GB-CENT: Gradient Boosted Categorical Embedding and Numerical Trees

Qian Zhao
University of Minnesota
Minneapolis, USA
zhaox331@umn.edu

Yue Shi
Yahoo Research*
Sunnyvale, USA
yueshi@acm.org

Liangjie Hong
Etsy Inc.†
Brooklyn, NY, USA
lhong@etsy.com

ABSTRACT

Latent factor models and decision tree based models are widely used in tasks of prediction, ranking and recommendation. Latent factor models have the advantage of interpreting categorical features by a low-dimensional representation, while such an interpretation does not naturally fit numerical features. In contrast, decision tree based models enjoy the advantage of capturing the nonlinear interactions of numerical features, while their capability of handling categorical features is limited by the cardinality of those features. Since in real-world applications we usually have both abundant numerical features and categorical features with large cardinality (e.g. geolocations, IDs, tags etc.), we design a new model, called GB-CENT, which leverages latent factor embedding and tree components to achieve the merits of both while avoiding their demerits. With two real-world data sets, we demonstrate that GB-CENT can effectively (i.e. fast and accurately) achieve better accuracy than state-of-the-art matrix factorization, decision tree based models and their ensemble.

Keywords

recommender systems; matrix factorization; low-dimensional embedding; gradient boosting; decision trees; numerical and categorical features; large cardinality

1. INTRODUCTION

Among a variety of machine learning models, Matrix Factorization (MF) and ensemble of decision trees are widely used in tasks of prediction, ranking and recommendation. Prior studies have shown their effectiveness [6, 4]. MF has been generalized from traditional Singular Value Decomposition (SVD) in linear algebra to low-dimensional embedding models, e.g. from FunkSVD [13] to SVDFeature [7] and Factorization Machine [24] (FM) in recommender systems. Decision trees like CART [3] are popular in practice because they are good at handling high-order nonlinear interactions among input features. Their performance can be further enhanced when many trees are ensembled. For instance, Gradient Boosting Machine (GBM) proposed by Friedman [12] can boost any weak learner which has better than random performance to be a strong predictor. Particularly, a widely used model Gradient Boosted Decision Trees (GBDT) is a GBM using trees as the weak learners. Random Forest [2], on the other hand, which represents another ensemble approach, bagging (v.s. boosting), achieves great performance by averaging many decision trees mainly because of the variance reduction compared with each individual tree.

In real-world applications, input features are generally divided into two types: numerical features and categorical features. From this perspective, both embedding based and decision tree based models seem to be good at handling one type but not so good at the other. For example, for embedding based models, it is natural to embed categorical user IDs into a low-dimensional space to represent user latent preferences, embedding the release year of a movie does not make as much sense. When the scale of a numerical feature is big and its relationship with the output is nonlinear, sophisticated preprocessing is necessary to make sure the embedding algorithm is stable and the model is fitted well. For decision trees, on the contrary, although it is relatively easy to split on low cardinality categorical features by one-hot encoding [23], when the cardinality of the feature becomes large (e.g. user IDs), it is very expensive to split on it and have to resort to approximated splitting algorithms [4]. In practice, such large cardinality categorical features are sometimes transformed to be numerical by either computing response statistics for the feature or embedding them into low-dimensional space before feeding into decision trees. This two-stage modeling approach is usually much more complex to maintain in practice than a single one.

In this paper, we design a new model that composes of both embedding and tree components to solve the above problem. It harnesses categorical features with large cardinality and handles high-order nonlinear interaction of the input features. From the application point of view, it is also very intuitive to interpret and hence makes it easier to inspect for potential modeling problem. Our major contributions in this work are summarized as below:

- We design a new predictive model, GB-CENT, that leverages the advantages of the latent factor models and gradient boosted decision trees in respect to their ca-

*Yue Shi is now at Facebook.
†Liangjie Hong was at Yahoo Research when this work was done.

WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052668

pability of exploiting categorical features and numerical features. We demonstrate that GB-CENT performs significantly better than state-of-the-art factorization and decision tree based models that do not explicitly differentiate categorical features from numerical features, such as SVDFeature, FM and GBDT.

- We demonstrate that GB-CENT out-performs state-of-the-art ensemble of matrix factorization and GBDT in terms of prediction accuracy.
- We show that GB-CENT is much faster to learn than the ensemble of matrix factorization and GBDT.

The rest of this paper is organized as follows. We discuss the related work in §2 and introduce our proposed method in §3, followed by experiments and evaluations on real world datasets in §4. Further discussions and future work can be found in §5 and §6 and we conclude the paper in §7.

2. RELATED WORK

MF techniques especially the optimization version of SVD (i.e. low dimensional approximation of the original matrix with respect to Frobenius norm) has been used by recommender system researchers on the rating matrix of users on items [21] to predict unobserved ratings in the matrix. In addition to the inner product of two low-dimensional factors, Funk et al. [13] added bias terms into the approximation function, which gives FunkSVD model. Koren et al. demonstrated that SVD++ [20] which further models the set of rated items by users with latent factors can improve rating prediction accuracy. Real systems usually have both explicit feedback of ratings from users but also much more implicit behavioral data of clicks, purchases etc. Hu et al. [17] introduced the confidence level parameter to model the associated uncertainty on user preference with implicit feedback data. Singh et al. proposed collective matrix factorization [25] to model multiple available relations among the entities in the domain, such as rating, clicking, tagging matrices in recommender systems, essentially treating it as weighted multi-objective optimization problem. Karatzoglou et al. [18] modeled user behavioral data as a User-Item-Context N-dimensional tensor instead of the traditional 2D User-Item matrix. They showed that the accuracy benefits of their Multiverse Recommendation by effectively modeling contextual information with tensor factorization. Agarwal et al. proposed regression based latent factor models in [1], where response to be predicted is modeled as a multiplicative function of row and column latent factors that are estimated through separate regressions on known row and column features. SVDFeature by Chen et al. [7] and Factorization Machine (FM) by Rendle [24] can model any second-order interaction of input features by embedding them in a low-dimensional vector space to predict the response variable. The difference between the two is that SVDFeature explicitly categorizes input features into three groups including a global group, user group and item group and only the features indicated in user and item groups are interacted with each other. In comparison, FM by default models the second order interaction between all pairs of input features. From above, we see a transition from traditional linear algebra based SVD to optimization based supervised embedding, similarly to frameworks of graph embedding for various kinds of relational graphs [26].

Quite different from embedding based models, decision trees such as CART [3] split the input space into a set of rectangular regions and model the distribution of response variables in the regions to make prediction. It takes into account high-order nonlinear interaction among input features when deeper trees are fitted. When using trees as components, powerful predictive models can be fitted through the frameworks of generalized additive modeling (GAM) [15] or GBM [12]. Different from GAM where backfitting algorithms [15, 11] are usually used to fit component functions iteratively until convergence, GBM greedily fits the negative gradient of the current predictive function with another component function, e.g. another tree for GBDT. GBDT has been shown to be very effective in practice. Several well optimized open-source implementations are available, such as *gbm* R package, *scikit-learn* [23] in Python and *xgboost* [4] in C++.

Many researchers have approached the problem of combining generalized MF and decision trees to solve real-world application problems. Zhou et al. proposed functional matrix factorization (fMF) [28] to attack cold-start recommendation. fMF constructs a decision tree for the initial interview questions of onboarding a new user and associate latent profiles for each node of the tree. It enables the recommender to query a user adaptively according to the user's prior response and gradually refine the user profiles. Karimi et al. further improved the speed of tree construction by a method called Most Popular Sampling [19] without harming the accuracy of rating prediction. Their second part of work in [19] tested the idea of improving prediction accuracy by factorizing at nodes of the tree, which is very similar in concept to what Zhong et al. [27] proposed – Contextual Collaborative Filtering via Hierarchical Matrix Factorization. They showed that instead of uniformly decomposing user rating matrix, rating prediction accuracy can be improved by first partitioning the matrix into sub-matrices under different contexts, factorizing those sub-matrices and ensembling at the end. They adopted Random Decision Trees (RDT) [10], an efficient random partition technique, for the partition process. Following the work of SVDFeature in [7], Chen et al. [5] further theorized the problem of utilizing auxiliary information in MF as general functional MF, by expanding the form of a latent feature from a single value to be an additive function of the auxiliary information with component functions defined in certain families. For example, when all the component functions are restricted to be the same decision tree, it reduces to be the model proposed by Zhong et al. [27]. They proposed to learn the model in gradient boosting framework and developed algorithms to automatically search for suitable feature functions from infinite functional space and hence could demonstrate improved prediction accuracy on real-world data sets.

3. GB-CENT

3.1 Model Description

We define the following notations to describe our model *Gradient Boosted Categorical Embedding and Numerical Trees* (GB-CENT). Consider a data set with N instances. Each instance is denoted as (x, y) where x is a tuple with two groups of features: categorical group a and numerical group b, i.e. $x = (a, b)$. y is the output or response variable we want to predict. Further, let $a = \{a_0, a_1, a_2, ..., a_k\}$ and $b = (b_1, b_2, ..., b_p)$, where $k + 1$ is the cardinality of a and p

is the length of b. p is fixed for all instances in the data set while k varies for different instances. Note that a is a set and we define $a_0 = Root$ where $Root$ is a special feature contained by all instances. For example, the instance could be a rating event by a user on a movie in a movie recommender system, where y denotes the value of the rating, a denotes the user's ID, the movie's ID, the movie's set of genres and b denotes the average rating, release year of the movie. Then, `CENT` model is defined as follows.

$$\hat{y(x)} = \underbrace{\underbrace{\sum_{i=0}^{k} w_{a_i}}_{bias} + \underbrace{(\sum_{a_i \in U(a)} Q_{a_i})^\intercal (\sum_{a_i \in I(a)} Q_{a_i})}_{factor}}_{CAT\text{-}E} + \underbrace{\sum_{i=0}^{k} T_{a_i}(b)}_{CAT\text{-}NT} \tag{1}$$

where $\hat{y(x)}$ is the prediction function, w is a real vector, Q is a real matrix with fixed d rows and T represents trees with only b as the input features. They are all components to learn from data and are **indexed with categorical features** a_i. That is, the cardinality of the set of all possible values of a in the data set determines the length of w, the number of columns in Q and the number of trees in the final `CENT` model. $U(a)$ and $I(a)$ represent the user side and the item side of a respectively which are indicated by practitioners and both default to be $a_{1:k}$ (i.e. modeling all second-order interactions among both sides of categorical features, similar to Factorization Machine [24]). On the other hand, for example, we can also choose to only model the interaction between user side categorical features (i.e. $U(a)$) of user-ID, gender, country and item side categorical features (i.e. $I(a)$) of movie-ID, genres, actors similar to `SVDFeature` [7]. For categorical feature values without many **supporting instances** (xs that have a particular categorical value a_i), it may not be necessary or appropriate to learn corresponding components because of a lack of evidences.

We call the former two components with parameters w and Q in `CENT` model as `CAT-E` (reads as "cat embedding") and call the latter components of T as `CAT-NT` (reads as "cat numerical trees"). Although we name this model `CENT`, the component functions are not new. The most related work in the literature is Chen et al.'s work [6] for KDD CUP 2012 competition[1], where factorization model and additive forests are ensembled to achieve leading performance in a collaborative followee recommendation task. Particularly, we point out what differentiate `CENT` from previous work as follows.

3.2 Model Comparison

Differences from Additive Forest [6]. The approach of explicitly modeling categorical features and numerical features with different forms of functions is novel. We show the benefits of this approach in the results section. `CAT-E` component is similar to the form of `SVDFeature` [7] or `Factorization Machine` [24] but leaving out numerical features. However, as mentioned in previous sections, we propose not to embed numerical features in low-dimensional space because of potential scale, nonlinearity problems and lack of application level understanding (e.g. embedding the release year of a movie is not intuitive). `CAT-NT` components explicitly model the interaction among only numerical features (i.e. the tree T_{a_0} corresponds to $a_0 = Root$) and between the categorical group and numerical group features through conditional tree learning, fully utilizing the benefits of decision trees in modeling high-order nonlinear interaction effects.

The categorical features themselves may be specified to be high-order interactions such as going from *uni-gram* to *n-gram* for texts, in which case higher-order interactions between categorical group and numerical group are naturally taken care of by `CENT`.

`CAT-NT` can learn trees for a context, a geolocation, a user group, an item group and even a specific user or item in recommender systems. This generalizes the additive forest model by Chen et al. [6] where a forest was learned for each item, except that we only use one tree instead of a forest of boosted trees here and each tree has only numerical features as the input letting `CAT-E` handle the effects of categorical features.

From generalization and memorization perspective of machine learning models, `CENT` model also has similarity with the recent work by Cheng et al. [8] on combining wide and deep learning in recommender systems. Specifically, `CAT-E` generalizes by embedding users and items into low-dimensional space capturing their general profiles, tastes, topics and mutual similarities, while `CAT-NT` memorizes each user or item's peculiarities in relevant numerical metrics, e.g. a user might only be interested in movies released after 2010 and with average rating higher than 4 stars.

Differences from GBDT. In order to clarify the differences between `CAT-NT` and `GBDT` [12, 11], we briefly introduce `GBDT` here. A boosted tree model is a sum of decision trees as shown in Equation 2 where M is the number of trees (a pre-specified parameter) and x is an instance as defined previously.

$$f_M(x) = \sum_{m=1}^{M} T_m(x) \tag{2}$$

Note that it uses both categorical a and numerical b as input. a is one-hot encoded [23], i.e. representing a as a binary-valued (0 or 1) sparse vector with its length equal to the cardinality of the categorical features in the data set and most of the entries being zeros except the entries corresponding to the categorical feature values in a. At m-th step of learning a `GBDT` following gradient boosting framework [12], a new tree T_m is learned to fit the negative gradients of $\sum L(y, f_{m-1}(x) + T_m)$ where L is certain loss function (see [11] for the negative gradients of different loss functions) and the summation is with respect to N instances in the whole data set.

To summarize, there are three major differences between `CAT-NT` and `GBDT`.

1. The number of trees in `CAT-NT` depends on the cardinality of categorical features in the data set, while `GBDT` has a pre-specified number of trees M.
2. Each tree in `CAT-NT` only takes numerical features b as input while `GBDT` takes in both categorical a and numerical b.
3. Learning a tree for `GBDT` uses all N instances in the data set while the tree for a categorical feature a_i in `CAT-NT` only involves its supporting instances.

[1] http://www.kddcup2012.org/

Algorithm 1: Training Gradient Boosted Categorical Embedding and Numerical Trees (GB-CENT)

Data: training set $(X, Y) = ((A, B), Y)$ with each row/instance as $(x, y) = ((a, b), y)$ where a is the categorical group of features and b is the numerical group of features
Optional Data: validation set (X', Y') in the same format as (X, Y)
Parameters: *maxTreeDepth, minNodeSplit, minTreeSupport, minTreeGain*

w, $Q \leftarrow$ run stochastic gradient descent algorithm with L as the loss function of `CAT-E`, with A as input and Y as the label (optionally monitored by A' and Y' to prevent overfitting)
$S(c) \leftarrow$ compute a mapping from all possible categorical feature values c in A to the number of supporting instances
$C \leftarrow$ get a list of all possible categorical feature values c sorted descendingly according to the support $S(c)$
$\hat{Y} \leftarrow$ predictions of Y from `CAT-E` (with w, Q; similarly on validation set which is optional)
for c *in* C **do**
  if $S(c) >= minTreeSupport$ **then**
    $B(c)$, $Y(c) \leftarrow$ the subset of supporting instances of c (similarly on validation set, optional)
    $\hat{Y(c)} \leftarrow$ the subset of predictions on the supporting instances of c (similarly on validation set, optional)
    $R(c) \leftarrow$ compute the negative gradients of L at $\hat{Y(c)}$, Y(c)
    $t \leftarrow$ run a regression tree learning algorithm with $B(c)$ as the input and $R(c)$ as the label, and $maxTreeDepth$ and $minNodeSplit$ as parameters to control the size of the tree.
    $new\hat{Y}(c) \leftarrow \hat{Y(c)} + t$ (similarly similarly on validation set, optional)
    if *tree-is-warranted(t, minTreeGain, other optional parameters)* **then**
      $T(c) \leftarrow t$
      $\hat{Y(c)} \leftarrow new\hat{Y}(c)$ (similarly on validation set, optional)
    end
  end
end

Result: w, Q, T where T is a mapping from categorical feature values to decision tree functions.

3.3 The Training Algorithm

Learning a `CENT` model corresponds to finding an approximate solution for the problem in Equation 3, where L is the specified loss function according to the distribution of the response variable y, e.g. least squares for regression and logistic loss for binary classification.

$$w^*, Q^*, T^* \leftarrow \underset{w,Q,T}{\operatorname{argmin}} L(Y, \hat{Y}) \tag{3}$$

There are many possible solutions to this problem. Since `CENT` model belongs to `GAM` [15] family, we can train the model through backfitting algorithms [15, 11], i.e. alternatively fitting one of the `CAT-E` and `CAT-NT` components fixing others until the model converges. Concerned by the complexity of this approach, we instead design an algorithm following the gradient boosting framework [12] by first fitting the `CAT-E` component until convergence and then greedily fitting trees for the `CAT-NT` components. The results show that this algorithm works and we leave designing better algorithms for `CENT` as future work, which is discussed at the end of the paper.

The detailed steps are illustrated in Algorithm 1 (multi-class classification problem may not apply here but it is straightforward to extend the algorithm for that by using different w and Q for each class). As shown there, `CAT-E` (with w and Q) is first trained by *stochastic gradient descent algorithm*, with L as the loss function, A as the input and Y as labels (see `SVDFeature` [7] for detailed steps). Then a regression tree (we used CART [3] here; future work is necessary to compare training single tree vs. an ensemble of multiple trees) for each categorical feature values is trained to fit the negative gradients of the current model. We use a heuristic here to order the fitting of the trees descendingly in the number of supporting instances. Note that the tree for $a_0 = Root$, i.e. a tree to model the interactions among only numerical features always has the biggest support and gets trained first because it is present in all instances.

The four parameters are all used to regularize the tree learning of `CAT-NT` because potentially there could be millions of possible categorical values in large scale systems. However, the cost of training each tree is substantially lower by subsetting compared with using the whole dataset and it gets cheaper and cheaper as the number of supporting instances is decreasing (which is shown in the results section). *minTreeSupport* controls whether training a tree for a categorical feature value at all. *maxTreeDepth* and *minNodeSplit* controls the maximum depth of the tree and how many instances are required in a node of the tree to be further splitted. After fitting a tree, a sub procedure **is-tree-warranted** with *minTreeGain* as a parameter is used to decide whether keeping the tree in the final model. By default we test whether the loss of the validation subset for the categorical feature decreases. Different kinds of procedures are experimented in the results section.

Note that `CAT-E` and `CAT-NT` both can be separately trained without much modification to Algorithm 1. To get `CAT-NT` without `CAT-E`, the only difference is to set the initial predictions from `CAT-E` $\hat{Y}$ to be zeros.

4. EXPERIMENTS AND RESULTS

To show the benefit of our `GB-CENT` model, we conducted experiments performing the tasks of *rating prediction* and *binary classification* on two real-world data sets. Following are the baseline models we compared with.

- `GBDT-OH`: One-Hot encode a and feed them into `GBDT` together with b.

Table 1: MovieLens and RedHat data sets and their corresponding feature design.

Data Set	MovieLens	RedHat
Statistics for Original Data Set	#users: 240,000 #items: 33,000 #instances: 22,000,000	#customers: 151,295 #activity categories: 7 #instances: 2,197,291
Number of Runs	20	5
Statistics for Each Run	#users: $\sim$ 12,000 #items: $\sim$ 14,000 #instances: $\sim$ 1,100,000	#customers: $\sim$145239 #activity categories: 7 #instances: $\sim$1757833
CAT-E Features	a: *userId, itemId, genre, language, country, grade* $U(a)$: *userId* $I(a)$: *itemId, genre, language country, grade*	a: *people_id, activity_category* $U(a)$: *people_id* $I(a)$: *activity_category*
CAT-NT Features	b: *year, runTime, imdbVotes, imdbRating, metaScore*	b: *char_38*
GBDT-OH	x, i.e. both a and b	x, i.e. both a and b
GBDT-CE	Similar to GBDT-OH except that a is first embedded into latent space based on CAT-E	Similar to GBDT-OH except that a is first embedded into latent space based on CAT-E
SVDFeature-S	user group: *userId* item group: a except *userId* and sigmiod transformed b	user group: $U(a)$ and sigmoid transformed b item group: $I(a)$
SVDFeature-D	Similar to SVDFeature-S except that b is discretized.	Similar to SVDFeature-S except that b is discretized.
FM-S	x, i.e. both a and sigmoid transformed b	x, i.e. both a and sigmoid transformed b
FM-D	Similar to FM-S except that b is discretized.	Similar to FM-S except that b is discretized.
Label/Response	real response: *rating, 1-5 stars*	binary response: *outcome, yes or no*

- `GBDT-CE` (state-of-the-art ensemble of `GBDT` and matrix factorization): First embed a into latent space based on `CAT-E`. Specifically, let d be the latent space dimension, then d new numerical features are generated for user group and d new numerical features are generated for item group, by taking the sum of each individual categorical feature's latent embedding vector. They are fed into `GBDT` together with b.
- `SVDFeature-S`: Sigmoid transform b and feed them into `SVDFeature` together with a. Specifically, we use the following sigmoid function to transform numerical features (Equation 4). This linear transformation function makes sure the scale of output is within $(-1, 1)$ while better retaining the difference of input values and hence is sometimes better suitable for feature transformation, compared with the nonlinear logistic sigmoid.
- `SVDFeature-D` (state-of-the-art ensemble of matrix factorization with decision trees): discretize b based on a single decision tree model and feed them into `SVDFeature` together with a. Specifically, we built a single decision tree with only b as input on the training set (maxTreeDepth=10, minNodeSplit=2000). Then its predicted leaf node is used as the discretized categorical feature. In most cases, it gives a full tree, i.e. around 1024 possible nodes. This approach of discretization takes the distribution of response variables into account when varying the feature values. It can help linear models learn nonlinearity by nonlinearly mapping numerical features into buckets. It is similar to treating decision trees as feature extractors of large sparse linear models as illustrated in [16].
- `FM-S`: Similarly sigmoid transform b using Equation 4 and feed them into `FM` together with a.
- `FM-D` (state-of-the-art ensemble of matrix factorization with decision trees): Similarly discretize b as `SVDFeature-D` and feed it into FM together with a.

$$f(x) = \frac{x}{1 + |x|} \tag{4}$$

4.1 Data Sets, Metrics and Software

Table 1 summarizes the data sets we used and their corresponding feature design. The first one is MovieLens latest ratings data set [14][2]. To have more abundant meta data on movies, especially numerical metrics, we requested IMDB API [3] for more movie statistics. This is enabled because there is a matching file for MovieLens movie IDs and IMDB IDs provided in the latest MovieLens data set. Instead of running evaluation once for each **original data set** using all users' data, we bootstrap the data sets by randomly sampling users without replacement to have more robust evaluation which essentially divides each original dataset into multiple **subsampled data sets** of nonoverlapping users. For each subsampled data set, we first sort it temporally and

[2]http://grouplens.org/datasets/movielens/latest/
[3]http://www.omdbapi.com/

Table 2: The mean metrics (with standard deviation in the parentheses) of different models on MovieLens data set with 20 runs and RedHat data set with 5 runs. Time(s) shows the total training time of each model in seconds. The plus and minus for RMSE and AUC indicate accuracy improvement or degradation compared with our model GB-CENT. For Time(s), they mean more or less training time compared with GB-CENT.

Data Set	Metric	GBDT-OH	GBDT-CE	SVDFeature-S	SVDFeature-D	FM-S	FM-D	CAT-E	CAT-NT	GB-CENT
MovieLens	RMSE	0.883 (0.007) -%1.8	0.863 (0.006) +%0.4	0.877 (0.009) -%1.1	0.867 (0.006) +%0.0	0.913 (0.024) -%5.3	0.888 (0.005) -%2.4	0.886 (0.011) -%2.1	0.900 (0.006) -%3.8	0.867 (0.006)
	Time (s)	282 +1.08	1034 +6.65	68 -%49.6	66 -%51.1	73 -%45.9	60 -$55.5	77 -%42.9	54 -%60.0	135
RedHat	AUC	0.955 (0.0005) -%3.6	0.981 (0.0003) -%1.0	0.975 (0.0002) -%1.6	0.976 (0.0003) -%1.5	0.986 (0.0009) -%0.5	0.987 (0.0003) -%0.4	0.967 (0.0002) -%2.4	0.942 (0.0006) -%4.9	0.991 (0.00006)
	Time (s)	857 +%35.8	3140 +3.97	130 -%79.3	241 -%61.8	204 -%67.6	181 -%71.3	561 -%11.0	98 -%84.4	631

further split it into *80% training*, *10% validation* and *10% testing* sets for each user, i.e. we always use user history data to predict user future behavior. The second one is RedHat data set which was used for Kaggle competition[4]. The task is to identify which customers who performed certain activities have the most potential business value for Red Hat. For this data set, we randomly pick 10% of the instances as validation set, another different 10% as the testing set, the other 80% as training set. This process was repeated five times (i.e. five folds). We call the process of training a model on the training set (using the validation set to prevent overfitting during training) and then evaluating the model on the testing set **one run of the evaluation procedure**.

For *rating prediction* task on MovieLens data set, we use Root Mean Squared Error (RMSE) as the metric. For *binary classification* task on RedHat data set, we use Area Under the Curve (AUC) as the metric.

We use open source implementation *xgboost* [4] through Java API to operationalize the GBDT-OH and GBDT-CE models in our experiments. The default parameter setting is used except that *tree_method* [12] is set to be *exact splitting* for all experiments, i.e. although *xgboost* supports approximate splitting to accelerate training, we always use exact splitting. In each run, 1000 trees/rounds are fitted and take the model from the best iteration. All the other models are implemented by the authors in Java [5]. We confirmed the performance of the implementation is consistent with the SVDFeature and FM implementations in [7] and [24]. All factorization models are fitted on training set with 50 maximum number of iterations. The dimension of latent factors d is 20. Learning rate is 0.001 for MovieLens data set and 0.01 for RedHat data set. We did not use regularization on w and Q given we already use validation set to avoid overfitting. For GB-CENT model, we set $minTreeSupport = 50$, $minTreeGain = 0.0$, $minNodeSplit = 50$ and $maxTreeDepth = 3$. The experiments are run on a single machine with 8 available cores (Intel i7-4790 CPU @ 3.60GHz) and 32 GB memory.

[4]https://www.kaggle.com/c/predicting-red-hat-business-value. The competition was complete on August 2016, but the data set is still publicly available.

[5]Open sourced in Github together with a generic recommender and predictor server called Samantha developed by the first author in GroupLens research lab: https://github.com/grouplens/samantha

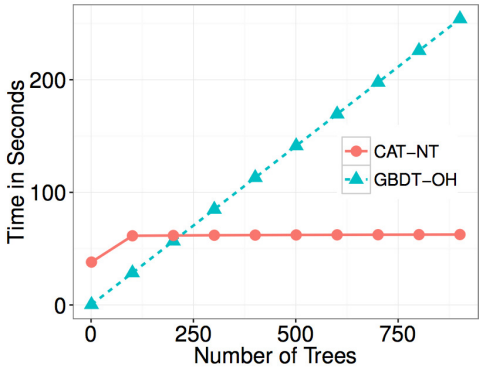

Figure 1: The time taken as more trees are boosted in GBDT-OH and CAT-NT on MovieLens data set in one run of the evaluation. xgboost is using 8 available cores on the tested machine.

4.2 Results

Accuracy. As shown in Table 2, GB-CENT achieves the best accuracy on both data sets. Note that on MovieLens data set, GBDT-CE is not significantly different from GB-CENT. SVDFeature-D achieves the same accuracy on MovieLens data set compared with GB-CENT, but it performs %1.5 worse on RedHat data set. This suggests that GB-CENT's accuracy generalizes better across tasks and data sets. It stably achieves better accuracy than other models. Note that the best AUC achieved in the Kaggle competition on RedHat data set is 0.995[6]. GB-CENT can achieve 0.996 when using all available features (although it is evaluated on a different subset as designed in Table 1). Our experiments in this paper used three features, two of which are categorical and the other one is numerical and achieve AUC score 0.991.

Training Time. Training GB-CENT is faster than both GBDT-OH and GBDT-CE. GBDT-OH takes 1.08 times longer to train than GB-CENT and GBDT-CE takes 6.65 times longer. Factorization models are faster than GB-CENT, but their accuracy generally suffers compared with GB-CENT. CAT-NT is much faster to train than GBDT-OH because each numerical tree in CAT-NT only involves its supporting instances as mentioned above. From Figure 1, in learning the first several trees, CAT-NT takes more time because of the computation

[6]https://www.kaggle.com/c/predicting-red-hat-business-value/leaderboard/private

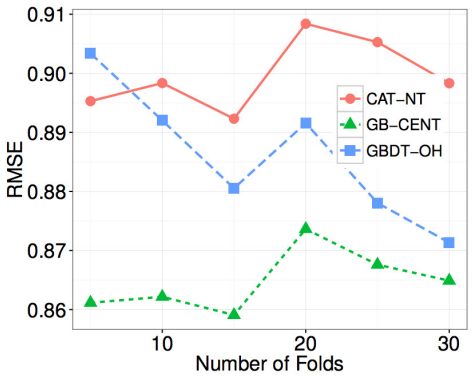

Figure 2: The testing RMSEs for GBDT-OH, CAT-NT and GB-CENT on MovieLens data set as more users' data are used, i.e. when categorical features' cardinality increases. Going from right to left, as the number of folds decreases, more users' data are included in one run.

and sorting of the number of supporting instances of categorical features (i.e. $S(c)$ in Algorithm 1). As more trees are introduced, CAT-NT takes less and less time and dramatically reduce the overall time compared with GBDT-OH. The ratio of learning the same number of trees (1000 here) is about 1 vs. 4.

Ensemble. Discretizing numerical features based on a decision tree works better than sigmoid transformation. Because of the linear limitation of factorization models, decision tree based discretization or feature extraction helps it capture the nonlinearity in the data set. Embedding categorical features into low-dimensional space, i.e. GBDT-CE is more accurate than using raw categorical features in GBDT-OH. However, it incurs substantial cost. The training time is much longer. GB-CENT behaves consistently with our design and hypotheses: it effectively (i.e. fast and accurately) handles both low-dimensional embedding and high-order nonlinearity of the data set.

Cardinality. Results in Figure 2 are also consistent with our hypothesis as mentioned in the introduction. We use the number of users as a proxy of the cardinality of categorical features because it increases when more users' data are used. It also reflects the scenario when a real application system has more and more active users. The figure shows that when the cardinality of the data increases, the gap between the GBDT-OH and CAT-NT has a shrinking trend while the gap between GBDT-OH and GB-CENT has a expanding trend. With large enough cardinality, CAT-NT itself can out-perform GBDT-OH as illustrated by the cross of the two lines in Figure 2. This demonstrates that CAT-NT in GB-CENT dynamically exploits the growing cardinality of the data set to achieve better and better accuracy.

4.3 Tree Regularization

An important part in training CAT-NT component of GB-CENT is to regularize the tree learning, which has two aspects: the size of the individual tree and whether a tree is warranted to keep in the final model. Parameter *maxTreeDepth* controls the first aspect and *minTreeSupport* controls the second, i.e. whether learning a tree at all for a categorical feature value. Table 3 illustrates how RMSE varies with different values of the two parameters on MovieLens data set. They tell that generally lower values for both parameters seem to be better. In other words, the size of an individual tree should be small (similar to the default setting of GBDT-OH) and the threshold to learn a tree should be low. We tested two approaches for the procedure *is-tree-warranted* as follows given that a tree has been learned.

Table 3: The effect of minTreeSupport and maxTreeDepth on MovieLens data set. minTreeSupport is held to be 50 when varying maxTreeDepth; maxTreeDepth is held to be 3 when varying minTreeSupport.

minTree-Support	RMSE	maxTree-Depth	RMSE
10	0.902	2	0.901
50	0.906	3	0.906
100	0.917	5	0.918
200	0.925	8	0.924
300	0.936	10	0.929
400	0.943	15	0.950

Table 4: The effect of tree regularization on MovieLens data set. minTreeSupport=50, maxTreeDepth=3.

Regulariza-tion	minTree-Gain	Number of Accepted Trees	RMSE
AAT	N.A.	7926	0.905
VSLR	0	7606	0.906
	1	7559	0.913
	3	7441	0.921
	5	6737	0.928
	8	6375	0.945

- **Validation Subset Loss Reduction (VSLR)**, which tests whether there is reduction in the average loss of the predictions on the subset of the validation set that the tree's categorical feature value (c) corresponds to. We set $minTreeGain = 0$ by default. Since this approach only tests a subset of the validation set, it is relatively cheaper than testing on the whole validation set.
- **Accept All Trees (AAT)**, i.e. all learned trees are accepted to add into the final model.

As shown in Table 4, it demonstrates a trend that learning more trees generally gives better accuracy. However, we think these parameters need to be tuned for specific data sets at hand. For example, although the table shows that smaller *minTreeGain* threshold is better, it is possible fitting trees for categorical features with few supporting instances might over-fit.

5. DISCUSSION

With above results, we demonstrate that GB-CENT model have advantages over SVDFeature, FM, GBDT and their ensembles. As motivated in the introduction section by the difference between numerical and categorical features, we think that they are fundamentally different from both algorithmic

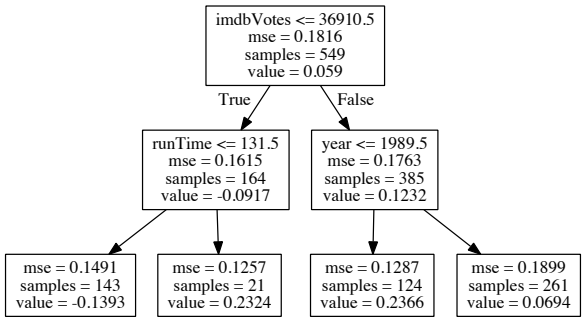

Figure 3: An example tree learned by CAT-NT for a specific user.

and application perspectives. Different from numerical features, categorical features have two faces that a model needs to capture in order to have better predictive performance: *low-dimensional embedding*, which captures *generalizability, latent grouping or similarity* of the entity that the categorical feature represents, and *high-order nonlinear interaction* in numerical metrics, which captures *specificity or peculiarity* of the same entity. GB-CENT captures both and hence is well suitable for applications with these properties such as recommender systems.

GB-CENT model also has another benefit: *interpretability*. This is not only true for CAT-E component which explicitly describes the match between the latent profiles of users and items after being embedded into a low-dimensional space. It is also true for CAT-NT. Figure 3 plotting a numerical tree for a user ID explicitly describes the user's particular preference on movies in a high-order and nonlinear way. It seems that this user prefers movies that are popular (with large *imdbVotes*) but released before 1989 or movies that are not very popular but with long run time (i.e. greater than 131 minutes). CAT-NT can also model the increasing level of specificity for categorical features with hierarchies such as geolocations of country, state and city. One can imagine that during the learning process, CAT-NT first learns a tree for a country and further boosts another tree for a state if the state has different enough regularity from its country and similarly for cities in the state. For GBDT-OH on the other hand, the trees learned are less interpretable mixed with both categorical (especially IDs) and numerical features because the splitting points mostly test whether it is a specific ID or not, implicitly doing grouping of IDs, which has been handled by CAT-E in GB-CENT.

6. FUTURE WORK

Previous literature [22] has shown that acuracy alone is not enough to evaluate a recommendation algorithm. We consider it necessary future work to comprehensively evaluate the recommendations made by GB-CENT model and especially conduct online field experiments to gain a better understanding on how users perceive these recommendations. An interesting question is whether GB-CENT can achieve a good balance between novelty [29] and accuracy because of the generalizability of CAT-E component and the specificity of CAT-NT component.

Another important future work we believe that needs to be done is to scale up the training algorithm of the model, especially CAT-NT. In real-world applications, there could be possibly millions of categorical feature values, which corresponds to learning millions of trees (note that predicting is cheap because only accessing trees for the sparse categorical values in the instance is necessary). Therefore, distributed learning algorithm is demanded. However, we consider that CAT-NT is friendly to parallelism because the trees are learned on millions of subsets of the original data set, which can be naturally fit into Map-Reduce framework [9]. Although gradient boosting is essentially a sequential process with dependencies among trees, it might be possible to break the dependency by grouping categorical feature values and train parallelly between groups and sequentially within groups without much loss of performance. What's more, GB-CENT has the potential to enable online learning of decision trees because each CAT-NT only relies sparse supporting instances.

7. CONCLUSION

We propose a predictive model GB-CENT with both low-dimensional embedding and decision tree components. The first component CAT-E (Categorical Embedding) embeds categorical features into a low-dimensional space and the second component CAT-NT (Categorical Numerical Trees) learns a numerical tree for each categorical feature value with enough supporting instances in the data set. GB-CENT performs significantly better than the state-of-the-art matrix factorization and GBDT models. It also outperforms the feature level ensemble of the two types of models. With these results, we demonstrate that we can be better off to differentiate modeling categorical features from numerical features. GB-CENT is a model specially designed for this with nice interpretability.

Particularly, we show the advantages of CAT-NT component in GB-CENT model over GBDT-OH. It is much less expensive to learn (time ratio for the same number of trees: 1 vs. 4) and achieves increasingly better accuracy as the cardinality of the categorical features in the data set becomes larger. Similarly, the accuracy gap between GBDT-OH and GB-CENT is expanding with the increasing cardinality.

8. ACKNOWLEDGEMENT

We thank Yahoo Research for its support on this work with an internship. We also thank GroupLens research for its continuing support. We thank Ting Chen, Yue Ning and Qingyun Wu for their helpful discussions.

9. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28. ACM, 2009.

[2] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[4] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

[5] T. Chen, H. Li, Q. Yang, and Y. Yu. General functional matrix factorization using gradient boosting. In *ICML (1)*, pages 436–444, 2013.

[6] T. Chen, L. Tang, Q. Liu, D. Yang, S. Xie, X. Cao, C. Wu, E. Yao, Z. Liu, Z. Jiang, et al. Combining factorization model and additive forest for collaborative followee recommendation. *KDD CUP*, 2012.

[7] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research*, 13(Dec):3619–3622, 2012.

[8] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. *arXiv preprint arXiv:1606.07792*, 2016.

[9] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[10] W. Fan, H. Wang, P. S. Yu, and S. Ma. Is random model better? on its accuracy and efficiency. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 51–58. IEEE, 2003.

[11] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[12] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[13] S. Funk. Netflix update: Try this at home, 2006.

[14] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2016.

[15] T. J. Hastie and R. J. Tibshirani. *Generalized additive models*, volume 43. CRC Press, 1990.

[16] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9. ACM, 2014.

[17] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[18] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[19] R. Karimi, M. Wistuba, A. Nanopoulos, and L. Schmidt-Thieme. Factorized decision trees for active learning in recommender systems. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 404–411. IEEE, 2013.

[20] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[21] Y. Koren, R. Bell, C. Volinsky, et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[22] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 extended abstracts on Human factors in computing systems*, pages 1097–1101. ACM, 2006.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[24] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[25] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658. ACM, 2008.

[26] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: a general framework for dimensionality reduction. *IEEE transactions on pattern analysis and machine intelligence*, 29(1):40–51, 2007.

[27] E. Zhong, W. Fan, and Q. Yang. Contextual collaborative filtering via hierarchical matrix factorization. In *SDM*, volume 12, pages 744–755. SIAM, 2012.

[28] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 315–324. ACM, 2011.

[29] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.