SVMs or decision trees, FM includes interactions between predictor variables. For example, FMs can learn that young users like to access from different devices, whereas preferences of old users are opposite.

B. Feature Engineering:

Feature engineering is the process of creating features that make machine learning more accurate and efficient using domain knowledge of the data.

[1]https://www.kaggle.com/c/criteo-display-ad-challenge
[2]https://www.kaggle.com/c/avazu-ctr-prediction
[3]http://spark-packages.org/package/zhengruifeng/spark-libFM

III. APPROACH

As introduced in Section 1, FFM is used as our based model for predicting users across cross-devices. Factorization Machines (FM) is defined as mention in equation 1. A second-order FM for i-th feature vector x_i of n x n matrix X is defined as in equation 1:

$$\widehat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_i, v_j > x_i x_j \quad (1)$$

where, $< v_i, v_j >$ is the dot product of two vectors of size k:

$$< v_i, v_j > := \sum_{f=1}^{k} < v_{i,f}, v_{j,f} > \quad (2)$$

The field-aware factorization machines (FFM) is explained in equation 3:

$$\widehat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_{i,f1}, v_{j,f2} > x_i x_j \quad (3)$$

where, $< v_{i,f1}, v_{j,f2} >$ are two vectors with length k. f_1 and f_2 are respectively the fields of i and j.

The two vectors are compared, and their features are loaded into the model without any feature engineering. The datasets from the device, cookie and IPs are used for feature engineering. The datasets device and cookie are merge using commen field *drawbridgeHandle*. As our model handles features, we can just load raw features in the model. The learning model reads the features of the matrices and updates the learning rate. There are 42 different types of features used out of which seven features are shown as an example in the figure 2. Equation 4 defined the mapping of two vectors field and feature that is dervied from FFM definition:

$$\widehat{y}(x) := \sum_{i=1}^{n} \sum_{j=i+1}^{n} < v_{i,f1}, v_{j,f2} > x_i x_j \quad (4)$$

Field name	Field Index	Feature Name	Feature Index
Device ID	1	id_1000002	1
Cookie ID	2	Id_10	2
Comp OS/type	3	Devos_7	3
	3	Devos_72	4
Device type	4	Devtype_2	5
	4	Devtype_3	6
IP	5	ip8352948	7
Is cell IP	6	0	8
Property ID	8	property_66021	9

Fig. 2: Interactions between fields and their features.

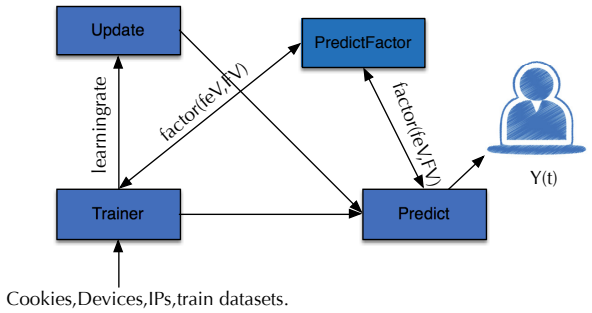

Fig. 3: Predictive learning Data Model.

The datasets are feed to the model. Before sending the data, the cookies and devices are merged. On the training phase, it calculates the factors in the feature vector and field vector. In the update, the learning rate is calculated, and learning of the model with each field present in the cell is calculated. The update algorithm performance is quite efficient. The learning rate is divided by the number of features in the opposite field. In fact, in this process it can save time to traverse the entire matrix. And finally, users are predicted in the prediction block as shown in figure 3

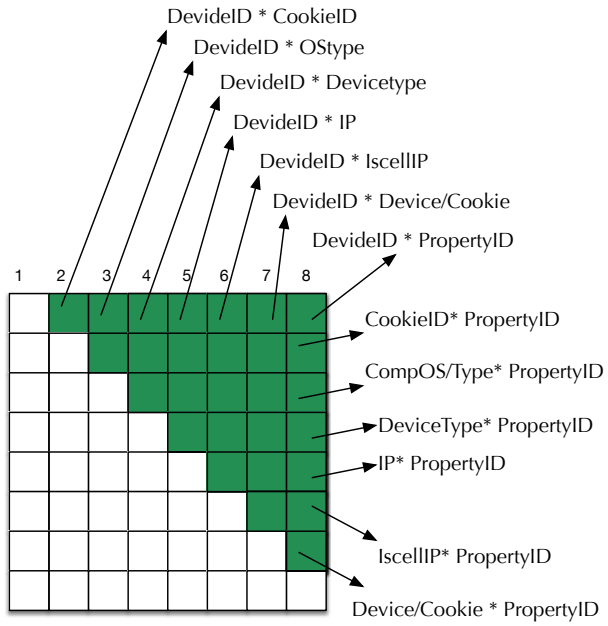

Fig. 4: Factors in the feature vector and field vector

Figure 4 depicts the interaction pairs of the model between field vector and feature vector. Within every cell of the matrix in Figure 4, the model had its own set of features for the factorization. For example, there are 8 unique device types, which is used as a feature for the field name device type. For unique 8 types of device, the mapping between fields and features is shown in figure 5. There are 89 unique device operating systems, 368566 unique properties, 443 unique categories, etc. The training file consists of 142770 devices and the test file consists of 61156. There is almost 2.1 million cookies records.

The traditional FFM provides an efficient method of factor field multiplication. Our model reads the entire field and divides the feature using the number of features in the opposite field, which makes it efficient by jumping the cells in the matrix.

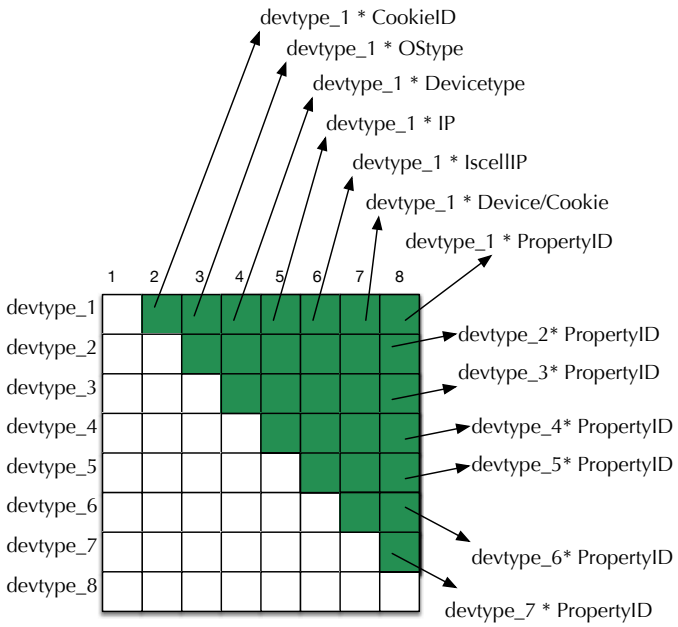

Fig. 5: Factors multiplication between fields and features of devicetype field.

The model deals with the properties directly without the need for any feature engineering. The model can adapt the properties directly from the raw data. Before feeding data to the model, the number of potential cookie/device pairs is reduced down to about 900k and then treats as a regression problem. There exist several properties for devices and cookies. This makes challenging for FFM to learn in term of time and memory complexity.

A. *FFM Learning rate:*

The issue with learning rate is that the other features interacting with the property feature will get updated once for every property and since they are all updates towards the same field this means we update the same weight again for every property. With the large-scale datasets where the properties of fields are massive, this tends to be an issue. For example, we could have over 200 properties for one sample and maybe just 1 for another. This can be compared to one of the samples getting a 200 times higher learning rate towards that field. So it would either end up with a too high or too low learning rate, neither works very well.

Our approach solves this problem by dividing the learning rate by the number of features in the opposite field. For example, if there are 200 properties in one sample and 1 for another. The learning rate is divided by 200, so the total amount of learning rate applied is the same as if there was 1 property only. This approach fits the training data to a point of extreme precision, but it could easily over-fit. Another key challenge is to overwhelm the over-fit if there are the small

weights adding up. We need to add Regularization [8] to reduce the impact of the smaller weights making a more robust solution, which does not over-fit.

B. Feature Engineering:

The model can handle features automatically from the original data, which makes the model scalable and efficient. This provides advantages over Support Vector Machines (SVMs) and Gradient Boost Machines (GBMs). We can achieve similar accuracy, using other machine learning algorithms like SVMs, GBMs, Decision trees, etc. But feature engineering is necessary to be performed before processing data to the model, which consumes unnecessary latency and performance degradation of the model. In fact, users need to understand the properties of the matrix to perform feature engineering. 42 different features are extracted that is listed in table I.

TABLE I: List of features used in the model.

S.N.	Fieldname	Type
1	Device type	Device
2	Device OS version	
3	Device Country Info	
4	$DeviceAnonymous\_c0$	
5	$DeviceAnonymous\_c1$	
6	$DeviceAnonymous\_c2$	
7	$DeviceAnonymous\_5$	
8	$DeviceAnonymous\_6$	
9	$DeviceAnonymous\_7$	
10	$deviceProperties$	
11	computer OS type	Cookie
12	Browser version	
13	Cookie country info	
14	$CookieAnonymous\_c0$	
15	$CookieAnonymous\_c1$	
16	$CookieAnonymous\_c2$	
17	$CookieAnonymous\_5$	
18	$CookieAnonymous\_6$	
19	$CookieAnonymous\_7$	
20	cookieProperties	
21	DeviceIPFreq count	Device IP
22	DeviceIPAnonymous Count 1	
23	DeviceIPAnonymous Count 2	
24	DeviceIPanonymous Count 3	
25	DeviceIPAnonymous Count 4	
26	DeviceIPAnonymous Count 5	
27	CountIPsForDevice	
28	CookieIPFreq count	Cookie IP
29	CookiesLeastFrequentIP	
30	CookiesSecondLeastFrequentIP	
31	CookiesThirdLeastFrequentIP	
32	CookieIPAnonymous Count 1	
33	CookieIPAnonymous Count 2	
34	CookieIPanonymous Count 3	
35	CookieIPAnonymous Count 4	
36	CookieIPAnonymous Count 5	
37	CountIPsForCookie	
38	IPaggIs cell IP	IP aggregation
39	IPaggTotal Freq	
40	IPaggAnonymous count c0	
41	IPaggAnonymous count c1	
42	IPaggAnonymous count c2	

IV. Results

Setup: Our system is comprised of 6 core, 64 GB of ECC DDR-2 RAM, 1 TeraBytes of storage. The operating system used is Windows 8. Java 1.7 is used to build the model.

Datasets used here are provided by Drawbridge during Kaggle competition. The datasets consist of relational information about users, devices, cookies, information on IP addresses and behavior. It consists of different sets of data. First device table provides basic information of the device. It provides high-level summary information regarding the device. Drawbridge Handle field uniquely identify a person behind the device and cookie. Device and cookie of the same person will have the same handle. Similarly, cookie table provides similar information about the cookie. Second, IP table describes the joint behavior of device or cookie on IP address. One device or cookie can appear on multiple IPs, so we put all the IPs into a bag. Third, IP aggregation table provides IP address information. Fourth, property and property category table provides the information regarding website (cookie) and mobile app (device) that user has visited before. It also provides the list of hash values for specific name of the website and mobile app. Also provides the list of categorical information of the websites and mobile apps. Detailed information for the datasets is provided in the actual Kaggle competition [9].

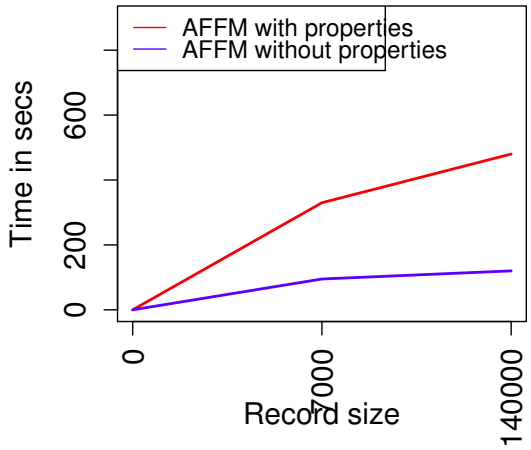

Fig. 6: Performance graph comparison of FFM with our model.

AFFM performance is compared with features and without features in figure 6. It is a fact that without features, save lots of computation memory and time. Varying the size of training data to evaluate the performance of the model. The training data consists of 142770 devices and the test file consists of 61156. There is almost 2.1 million cookies record. Performing features selection on devices, cookie and IP tables will be time and memory consuming. One of the team member Gilberto [4] used Decision tree's but for this paper we will focus on the results got from FFM. AFFM provides better compromise

[4]https://www.kaggle.com/titericz

accuracy with good performance. There is always a trade-off between accuracy and performance.

Regularization is added to our model so that it reduces overfitting. The AFFM produces a score of 0.864845 in the private Leaderboard. The actual leaderboard score for the competition is 0.789924 but after the competition had ended we realized everyone removed -1 value handles, so trying that approach lead scores to 0.864845. Our model is tested without properties and the scores in leaderboard is 0.852774.

V. Conclusion

In this paper, we presented our FFM model with auto feature engineering capability. Our model can perform on the raw datasets. It has inbuilt ability to calculate the feature in the field and drops the learning rate of the model. Our model does not read all the unique values of cell present for that features. The model has a balanced learning rate which allows it to use raw properties of the devices and the cookies. The model provides the score of 0.864845 in the leaderboard. We are able to manage to improve the model performance by performing the auto feature engineering and balanced learning rate. This emphasizes the importance of feature engineering and learning rate.

References

[1] S. Rendle, "Factorization machines with libfm," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, pp. 57:1–57:22, May 2012. [Online]. Available: http://doi.acm.org/10.1145/2168752.2168771

[2] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[3] Y. J. Y. Zhuang and W.-S. Chin. Field-aware factorization machines. [Online]. Available: http://www.csie.ntu.edu.tw/~r01922136/slides/ffm.pdf

[4] Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. E. Guestrin, and J. Hellerstein, "Graphlab: A new framework for parallel machine learning," *arXiv preprint arXiv:1408.2041*, 2014.

[5] J. Canny and H. Zhao, "Bidmach: Large-scale learning with zero memory allocation," in *BigLearning, NIPS Workshop*, 2013.

[6] M. Jahrer, A. Toscher, J. Lee, J. Deng, H. Zhang, and J. Spoelstra, "Ensemble of collaborative filtering and feature engineered models for click through rate prediction," in *KDDCup Workshop*, 2012.

[7] H. Xinran, P. Junfeng, J. Ou, X. Tianbing, L. Bo, X. Tao, S. Yanxin, A. Antoine, H. Ralf, B. Stuart *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 1–9.

[8] Y. Kodratoff, *Introduction to machine learning*. Morgan Kaufmann, 2014.

[9] K. 2015. Icdm 2015: Drawbridge cross-device connections data description. [Online]. Available: https://www.kaggle.com/c/icdm-2015-drawbridge-cross-device-connections/data