We present in this paper a
machine learning framework
based on logistic regression
for display advertising.

The resulting system
is easy to implement and deploy
is highly scalable
provides models with state-of-the-art accuracy.

1. INTRODUCTION

Machine learning framework
effective with a small memory footprint.

Logistic Regression

A two-phase feature selection algorithm
generalized mutual information method
to select the feature groups to be included in the model
feature hashing
to regulate the size of the models.

2 Related work
3 Differences between click and conversion rates with respect to features
analyze the delay between clicks and conversions
4 Logistic Regression model
features
hashing trick used in our framework
smoothing and regularization are asymptotically similar
5 Results
6 Modified version of mutual information that is used to select feature groups
7 Algorithm for exploration
8 Map-reduce implementation
9 Summary & Conclusion

~~2 RELATED WORK [Details]~~

3 DATA AND FEATURES

Display advertising data
Yahoo!'s Right Media Exchange
network-of-networks

Click & Post-Click
CTR & C(V)R

advertiser, publisher, user, time

Feature family	Feature members
Advertiser	advertiser (id), advertiser network, campaign, creative, conversion id, ad group, ad size, creative type, offer type id (ad category)
Publisher	publisher (id), publisher network, site, section, url, page referrer
User (when avail.)	gender, age, region, network speed, accept cookies, geo
Time	serve time, click time

Table I. Sample of features considered divided into feature families.

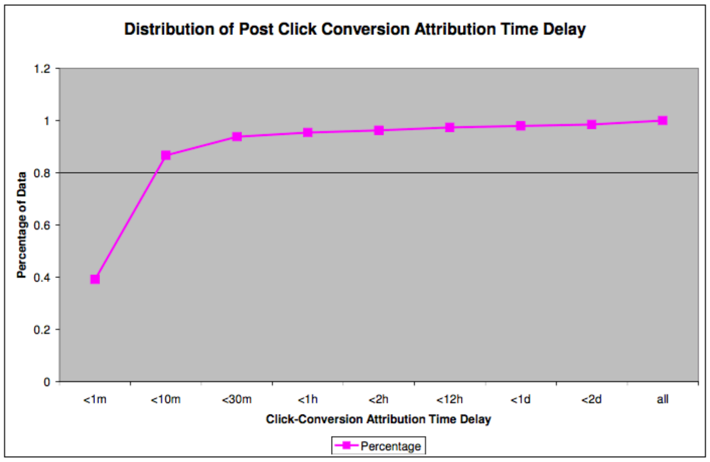


Fig. 1. Distribution of click conversion attribution time delay.

Ignore approximately 1.5% of the conversion

Incorrectly label a click event as negative (no conversion)

Time frame set for PCC attribution is limited to 2 days

Ad clicks (97%) occur within a minute of the ad impression

4 MODELLING

The predicted probability of an example x belonging to class 1 is:

$$\Pr(y = 1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$

The logistic regression model is a linear model of the log odds ratio: (1)

$$\log \frac{\Pr(y = 1 \mid \mathbf{x}, \mathbf{w})}{\Pr(y = -1 \mid \mathbf{x}, \mathbf{w})} = \mathbf{w}^\top \mathbf{x}$$

The weight vector w is found by minimizing the negative log likelihood with an L_2 regularization term on the weight vector: (2)

$$\min_{\mathbf{w}} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i))$$

- solve with any gradient based optimization technique
- L-BFGS

All features are categorical

- Real valued features made categorical through discretization

Standard way of encoding categorical features is to use several binary features

- one-hot | 1-of-c | dummy encoding

$$d = \sum_{f=1}^{F} c_f$$

- dimensionality (d) can get very large

Hashing Trick which popularize with Vowpal Wabbit learning software

Hashing Trick

- make use of the one-hot
- instead of a c-dimensional code, d-dimensional one
  - where d is the number of bins used with hashing
  - if d < c
    - compressed representation
    - collisions are bound to occur
    - not a major concern

- two possible strategies:
  - Hash each feature f into d_f dimensional space and concatenate
    - number of features x d_f
  - Hash all features into same space, different hash function for each feature

- used latter approach

ALGORITHM 1: Hasing trick

```
Require: Values for the $F$ features, $v_1, \ldots, v_F$.
Require: Family of hash function $h_f$, number of bins $d$.
  $x_i \leftarrow 0, \quad 1 \le i \le d$.
  for $f = 1 \ldots F$ do
    $i \leftarrow [h_f(v_f) \mod d] + 1$.
    $x_i \leftarrow x_i + 1$
  end for
  return $(x_1, \ldots, x_d)$.
```

The values v_f can be of any type
there is a hashing function h_f for every feature f
can be implemented:
- using a single hash function and having f as the seed
- concatenating f to the value v_f to be hashed.

Other ways of reducing the dimensionality of the model
- discarding infrequent values
  - more data processing
  - use of a dictionary

Appeal of the hashing trick:
- simplicity
- no additional data processing
- no data storage
- straightforward to implement

Collisions
- first has been observed n_1 times, all on negative examples
- second has been observed n_2 times, all on positive examples,
- when there is only one feature in the system (no fallback on other features)
- no collision:
  - the weight for the first value would be $-\infty$
  - the weight for the second value would be $+\infty$.

  - log likelihood is zero where either value is present

- with collision:
  - log likelihood is:

$$-n_1 \log \frac{n_1}{n_1+n_2} - n_2 \log \frac{n_2}{n_1+n_2}$$

  - all the weights are at 0
    - except the one where there is a collision
      - has a value $\log(n_2/n_1)$.

  - log likelihood is large only when both n_1 and n_2 are large
  - Worst case one because:
    - (1) Two values were extremely predictive;
      - if not predictive, collision not harm the log likelihood
      - zero weight in all cases
    - (2) No redundancy in features
      - if the system includes redundant features:
        - a collision on one value could be mitigated by another value

Regarding the last point:

alleviate the collision issue by making use of multiple hash functions

Bloom filter

does not improve the results

Linear model learn effects independently for each feature

not learn that the CTR of a bank of America ad is particularly high on finance.yahoo.com

new conjunction feature that is the cartesian product of advertiser and publisher

Conjunction between two categorical variables of cardinality c_1 and c_2

another categorical variable of cardinality $c_1 \times c_2$

if c_1 and c_2 are large, the conjunction feature has high cardinality

Conjunctions between all features = considering a polynomial kernel of degree 2

Due to the large cardinality of the representation, there will most likely be **pairs** of **variable** values that are **unseen** in the **training** data. And these pairs, take (advertiser, publisher) for instance, are biased by the current serving scheme, as specific advertisers are selected for a given publisher. The **hashing trick** helps **reduce** the **dimensionality** of data, **but** might do a **poor job** on these **unseen pairs** of values **due** to **collisions**. This can be problematic, especially in an exploration setting where predictions on in- frequent values are required. A **possible solution** is to **represent** the **variable** values **using** a **low-dimensional** representation, for example through the use of matrix factorization or a related approach. This type of representation is **not studied** in this **paper**. A promising future work direction would be to combine a low-dimensional representation with the hashing trick: the former would capture the general trend in the data while the latter could be use to refine the model for the frequent pairs observed in the training set.

Multi-task learning = single model with conjunctions as presented in this paper

~~[Details]~~

Subsampling

Data is large – around 9B impressions daily

Training data is very imbalanced – the CTR is lower than 1%.

Sub-sample the negative class at a rate $r \ll 1$

$\Pr'$: Probability distribution after subsampling

$$\begin{aligned}
\frac{\Pr(y=1\mid \mathbf{x})}{\Pr(y=-1\mid \mathbf{x})} &= \frac{\Pr(\mathbf{x}\mid y=1)\Pr(y=1)}{\Pr(\mathbf{x}\mid y=-1)\Pr(y=-1)} && (4)\\
&= \frac{\Pr'(\mathbf{x}\mid y=1)\Pr'(y=1)}{\Pr'(\mathbf{x}\mid y=-1)\Pr'(y=-1)/r} && (5)\\
&= r\frac{\Pr'(y=1\mid \mathbf{x})}{\Pr'(y=-1\mid \mathbf{x})} && (6)
\end{aligned}$$

The above equation relies on the fact that the class conditional distribution are not affected by the subsampling: $\Pr(x \mid y) = \Pr'(x \mid y)$.
Combining equations (1) and (6), it turns out that the log odds ratio is shifted by log r.
Thus, after training, the intercept of the model has to be corrected by adding log r to it.

Instead of shifting the intercept
- give an importance weight of 1/r to the negatives samples,
- experiments with this solution showed a lower test accuracy
- generalization error bounds are worse with importance weighting

Regularization and Smoothing

Single feature.
- We draw in this section a connection between
  - Tikhonov regularization as used in this paper
  - Laplace smoothing

Suppose that our model contains a single categorical feature.
Let us consider the j-th value of that feature.

$$w^* = \arg\min_w \sum_{i \in I_j} \log(1 + \exp(-y_i w)) + \frac{\lambda}{2} w^2 \qquad (7)$$

with $I_j = \{i, x_i = j\}$
When there is no regularization ($\lambda = 0$), the closed form solution for w^* is:

$$w^* = \log \frac{k}{m-k} \quad \text{with } k = |\{i \in I_j,\ y_i = 1\}| \text{ and } m = |I_j|.$$

This leads to a prediction equal to k/m, which is the empirical probability $P(y = 1|x = x_j)$.
just re-derived that the logistic loss yields a Fisher consistent estimate of the output probability.

This empirical probability may have a large variance when m is small.
This is the reason people often use a Beta prior to get a biased but lower variance estimator:

$$\frac{k+\alpha}{m+2\alpha}. \qquad (8)$$

This estimator is referred to as the Laplace estimator.

The regularizer in (7) is another way of smoothing the probability estimate toward 0.5.
These two methods are in general not equivalent, but they are related.
The following proposition shows that the smoothing is similar asymptotically.

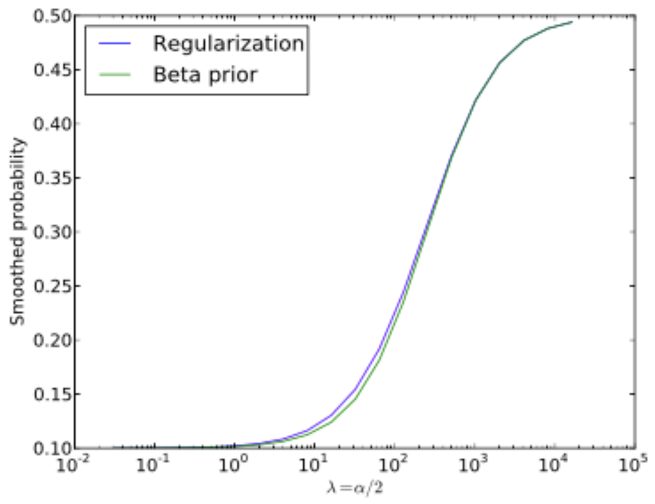


Figure 2: Smoothed probability from 100 success and 1000 trials. The smoothing is achieved by regularization (7) or Beta prior (8).

Hierarchical feature.

Consider now the case of two features having a hierarchical relationship
such as advertiser and campaign.
Suppose that we have a lot of training data for a given advertiser,
but very little for a given campaign of that advertiser.
Because of the regularization, the weight for that campaign will be almost zero.
Thus the predicted CTR of that campaign will mostly depend on the advertiser weight.
This is similar to the situation in previous section
except that the output probability is not smoothed toward 0.5
but toward the output probability given by the parent feature.

The advantage of the logistic regression approach with regularization is that
it implicitly performs hierarchical smoothing:
unlike the works mentioned above, we do not need to specify the feature hierarchy.

Bayesian Logistic Regression

$$\min_{\mathbf{w}} \quad \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \log(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i))$$

it is convenient to have a Bayesian interpretation of logistic regression
Above equation (2) can be interpreted as the Maximum A Posteriori (MAP) solution of a probabilistic model with a logistic likelihood and a Gaussian prior on the weights with standard deviation 1/squareRoot(λ)

There is no analytical form for posterior distribution:

$\Pr(w \mid D) \propto {}^{n}_{i=1} \Pr(y_i \mid x_i, w) \Pr(w)$,

But it can be estimated by a Gaussian distribution using the Laplace approximation

$$\Pr(\mathbf{w} \mid D) \approx \mathcal{N}(\mu, \Sigma) \quad \textbf{with} \quad \mu = \arg\min_{\mathbf{w}} L(\mathbf{w}) \quad \textbf{and} \quad \Sigma_{ii}^{-1} = \frac{\partial^2 L(\mathbf{w})}{\partial w_i^2}$$

That approximation with a diagonal covariance matrix gives:
and $L(w) := -\log \Pr(w \mid D)$ is given in equation (2).

5 RESULTS

Experimental Setup
Most on the RMX data
Some on logs from Criteo – in figures 3, 4, 5 and section 5.6

The training and test sets are split chronologically,
    both periods ranging from several days to several weeks
After subsampling the negative samples
    number of training samples is of the order of one billion
Number of base features is about 30 from which several hundreds conjunctions are constructed
Number of bits used for hashing was 24 resulting in a model with 16M parameters

Test metrics:
    the negative log likelihood (NLL),
    root mean squared error (RMSE),
    area under the precision / recall curve (auPRC)
    area under the ROC curve (auROC)

metric is said to be normalized = metric relative to the best constant baseline.

Hashing Trick
1st component to evaluate is the use of Hashing Trick
Can't run the logistic regression w/o dimensionality reduction
Alternative to Hashing Trick
    keep only most important values
    two simple heuristics for finding important values:
        Count: Most frequent
        Mutual Information: Most helpful in determining the target
            drawback is model needs to be stored as dictionary instead of array
            dictionary maps to each important value to its weight

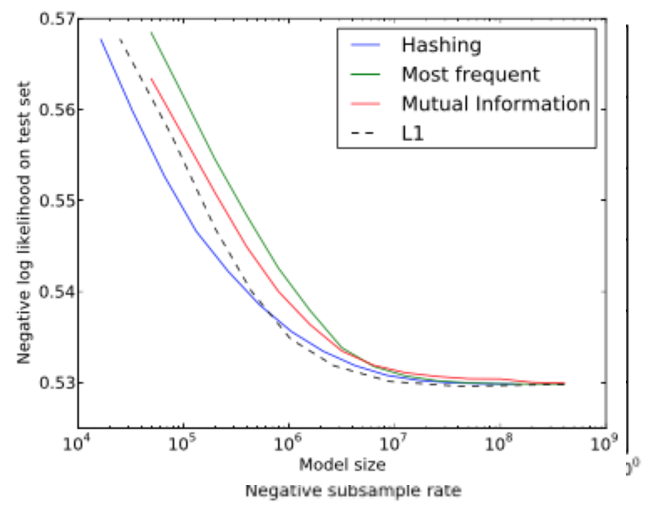


Figure 3: Log likelihood on a test as a function of the model size for different dimensionality reduction strategies.

model size is 4d for hashing and 12d for dictionary based models,
    where d is the number of weights in the model

Same model size, the hashing based model is slightly superior:
    Convenience: no need to find the most important values and keep them in a dictionary
    Real-time efficiency: hashing is faster than a dictionary look-up

For the sake of the comparison, figure 3 also includes a method where the most important values are selected through the use of a sparsity inducing norm. This is achieved by adding an L_1 norm on w in (2) and minimizing the objective function using a proximal method. Each point on the curve corresponds to a regularization parameter in the set {1, 3, 10, 30, 300, 1000, 3000}: the larger the parameter, the sparser the model. The L_2 regularization parameter is kept at the same value as for the other methods. Even though the resulting model needs to be stored in a dictionary, this method achieves a good trade-off in terms of accuracy vs model size. Note however that this technique is not easily scalable: during training, it requires one weight for each value observed in the training set. This was feasible in figure 3 because we considered a rather small training set with only 33M different values. But in a production setting, the number of values can easily exceed a billion.

Effect of Subsampling

Two different levels of subsampling that can be done:

	1%	10%
auROC	-2.0%	-0.5%
auPRC	-7.2%	-2.1%
NLL	-3.2%	-2.3%

    Subsampling the negatives as discussed in section 4.7
    Subsampling the entire dataset

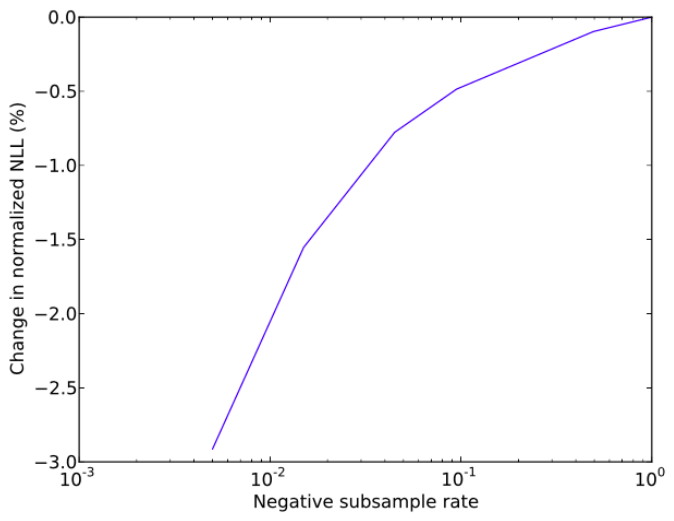


Table II: Test performance drop as a function of the overall subsampling rate.

Figure 4: Change in log likelihood by subsampling the negatives. The baseline is a model with all negatives included.

Subsampling the negatives.

Keep all the positives and sub-sample the negatives

Figure 4 shows how much the model is degraded as a function of the subsampling rate

Empirically that a subsample rate of 1% was a good trade-off

Negatives are subsampled at 1% in the rest of this paper

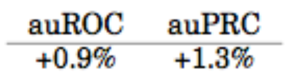
Overall subsampling.

Entire data has been subsampled in this experiment at 1% and 10%.

Table II show that there is a drop in accuracy after sub- sampling

In summary, even if it is fine to subsample the data to some extent – the negatives in particular – the more data the better and this motivates the use of the distributed learning system that will be presented in section 8.

Comparison w/ a feedback model



auROC	auPRC
+0.9%	+1.3%

Both models are similar, with a slight advantage for our proposed model

Table III. Comparison with a click feedback model.

Comparison with a hierarchical model

auROC	auPRC	NLL
+ 3.1%	+ 10.0%	+ 7.1%

We compare our approach to the state-of-the-art LMMH (Log-linear Model for Multiple Hierarchies) method [Agarwal et al. 2010] that has been developed in the same context: CTR estimation for display advertising with hierarchical features.

Table IV. Comparison with LMMH

Value of publisher information

Both for CTR and CVR predictions

A model w/o any publisher features and w/ publisher features

The model w/ publisher features

improves the normalized negative log likelihood

by 52.6% for CTR prediction

by 0.5% for CVR prediction

Publisher features are very useful for CTR prediction as it helps in the ad matching. But once the user clicks on the ad, the publisher's page does not have much impact on that user's conversion behavior. Potentially the PCC could thus be consider- ably simplified as no publisher-side information would be needed.

In the absence of explicit user attributes, publisher information might serve as a proxy for the user features. The website for Disney games will surely attract more kids than adults with no kids.

Multiple hash functions

Table V shows multiple hash functions does not result in any significant improvements

Might be that conjunctions already induce redundancies

2	4	8	12
-0.21%	+ 0.44%	+ 0.13%	-0.14%

Table V: Improvement in log likelihood on the training set as a function of the number of hash functions, the total number of bits remaining constant

6 FEATURE SELECTION

Tackle the problem of feature selection for categorical variables

different levels of selection in the context of categorical variables:

Feature selection. select some of the features to be included

Value selection. discard some of the least important values

L_1 regularization:

a way of reducing the number of parameters in the system

thus a value selection method

not suppress all the values of a given feature

therefore can't be used for feature selection

For this, a possible regularization is the so-called l_1/l_2 regularization or group lasso

As for mutual information and other filter methods for feature selection
    Look for criterion that measures the utility of a feature in a binary classification task

However, the goal is to do so conditioned on already existing features, that is we want to estimate the additional utility of a feature when added to an already existing classifier

Conditional Mutual Information

We assume that
    already have an existing logistic regression model
    with a base set of features
    Let s_i be the score predicted by this model

[Details][18]

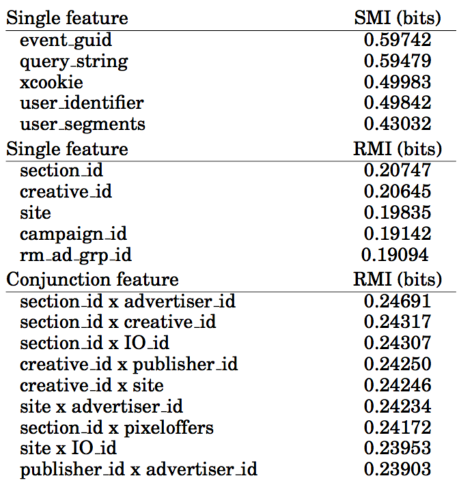
$$\sum_{i=1}^{n} \log \frac{\Pr(y_i \mid x_i)}{\Pr(y_i)}, \qquad (14)$$

which is exactly the mutual information between the variables x and y

The proposed method can thus be seen as an extension of mutual information to the case where some predictions are already available.

Reference Distribution [Details]

There is an overfitting danger:
    it is particularly true for features with a lot of values
    can improve the likelihood on the training set,



Single feature	SMI (bits)
event_guid	0.59742
query_string	0.59479
xcookie	0.49983
user_identifier	0.49842
user_segments	0.43032
Single feature	**RMI (bits)**
section_id	0.20747
creative_id	0.20645
site	0.19835
campaign_id	0.19142
rm_ad_grp_id	0.19094
Conjunction feature	**RMI (bits)**
section_id x advertiser_id	0.24691
section_id x creative_id	0.24317
section_id x IO_id	0.24307
creative_id x publisher_id	0.24250
creative_id x site	0.24246
site x advertiser_id	0.24234
section_id x pixeloffers	0.24172
site x IO_id	0.23953
publisher_id x advertiser_id	0.23903

    but not necessarily on the test set

This problem has also been noted with the standard mutual information

To prevent this issue, a regularization term, λw_k^2, can be added to L_k. [Details][18]

Table VI: Top features for click prediction along with their mutual information. First table: standard mutual information; second and third table: modified mutual information (RMI). Bottom section contains the top conjunction features.

Results

We utilized the method described above for
determining feature relevance for click prediction
Our main motivation was
decreasing the model complexity
as the available number of possible features is too large
to be used in their entirety during prediction/modeling
Practical considerations, such as memory, latency, and training time constraints,
make feature selection a clear requirement in this task.

The evaluation is divided into two parts:
we first verify that computing the mutual information using a reference distribution
gives more sensible results than the standard mutual information;
and then we evaluate the use of the conditional mutual information
in a forward feature selection algorithm

Use of a reference distribution. [Details]

We applied the standard MI (SMI) ranking algorithm for feature selection.
The results, summarized in Table VI (top) reflect our main concern.
The spurious features, or features that are informative about the data point per se
rank substantially high
The calculated MI score is correct in that it reflects the information content of these features;
however, these features are too specific to the training data distribution.

The proposed extension of the MI score utilizing a reference distribution (RMI)
provides a more appropriate ranking as shown in Tables VI (mid-bottom).
The reason for this is that the information content is calculated with respect to (expectations on) the reference distribution and thus feature values that are not seen in the new distribution are basically considered less important and their impact on the information score is reduced.

More specifically, attributes such as event_guid that identifies the data point have maximal information content according to the training distribution (SMI), but near zero information content when calculated with a reference distribution (RMI).
A similar effect was observed for other features that have low relevance for prediction such as query_string and receive_time which, unless parsed, are too specific, xcookie and user_identifier which clearly do not generalize across users (but could be quite informative about a small fraction of the test data), and user_segments which indexes user categories.

The results for other features are more subtle but follow the same underlying principle
where a reference distribution is utilized to avoid spurious dependencies
often found when utilizing empirical distributions

Learning Performance Results.

explore the question of automatically finding new conjunction features and whether these features actually offer any performance gains

use the conditional mutual information within a forward feature selection algorithm:
(1) Start with a set of base features and no conjunction features;
(2) Train a model with all the selected features;
(3) Compute the conditional mutual informations for all conjunctions not yet selected;
(4) Select the best conjunction;
(5) Go back to (2).

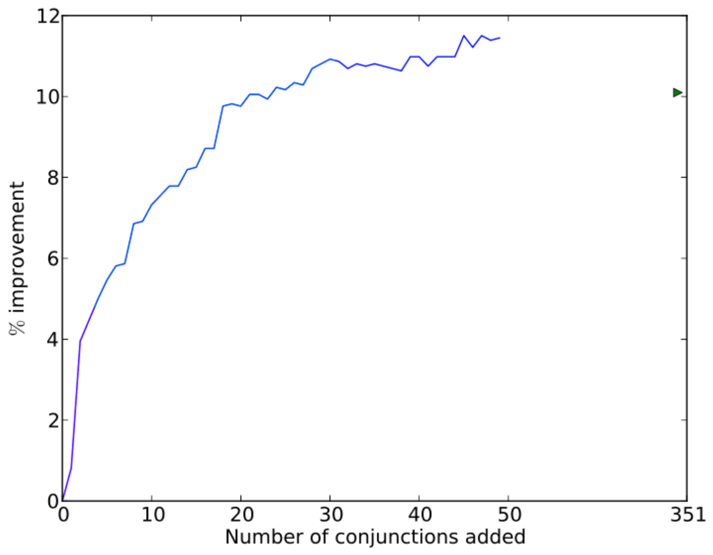


Figure 5

The results of this procedure are shown in Figure 5.
It can be seen that selecting 50 features in this way has a two-fold advantage
over including the 351 possible conjunctions
it results in a model with less features
and it generalizes better.

7 Non-Stationary

Display advertising is a non stationary process
as the set of active advertisers, campaigns, publishers and users is constantly changing

quantify in sections 7.1 and 7.2 these changes
update the model in order to take into account the non-stationarity of the data in section 7.3
discuss Thompson sampling in section 7.4
as a way to address the explore / exploit trade-off necessary in dynamic environment

Ad Creation Rate [Details]

Ad Life-Time

Model Update

data collected from a given month is used for training
that of the following month is used as test data.

In order to evaluate the impact of the new ads on the model performance
we divided the test data into daily slices.

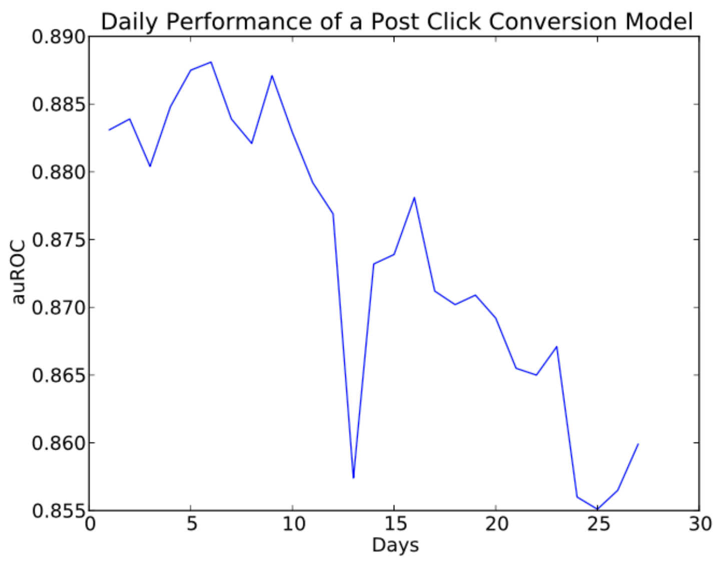


Fig. 9. The performance (auROC) of the model degrades with time.

~~[Details]~~

~~8 Large Scale Learning [Details]~~

9 Conclusion

Presented a framework for modeling response prediction in display advertising.
advantages over the alternatives:

- simplicity:
  - easy to implement
  - trivial to update
  - lightweight to use on real servers
- scalability :
  - easy to parallelize map-reduce architecture
- efficiency :
  - as good better than the state-of-the-art alternatives

Conducted on data from two large (and distinct) display advertising companies