Technical report

Predicting clicks in online display advertising with latent features and side-information

Bjarne Ørum Fruergaard
b.fruergaard@adform.com,
Adform Aps, Hovedvagtsgade 6 th., 1103 København K, Denmark

December 1, 2014

Abstract

We review a method for click-through rate prediction based on the work of Menon et al. [1], which combines collaborative filtering and matrix factorization with a side-information model and fuses the outputs to proper probabilities in $[0, \ 1]$. In addition we provide details, both for the modeling as well as the experimental part, that are not found elsewhere. We rigorously test the performance on several test data sets from consecutive days in a click-through rate prediction setup, in a manner which reflects a real-world pipeline. Our results confirm that performance can be increased using latent features, albeit the differences in the measures are small but significant.
1 Introduction

With the growing popularity of the Internet as a media, new technologies for targeting advertisements in the digital domain, a discipline generally referred to as computational advertising, have opened up to new business models for publishers and advertisers to finance their services and sell their products. Online advertising entails using banner ads as a means to attract user attention towards a certain brand or product. The clicks, known as click-throughs, take a user to a website specified by the advertiser and generates revenue for the page displaying the banner, which we call the publisher.

In real-time bidding (RTB) banner ads are determined and placed in real-time based on an auction initiated by the publisher between all potential advertisers, asking them to place a bid of what they are willing to pay for the current impression (displaying the ad), given information about the page, the user engaging the page, a description of the banner format and placement on the page. The advertiser with the highest bid wins the auction and their banner is displayed to the user. RTB thus requires advertisers, or more commonly, the demand side platforms (DSPs) acting on behalf of the advertisers, to be able to estimate the potential value of an impression, given the available information. A key measure for evaluating the potential values of impressions is the click-through rate (CTR), calculated as the ratio of the number of clicks over the total number of impressions in a specific context. What we are investigating in the present work, is a model for predicting CTRs, even in the face of contexts without any
previous clicks and/or very few impressions available, such that the empirical CTR can be unknown or very poorly estimated.

1.1 Dyadic prediction

We frame our main objective of estimating click-through rates for web banner advertisements in the general scope of a dyadic prediction task. Dyadic prediction concerns the task of predicting an outcome (or label) for a dyad, \((i,j)\), whose members are uniquely identified by \(i\) and \(j\), but which may include additional attributes of the dyad \((i,j)\) being observed.

In this paper we are interested in predicting the binary labels being either click or not click, in general referred to as click-through rate prediction, given the pair of a domain and a web banner advertisement. In the following, we give a formal introduction of this problem.

We are given a transaction log of banner advertisements being shown to users. In the logs, various dimensions are recorded, including a banner ID and a domain ID, as well as a number of other attributes, which we shall elaborate more on later. For each record in the log, henceforth called a view, it is recorded whether the banner was clicked or it was displayed without any subsequent click (non-click). Let \(i = 1, \ldots, M\) index the banner dimension and \(j = 1, \ldots, N\) the domain dimension. We can then construct a matrix, \(X\), summarizing the records in the log in terms of empirical click-through rates, i.e., let the entries of the matrix be defined by

\[
X_{ij} = \begin{cases} 
\frac{C_{ij}}{V_{ij}} & \text{if } V_{ij} > 0 \\
? & \text{otherwise}
\end{cases}
\]  

(1.1)

Here \(C_{ij}\) is the number of clicks and \(V_{ij}\) is the number of views involving dyad \((i,j)\). Note that per definition, both clicks and non-clicks count as views, so we must always have \(V_{ij} \geq C_{ij}\). The “?” denotes unobserved pairs, where there is no historical data in the log, hence for such dyads \(X_{ij}\) is undefined.

With this formulation, our click-through rate prediction task is to learn models estimating \(X\). Naturally, any such model should be able to predict the missing entries “?” as well as being able to smoothen predictions, such that the model does not get over-confident in situations with too few views. For instance, if \(C_{ij} = 1\) and \(V_{ij} = 3\), a CTR estimate of \(X_{ij} = \frac{1}{3}\) is probably too extreme, as well as the case \(C_{i'j'} = 0 \Rightarrow X_{i'j'} = 0\), where the natural assumption should rather be that not enough pairs \((i',j')\) have yet been observed.

One possible approach to the above is where additional features about the entities \(i\) and \(j\) are known. This side-information can then be used as predictors in a supervised learning model, such as logistic regression. We refer to this approach as feature-based.
In the complete lack of side-information, one can instead borrow ideas from collaborative filtering. In collaborative filtering the classic setup, e.g., the Netflix movie rating problem [4], is where dyads are (user,item) pairs and each observed pair is labeled with a rating, for instance on the scale 1 to 5. The task is then to predict the ratings for unobserved pairs, which can then be used as a component in a recommender system. In our case we can identify a similar collaborative filtering task, but where instead of ratings we have binary outcomes and the dyads are (banner,domain) pairs. The assumption in collaborative filtering is that for a particular (banner,domain) pair, other row objects (other banners) as well as other column objects (other domains) contain predictive information. I.e., we are assuming that some information is shared between entities and we need to learn a model of this shared information.

In this work we investigate a model that fuses ideas from collaborative filtering via matrix factorization and a mapping to valid probabilities in $[0,1]$, called a latent feature log-linear model (LFL) with a feature-based model for explicit features, that we refer to as a side-information model.

### 1.2 Related work

The model that we investigate in this work was introduced in [11] and builds on the latent feature log-linear model (LFL) from [12]. Our work can be seen as a supplement to [11], as we think this work is lacking in details, which we thus try and provide. Also, we offer different conclusions about the applicability of this model to a dataset of our own, but for the same application as [11]. [11] does not share any of their data so we can unfortunately not reproduce their results.

The modeling of click-through rates has been extensively investigated in the domain of search engine advertising, i.e., the sponsored advertisements that appear in web search engines as a result of user queries for relevant content retrieval. Many methods proposed in this domain are feature-based, e.g., [5, 7, 14] based on logistic regression. Other techniques are maximum likelihood based [3, 9], i.e., they operate directly with the empirically observed counts, which makes it a problem to predict in cold-start settings. Since in search engines, the user directly reveals an intent through his or her query, the features in most of these studies include somehow to predict click-through rates of pairs of (word,ad), which could indeed also be modeled using the LFL framework [12], but to our knowledge this has yet to be investigated.

In the setting that we are looking at, namely placement of banner ads, there is no direct query from the user, so modeling click-through rates cannot be based on (word,ad) pairs and have to be based on other often much weaker predictors of intent. Feature-based approaches are also popular in this setting, see e.g. [10]. Latent feature models are also not much explored in this area, hence a motivation for this work is to combine the best of combining latent and explicit features and share our findings.
Our focus in this work is on combining the LFL model [12] with a logistic regression model on the explicit features as in [11]. This combined model has the advantage that it in the face of weak explicit predictors, recommender effects from the latent features can kick in.

2 Response prediction

The model we apply for response prediction is based on the work in [11], a collaborative filtering technique based on matrix factorization, which is a special case of the latent feature log-linear model (LFL) [12] for dyadic prediction with binary labels. Menon et al. demonstrate that their model incorporating side-information, hierarchies and an EM-inspired iterative refinement procedure overcome many collaborative filtering challenges, such as sparsity and cold-start problems, and they show superior performance to models based purely on side-information and most notably the LMMH model [1]. In the following we introduce the confidence-weighted latent factor model from [11].

2.1 Confidence-weighted factorization

A binary classification problem of the probability of click given a dyadic observation \((i, j)\) for page \(p_i\) and banner \(b_j\), \(p(\text{click}|(i, j))\), can be modeled with the logistic function and a single weight, \(\omega_{ij}\), per dyad. I.e., \(p_{LR}(\text{click}|\omega_{ij}) = \sigma(\omega_{ij})\). However, such a model is only capable of classifying dyads already observed in training data and cannot be applied to unseen combinations of pages and banners. Therefore we assume a factorization of \(\omega_{ij}\) into the factors \(\alpha_i\) and \(\beta_j\) each representing latent feature vectors of the page and banner dimensions, respectively, such that \(\omega_{ij} \approx \alpha_i^T \beta_j\). Henceforth, we will refer to this estimator as \(p_{MF}^{ij} : = p_{MF}(\text{click}|\alpha_i, \beta_j) = \sigma(\alpha_i^T \beta_j)\). With data being \(d = 1, ..., D\) observations of dyads, \(x_d = (i, j)\), with binary labels, \(y_d \in \{0, 1\}\), learning can be formulated as the regular logistic regression optimization problem:

\[
\min_{A, B} \quad - \sum_{d=1}^{D} y_d \log(p_{MF}^{i_d j_d}) + (1 - y_d) \log(1 - p_{MF}^{i_d j_d}),
\]

\[\text{(2.1)}\]

i.e., a maximum-likelihood solution for Bernoulli-distributed output variables using the logistic function to non-linearly map continuous values to probabilities. With the latent variables \(A\) and \(B\) being indexed by \((i, j)\), we can rewrite Eq. \[\text{(2.1)}\] to a confidence-weighted factorization:

\[
\min_{A, B} \quad - \sum_{(i, j) \in \mathcal{O}} C_{ij} \log(p_{MF}^{ij}) + (V_{ij} - C_{ij}) \log(1 - p_{MF}^{ij}),
\]

\[\text{(2.2)}\]
where \( C_{ij} \) is the number of clicks \((y_d = 1)\) involving dyad \((i, j)\) and \((V_{ij} - C_{ij})\) the number of non-clicks \((y_d = 0)\) involving dyad \((i, j)\) in the training data. This reformulation can be a significant performance optimization, since the number of distinct dyads can be much smaller than the total number of observations. E.g., in the case of click-through data, we can easily have many thousands of click and (particularly) non-click observations per dyad, hence the number of operations involved in the summation of Eq. (2.2) is significantly reduced compared to Eq. (2.1).

### 2.1.1 Regularization, learning and bias weights

Optimization of Eq. (2.2) is jointly non-convex in \( A \) and \( B \), but convex for \( A \) with \( B \) fixed, and vice versa. In practice that means we can only converge to a local minimum. Introducing regularization into the problem alleviates some non-convexity by excluding some local minima from the feasible set and additionally helps controlling overfitting. \cite{11} suggests an \( \ell_2 \) norm penalty, thereby effectively smoothing the latent factors:

\[
\min_{A, B} \Omega_{\ell_2}(A, B) = \sum_{(i,j) \in \mathcal{O}} C_{ij} \log(p_{ij}^{MF}) + (V_{ij} - C_{ij}) \log(1 - p_{ij}^{MF}), \tag{2.3}
\]

where \( \Omega_{\ell_2}(A, B) = \lambda \left( \sum_{i=1}^{I} ||\alpha_i||_2^2 + \sum_{j=1}^{J} ||\beta_j||_2^2 \right) \). In this work we also try optimization with an \( \ell_1 \) norm regularizer:

\[
\min_{A, B} \Omega_{\ell_1}(A, B) = \sum_{(i,j) \in \mathcal{O}} C_{ij} \log(p_{ij}^{MF}) + (V_{ij} - C_{ij}) \log(1 - p_{ij}^{MF}), \tag{2.4}
\]

with \( \Omega_{\ell_1}(A, B) = \lambda_\alpha \sum_{i=1}^{M} |\alpha_i|_1 + \lambda_\beta \sum_{j=1}^{N} |\beta_j|_1 \), thereby promoting sparse latent features.

For the \( \ell_2 \) regularized problem Eq. (2.3), a batch solver such as L-BFGS (see \cite[Chapter 9]{13}) can be invoked. For the \( \ell_1 \) regularized problem Eq. (2.4), special care must be taken due to non-differentiability. The quasi-newton method OWL-QN \cite{2} can be used instead in this setting. For really large problems, an on-line learning framework, such as stochastic gradient descend (SGD) is more scalable; again requiring special handling of the \( \ell_1 \) regularizer; see \cite{15} for details.

In general with classification problems with skewed class probabilities, e.g., observing many more non-clicks than clicks, we can add bias terms to capture such baseline effects. We follow the suggestion from \cite{12} and add separate bias terms for each row and column object, i.e., in our case per-page and per-banner bias weights. Hence, without loss of generality, when we refer to \( \alpha_i \) and \( \beta_j \), we assume they have been appended \([\alpha_0, 1]^T\) and \([1, \beta_0]^T\), respectively, thereby catering for the biases. Furthermore, when we speak of a rank \( k \) latent feature model, we actually refer to a rank \( k + 2 \) model consisting of \( k \) latent features as well as the two bias dimensions.
2.2 Feature-based response prediction

A different approach to response prediction is a model based on explicit features available in each observation. In the case of click-through data, such information could for instance be attributes exposed by the browser (e.g., browser name and version, OS, Screen resolution, etc.), time-of-day, day-of-week as well as user profiles based on particular user’s previous engagements with pages, banners, and with the ad server in general.

Again, we can use logistic regression to learn a model of binary labels: For \( d = 1, \ldots, D \) observations we introduce feature vectors, \( x_d \), and model the probability of click given features with the logistic function, i.e., \( p^L_R = p^{LR}(\text{click}|\omega^{LR}) = \sigma(\omega^{T}_R x_d) \). The optimization problem for learning the weights \( \omega^{LR} \) becomes

\[
\min_{\omega^{LR}} \Omega_{\ell_1}(\omega^{LR}) - \sum_{d=1}^{D} y_d \log(p^L_R) + (1 - y_d) \log(1 - p^L_R),
\]

(2.5)

where \( \Omega_{\ell_1}(\omega^{LR}) = \lambda^{LR}|\omega^{LR}|_1 \) is added to control overfitting and produce sparse solutions. As discussed in Section 2.1.1 adding bias terms can account for skewed target distributions, and may be included in this type of model, e.g., as a global intercept, by appending an all-one feature to all observations. Alternatively, if we want to mimic the per-page and per-banner biases of the latent factor model, we do so by including the page indices and banner indices encoded as one-of-\( M \) and one-of-\( N \) binary vectors, respectively, in the feature vectors.

2.3 Combining models

With dyadic response prediction as introduced in Section 2.1 the model can be extended to take into account side-information available to each dyadic observation. I.e., introducing an order-3 tensor \( X \) with entries \( x_{ij} = X_{ij} \) being the feature vectors of side-information available to the \((i, j)\) dyad, we follow \[11\] and model the confidence-weighted factorization with side-information as

\[
p^{SI}_{ij} = p^{SI}(\text{click}|\alpha_i, \beta_j, \omega^{SI}) = \sigma(\alpha^T_i \beta_j + \omega^{T}_{SI} x_{ij}).
\]

Learning such a model by jointly optimizing both \( \alpha \), \( \beta \), and \( \omega^{SI} \), is non-convex and may result in bad local minima \[12\]. To avoid such bad minima, \[11\] \[12\] suggest a simple heuristic; first learn a latent-feature model as the one detailed in Section 2.1, then train the side-information model as in Section 2.2, but given the log-odds \( \alpha^T_i \beta_j \) from the latent-feature model as input. I.e., \( p^{SI}_{ij} \) can be rewritten as

\[
p^{SI}_{ij} = \sigma((1; \omega^{SI})^T [\alpha^T_i \beta_j; x_{ij}]),
\]

(2.6)

hence, having first learned \( \alpha^T_i \beta_j \) c.f. Section 2.1, \( \omega^{SI} \) can be learned by extending the input features with the log-odds of the latent-feature model and fixing
the corresponding weights to one. This training heuristic is a type of residual fitting, where the side-information model is learning from the differences between observed data and the predictions by the latent-feature model.

In practice we have found the above procedure to be insufficient for obtaining the best performance. Instead we need to alternate between fitting the latent features and fitting the side-information model, each while holding the predictions of the other model as fixed. This leaves how to train the latent feature model using the current side-information model prediction as fixed parameters open. Therefore we in the following show how this can be achieved.

For Eq. (2.3), which we will use as the working example, the observations are summarized for each unique dyad, \((i,j)\), in terms of the click and non-click counts, regardless of the side-information in each of those observations. Therefore we now address the question: Given \(\omega_{LR}\) from Section 2.2, how do we obtain the quantities \(\omega_{SI}x_{ij}\) in Eq. (2.6)?

Initially, we define the notation \(x^{(i,j)}_d\) as indexing the \(d^{th}\) explicit feature vector involving the dyad \((i, j)\). Hence, for dyad \((i, j)\) there are \(V_{ij}\) (potentially) different feature vectors \(x^{(i,j)}_d\), \(d = 1, ..., V_{ij}\), involved. Assuming a model learned on the explicit features alone according to \(p_{LR}^d\) from Section 2.2, the overall predicted click-through rate for the observations involving dyad \((i, j)\) becomes

\[
p_{ij}^{LR} := \frac{1}{V_{ij}} \sum_{d=1}^{V_{ij}} \sigma(\omega_{LR}^T x^{(i,j)}_d),
\]

(2.7)

which is obvious from the fact that the sum calculates the predicted number of clicks and \(V_{ij}\) is the empirical number of observations. I.e., Eq. (2.7) is just the average predicted click-through rate taken over the observations involving dyad \((i, j)\). Using this result we can now make sure the combined model yields \(p_{ij}^{LR}\) when either \(\alpha_i = 0\) or \(\beta_j = 0\) (or both) by fixing the term \(\omega_{SI}x_{ij}\) according to the log-odds of \(p_{ij}^{LR}\). Hence,

\[
\omega_{SI}^T x_{ij} = \log(p_{ij}^{LR}) - \log(1 - p_{ij}^{LR})
\]

(2.8)

should be used as fixed inputs while learning the latent-factor model and thus accounts for the predictions of the feature-based model the same way as bias terms account for baseline effects.

3 Data and experiments

We will run experiments on datasets extracted from ad transaction logs in Adform, an international online advertising technology provider. Due to the sequential nature of these data, we will report results from training a model on 7 consecutive days and then testing on the 8th. For measuring performance
we report area under the ROC curve (AUC) scores as well as the logistic loss, evaluated on the held-out data (i.e., the last day). We evaluate different instantiations of the model over a period of in total 23 test days each using the previous 7 days for training and therefore can also report on the consistency of the results.

The data consists of observations labeled either click or not click and in each observation the domain and the banner id are always known. The additional features, that are features for the side-information, include various categorical variables all encoded as one-of-K. These include the web browsers UserAgent string (a weak fingerprint of a user), an indicator vector of the top-50k (for a single country) websites the user has visited (URLs visited) the past 30 days, the full URL of the site being visited (top-50k indicator vector, per country), a binned frequency of the times the user has clicked before (never, low, mid, high), as well as cross-features of the above mentioned and each banner id, thereby tailoring coefficients for each ad. The resulting number of features ($P$) is between 500k-600k and the number of positive observations is around 250k ($N_1$), i.e., the problem is overcomplete in features and thus $\ell_1$ on the side-information model is added as a means of feature selection. The negative class $N_0$ is down-sampled by a factor of 100 bringing it down to around 1.5M-2.5M. The resulting model can be corrected in the intercept [8]. In our experience down-sampling the negative class this drastically and calibrating the model by intercept correction does not impact downstream performance.

We train different variations of the models to investigate in particular the usefulness of latent features in addition to a model using only explicit features. The different models are:

**LR** Logistic regression on the side-information alone. The corresponding regularization strength we call $\lambda_{LR}$.

**LFL_0** The latent feature log-linear model using only the bias features, i.e., corresponding to a logistic regression for the two indicator features for domain and banner, respectively. The corresponding regularization weights we refer to as $\lambda_{\alpha_0}$ and $\lambda_{\beta_0}$, but as we describe later, in practice we use the same weight, $\lambda_0$, for both.

**LFL_K** The latent feature log-linear model with $K \geq 0$ including bias features. The corresponding regularization weights we call $\lambda_\alpha$ and $\lambda_\beta$, which we also find in practice can be set to be equal, and for this introduce the weight $\lambda_{\alpha\beta}$.

**LR+LFL_K** The combined model with $K \geq 0$ for the LFL model combined with the side-information model.
3.1 Tuning hyper-parameters

The combined model with both latent features and side-information we dub \( \text{LR} + \text{LFL}_K \). This model has up to 5 (!) hyper-parameters that need tuning by cross-validation: \( (\lambda_{LR}, \lambda_{\alpha_0}, \lambda_{\beta_0}, \lambda_{\alpha}, \lambda_{\beta}) \). \cite{12} does not report whether they use individual \( \lambda_{\alpha}, \lambda_{\beta}, \lambda_{\alpha_0}, \) and \( \lambda_{\beta_0} \) weights, but we consider this highly infeasible. What we have found to be most effective, is to use the same weight \( \lambda_{\alpha\beta} \) for the latent dimensions as well as a shared bias weight \( \lambda_0 \), which narrows the search space down to three hyper-parameters that must be tuned.

Tuning three hyper-parameters is still a cumbersome task, in particular for large datasets, where an exhaustive search for a reasonable grid of three parameters becomes too time consuming. Instead we have had success using the following heuristic strategy for tuning of these parameters:

1. First run experiments for the logistic regression model alone and find a suitable regularization weight \( \lambda_{LR} \).
2. Run experiments for the LFL_0 model (i.e., bias weights only) and find a suitable \( \lambda_0 \).
3. Run experiments for a number of LFL_K models with \( K > 0 \), with bias weights regularized by \( \lambda_0 \) fixed from (2), and find a suitable \( \lambda_{\alpha\beta} \).
4. Finally, train the combined LFL+LR_K model with different \( K \geq 0 \) and \( \lambda_0 \) fixed, but varying \( \lambda_{\alpha\beta} \) as well as \( \lambda_{LR} \) both in the neighborhood of the values found in (1) and (3). If the results indicate performance could be improved in any direction away from that region, we run more experiments with the hyper-parameters set in that direction.

To verify the validity of this approach, we have run experiments with the hyper-parameters set to their optimal settings as per the above procedure, and varying one at the time, including separate weights for the latent features and biases. In this way we do not find any increase in the performance along any single direction in the space of hyper-parameters.

4 Results and discussion

4.1 Validation set results and initialization

We use the first 8 days, i.e., train on 7, test on the 8th, to find a reasonable range of hyper-parameters that we will test over the entire period. I.e., we use the first test day of a total of 23 days (30 days worth of data, where the first 7 are only used for training) as our validation set. At the same we initialize models with different parameters, that we use for warm starting the training on subsequent data. In the following we provide our results where we are testing performance on a single day, thereby gaining insights into both hyper-parameter
Figure 1: Results using $\ell_2$ regularization for LFL$_K$ modeling on a single day with varying regularization strengths and model orders. (a) A sweep in over the LFL$_0$ regularization strength $\lambda_0$ in the vicinity of the optimum. (b) Using $\lambda_0 = 3.0$ the AUC is plotted as intensities in a grid with varying model orders and the shared regularization strength $\lambda_{\alpha \beta}$ for the latent dimensions. The annotations marks the optimum. (c) Same as (b), except for logistic loss, i.e., the lower the better.

values, model order ($K$) and regularization type ($\ell_1$ and $\ell_2$) for the latent features.

In Fig. 1 we show the results using $\ell_2$ regularization (see Eq. (2.3)) and varying $\lambda_0$ with an LFL$_0$ model (a) and $\lambda_{\alpha \beta}$ in (b-c) with $\lambda_0 = 3.0$ fixed, different model orders and no side-information model. Results are not shown for experiments where $\lambda_0$ and $\lambda_{\alpha \beta}$ were varied in larger grids, i.e., these plots focus of where the performances peak. What we also learn from these plots (b-c), is that higher model orders are advantageous, but that this increase levels off from between $K = 5$ to $K = 20$. This is in contrast to [11] reporting $K \geq 200$ being advantageous. We have also run experiments not shown here with $K = 100$ and $K = 200$ and seen no further increase, and if anything at all, a slight decrease in performance.

The same experiments using $\ell_1$ regularization are summarized in Fig. 2. Both the experiments for the bias regularization $\lambda_0$ (a) as well as those for the latent factors (b-c) do not show as good performances as in the case of $\ell_2$ regular-
Figure 2: Results using $\ell_1$ regularization for LFL$_K$ modeling on a single day with varying regularization strengths and model orders. Also see the caption for Fig. 1.
Figure 3: Results using the $\ell_2$ regularized version for LR+LFL$_K$ on the first day and varying regularization strengths as well as model orders. $\lambda_0 = 3.0$ remains fixed. In the top we show AUC intensities and in the bottom logistic losses. Little x’s are used to mark those specific configurations that we run experiments with across all the test days.

In Fig. 3 we show experiments with varying $\lambda_{LR}$, $\lambda_{\alpha\beta}$ as well as different models orders $K$ for combined LR+LFL$_K$ models. We confirm a trend towards better performance using higher $K$, but again saturating beyond $K = 5$. We further notice that peak performances in terms of AUC do not necessarily agree completely with those for LL. There may be other explanations as to why that is, but we believe this is a consequence of the LL being sensitive to probabilities being improperly calibrated, while the AUC is not. Inspection of the different models seem to confirm this; where the models perform better in terms of logistic loss, the predicted click-through rate for the test set is (slightly) closer to the empirical, than for those models which maximize AUC. We expect that a post-calibration of the models beyond just an intercept correction could be beneficial for the reported logistic losses, but also note that this would not change the AUCs.

As mentioned in Section 2.3, we find that alternating between fitting the latent model and the side-information model is necessary. For the experiments Fig. 3 we have alternated 7 times which we have confirmed in practice ensures the
performance has leveled off. An example supporting this claim is shown in Fig. 4 and serves to illustrate the general observation we make in all our experiments.

4.2 Results on 22 consecutive days

With just a subset of LR+LFL$_K$ models hand-picked (marked by little x’s in Fig. 3) from the experiments on the first test day, we run experiments on a daily basis while initializing with the models from the previous day. This sequential learning process reflects how modeling would also be run in production at Adform and by warm starting the models in the previous days’ coefficients, we do not expect that running multiple epochs of alternated fitting is required, i.e., this only needs to be done once for initialization.

In the following, the AUCs and logistic losses we report are daily averages of the performances for each banner. As opposed to the performances over an entire test data set that we have reported up until now, making daily averages per banner prevents the performance numbers from being entirely dominated by a single or a few banners, and instead assigns per-banner performances equal weights.

Reporting performances based on slices of data per banner further allows analysis of under which circumstances the latent feature models add a statistical significant improvements. In Fig. 5 we show the difference in AUC banner averages per day in the total of 22 days we use for testing. The upper shows the performances for all the banners with 1 or more clicks in each test set (day), while the lower is averaged daily performances for only the banners with 10 or more clicks. It is apparent from these two figures, that AUC scores based on very few clicks add significant variance to the daily averages and the difference between the model orders is hard to spot. We also note, that since we cannot evaluate AUCs score for banners without clicks in the test set, these are ignored entirely. For logistic loss, however, we can still report a performance for banners
Figure 5: Daily average AUC differences (i.e., increase) for LR+LFL_K models using the optimal settings for different model-orders (colored lines) relative to the side-information model alone. Shaded, gray lines in the background trace all of the different configurations tested. In the above the averages include every banner with 1 or more clicks on a day (between 900-1000 banners qualify each day), while in the lower all the banners on a particular day with less than 10 clicks is filtered from the averages (between 400-500 banners qualify each day), hence decreasing the variance in the measures.
Figure 6: **Top row:** Box plots of the relative AUC differences corresponding to each of the models also displayed in Fig. 5. **Bottom row:** Box plots also for the relative logistic loss differences with the only difference being that in the left-most box plot, the losses for all banners each day are included in the averaging (1100-1200 daily). **Legend:** Boxes show 1st and 3rd quartiles, whiskers are 1.5 IQR (interquartile range), medians are red lines and green circles mark the means. Outliers are marked with pluses. The notches are 5000 bootstrap sample estimates of the 5%-95% confidence interval for the medians.

without clicks in the test. The LR model used as the reference (0.0) in Fig. 5 uses \( \lambda_{LR} = 4.0 \), which we found as optimal over the entire 22 day period testing a grid from 2.0 to 7.0 in increments of 0.5.

In order to further quantify and investigate the impact different model orders has on performance, we summarize in Fig. 6 the relative differences over the 22 test days in box plots. Again, we show performances relative to the side-information model and for different inclusion criteria, based on number of clicks in the test sets. In all cases, we see an increase in performance, as the model order is increased, and this increase levels off from \( K = 10 \) to \( K = 50 \). The notches on boxes are 5k sample bootstraps of the medians, hence based on these we can say something about the statistical significance of these results. I.e., non-overlapping notches correspond to \( p < 0.05 \) for a two-tailed null-hypothesis test. First of all, all model orders, including \( K = 0 \), improve performances compared to the side-information model alone.

For both the AUCs and the logistic losses we see wide confidence intervals on
the medians, when banners with very few clicks (< 10) per day are included. We still observe an increase in performance as the model order increases, but only in the case of logistic loss do the model orders \( K = 20 \) and \( K = 50 \) barely clear overlapping with the notches of \( K = 0 \).

In the case of including only banners with more than 10 clicks in the summary statistics, the confidence intervals of the medians shrink, in particularly in the case of logistic loss. However, the relative gains (means and medians) are also slightly lower. I.e., there is a trend, albeit barely statistically significant, that there are higher gains among the banners with few clicks in the test sets, than for those with more. Apart from this, there is now also statistically significant differences between the medians for the higher model orders and \( K = 0 \); in the case of AUC this includes \( K = 20 \) and \( K = 50 \), and in the case of logistic loss, \( K \geq 3 \) are statistically better.

It is worth noting that, regardless of the slice based on number of clicks in the test sets, the results agree that using the LR+LFL\(_K\) model yields higher performance than the LR model alone.

For the results in Fig. 6, while we find evidence that supports that latent features improves click-through prediction, the question remains how much this improves real-world performances. Indeed the increments which the latent features introduce in the two measures we report here seem very small. When measuring AUC scores, in particular, we are however not the first to report small, but significant improvements on the third decimal. As McMahan \[9\] (on web search ads) puts it:

*The improvement is more significant than it first appears. A simple model with only features based on where the ads were shown achieves an AUC of nearly 0.80, and the inherent uncertainty in the clicks means that even predicting perfect probabilities would produce an AUC significantly less than 1.0, perhaps 0.85.* \[9, p.532\]

Our data as well as our experiences in web banner ads support this statement, and we often also identify new features or model changes with these low levels of improvement, but which however remain consistent.

Another possibility as an alternative to off-line measures on held-out data, such AUC and logistic loss, is live A/B testing. Yet, before taking new features, models or technology into production, a prerequisite to us at least, is to demonstrate consistent off-line data performance improvements. For the present work, we have not had the opportunity to test it live.

5 Conclusion

In this work we have reviewed a method for click-through rate prediction which combines collaborative filtering and matrix factorization with a side-information
model and fuses the outputs to proper probabilities in $[0, 1]$. We have provided
details about this particular setup that are not found elsewhere and shared re-
results from numerous experiments highlighting both the strengths and the weak-
nesses of the approach.

We test the model on multiple consecutive days of click-through data from
Adform ad transaction logs in a manner which reflects a real-world pipeline and
show that predictive performance can be increased using higher-order latent
dimensions. We do see a level-off in the performances for $\approx K \geq 20$, whereas
$K \geq 200$ was suggested in another work [11], but may be due to differences
in the data sets; in particular how many side-information features are available
and used.

Our numerous experiments detail a very involved phase for finding proper re-
gions for the various hyper-parameters of the combined model. This is par-
ticularly complicated, since the latent feature model and the side-information
model need to be trained in several alternating steps, for each combination of
hyper-parameters. This we think is one of the most severe weaknesses of this
modeling approach. We circumvent some of the complexity of finding good
hyper-parameters by using shared regularization strengths for both entities of
the latent model and demonstrate, that in a sequential learning pipeline, it is
only for initialization of the model, i.e., on the first training set, that we need
multiple alternating steps.

For future studies, it would be particularly useful if the hyper-parameters could
instead be inferred from data. Yet, as we also show in our results, the objective
differences (i.e., the evidence) that separate good models from the bad, are
small, hence we expect any technique, such as Type II maximum likelihood,
would be struggling to properly navigate such a landscape.

References

[1] Agarwal, D., Agrawal, R., Khanna, R., and Kota, N. Estimating rates
of rare events with multiple hierarchies through scalable log-linear mod-
els. Proceedings of the 16th ACM SIGKDD international conference on
Knowledge discovery and data mining - KDD '10, page 213, 2010.

[2] Andrew, G. and Gao, J. Scalable training of L1-regularized log-linear mod-
els. Proceedings of the 24th international conference on Machine learning,
pages 33–40, 2007.

[3] Ashkan, A., Clarke, C. L. a., Agichtein, E., and Guo, Q. Estimating Ad
Clickthrough Rate through Query Intent Analysis. 2009 IEEE/WIC/ACM
International Joint Conference on Web Intelligence and Intelligent Agent
Technology, pages 222–229, 2009.
[4] Bennett, J. and Lanning, S. The netflix prize. In KDD Cup and Workshop in conjunction with KDD, 2007.

[5] Chakrabarti, D., Agarwal, D., and Josifovski, V. Contextual advertising by combining relevance with click feedback. Proceedings of the 17th international conference on World Wide Web, pages 417–426, 2008.

[6] Dembczynski, K., Kotlowski, W., and Weiss, D. Predicting ads’ click-through rate with decision rules. Workshop on Targeting and Ranking in Online Advertising, 2008.

[7] Graepel, T., Borchert, T., Herbrich, R., and Com, R. M. Web-Scale Bayesian Click-Through Rate Prediction for Sponsored Search Advertising in Microsoft’s Bing Search Engine. Search, (April 2009), 2010.

[8] King, G. and Zeng, L. Explaining Rare Events in International Relations. International Organization, 55(3):693–715, September 2001.

[9] McMahan, H. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. International Conference on Artificial Intelligence and Statistics, 2011.

[10] McMahan, H., Holt, G., and Sculley, D. Ad click prediction: a view from the trenches. Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013.

[11] Menon, A., Chitrapura, K., and Garg, S. Response prediction using collaborative filtering with hierarchies and side-information. Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining KDD, 2011.

[12] Menon, A. A. K. and Elkan, C. A log-linear model with latent features for dyadic prediction. 2010 IEEE International Conference on Data Mining, pages 364–373, December 2010.

[13] Nocedal, J. and Wright, S. J. Numerical Optimization. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, 1999.

[14] Richardson, M., Dominowska, E., and Ragno, R. Predicting clicks: estimating the click-through rate for new ads. Proceedings of the 16th international conference on World Wide Web, pages 521–530, 2007.

[15] Tsuruoka, Y., Tsujii, J., and Ananiadou, S. Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty. Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - ACL-IJCNLP ’09, 1:477, 2009.