Ensemble of Collaborative Filtering and Feature Engineered Models for Click Through Rate Prediction

Michael Jahrer, Andreas Töscher
Opera Solutions
8580 Köflach, Austria
michael.jahrer@operasolutions.com
andreas.toescher@operasolutions.com

Jeong-Yoon Lee, Jingjing(Bruce) Deng,
Hang Zhang, Jacob Spoelstra
Opera Solutions
San Diego, CA 92130, USA
{jlee,bdeng}@operasolutions.com
{hzang, jspoelstra}@operasolutions.com

ABSTRACT

The challenge for Track 2 of the KDD Cup 2012 competition was to predict the click-through rate (CTR) of web advertisements given information about the ad, the query and the user. Our solution comprised an ensemble of models, combined using an artificial neural network. We built collaborative filters, probability models, and feature engineered models to predict CTRs. In addition, we developed a few models which directly optimized AUC, including the collaborative filters and ANN models. These models were then blended using AUC optimized ANN such that the final output of the system had significantly improved performance over the constituent models on test data. We achieved an AUC score of 0.80824 on the private leaderboard and finished second in the competition.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-Data Mining

General Terms

Application

Keywords

Collaborative Filtering, Ensemble Learning, KDD Cup, AUC Optimization, Ranking, Search Engine

1. INTRODUCTION

Internet advertising has become a major source of revenue for web-based businesses. The online advertising revenue in the first half of 2009 reached over 10.9 billion US dollars [1]. Accurate predictive modeling of the click-through rate (CTR) of advertisements has a significant impact on the Internet industry. It allows Internet companies to identify most relevant ads for each user, to estimate the value of each click, and to improve user experiences.

The task of Track 2 of the 2012 KDD Cup competition was to predict CTRs of ads displayed alongside results of online search by users. Tencent, one of the Internet giants in China, sponsored this task and provided one of the largest publicly accessible datasets. Performance in the competition was measured by Area Under Curve (AUC) on a private test set, which was a (fixed) random subset of the whole test set. The AUC on the remainder of the test set, called the public test set, was used for the rankings on the public leaderboard.

The training data for Track 2 consisted of 149,639,105 records of user queries, a total of 10 Gb of data. Multiple queries with the same properties and their outputs were rolled up into one record in the training data. Expanded to individual queries, this produced 235 million training records.

Each query and its output were described by 10 variables as follows:

- `adUrlID` - This is a property of the ad. The URL was shown together with the title and description of an ad. It was usually the shortened landing page URL of the ad, but not always.
- `adID` - The unique ID of an advertisement.
- `advertiserID` - The unique advertiser ID of the `adID`. It is a property of the ad. Some advertisers might produce more attractive advertisements than others.
- `depth` - The number of ads displayed to the user in a query session. The maximum depth is 3.
- `position` - The position of the `adID` in a query session. The maximum position is 3.
- `queryID` - The unique ID of the query.
- `keywordID` - A number representing a keyword used in the ad.
- `titleID` - A number representing the ad title.
- `descriptionID` - A number representing a description of the ad.
- `userID` - The unique ID of a user who conducts the query.

Table 1: Overlapping between unique IDs in training and test data.

Field	Train	Test	Test new IDs (%)
adUrlID	26,273	17,692	4.2
adID	641,707	300,012	9.6
advID	14,847	11,157	3.6
queryID	24,122,076	3,801,978	55.8
kwdID	1,188,089	495,421	12.4
titleID	3,735,796	1,262,498	25
dscptID	2,934,101	981,985	24.2
userID	22,023,547	3,263,681	57.7

The IDs above were all hash-mapped to integers. Each of the `queryID`, `keywordID`, `titleID`, and `descriptionID` was associated with a set of keywords, which were also hash-mapped to integers and provided in four extra data files, to provide detailed description of the query and advertisement, respectively. If a user could not be identified, 0 was assigned to the `userID` (30% of data). In addition, The `gender` (Male=1, Female=2, and Unknown=0) and `age` ((0,12]=1, (12,18]=2, (18, 24]=3, (24,30]=4, (30,40]=5, and (40+,]=6) of each `userID` were provided in another data file.

Two variables described the response of a user to an advertisement: the number of clicks and the number of impressions. For each user, query and the resulted ad, the number of impressions indicated the times that the ad was displayed to the user. The number of clicks was the times that the user clicked the ad. Consequently, the CTR was the ratio between the numbers of clicks and impressions. The average CTR of the training data was 0.0387.

The test data conprised 20,297,594 records similar to the training data. The columns representing the number of clicks and the number of impressions were withheld.

A full description of the dataset is available at [9].

One challenge was that not all entities in the test set were present in the training data. Table 1 showed the overlap between the unique IDs in training and test data. This provided guidance for our model selection. As shown in Table 1, the `advertiserID` has the lowest percentage of new IDs in test data (3.6%). More than 55% of the `queryID` and `userID` in test data were new IDs.

There has been a number of recent reports on the task of predicting CTRs of online content. There are two categories of models for CTR prediction: models on raw historical data using collaborative filters and probability models, and models using features extracted from raw data. Menon et al. [8] built collaborative filters to predict CTRs of ads. Dembczynski et al. [2] proposed a maximum likelihood estimation (MLE) technique to learn the parameters of a probability model. However, this kind of models are suitable only for existing ads. To predict CTRs of new ads, a method based on decision rules using features of ads was proposed.

Feature engineered models include the models by Gupta [4] which predicted CTRs of job lists. Zhang et al. [14] used artificial neural networks (ANN) to identify significant factors that have impacts on the CTR, and reported that the CTR is significantly sensitive to factors including the total rank of ads that have been clicked by a user, length of the query, type of browser, and popularity of ads. Wang and Chen [13] extracted features from raw data and built advanced statistical models (support vector machines (SVM), ANN and decision trees) on the features. Richardson et al. [11] extracted features, such as the length of the ad and the words it uses, to build models predicting CTRs of new ads.

As described in this paper, we built collaborative filters, probability models, and feature engineered models to predict CTRs of online search advertisements. We developed a few models which directly optimized AUC, including the collaborative filters and ANN models. These models were then blended using AUC optimized ANN such that the final output of the predictive system had significantly improved performance over the constituent models on test data.

The remainder of this paper was organized as follows. First, we introduced how we split the training data into training and validation sets. We then described the models that we built on raw data (collaborative filters and probability models) and on features extracted from raw data. Third, a blending algorithm and its performance was reported. The last section concludes this paper.

2. TRAINING DATA SELECTION

To train, validate, and blend models, we randomly split the training data into three sets: `ValidTrain` (97%), `Valid1` (1.5%), and `Valid2` (1.5%), denoted as Ω_{VT}, Ω_{V1}, and Ω_{V2}, respectively. We also denote the original training and test data as Ω_{Trn} and Ω_{Tst}, respectively. Figure 1 shows the data split.

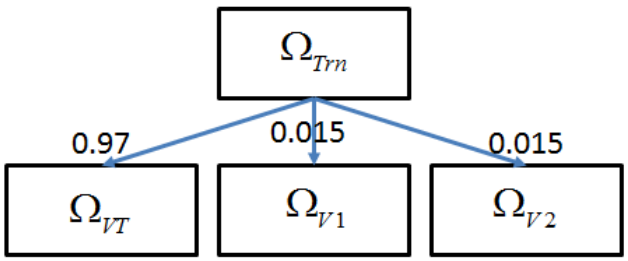

Figure 1: Splitting train data into Ω_{VT}, Ω_{V1}, and Ω_{V2}

In general, single models were trained on Ω_{VT}, and validated using Ω_{V1}. For single models, the optimal parameters that maximized AUC scores on Ω_{V1} were chosen. Blending models were trained using predictions of single models on Ω_{V1}, and validated with predictions of single models on Ω_{V2}. For blending models, parameters were optimized to maximize AUC scores on Ω_{V2}. Finally, the optimized blending models were used to make predictions for Ω_{Tst}.

The data split approach was evaluated by a simple bias model comprising the global mean and 5 bias terms indexed by `userID`, `adID`, `advertiserID`, `position`, and `adUrlID`.

Table 2: AUC of a simple bias model on Ω_{V1}, Ω_{V2}, and Ω_{Tst}.

Ω_{V1}	Ω_{V2}	Ω_{Tst}
0.77798	0.77697	0.74731

Our criteria for a good split was that model performance on Ω_{V1}, Ω_{V2}, and Ω_{Tst} be consistent, although the performance on Ω_{Tst} is expected to be slightly worse than on Ω_{V1} and Ω_{V2}. Table 2 shows that these criteria were met. This split was used in all models we built.

Ideally, to maximize use of the training data, we would have retrained the model on Ω_{VT}+Ω_{V1} to predict for Ω_{V2}, and then retrained the model on Ω_{Trn} to predict for Ω_{Tst}, using the optimized model parameters. However, with hundreds of millions of observations in the training set it was too time-consuming to retrain the models . On the other hand, since Ω_{V1}+Ω_{V2} was ony 3% of Ω_{Trn}, we assumed that the difference among models trained on Ω_{VT}, Ω_{VT}+Ω_{V1}, and Ω_{Trn} would be negligible.

3. ALGORITHMS

As introduced in Section 1, two categories of models existed for CTR prediction: collaborative filters and probability models built on raw data, and feature engineered models. In this section, we described these two types of models that we developed during this competition.

3.1 Models on Raw Data

Factor models reveal their full potential when trained on huge amounts of raw data. The dataset from Tencent contained about 150 million records, but they were summarized at the impression level. The models below benefited from unrolling the impressions in the dataset to around 235 million individual event records. The bias and factor models were trained by pairwise stochastic gradient descent (SGD) on raw unrolled data (see 2.2.2 in [6]). η was the learning rate, λ was the L2-regularization. The learning rate η was multiplied after every epoch by 0.85.

3.1.1 Bias Model

This model learned biases b_i^m for each unique ID in the dataset. Per training instance i there were M=10 different ID sources (see Section 1). d_i^m was the value of the m-th ID from sample i. $m \in [1, M]$. For example, the value of d_1^1=7686695 (i=1 was the first sample in training set and m=1 was the `adID`). The model has biases b_j^m for each ID source ($m = 1...M$) and within each source for each unique ID ($j = 1...|d_*^m|$). The prediction score of the i-th sample was a sum of all biases:

$$\hat{r}_i = \sum_{m=1}^{M} b^m_{(d_i^m)} \quad (1)$$

Despite the fact that the bias model learned no interaction between IDs, it performed quite well. It had $AUC = 0.76461$ on public leaderboard when a uniform learning rate ($\eta = 0.01$) and regularization parameter of $\lambda = 0.02$ were used.

Table 3: ID-dependent learning rates and regularization values found by optimizing the bias model.

ID NAME	η	λ
ADURLID	0.000013	0.01
ADID	0.0001	0.0135
ADVERTISERID	0.0001	0.0379
DEPTH	0.000013	0.0379
POSITION	0.009	0.002
QUERYID	0.0025	0.0379
KEYWORDID	0.0001	0.002
TITLEID	0.0001	0.0135
DESCRIPTIONID	0.0001	0.137
USERID	0.0025	0.0075

Table 4: Models on raw upsampled data.

MODEL	Ω_{V1}	Ω_{V2}	Ω_{Tst}
BIAS MODEL	0.82719	0.82754	0.788
FACTOR MODEL	0.82989	0.829394	0.7913
AFM MODEL	0.7673	0.7698	0.740
BAYESIAN MODEL	0.8430	0.8460	0.752
GBM	0.7860	0.7836	0.757
SVM^{perf}	0.7961	0.7924	0.764
ANN	0.8012	0.8251	0.765

The bias model benefited greatly from an ID-specific learning rate and L2-regularization values. We found the values in Table 3 by coordinate search. The optimized bias model had leaderboard $AUC = 0.78784$.

3.1.2 Factor Model

The factor model learned pairwise interactions for each of the 10 ID sources. Figure 2 depicts the interaction pairs of the model. Within every cell of the matrix in Figure 2, the model had its own set of features for the factorization.

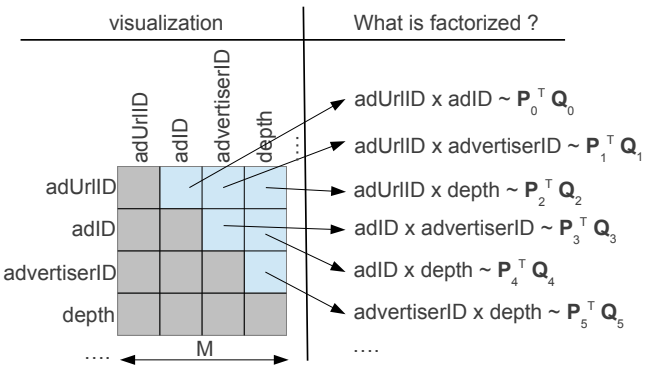

Figure 2: Factors in the factor model. Every cell had its own factor matrices.

The final prediction was the sum of all pairwise feature dot products. With a total of 10 factors, we had 45 dot products in the sum. We used only two features per ID ($F = 2$).

$$\hat{r}_i = \sum_{m=1}^{M} \sum_{n=m+1}^{M} \overbrace{\mathbf{p}^{c}_{(d_i^m)}}^{vector1}{}^{T} \overbrace{\mathbf{q}^{c}_{(d_i^n)}}^{vector2}, c = unique\ per\ cell \quad (2)$$

Training was done by pairwise SGD using parameters $\eta = 0.014$ and $\lambda = 0.013$. The factor model achieved a leaderboard $AUC = 0.79137$. We observed problems with overfitting even when the regularization was increased. In order to combat overfitting, all factor models were trained for only a single SGD epoch through the dataset.

3.1.3 AFM - Asymmetric Factor Model

The idea of the AFM (cp. NSVD in [10]) on this dataset was that a user could be described by the set of ads in the training set which she had interacted with $N(u)$. The advantage of this approach was that we could also include information from the test set samples in training and different from the factor models, the AFM is capable of predicting for new IDs . For training we used $N(u)$=train+valid, and for re-training $N(u)$=train+valid+test. We pre-calculated a user-to-sample list, which simplified training.

The prediction of a sample i was

$$\begin{aligned} \hat{r}_i &= \overbrace{\mathbf{p}^{T}}^{\text{"item feature"}} \cdot \overbrace{\mathbf{q}}^{\text{"user feature"}} \\ \mathbf{p} &= \{sum\ of\ 7\ features\ of\ sample\ i\} \\ \mathbf{q} &= \mathbf{q}_u + \sum_{j \in N(u)} \sum_{k=1}^{7} \{sum\ of\ 7\ features\ of\ sample\ j\} \end{aligned} \quad (3)$$

Basically, the prediction $\hat{r}_i$ was a dot product of an item by a user feature. The item feature vector $\mathbf{p}$ of sample i was composed from a sum of 7 features. The 7 features were: `adUrlID`, `adID`, `advertiserID`, `queryID`, `keyWordID`, `titleID`, `descriptionID`. We rejected here the `depth` and `position` features. The user feature vector $\mathbf{q}$ of sample i was the user feature $\mathbf{q}_u$ itself and a sum of the 7 item features from the samples of the user ($N(u)$ was an integer set). The sum (second term) of $\mathbf{q}$ was normalized by 1 divided by the square root of the number of elements in the sum (see [10]). Training was done with SGD to minimize pairwise rankings, as was done in the bias and factor models. The Leaderboard AUC for this model was around 0.74, which was a weak model itself, but it had some small contribution in our final blend.

3.1.4 Bayesian Model

We applied Bayesian rules to predict the probability that a user clicks an ad, given a list of features. This probability was just the posterior probability described as follows:

$$\begin{aligned} P(Y=1|f_1,...,f_n) &= \frac{P(f_1,...,f_n,Y=1)}{P(f_1,f_2,...,f_n)} \\ &= \frac{\prod_{i=0}^{n} P(f_i|Y=1)P(Y=1)}{\prod_{i=0}^{n} P(f_i|Y=1)P(Y=1) + \prod_{i=0}^{n} P(f_i|Y=0)P(Y=0)} \end{aligned} \quad (4)$$

where Y was a response event (1: clicked, 0: not clicked). The features f_i's were the unique IDs in the raw data. The prior probability $P(Y=1)$ and $P(Y=0)$ could be estimated from the training data.

The conditional probability $P(f_i|Y=1)$ could be estimated by just counting the training data records for the proportion of cases when $Y=1$ that feature f_i was also involved, i.e.,

$$P(f_i|Y=1) = \frac{\sum_{i=1}^{n} I(f_i, Y=1) + N_1}{\sum_{i=1}^{n} I(Y=1) + N_2} \quad (5)$$

where N_1 and N_2 were set to be 0.38 and 10 respectively to mimic the average CTR (0.038) in the whole training set. We tested different combinations of IDs as the features in Eqn. 4. When all IDs were used, we achieved the optimal performance of 0.7518.

There was a strong assumption in Equation 4 that the probabilities of each ID conditioning on Y are independent. This is done to simplify the computation, even though we know it not to be the case (hence the *naive* in the name!). Further work could be done to derive a better estimate of the joint conditional probability of all the n IDs in order to improve the performance of this model.

3.2 Feature Engineering

We extracted features as derived variables from the original data so that a big variety of advanced statistical models, including Gradient Boosted Machines (GBMs), ANNs, and Support Vector Machines (SVMs), could be built using these features. There were two sets of features extracted: 24 risk values of each unique ID, its related keywords and some 2-D combinations, and features based on the similarities between a query and an advertisement.

3.2.1 Risk Features

The risk value of each unique ID was the conditional probability that when the ID was present in a record, the search advertisement would be clicked. Mathematically, the risk value of ID_i was defined as:

$$Pr(Y=1|ID_i) = \frac{\sum_{j=1}^{n} (c_j + N_1) \times I(ID_i \in R_j)}{\sum_{j=1}^{n} (n_j + N_2) \times I(ID_i \in R_j)} \quad (6)$$

where $Y=1$ was the event that the advertisement was clicked, n was the total number of records, $I(\cdot)$ was an identity function, R_j was the j^{th} record, and N_1 and N_2 were two smoothing parameters. We chose $N_1 = 0.38$ and $N_2 = 10$ as in 3.1.4. We had 8 1-D risk values for `adUrlID`, `adID`, `advertiserID`, `depth`, `position`, `userID`, `gender`, and `age`.

Since each `queryID`, `keywordID`, `titleID`, and `descriptionID` was also associated with a set of keywords, we calculated the risk values for each keyword. Then, for each record in Ω_{V1}, Ω_{V2}, and Ω_{Tst}, we used two variables to represent the risks of the keyword set of `queryID`, `keywordID`, `titleID`, and `descriptionID`, respectively. These two variables were the average and maximum risk values of the associated keywords. In total, we had 8 risk variables for keywords.

Based on an assumption that different genders and ages might have different preferences on `adID`, `advertiserID`, `depth`, and `position`, we also calculated the 2-D risks for `adID-gender`, `adID-age`, `advertiserID-gender`, `advertiserID-age`, `depth-gender`, `depth-age`, `position-gender`, and `position-age`.

3.2.2 Similarity Features

Assuming that the underlying driver of the click action from a user was the similarity between the query and the advertisements, we calculated the similarity between the keywords associated with the `queryID`, and the keywords associated with the `keywordID`, `titleID`, and `descriptionID`, respectively. We used four statistics to measure the similarity between the keywords of ID_1 and ID_2.

- The proportion of the ID_1 keywords that are covered by the keywords associated with ID_2.
- The proportion of the ID_1 2-keyword phrases that are covered by the 2-keyword phrases of ID_2.
- If there exist common ID_1 and ID_2 keywords, their earliest position in ID_2 keywords. Otherwise, assign value 99.
- If there exist common ID_1 and ID_2 2-keyword phrases, their earliest position in ID_2 2-keyword phrases. Otherwise, assign value 99.

Here, ID_1 was the `queryID`, and ID_2 was `keywordID`, `titleID`, and `descriptionID`, respectively. A 2-keyword phrase referred to any two consecutive keywords in the keyword set of an ID, in the same order as they are in the keyword set. In total we created 12 variables to describe the similarities between queries and the ads.

3.3 Models on Engineered Features

We then built GBM, ANN, and SVM models on the feature matrix of records in Ω_{V1}, and optimized the model parameters based on their performance on the feature matrix of records in Ω_{V2}. The optimized models were then used to make predictions for Ω_{Tst}. Table 4 shows the performance of these three models on Ω_{V1}, Ω_{V2}, and Ω_{Tst}.

3.3.1 GBM

GBM is a machine learning technique which produces a predictive model in the form of an ensemble of weak predictors, typically trees, such that the performance can be significantly improved from the single weak predictors. Readers are referred to [3] for details of the algorithm. In this competition, we directly employed the "gbm" package in R [12] to build the model and make predictions.

We utilized GBM to fit regression models using CTR as the target value and RMSE as the loss function. The target CTR was in the range of [0, 1], but the predicted CTR from a regression model might violate this range limitation. This was not a problem for this task since the the performance measurement was AUC, meaning that we were only interested in rank-ordering the predicted CTRs. Therefore we did not post-process the output to map the predicted CTR back to the [0, 1] range.

There were three parameters to be tuned carefully: the number of trees, the shrinkage parameter, and the depth of the trees. We trained our GBM models on the feature matrix of Ω_{V1}, and chose the parameters that maximize the AUC scores on Ω_{V2}. In the final, the optimized parameters were # of trees=1000, shrinkage parameter=0.01, and depth=8. The performance of this GBM model on Ω_{Tst} was 0.757, as shown in Table 4.

3.3.2 SVM

We used SVM^{perf} with AUC optimization [7]. Since the size of the training data is large, we used a linear kernel for the SVM models. We chose the regularization parameter, $c = 500$, that maximized the AUC scores on Ω_{V2}.

3.3.3 ANN

We implemented an ANN with AUC optimization. To this end, we used the rank statistic, R [5], which is the estimator of AUC as an objective function.

$$R = \frac{1}{PQ} \sum_{j,k}^{P,Q} s(y_j^+ - y_k^-) \quad (7)$$

where P and Q were the numbers of minority and majority class observations respectively. Variables y^+ and y^- were the predictions for the minority and majority class observations respectively. Function $s(.)$ was the sigmoid function $1/(1 + e^{-x})$.

We used one sigmoidal hidden layer, and chose the optimal learning rate, number of hidden nodes, and number of epochs that maximize the AUC score on Ω_{V2}.

4. BLENDING

We used a neural network for blending with one sigmoidal-activation hidden layer and one linear output node. The network was trained to minimize a pairwise ranking error, as described in [6]. The final blend combined 32 predictors as described in the previous sections plus 25 handcrafted features. The structure of the network we used is depicted in Figure 3.

The 25 handcrafted features consisted of 21 inputs, constructed using one-hot encodings as follows:

- `position:` We used 3 inputs to encode the position.
- `gender:` We used 3 inputs to encode the gender.
- `age:` We used 6 inputs to encode the age.
- `tokenOverlap:` We encoded the query keyword token overlap and encode it using 3 inputs. The first being 1.0 for no overlap, the second being 1.0 for one common keyword and the third input being 1.0 for more than 2 common keywords.
- `tokenOverlap:` The query title keyword was also one-hot encoded using 3 inputs.
- `tokenOverlap:` Also the query description overlap was encoded using 3 inputs.

The remaining 4 handcrafted features were logarithms of the number of tokens: `queryID`, `keywordID`, `titleID` and `descriptionID` tokens.

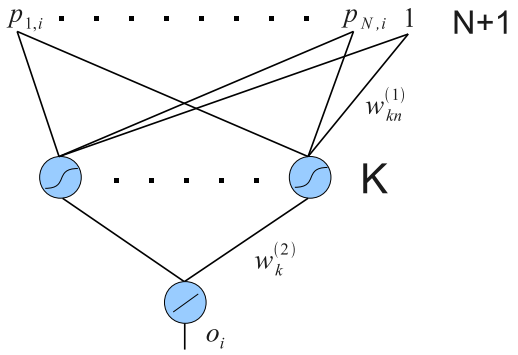

Figure 3: The architecture of the neural network used for blending. The first 25 inputs (p_1 to p_{25}) stem from handcrafted features, the remaining ones are predictors normalized to be in the range from -1 to 1.

The output of the neural network for the training sample i was given by:

$$h_{ki} = tanh\left(\sum_n w_{kn}^{(1)} p_{ni}\right) \tag{8}$$

$$o_i = \sum_k w_k^{(2)} h_{ki} = \sum_k w_k^{(2)} tanh\left(\sum_n w_{kn}^{(1)} p_{ni}\right) \tag{9}$$

We considered pairwise rankings which leads to the following error function:

$$E = \sum_{c_i \neq c_j} ((c_i - c_j) - (o_i - o_j))^2 + \lambda(\sum_k w_k^{(2)}) + \lambda \sum_{k,n} w_{kn}^{(1)} \tag{10}$$

The weights were initialized using small random values drawn from a uniform distribution $\mathcal{U}(-0.001, 0.001)$. We used SGD to learn the weights. Ω_{V1} and Ω_{V2} were used for training and validation respectively. Every training sample with more than 1 impressions was split into multiple training samples. The final blend used a neural network with 20 hidden units and was trained for 20 epochs. The learning rate was 0.0005 and the L2 regularization was 0.001. Additionally we used a multiplicative learning rate decay of 0.85 which was applied after each full training epoch. This results in a AUC of 0.80524 on the public and 0.80824 on the private leaderboard.

5. CONCLUSION

In this paper we presented our ensemble of collaborative filters, Bayesian models, and feature engineered models for predicting CTR of online searching advertisements. This ensemble of models put our team, Opera Solutions, in second place on the private leaderboard of the KDDCup 2012 Track 2 competition.

The collaborative filtering models included AUC optimized bias models and factorized models. These models used different learning rates and regularization parameters for different terms. The Bayesian models estimates the posterior probability that an ad will be clicked conditioning on all the existing features, the IDs of the property fields of the query and the ad. The feature engineered models extracted informative features from the raw data and could be used to build advanced statistical/machine learning models on the feature matrices. Specifically, we implemented AUC optimized ANN models, and used AUC optimized SVM models. RMSE optimized GBM model was also trained on the feature matrices to make predictions.

An AUC optimized ANN model was trained to blend the individual models. The blending model achieved significantly better performance on the public leaderboard than any single model.

As was seen in a number of recent data mining competitions, our results once again underscore the advantages of an ensemble approach to maximizing model predictive performance. We benefited from combining not only a set of different models (bagging), but specifically from using diverse modeling techniques on the same data. In addition, we found that modifying the objective function of the ANN and SVM to specifically optimize AUC, rather than the usual squared error, produced more robust results.

Although we tried to optimize our single and ensemble models, we still managed to find ways to improve the model performance by changing parameters until the last minute of the competition. This emphasizes the importance of parameter tuning and leads us to believe that there is still room for improving our CTR predictive systems.

6. REFERENCES

[1] IAB internet advertising revenue report. Technical report, Interactive Advertising Bureau and PricewaterhouseCoopers LLP, 2009.

[2] K. Dembczynski, W. Kotlowski, and D. Weiss. Predicting ads' click-through rate with decision rules. In *WWW 2008 Beijing*, 2008.

[3] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[4] M. Gupta. Predicting click through rate for job listings. In *WWW 2009 Madrid*, 2009.

[5] A. Herschtal and B. Raskutti. Optimising area under the ROC curve using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, Banff, Canada, 2004.

[6] M. Jahrer and A. Töscher. Collaborative filtering ensemble for ranking. *JMLR: Workshop and Conference Proceedings*, 18:153–167, 2012.

[7] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

[8] A. K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, and N. Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *KDD'11, August 21-24, 2011*, San Diego, USA, 2011.

[9] Y. Niu, Y. Wang, G. Sun, A. Yue, B. Dalessandro, C. Perlich, and B. Hamner. The tencent dataset and KDD Cup'12. In *Proceedings of KDD-Cup Workshop*, 2012.

[10] A. Paterek. Improving regularized singular value

decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, 2007.

[11] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW 2007 Banff, Alberta, Canada*, 2007.

[12] G. Ridgeway. Generalized boosted regression models, 2007. http://cran.r-project.org/web/packages/gbm.

[13] C.-J. Wang and H.-H. Chen. Learning user behaviors for advertisements click prediction. In *SIGIR 2011 Workshop: Internet Advertising*, Beijing, China, 2011.

[14] Y. Zhang, B. J. Jansen, and A. Spink. Identification of factors predicting clickthrough in web searching using neural network analysis. *Journal of the American Society for Information Science and Technology*, 60:1–14, 2008.