

# Social Network and Click-through Prediction with Factorization Machines

Steffen Rendle  
Social Network Analysis  
University of Konstanz  
78457 Konstanz, Germany  
steffen.rendle@uni-konstanz.de

## ABSTRACT

The two tasks of KDDCup 2012 are to predict the followers of a microblogger (track 1) and to predict the click-through rate of ads (track 2). On first glance, both tasks look different however they share two important challenges. First, the main variables in both problem settings are of large categorical domain. Estimating variable interactions between this type of variables is difficult with standard machine learning models and factorization models have become popular for such data. Secondly, many additional predictor variables are available which requires flexible models based on feature engineering to facilitate model definition.

In this work, it is shown how Factorization Machines (FM) can be used as a generic approach to solve both tracks. FMs combine the flexibility of feature engineering with the advantages of factorization models. This paper shortly introduces FMs and presents for both tasks in detail the learning objectives and how features can be generated. For track 1, Bayesian inference with Markov Chain Monte Carlo (MCMC) is used whereas for track 2, stochastic gradient descent (SGD) and MCMC based solutions are combined.

## Categories and Subject Descriptors

H.3.3 [Information Systems]: Information Search and Retrieval—*Information filtering*

## General Terms

Algorithms, Experimentation, Measurement, Performance

## Keywords

Social Network Analysis, Clickthrough Prediction, Factorization Machine

## 1. INTRODUCTION

KDDCup 2012 [4] consists of two large-scale prediction tasks from the microblogging website Tencent Weibo<sup>1</sup>. The first task is to predict which microblogger a user is following. More specifically, for each user a set of recommended microblogs is given and the prediction task is to rank the microblogs of this set by the likelihood that the user follows a microblog. The two main variables involved in this problem are user and microblog which are categorical variables of large domain. Besides this, other information is present: e.g. user attributes such as gender and age, the social network about followers/ followees and time information about each recommendation of a microblog. The second task is to predict the clickthrough rate of ads given a user and a query. The main variables in this prediction problem are the ad, the user and the query which again are variables of large categorical domains. This task also includes additional information, e.g. age, gender, query tokens, or the position of an ad.

Even though the applications in track 1 and track 2 are different, in both problems the main variables are of large categorical domain. In recent years, factorization models optimized with machine learning techniques have shown great success in such problems, starting from matrix factorization for problems over two categorical variables (e.g. [8]), to tensor factorization over several categorical domains (e.g. [2, 7]) and specialized models for specific tasks, e.g. SVD++ [3]. Factorization machines (FM) [5] are a generic factorization model which allow to use real-valued predictor variables the same way as e.g. linear regression, SVMs, etc. It has been shown that FMs subsume a wide variety of factorization models by simple feature engineering. For inference with FMs, SGD and coordinate descent as well as Bayesian inference with MCMC have been proposed.

In this work, it is shown how FMs can be applied to both tasks of KDDCup 2012. First, FMs are shortly recapitulated (for more details see [5]). Then for each task a detailed description about the target function, predictor variables and feature engineering is given.

The presented solution based on FMs has three key components:

1. For both tracks, using factorized interactions is important to estimate variable interactions reliably.
2. The usage of attribute information is important: in track 1, attributes about the user and in track 2, attributes of users and queries are used.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDDCup '12 Beijing, China

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$15.00.

<sup>1</sup><http://t.qq.com/>

3. In track 1, sequential indicators largely improve prediction quality.

## 2. FACTORIZATION MACHINES

The approaches presented in this work are based on factorization machines. In this section, the basic concepts of FMs are recapitulated. The presentation is restricted to the techniques used later. For more details about FMs, the reader is referred to [5].

### 2.1 Model

Factorization machines are a generic machine learning model that works with real valued input vectors  $\mathbf{x} \in \mathbb{R}^p$ . It models variable interactions up to order  $d$ . The second-order factorization machine (FM) model is defined as

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{j=1}^p w_j x_j + \sum_{j=1}^p \sum_{j'=j+1}^p x_j x_{j'} \sum_{f=1}^k v_{j,f} v_{j',f} \quad (1)$$

where  $k$  is the dimensionality of the factorization and the model parameters  $\Theta = \{w_0, w_1, \dots, w_p, v_{1,1}, \dots, v_{p,k}\}$  are

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{p \times k}. \quad (2)$$

This model is closely related to nested polynomial regression of order  $d$  with the very important difference that FMs use factorized interactions whereas polynomial regression has independent parameters per interaction. This makes FMs an appealing model for prediction problems involving variables of large categorical domain where interactions typically are hard to estimate.

Like any other model using real valued input variables, FMs can work with categorical variables by using binary dummy variables. Likewise any other variable encoding can be used, e.g. for set-categorical variables. In [5] several examples show how variable encodings are related to specialized factorization models.

### 2.2 Learning and Inference

For inference over model parameters  $\Theta$ , several learning algorithms have been proposed from point estimators using coordinate descent/ alternating least-squares and stochastic gradient descent to Bayesian inference based on Markov Chain Monte Carlo sampling. With efficient algorithms, all the proposed learning methods have the same computational complexity. The proposed solutions for KDDCup 2012 mainly uses MCMC inference and for track 2 also SGD. The advantage of MCMC inference is that (1) regularization values are integrated and have not to be tuned in a time-consuming grid-search and (2) uncertainty over model and hyperparameters is taken into account.

Like other supervised learning methods, the training data of FMs is a multiset  $S$  of cases  $(\mathbf{x}, y)$ , where  $\mathbf{x} \in \mathbb{R}^p$  are the predictor variables and  $y$  is the target. The data of  $n$  training cases can be written as a *design matrix*  $X \in \mathbb{R}^{n \times p}$ . In addition, for track 2, it is assumed that every case is assigned with a case weight  $c \in \mathbb{N}^+$ . It is straightforward to extend the algorithms in [5] with weights. This can be done either by changing the learning algorithms or by oversampling cases with sample probability  $c$ . For SGD, oversampling was used and for MCMC the weights were added to the posterior distributions.

#### 2.2.1 Stochastic Gradient Descent (SGD)

For deriving a point estimator of the model parameters, the task is to minimize an objective consisting of loss  $l$  and regularization (here L2)

$$\text{OPT}(S) := \underset{\Theta}{\operatorname{argmin}} \left( \sum_{(c, \mathbf{x}, y) \in S} c l(\hat{y}(\mathbf{x}|\Theta), y) + \sum_{\theta \in \Theta} \lambda_\theta \theta^2 \right) \quad (3)$$

where the loss  $l$  can be e.g. least-squares for regression

$$l^{\text{LS}}(y_1, y_2) := (y_1 - y_2)^2, \quad (4)$$

or logit loss for binary classification ( $y \in \{-1, 1\}$ )

$$l^{\text{C}}(y_1, y_2) := \ln(1 + \exp(-y_1 y_2)). \quad (5)$$

Stochastic gradient descent (SGD) is a popular algorithm to minimize such an objective. In SGD a case  $(c, \mathbf{x}, y) \in S$  is drawn<sup>2</sup> and model parameters  $\theta$  are updated with:

$$\theta \leftarrow \theta - \eta \left( \frac{\partial}{\partial \theta} l(\hat{y}(\mathbf{x}), y) + 2 \lambda_\theta \theta \right) \quad (6)$$

where  $\eta \in \mathbb{R}^+$  is the learning rate. When SGD is applied, several parameters have to be tuned: the learning rate  $\eta$ , regularization  $\lambda$  and the initialization of parameters (here the standard deviation of a normal distribution  $\mathcal{N}(0, \sigma)$ ).

#### 2.2.2 Markov Chain Monte Carlo (MCMC)

MCMC makes Bayesian inference over the predictive distribution by marginalizing over the model parameters and hyperparameters

$$p(y|\mathbf{x}, S) = \int p(y|\mathbf{x}, \Theta, \lambda) p(\Theta, \lambda|S) d\{\Theta, \lambda\} \quad (7)$$

This distribution is approximated by sampling from the posterior distribution  $p(\Theta, \lambda|S)$ . In [1], an efficient Gibbs sampler for both model parameters and hyperparameters has been proposed. In [5], an extension for grouped parameters and classification is presented.

Applying MCMC is much easier than SGD as the regularization is integrated and there is no learning rate. The only hyperparameter to tune for MCMC inference is the standard deviation for initialization where a proper choice accelerates the convergence of the sampler. The standard deviation is typically quite stable among different choices of features and different numbers of latent dimensions. For each track of KDDCup12, the standard deviation was only searched in the first experiments and then kept constant.

#### 2.2.3 Accelerated Learning

Feature generation in general can lead to very large design matrices  $X \in \mathbb{R}^{n \times p}$  with many non-zero elements which might slow down inference for any machine learning model applied to such data. Often the design matrices involve repeating patterns and such structure can be used to accelerate learning. In KDDCup12, repeating pattern occur for example in track 1 when all other users a user follows are used as predictor variables. For KDDCup12, an accelerated MCMC learning algorithm is used that makes use of such patterns. The enhancement is exact (no approximation) and

<sup>2</sup>With oversampling cases are not drawn uniformly but proportional to their case weight  $c$ .

thus generates the same solution as the standard algorithm in [5]. The technical details about this speedup algorithm are not in the scope of this report on KDDCup12.

A second enhancement to speed up the convergence is to remove variable interactions that are supposed to be non-informative. Let  $\mathcal{B} = \{B_1, B_2, \dots\}$  be a partition of variable indices  $\{1, \dots, p\}$ , i.e.  $B_i \cap B_j = \emptyset$  for  $i \neq j$  and  $\bigcup_i B_i = \{1, \dots, p\}$ . All interactions between variables  $x_j$  and  $x_{j'}$  are dropped from the FM, if both  $j$  and  $j'$  are in the same block  $B_i$ . That means only interactions between variables of different blocks remain:

$$\hat{y}^*(\mathbf{x}) := w_0 + \sum_{j=1}^p w_j x_j + \sum_{l=1}^{|\mathcal{B}|} \sum_{l' > l}^{|\mathcal{B}|} \sum_{j \in B_l} \sum_{j' \in B_{l'}} x_j x_{j'} \sum_{f=1}^k v_{j,f} v_{j',f}. \quad (8)$$

Examples for a reasonable partition is to place the indices of all predictor variables that belong to one (set) categorical variable into one block or all variables describing the user. Applying eq. (8) to such a partition corresponds to removing all interaction between variables within the same block. E.g. the interaction of a user's age with the user's ID could be removed. Removing interactions decreases the expressiveness of the model in general but facilitates learning model parameters provided that the assumptions induced by the blocks are (approximately) correct.

In the presented approaches for KDDCup12, by default, all predictor variables generated from a categorical or set-categorical predictor variable are placed in an individual block (per (set) categorical variable). E.g. all dummy variables that describe the friends of a user are placed in one block and thus the model does not try to estimate interactions between two friend variables. For all MCMC-based models in track 1 and track 2, interactions between variables in the same block are removed. In track 1, a second model is described where all interactions between any user variable is removed. Note that with blocks, only interactions between variables in the same block are dropped, but all other interactions between variables of different blocks remain.

### 3. TRACK 1: FOLLOWER PREDICTION IN A SOCIAL NETWORK

The prediction problem of track 1 is to predict if a user accepts a recommendation to follow the posts of an 'item' – items are important microbloggers and a subset of all users. The data consists of (user, item, time)-tuples with a binary target value that states whether the user has accepted the recommendation of the item at that time. The test set has the same tuples but the target is missing and the task is to rank all items recommended to the user in the test set by the likelihood that the user accepts the recommendation. The problem itself is a binary classification problem but the evaluation protocol uses the ranking *measure mean-average precision at 3* (MAP3). More details about the task can be found in [4].

The proposed approach models the original binary classification problem. That means a FM for classification of a (user, item, time)-tuple is set up.

## 3.1 Predictor Variables

In the following, the predictor variables generated for track 1 are described. Three different types of variables are presented: (1) the main indicators of the prediction problem, (2) additional background knowledge about users and (3) sequential indicators derived from the time variable.

### 3.1.1 Main Indicators

The two main predictor variables for modeling if a user accepts a recommendation of an item are:

- user ID
- item ID

Both variables are of categorical domain and are encoded in the FM with binary dummy variables – as it will be done with any categorical variable. The time variable is not used directly as the training and test sets do not overlap in time. However, sequential features are derived from the time variable (see sec. 3.1.3).

The dataset provides rich additional information about the user and item in form of attributes and follower interactions. In the proposed model, additional user information is used intensively. Item information is not used at all because the number of items is low and thus it is assumed that the item IDs are informative enough.

### 3.1.2 User Attributes & Social Network

The dataset provides information about each user in terms of attributes and in the follower-graph. All the provided user variables are used in the model:

- age of the user
- gender of the user
- interaction of age and gender
- number of tweets
- tags
- keywords
- set of all users that the user follows (from the social network)

All features are treated as categorical variables which also allows to directly handle missing values as levels. The additionally provided *user action* information is not used.

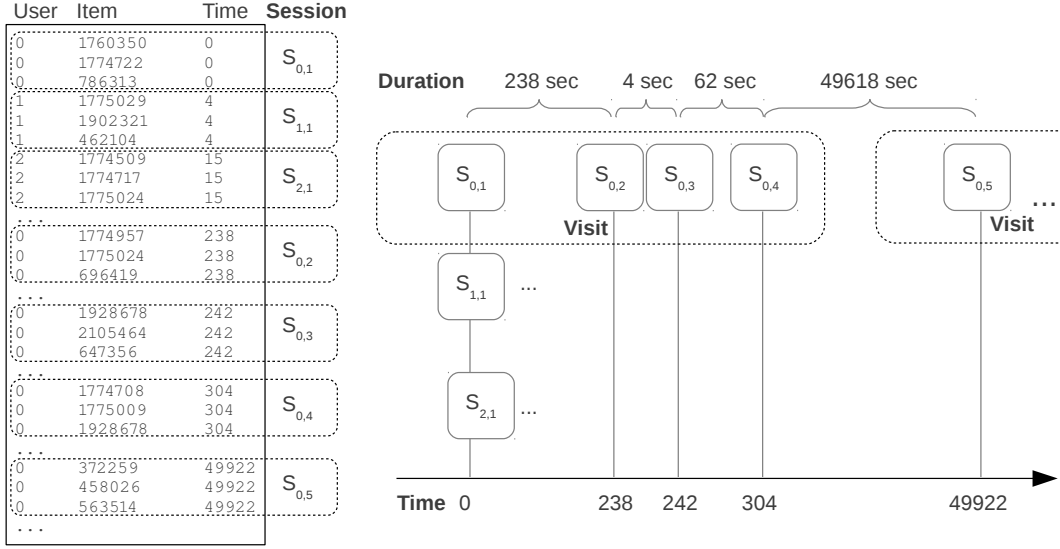
From these variables especially the social network and the keywords have been found to be predictive. Tags and number of tweets have only a minor effect.

### 3.1.3 Sequential Indicators

In the dataset each recommendation is assigned a time stamp in seconds. In the following the term *session* is used for all the recommendations that are made for a certain user at a certain time (see fig. 1 for an illustration). In the dataset, there are typically three items in a session. Let  $t(s)$  be the time assigned to a session  $s$ . For a user  $u$ , one can write his actions as a sequence of  $n_u$  sessions:  $s_{u,1}, s_{u,2}, \dots, s_{u,n_u}$  where  $\forall j < l : t(s_{u,j}) < t(s_{u,l})$ .

The *duration* of a session  $d(s_{u,j})$  is the time difference between the current session and the next session of the same user:

$$d(s_{u,j}) := t(s_{u,j+1}) - t(s_{u,j})$$



**Figure 1: Raw data (left) from the recommendation logs. Items with the same user and time stamp are grouped into *sessions*. The time difference between two contiguous sessions of the same user is the *duration* of a session (right). Sessions without a large gap are grouped in *visits*.**

This variable is highly predictive which also makes intuitively sense: if a user sees the recommendations only very shortly, it is unlikely that there is enough time for accepting any recommendation. Not only the duration of the current session but also the one of the previous and next sessions are predictive. One explanation is that many short sessions could indicate that the user is currently not in the mood of checking recommendations. If some of the sessions are longer, the user might be aware of recommendations. Another variable that can be derived for capturing this effect is the number of sessions in the next or previous N seconds.

Finally, the user might be more or less open for recommendations depending on how long (s)he uses the system. Reasons could be that the trust in the recommendations grows/ decreases with more interactions. This can be captured by a variable holding the session index – remember that this index is sorted by time. The session index has also been found very predictive.

Another direction to generalize sessions is to summarize them to *visits* where a visit is defined as a sequence of sessions where no contiguous pair of sessions has a time difference larger than 60 minutes. The users actions can be described as a sequence of visits. In contrast to session index, visit index has only a minor effect.

In total, to model a given  $(u, i, t)$ -event, the session index  $j$  and visit index  $l$  is computed and the following indicators are used:

- duration of this session:  $d(s_{u,j})$
- duration of previous session:  $d(s_{u,j-1})$
- duration of next session:  $d(s_{u,j+1})$
- duration of next next session:  $d(s_{u,j+2})$
- duration of next next next session:  $d(s_{u,j+3})$
- session index:  $j$

- session index in descending order:  $n_u - j$
- number of sessions in the next 60 seconds:  $|\{j' > j : |t(s_{u,j}) - t(s_{u,j'})| < 60\}|$
- number of sessions in the previous 60 seconds:  $|\{j' < j : |t(s_{u,j}) - t(s_{u,j'})| < 60\}|$
- visit index:  $l$

All numbers are treated as categorical variables and are encoded for FMs with binary dummy variables. The duration variables are converted to log-scale, truncated and treated as categorical variables.

### 3.2 Results

Several subsets of these features and two partitions of variables have been tested. For the partitions, in general, all predictor variables belonging to a (set) categorical variable are assigned to an individual block. Within these blocks all interactions are removed. This is the basic model that is referred to as the *FM with user interactions*. A second model is build where the pairwise interactions between all user variables are dropped, e.g. the interaction between user ID and age is removed. This model is referred to as *FM without user interactions*. Note that this means that interactions between two user variables are dropped but interactions between each user variable and all non-user variables (e.g. items or sequential indicators) remain. For the *FM without user interactions* a simple feature selection suggests to use all of the mentioned indicator variables. For the *FM with user interactions*, feature selection suggests to drop two features: *duration of next next next session* and *visit ID*.

Both FMs are learned with MCMC which facilitates hyperparameter selection because neither any regularization value nor any learning rate has to be chosen. Instead, MCMC makes inference over regularization parameters. Two hyperparameters remain that have to be selected: the standard

Method	$k$	# Samples	MAP3 (public)	MAP3 (private)
FM with user interactions	32	128	0.42405	0.41111
		256	0.42514	0.41192
FM without user interactions	22	128	0.42663	0.41491
		256	0.42802	0.41577
		384	0.42833	0.41582
Ensemble	n/a	n/a	0.42909	<b>0.41622</b>

**Table 1: Two variants of FMs and an ensemble of both models for track 1. The first FM includes interactions within user variables (e.g. interactions between user ID and user age). In the second one, these interactions have been removed.**

deviation of the initialization and the number of factors. For standard deviation, a value of 0.05 has shown fast convergence. For the number of factors,  $k = 22$  is used for the model without interactions within user attributes and  $k = 32$  for the model with interactions.

Table 1 shows the results of these two variants of FMs. It can be seen that the removed interactions seem in fact to be unimportant and removing them helps the MCMC sampler. The table shows also MAP3 measurements for different MCMC sample sizes. The final model is an ensemble of both FM models. The ensemble is a linear combination with a larger weight on the *FM without user interactions*. It slightly improves the score of the best standalone model.

## 4. TRACK 2: CLICKTHROUGH PREDICTION

The target in track 2 is to predict the clickthrough rate of ads. For evaluation, a weighted AUC (wAUC) is used and thus the predicted number is not of primary interest (for evaluation) but only the rank. Nevertheless, the proposed approaches are based on learning the clickthrough rate directly<sup>3</sup>. Thus a regression problem is set up where the clickthrough rate is used as target variable. For the training dataset, as regression target the empirically observed clickthrough rate is used, i.e. *number of observed clicks / number of impressions*. The empirically estimated clickthrough rates are highly unreliable if the number of impressions is low and thus a weighted regression is used where the weight is the number of impressions. Moreover using the number of impressions as weights also reflects the evaluation measure wAUC that uses the same weights.

### 4.1 Predictor Variables

This task includes several predictor variables about the impression and click-events, among them the ad ID, user ID, query ID but also IDs for URLs, descriptions, etc. Moreover, there is additional information about some of the ID variables, e.g. the tokens that form a query, tokens in the description, etc.

#### 4.1.1 Main Indicators

Personalized clickthrough prediction can be characterized by the ID variables for the user, query and ad. Additionally, for each ad impression, the position in the presented list is available. There are also other ID variables present such as for URLs and titles but in the proposed approach only the following main variables are used:

- ad ID
- user ID
- query ID
- position of the ad which reflects the position in the impression list.

All of them are treated as categorical variables.

In the dataset of KDDCup12, user ID and query ID are of very large domain with each more than 20 million levels (see table 2). Even more important, a large fraction of the levels appearing in the test set do not occur in the training set. This makes it impossible to directly estimate values for these levels.

#### 4.1.2 Attributes

To overcome the problem that many levels for user ID and query ID never appear in the training set, attribute information can be used. The user attributes are age and gender and for each query, a description with a set of token IDs exist. This information is supposed to be especially important for users/ queries where the main indicators are not predictive. From the attributes in the KDDCup12 dataset, the proposed model uses the following four variables:

- user gender
- user age
- query tokens
- title tokens

The user attributes are treated as categorical variables and the tokens as set categorical variables.

### 4.2 Models

The goal is to predict the clickthrough rate of an ad given the user and the query. In general, the problem is described in enough detail by using the IDs of user, query and ad. However as shown in table 2, most levels of user and query that appear in the test set are not present in the training set, so IDs are not predictive on a larger part of the test data. To reflect this, three different models are developed, one relying only on IDs, one relying only on attributes of users and queries but not on IDs and a last one relying on both attributes and IDs. A summary of the performance of these approaches can be found in table 3.

<sup>3</sup>This holds for the proposed standalone models. For ensembles, ranks are used.

Variable	Levels overall	Levels in test	Levels only in test
Ad ID	670,560	300,012	28,853
User ID	23,907,495	3,263,681	1,883,948
Query ID	26,243,385	3,801,978	2,121,309

**Table 2: In track 2, most levels of the user ID and query ID variable that appear in test set are not present in the training dataset. Values for these levels cannot be estimated.**

Model	Inference	wAUC (public)	wAUC (private)
ID-based model ( $k = 0$ )	SGD	0.78050	0.78086
Attribute-based model ( $k = 8$ )	MCMC	0.77409	0.77555
Mixed model ( $k = 8$ )	SGD	0.79011	0.79321
Final ensemble	n/a	0.79857	<b>0.80178</b>

**Table 3: Track 2 results for the best standalone approaches and for a ranking ensemble over variants of the three standalone approaches.**

#### 4.2.1 ID Model

With user ID or query ID as predictor variable, it is very easy to overfit the training data because many of the levels that appear in test set never appeared in the training dataset (and thus cannot be learned). In basic experiments, L2-regularization alone was not effective to prevent overfitting, so SGD with early-stopping is applied to models that include user IDs and query IDs.

The first model uses the three main indicators that are ad ID, user ID, query ID and the position of the ad. This model contains no variable interactions ( $k = 0$ ) and is trained with SGD. For regularization only early stopping is applied, i.e. no L2 regularization is used. This *ID model* scores 0.78050 (public) and 0.78086 (private).

#### 4.2.2 Attribute Model

Whereas the first model focuses only on the IDs, the second one replaces the user ID and query ID with the attributes of the user as well as query and title tokens. Now the number of observations per level in the test set is comparable to the ones in the training set. Thus MCMC is applied for learning. As factorization dimension  $k = 8$  is chosen. This *Attribute Model* scores 0.77409 (public) and 0.77555 (private).

#### 4.2.3 Mixed Model

The third kind of model uses both attributes and IDs. Again because of the user ID and query ID variable, SGD with early stopping is used for inference. In total, this *mixed model* uses the variables: ad ID, query ID, user ID, position, user attributes and query tokens. Several instances of this model have been learned and the best one (with  $k = 8$ ) achieves a score of 0.79011 (public) and 0.79321 (private).

### 4.3 Ensembles

The models created so far focus on different aspects of the problem – especially the ID and attribute model – and thus ensembles are promising. The ensembling model chosen here is just a linear combination of base classifiers where the weights are chosen heuristically (usually uniformly) by investigating the accuracy of the single models on the public leaderboard<sup>4</sup>. The reason for this choice is that it was difficult to sample a representative holdout set – the true test set

<sup>4</sup>The results of single models on the public leaderboard were often not as expected from validation split scores. Moreover,

is a time split but the provided data does not contain time, so it was impossible to use the same sampling procedure.

#### 4.3.1 Rank Ensembles

Ensembles can be built in different ways, not only regarding the ensembling method but also regarding the predictor variables. The most common way of ensembling is to use the predicted target values of single models as predictor variables in the ensemble. As the single methods use regression as prediction target, the obvious choice would be to ensemble over clickthrough predictions. However, the models are very diverse and might produce results on different scales which could make unsupervised linear ensembling of clickthrough predictions unreliable. Instead, following [6], ensembles over rank estimates are build. This reflects also the evaluation measure wAUC well which only takes the rank into account but not the regression value itself. For a rank ensemble, for each base classifier, the predictions are replaced by their rank and then the predicted ranks are linearly combined.

#### 4.3.2 SGD Ensembles

As SGD produces point estimators of parameters and thus also for predictions, several model instances are learned for the SGD based approaches and ensembled. E.g. for the mixed model, the final ensemble includes three variants with slightly different choices of  $k$ , early stopping iteration, learning rate and regularization. Ensembling just these three variants of the mixed model results in a wAUC of 0.79269 (public) and 0.79616 (private).

#### 4.3.3 Final Ensemble

An ensemble of all three types of models (ID model, attribute model and mixed model) and different variations for hyperparameters of the SGD trained models (ID model and mixed model) was the final submission that scored 0.79857 (public) and 0.80178 (private).

## 5. CONCLUSION

In this work, it was shown how Factorization Machines (FM) can be applied to predict the follower-relation in a small variations in hyperparameters sometimes had an unexpected large difference on the public leaderboard. Thus, the public leaderboard scores were only used very conservatively for tuning the approaches, model selection and ensemble weights.

microblogging community and the clickthrough-rate of ads. Both problems have commonalities as their main variables are of large categorical domain. FMs is a model class that is especially appealing for prediction problems over such data because it allows to estimate variable interactions reliably due to factorized interactions. Both prediction problems offer a wide variety of additional variables, e.g. attributes of users, the social network involved or query tokens. Moreover important variables that encode sequential information can be derived in track 1. In contrast to other factorization models, all the presented variables can be used in FMs by simple preprocessing. In general, only feature vectors have to be created and the generic FM model can be applied.

## 6. REFERENCES

- [1] C. Freudenthaler, L. Schmidt-Thieme, and S. Rendle. Bayesian factorization machines. In *NIPS workshop on Sparse Representation and Low-rank Approximation*, 2011.
- [2] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.
- [3] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.
- [4] Y. Niu, Y. Wang, G. Sun, A. Yue, B. Dalessandro, C. Perlich, and B. Hamner. The tencent dataset and KDD-Cup'12. In *KDD-Cup Workshop*, 2012.
- [5] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.
- [6] S. Rendle and L. Schmidt-Thieme. Factor models for tag recommendation in bibsonomy. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2009.
- [7] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90, New York, NY, USA, 2010. ACM.
- [8] N. Srebro, J. D. M. Rennie, and T. S. Jaakola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.