Feature Selection for FM-based Context-Aware Recommendation Systems

Xueyu Mao*
University of Texas at Austin, Austin, TX 78712
Email: xmao@cs.utexas.edu

Saayan Mitra, Viswanathan Swaminathan
Adobe Research, San Jose, CA 95110
Email: {smitra, vishy}@adobe.com

***Abstract*—Context-Aware Recommendation Systems has gained lots of attention in both industry and academic research. Factorization Machines (FM) based recommendation has been successfully used in sparse industrial datasets for user personalized video recommendations. FM is a collaborative filtering technique for predicting a target such as user rating, given observations of interaction between some users and items. The model can incorporate any available auxiliary information about the user, the item, or the interaction which serves as context or features of the data. In this paper, we propose a framework to automatically select features on FM-based recommender systems to improve the prediction quality. FM requires the input data as a one-hot encoded feature vector in a binary space domain. We use the values of the FM parameters in the binary space to determine the importance of the context. We consider the density of the important features in the binary space to rank and select the relevant features in the original data. Experiments on multiple datasets have been conducted to validate the efficiency and robustness of our method.**

***Index Terms*—video recommendations, feature selection**

I. Introduction

Recommendation systems [1] have been an active topic in recent years, especially since the Netflix Challenge [2]. Several approaches have been proposed for personalized recommendations, among which collaborative filtering [3] is one of the the most widely applied techniques. Collaborative filtering techniques use the users' history and predict his/her reaction to the items, that the user is yet to interact with. In a video recommendation setting, a user's behavior is captured in the video session. Typically, additional information is available pertaining to a session namely the user metadata (e.g. age, gender), video metadata (e.g. genre), and various other session context (e.g. the app used, the device, the location). This auxiliary information which serves as context could be useful for recommendation and this motivates the research on Context-Aware Recommendation Systems (CARS) [4–7]. Although having more context seemingly is an asset, it can be a curse too by making the input data high dimensional and very sparse. Not all context features are useful and they make the data noisy, thus degrading the model's accuracy. Also, it is unlikely that all the desired features to be collected will be available in every session. This necessitates the selection of useful context features for recommendation. CARS is widely used for recommendation in industrial systems such as [8], based on a prediction model called Factorization Machines (FM) [9] which has been proven to be efficient in personalized recommendations with auxiliary information [10]. In this paper, we propose a framework for context feature selection on FM-based personalized recommendation systems, which serves as the input to the model to accelerate and improve the model training. Traditionally feature selection methods are divided into three categories as noted in [11]. **Wrapper methods** evaluate each subset of features by brute force to choose the best set, but it is computationally expensive [12]. **Filter methods** use heuristic measures (i.e. mutual information) to score features, but it may be hard to apply to different datasets with the same metric [13]. **Embedded methods** do feature selection during model training, which does not explicitly produce the important features for future use. Embedded methods have been popular with FM for feature engineering. They typically (a) drop the linear weights and only keep the interaction weights [14]; (b) add regularization terms (ℓ_1, ℓ_2, $\ell_{2,1}$) in objective functions to make the model sparser, making some of the weights 0 (or near 0); (c) or use gradient boosting techniques to learn the feature function [15]. Our method is a combination of embedded and wrapper methods. We first choose features based on predictive power, and then, similar to wrapper methods, we subsample the set of features selected in the first step, in addition to a subsetting step for data preprocessing as illustrated in Fig. 1. Our method includes four steps:

- Data preprocessing, which includes a subsetting scheme and data normalization;
- Feeding normalized data to FM or a fast FM-variant to get feature indices set in binary space;
- Mapping the binary features back to original feature space, and ranking these features;
- Selecting some features and subsampling in a uniformly random way to get the final features.

Details will be introduced in section III.

II. Preliminaries

FM was proposed by Rendle [9], which combines the advantages of both a regression model and matrix factorization. The model equation of a degree 2 FM is given by:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j,$$

where, $\mathbf{x} \in \mathbb{R}^d$ is a feature vector representation of a datapoint. $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^d$, $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_d]^T \in \mathbb{R}^{d \times k}$

*This work was done while interning at Adobe Research.

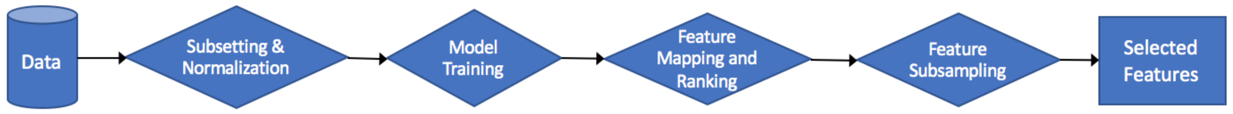


Fig. 1: Feature selection framework.

are model parameters, d is the dimension of the input feature vector, $k \in \mathbb{Z}^+$ is the rank of the factorization. $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ is the inner product of two latent factors, which models the interaction between the i-th and j-th variable.

In real data, most features are categorical, and FM would encode them as (sparse) binary vectors. Numerical data (e.g. time of day), can encoded as categorical by segmenting them into different buckets. Without loss of generality, we suppose all the context features are categorical in this paper. Let users be denoted by $U = \{u_1, u_2, \cdots\}$, items by $I = \{i_1, i_2, \cdots\}$, and context by $\mathcal{C}_3, \cdots, \mathcal{C}_m$. Then FM estimates the target y:

$$y: \; U \times I \times \mathcal{C}_3 \times \cdots \times \mathcal{C}_m \to \mathbb{R}.$$

Let the encoded input data be $\mathbf{X} \in \mathbb{R}^{n \times d}$, where each row of $\mathbf{X}$, denoted by $\mathbf{x}_i$, encodes a user session.

$$\mathbf{x}_i = (\overbrace{0 \cdots 1 \cdots 0}^{|U|}\overbrace{0 \cdots 1 \cdots 0}^{|I|}\overbrace{0 \cdots 1 \cdots 0}^{|\mathcal{C}_2|} \cdots \overbrace{0 \cdots 1 \cdots 0}^{|\mathcal{C}_m|})^T,$$

If we normalize each feature in the binary space, the weights of FM model can roughly represent the importance of each feature as the input would either be 1 or 0. If a feature is 1, it's weight denotes how much this feature will contribute to the target. Thus we can infer the predictive power of a feature based on it's weight. This is also the key difference between our method and other embedded methods like LASSO [16] and Elastic Net [17], while they just consider each context feature as a single variable associated with a corresponding coefficient, FM essentially breaks each context feature into multiple variables in binary space with coefficients for every value of the context.

III. Feature Selection Framework

A. Data Preprocessing

*1) **Data Subsetting:*** Given that real datasets are voluminous, it is crucial that we come up with a subsetting method that only takes a small fraction of the data for feature selection training, but still gives a comparable result to using all the data. We subset the input data in a way which preserves the distribution of sessions among users. In other words, we proportionately eliminate sessions per user. This also eliminates one-time users and makes the training data denser.

*2) **Data Normalization:*** Next, we need to normalize each feature (column) to have unit variance (e.g. divide each element by the standard deviation of the column), so that we can infer the importance of each feature by its weight. This weight eliminates the effect of variance in a feature. An extreme case is when the variance is near 0, it is not useful for prediction as it acts as constant, but normalizing would make the weight near 0.

B. Determining Feature Indices in the binary space from trained FM models

First, we use FM or a fast FM-variant (such as convex-FM [18]) to determine the weight of each linear variable $\mathbf{w}$ and interaction pair Σ, where $\Sigma_{i,j} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle$.

Next, we choose the top $b\%$ ($b \in (0, 100]$) feature indices in terms of the magnitude of the weight in binary space for both linear terms (forming index set S_1) and interaction terms (forming index set S_{21} and S_{22} for row and column). That is,

$$S_1 = \{h | \; |w_h| > \textit{quantile}(|\mathbf{w}|, b\%), h \in [d]\}$$
$$[S_{21}, S_{22}] = \{(h_1, h_2) | \; |\Sigma_{h_1,h_2}| > \textit{quantile}(|\Sigma|, b\%), h_1, h_2 \in [d]\}.$$

Then, output $S = \textit{intersection}(S_1, \textit{union}(S_{21}, S_{22}))$ as the important indices set, which includes features that are important in both linear terms and interaction terms. Also, if $i, j \in \mathcal{C}_l$ for $l = 3, \cdots, m$, we set $\Sigma_{i,j} = 0$. This is because while predicting, only one variable in each category would be 1, and interaction between the feature's own category will not contribute to the prediction. Also, b acts as a threshold for the number of features we want to select.

C. Mapping and Feature Ranking

When learning an FM model, we save a map that records the relationship between the original feature space and the encoded binary features. Suppose there are d_l categorical values for context $\mathcal{C}_l$, i.e., $|\mathcal{C}_l| = d_l$. Suppose there are r_l indices appearing in set S, then there are $t_l = r_l/d_l$ fraction of indices of feature l that are important. We generate the following two lists:

1) List of number of indices in the binary vector selected for each context feature: $\mathbf{r} = [r_3, \cdots, r_m]$;
2) List of fraction of indices in the binary vector selected for each context feature: $\mathbf{t} = [t_3, \cdots, t_m]$.

Then we merge the ranked lists $\mathbf{r}$ and $\mathbf{t}$, by simply ranking $\mathbf{r} \circ \mathbf{t}$, which is the element-wise multiplication of $\mathbf{r}$ and $\mathbf{t}$. Note that other reasonable merging methods like element-wise sum will also work. This gives us a **ranking of the features** in original feature space.

TABLE I: Feature Mapping

Binary Index	Context	Value
1	Device	iPhone
2	Device	Macbook
3	Device	Apple TV
4	Time	Daytime
5	Time	Evening
6	Zip code	78712
7	Zip code	95110
8	Zip code	78731
9	Zip code	95121

Example III.1. For the feature mapping given in Table I, there are 3 contextual features: {Device, Time, Zip code} and they have 3, 2, 4 categorical values respectively. Assume that the rows that are highlighted in red are the selected indices: $S = \{1, 4, 5, 6, 7, 8\}$. Then, the two generated lists are: $\mathbf{r} = [1, 2, 3]$; $\mathbf{t} = [1/3, 2/2, 3/4]$. Merging these two lists, we get $\mathbf{r} \circ \mathbf{t} = [1/3, 2, 9/4]$. Based on the merged list, the final ranking of the features is: Zip code, Time, Device.

D. Feature Subsampling

We want to account for prediction error affecting the ranking leading to not selecting the most important features. For example, if we want 10 features, the 11th on the ranking may be more important than the 10th but not likely to be more important than the 1st. To handle this, we adapt the idea of wrapper method, but on a much smaller set.

Suppose we want to choose p features. In this case, we first choose top $q = 2p \sim 4p \ll (m - 2)$ significant features from the ranked feature list, then uniformly subsample p features from these q features. We actually train the model for each of these selected feature sets and choose the set with the best results.

It may be noted that although our framework can automatically select the features, the selected features can be optionally improved with domain knowledge. For example, if the final features include both context "Zip code" and "State", we might choose one of them, if that does not affect the predictive power of the model. Given that typically $p \ll (m - 2)$, it is not hard for a human expert to check such a small set of features, compared to choosing from hundreds or thousands of them. It is also possible to adapt our framework to automatically decide the number of context features to select, say, after feature ranking, instead of subsampling, we choose important features from the top of the ranked list and stop if the gap between the values in $\mathbf{r} \circ \mathbf{t}$ is larger than a certain threshold.

IV. Industrial Dataset Evaluation

In this section, we present the results on industrial datasets, which has hundreds of context features to show the effectiveness and robustness of our framework. The target of the dataset is session progress, as introduced in [19], which is a real valued number in $[0, 1]$, which can be used for recommendation. Session progress is an implicit measure of engagement of the user with the video, which measures how much of the video was watched. Our metric for measuring the accuracy of the prediction is root-mean-square error (RMSE), which is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2},$$

where y_i is the session progress of the i-th session, $\hat{y}_i$ is the predicted session progress of the i-th session, n is the total number of sessions.

Our objective for FM training is to find a model that has the smallest RMSE on test set, and our goal for feature selection is to select a small subset of context features for training, ensuring that the RMSE is better than using all context features or not using any context features at all, and comparable to the set of features selected by a human expert. We have done experiments on different subsets of the data.

A. Dataset Structure

The structure of the datasets we used is shown in Fig. 2. For each dataset, we first filter videos by a minimum number of views: w_{min}, e.g., if $w_{min} = 10$, we only consider the videos viewed more than 10 times in the data. Then, we further filter the subsets by a minimum user session number u_{min}, e.g., if $u_{min} = 5$, we only keep the users that have at least 5 sessions. Note that these subsets are used as the input data, which is different from our subsetting method for data preprocessing.

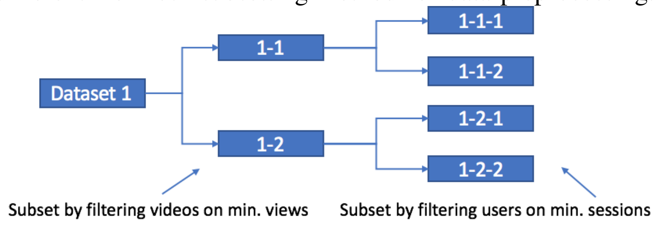

Dataset 1
1-1
1-2
1-1-1
1-1-2
1-2-1
1-2-2
Subset by filtering videos on min. views
Subset by filtering users on min. sessions

Fig. 2: Structure of Dataset 1.

The statistics of datasets we are using to select features and the parameters for preprocessing, constructing important feature indices set: c and b, are shown in Table II.

TABLE II: Statistics of datasets and choice of parameters.

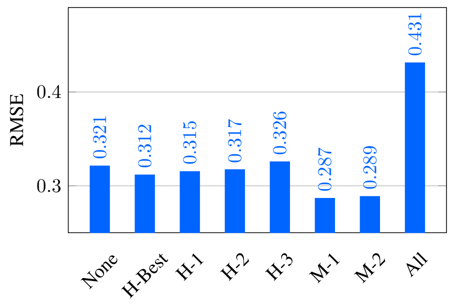
Dataset	size	preprocessed size	# of features	c	b
1-1-1	136.7 MB	55MB	968	50	50
2-1-1	1.6 GB	72.8 MB	994	10	50
3-1-1	6.6 GB	184.6 MB	171	10	50

B. Multiple Runs and Subsampling

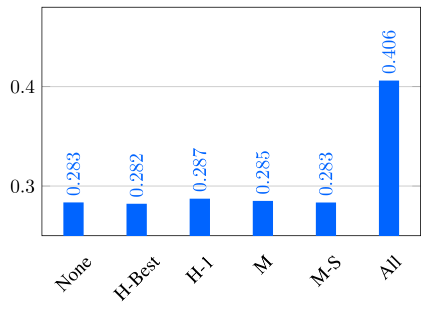
First we test our framework over different runs, as there is a degree of randomness built into our algorithm. In Fig. 3(a), where "H-i" represents a small set of features selected by a human domain expert i, "None" represents no context features used, "All" means that all the contextual features are used, and "Machine (M)" represents the set of features selected by our framework. M-S denotes features selected by machine with subsampling. We can see that the features selected by our framework provide better performance than the empirically chosen ones and is stable across multiple runs. It also demonstrates that if we use all the context features in the model, it would degrade the performance. Next, we check if subsampling can further help in prediction, or at least not affect the prediction negatively while using even fewer features. Fig. 3(b) shows the result for Dataset 2-1-1. We see that subsampling helps prediction with fewer context features.

C. Robustness among subsets

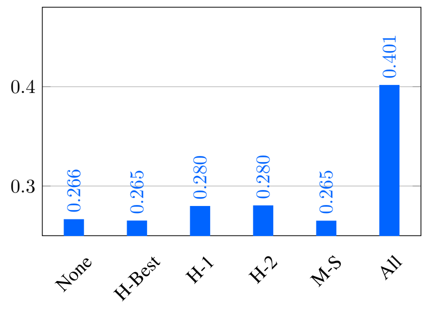
The results in Fig. 4 for Dataset 1 and Fig. 3(c) for Dataset 2 show that the features selected using one 1st level subset (filtering videos) has good performance on other 1st level subsets. For Dataset 1, features are selected using Dataset 1-1-1 and tested on Dataset 1-2-1 and 1-3-1, we see that the prediction quality is maintained. For Dataset 2, features are selected using Dataset 2-1-1 and test on 2-2-1, we observe the the results are robust in this case too. The results in Fig. 3(d), Fig. 3(e), and Fig. 3(f) show that features selected using



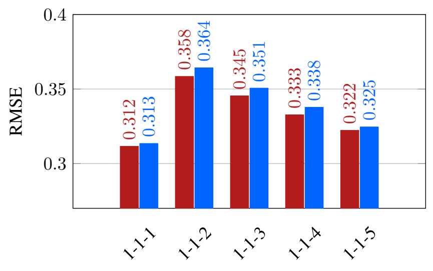
(a) Multiple run on Dataset 1-1-1.



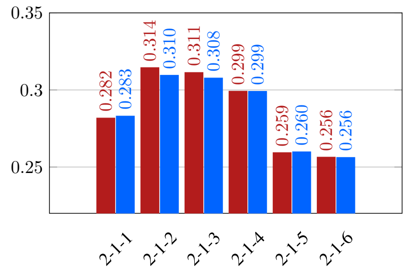
(b) Subsampling on Dataset 2-1-1.



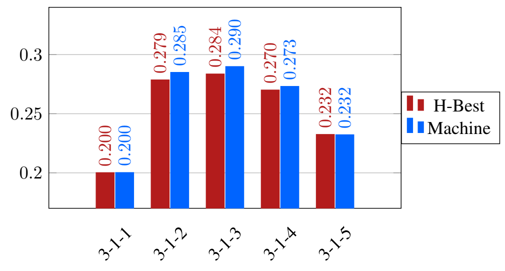
(c) Feature selected with Dataset 2-1-1.



(d) Feature selected with Dataset 1-1-1.



(e) Feature selected with Dataset 2-1-1.



(f) Feature selected with Dataset 3-1-1.

Fig. 3

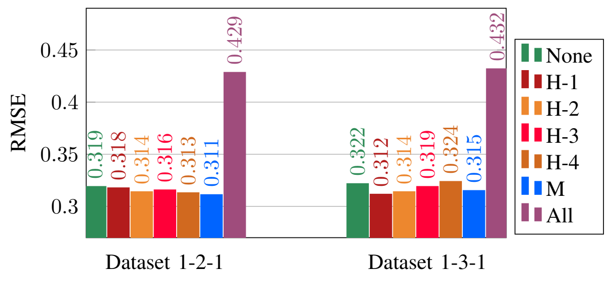


Fig. 4: Feature selected with Dataset 1-1-1.

one 2nd level subset (filtering users) performs well on other 2nd level subsets. We see that for all 3 datasets, the features have competitive performance with the best empirically chosen features, proving the robustness of our framework.

V. Future work and Conclusion

In this paper, we introduced a context selection method for FM model training. We proposed a novel feature selection framework to automatically select a small set of context features to help prediction, along with a preprocessing step of subsetting the data to accelerate training. We validated the efficiency and robustness of our framework with extensive experiments on industrial datasets. Our method is similar to covariance learning in graphical models [20]. In the future, we would like to model the problem as a graph learning problem where each vertex in the graph is a feature variable.

References

[1] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[2] J. Bennett and S. Lanning, "The netflix prize," in *Proc. KDD cup and workshop*, vol. 2007, p. 35, 2007.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. WWW*, pp. 285–295, ACM, 2001.

[4] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender systems handbook*, pp. 217–253, Springer, 2011.

[5] I. Cantador, I. Fernndez-Tobas, S. Berkovsky, and P. Cremonesi, "Cross-domain recommender systems," in *Recommender Systems Handbook*, ch. 27, pp. 919–959, Springer, 2015.

[6] A. Chen, Context-aware collaborative filtering system: Predicting the users preference in the ubiquitous computing environment, in *Location-and Context-Awareness*, Springer Berlin Heidelberg, pp. 244-253, 2005.

[7] L. Song, C. Tekin, and M. van der Schaar. Online learning in large-scale contextual recommender systems, in *IEEE TSC*, 9.3 (2016): 433-445.

[8] G. Wu, V. Swaminathan, S. Mitra, and R. Kumar, "Context-aware Video Recommendation based on Session Progress Prediction," in *ICME*, 2017.

[9] S. Rendle, "Factorization machines," in *ICDM*, pp. 995–1000, 2010.

[10] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proc. SIGIR*, pp. 635–644, ACM, 2011.

[11] R. Ronen, N. Koenigstein, E. Ziklik, and N. Nice "Selecting Content-Based Features for Collaborative Filtering Recommenders," in *Proc. RecSys*, pp. 407–410, ACM, 2013

[12] N. Koenigstein, and U. Paquet, "Xbox movies recommendations: Variational bayes matrix factorization with embedded feature selection," in *Proc. RecSys* , pp. 129–136, ACM, 2013.

[13] X. Geng, T.-Y. Liu, T. Qin, and H. Li, "Feature selection for ranking," in *Proc. SIGIR*, pp. 407–414, ACM, 2007.

[14] J. Xu, K. Lin, P. Tan, and J. Zhou, "Synergies that matter: Efficient interaction selection via sparse factorization machine," in *Proc. SDM*, pp. 108–116.

[15] C. Cheng, F. Xia, T. Zhang, I. King, and M. R. Lyu, "Gradient boosting factorization machines," in *Proc. RecSys*, pp. 265–272, ACM, 2014.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B*, pp. 267–288, 1996.

[17] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," in *JRSSB*, vol. 67, no. 2, pp. 301–320, 2005.

[18] M. Yamada, A. Goyal, and Y. Chang, "Convex factorization machine for regression," *arXiv preprint arXiv:1507.01073*, 2015.

[19] G. Wu, V. Swaminathan, S. Mitra, and R. Kumar, "Online video session progress prediction using low-rank matrix completion," in *ICMEW*, pp. 1–6, IEEE, 2014.

[20] M. I. Jordan, *Learning in graphical models*, vol. 89. Springer Science & Business Media, 1998.