Chapter 11

Look-alike Modeling for Conversion-Oriented User-Targeting of Ads
using Associative Classification

With the online advertising world constantly becoming more competitive but at the same time generating more revenue and expanding to new markets, internet companies are trying to create new user-targeting methods by which advertisers can get the best returns on their ad-spending. From rule-based targeting to Behavioral Targeting (BT), various targeting methods have been in use according to advertisers’ requirements.

As opposed to offline advertising, online advertising offers significantly finer granularity, which has been leveraged in state-of-the-art targeting methods, like Behavioral Targeting (BT). But, there is still a lot of scope for creating new targeting methods, especially those which create models which are customized according to each advertiser’s requirements and each campaign’s characteristics. Recent improvements in the scalability of software systems, especially Map-Reduce Systems like Hadoop, make the possibility of creating such customized targeting models more realistic. And, our efforts on Look-alike Modeling are a step in that direction.

The advertising process involves labeling users as those who have converted (converters) and those who have not (non-converters). An advertiser’s ad campaign $C$ emits labels for users as belonging to the set of users who either viewed the ad and then converted (targeted converters), viewed the ad but did not converted (targeted non-converters), or did not view the ad at all (untargeted users). In Look-alike Modeling given data about converters and non-converters obtained from advertisers, we would like to train models automatically for each ad campaign. Such custom models would help target more users which are similar to the set of converters the advertiser provides. Moreover, any custom model must integrate into the existing campaign and aim to complement the existing campaign by looking for potential converters.

Look-alike Modeling leverages historical user behavior, such as ad clicks and views, page views, search queries and clicks, to select the most relevant ads to display. Behavioral data about campaigns is generally very large in scale, with each campaign having around 0.5 million targeted users and 1 million or more features. Thus, these datasets are not only highly sparse, but extremely skewed (towards the negative class, i.e. non-converters) too. Of the $0.5$ million targeted users per campaign, the number of converters is extremely small as the typical conversion-rate is far below $10^{-4}$ valid conversions per impression. The rest of the users are non-converters.
Given a campaign related to a product, e.g. a cellular phone product \( C \), the motivation behind Look-alike Modeling is to find users who are potential converters, \( i.e. \) those users who are most likely to buy \( C \). The past user-behavior along with the conversion-related information regarding users is used to train models. Thus, the targeting model is very fine-grained in that it is customized to predict potential converters for campaign of product \( C \). This is distinctly different from BT where the goal is to predict a user’s CTR for coarse-grained categories like “Technology, Consumer Electronics, Cellular Telephones.”

Campaigns with very few conversions are called as tail campaigns, and those with many conversions are called head campaigns. A well-grounded statistical model should predict the conversion probability of a user for a particular ad from user behavior. Training a traditional Associative Classifier (even with high support values) based on such frequent itemsets is not practical, because the feature space is very large (around \( 10^6 \) features) and datasets are big (around 500K records). We propose a Hadoop-based Associative Classification approach \([\text{MRH}^*11]\) to Look-alike Modeling for tail campaigns. Pairs of features are used to derive rules to build a Rule-based Associative Classifier. The top \( k \) rules, sorted by frequency-weighted log-likelihood ratio (F-LLR), are then be applied to any test record to score it.

### 11.1 User Modeling

A main component of User Modeling is each user’s history, which is the sequence of events triggered by that particular user. For extracting features, we use events that occurred prior to the target event (conversion in our case). 14-day and seven-day window periods, before the trigger date (date when targeting is to be done), are used for training and scoring respectively. The tasks necessary to train a model are depicted in Fig. [11.1] Targets in Look-alike Modeling are generally extracted from beacons. Beacons are events which are fired when a user visits a particular page. Target collection is automated as the beacon events arrive through when users interact with the advertiser’s site. Although we restrict our discussion to beacon targets, there are other types. The class labels for training are derived from targets.

![Figure 11.1 Tasks required for training a model](image)

User event histories are extracted from server logs. User histories are then processed independently in a sequence of steps: labeling, feature extraction, and training set creation. Because the steps operate
on records independently, the data processing scales up linearly with the number of machines used such as in a Hadoop cluster. The labeling step attributes the target event to the appropriate time-stamp and determines the class labels.

A challenging aspect of target extraction is attribution, which is to determine when the event of interest could have been triggered. For example, the conversion event occurs at time $t_1$. But, the user converts immediately after viewing the ad, at $t_0 < t_1$. The time at which the event occurred is generally different from $t_0$, the time at which the event triggered. The sequence of events leading to the target event is: view the ad, click on the ad, visit the advertiser’s landing page, and then convert. Though we train the model to predict conversions, the appropriate trigger time is before the ad view-at some time $t$ ($\exists t < t_0 < t_1$). There is no use of targeting the user with an ad just prior to the conversion at $t_1$, because he has already seen the ad at $t_0$. Thus, attribution must determine the state of the user at the moment when showing an ad could have triggered interest which eventually leads to conversion.

A user profile is a sequence of events triggered by that particular user. Initially, the user starts as a blank slate. We denote a user $u \in U$ as an ordered sequence of events as illustrated in Eq. 11.1.

$$u = \langle e_1 = v_1, \ldots, e_n = v_n \rangle$$

where each $e_i$ denotes the event triggered by the user and $v_i$ denotes the corresponding intensity (no of time event was triggered). The true state of the user $s$ can be inferred using the sequence of observed events $\langle e_1 = v_1, \ldots, e_n = v_n \rangle$. This state is used for user-modeling and conversion-prediction.

11.1.1 Events and Features

The features used in the modeling process are derived from the events in a user’s profile to construct a feature vector for the user. The different event types and their corresponding events are:

- **Categorized Pages visited**: Pages are clustered based on navigational patterns. The feature is the id of the cluster and the category of the page, from an existing page categorizer.

- **Actual Pages visited**: These are actual pages visited by a user. The feature id is the page id of the page visited.

- **Actual Search queries**: Searches issued, clicks on search links, clicks on search advertising links. Features are the event type, the query, and the unigrams in the query.

- **Categorized Search queries**: Search query terms are categorized according to a hierarchical taxonomy. The feature is the category of query.

- **Actual Graphical Ads**: Views and clicks on ads. The feature is the event type and the id of the ad.

- **Categorized Graphical Ads**: Ads are clustered based on navigational patterns. Views and clicks on ads constitute a feature. The feature is the id of the cluster and the category of the ad, from an existing ad categorizer.
The value for each of these features is the number of times (intensity) a user has triggered the corresponding (visited a page, searched for a query, or clicked an ad) in the train or score window, depending on whether training or scoring (testing) is being done. An example feature and its value (intensity) is illustrated in Fig. 11.2. The user visits the Yahoo! Movies page five times during the train/score window period. This translates to a feature indicating a categorized page view on the movie category with a value of five.

![User visits Yahoo! Movies five times during train/score window period](image)

**Figure 11.2** Example feature

### 11.2 Look-alike Modeling

A user’s history is a sequence of events. Feature extraction uses events that occurred prior to conversion. 14-day and seven-day window periods are used for training and scoring respectively. The class labels in the training and test sets are conversions. The features are user activities preceding the conversion, and pertain to different event types: page views, search queries and clicks, and graphical ad views and clicks. If a user performs any one of these actions during the train/score window period, the corresponding feature is added to the user’s profile, with a value of one. e.g. `pageView_yahooMovie = 1`, indicates that the user visited the Yahoo! Movies page in the train/score window period.

Building targeting models for conversions is difficult for several reasons:

- Highly imbalanced large data. Although the data size is large, the typical conversion-rate is far below $10^{-4}$ valid conversions per impression so the number of positive examples is very small.

- Impressions cost money for the advertisers. Constructing an ad campaign solely to collect training data is very expensive and infeasible.

- Extremely high feature dimension. Event-based features are a very large feature space, well over $10^5$ features per model (even $10^6$ features for a few models). This feature dimensionality is greatly reduced using our feature extraction method.
Training a traditional Associative Classifier \cite{Tha07} (even for high support values) based on such frequent itemsets is not practical, because the feature space is very large (around $10^6$ features) and datasets are big (around 500K records). But, associations between features in the user-targeting domain do exist, and are exploited by us to create Look-alike models using a Hadoop-based algorithm. For the $O()$ analysis below, $n$ is the average number of features per train/test example and $|D|$ is the train set size. We enumerate all pairs of features in the training set which occur in at least five positive-class records, i.e. conversion-oriented records (Mapper–Algorithm 19 – $O(|D| \times n^2)$ complexity). The pairs of features are modeled as Associative Classification rules of the form $X \rightarrow y, A$, where $X$ is the pair of features, $y$ is the class, and $A$ is the affinity of $X$ towards $y$ measured using an information metric – Frequency-weighted LLR (F-LLR) in our case.

Individual features by themselves can also be $X$, i.e. give rise to rules on their own. But, such rules are very few as compared to those rules derived from pairs of features. For all rules $y$ is the positive class, i.e. conversion, as the main goal of the rules is to show how much affinity (gauged using F-LLR) the pairs of features, i.e. $X$ have towards conversion. F-LLR (a new information we propose) of a feature $f$ is F-LLR $= P(f) \times \log \left( \frac{P(f|\text{conversion})}{P(f|\text{non-conversion})} \right)$. The F-LLR for $f_{ij}$ is then calculated in the same manner as that for an individual feature (Reducer–Algorithm 20 – $O(n)$ complexity).

The rules are then sorted in descending order by their respective F-LLRs, and the top $k$ rules are applied on a test record $r$ in order to score it (Algorithm 21 – $O(n^2)$ complexity). For each rule if there is a match with $r$, then the product of the F-LLR of the rule and the minimum value, in $r$, of the features in the precedent of the rule is added to the score. We get the final score for $r$ after all the top $k$ rules have been applied to $r$. The Associative Classification rules model the affinity (association) of each feature/feature-pair towards conversion. This affinity, in the form of F-LLR, is used for classification.

In practice, it is also desired to refresh models often, given the dynamic nature of user behavior and the volatility of ads and pages. Constant refreshing of models would also help find more users who have high probability of converting by taking into account previously predicted converters, in addition to the seed-list of converters provided by the advertiser.

\section{Experimental Results}

We use three baseline models, namely Random Targeting, Linear SVM-based \cite{BHRP10} and GBDT-based Look-alike Modeling. We have used lifts ($\text{lift} = \frac{\text{new model metric} - \text{baseline metric}}{\text{baseline metric}}$) of two metrics for the experimental analysis. The metrics are the conversion rate (at 10\% reach – typical operating point) and the AUC. We have evaluated the Associative Classification-based Look-alike Modeling along with the SVM and GBDT-based methods on two pilot campaigns (300K records each, one record per user) for a real advertiser. The training and scoring window periods were 14 and seven days respectively. Campaign $C_1$ has very few positive class (conversion) examples in the training set, and is a typical tail campaign. On the other hand, campaign $C_2$ is a typical head campaign, and has many positive class
for each record \( r \in D \) do
conv = target (class) of \( r \)
for each feature \( f_i \in r \) do
\( v_i \) = value of \( f_i \)
if conv = 1 then
print \( f_i \) \( \backslash t \ v_i \) | conversion
else
print \( f_i \) \( \backslash t \ v_i \) | non-conversion
end if
\( \text{feature}[i] = f_i \)
\( \text{value}[f_i] = v_i \)
end for
\( \text{for} \{ i = 0; i < \text{feature}.\text{sizeof}() - 1; i + + \} \text{ do} \)
\( \text{feat}_1 = \text{feature}[i] \)
\( \text{value}_1 = \text{value}[\text{feat}_1] \)
\( \text{for} \{ j = j + 1; j < \text{feature}.\text{sizeof}(); j + + \} \text{ do} \)
\( \text{feat}_2 = \text{feature}[j] \)
\( \text{value}_2 = \text{value}[\text{feat}_2] \)
if \( \text{value}_2 < \text{value}_1 \) then
\( \text{value}\_\text{pair} = \text{value}_2 \)
else
\( \text{value}\_\text{pair} = \text{value}_1 \)
end if
if conv = 1 then
print \( \text{feat}_1, \text{feat}_2 \) \( \backslash t \ \text{value}\_\text{pair} \) | conversion
else
print \( \text{feat}_1, \text{feat}_2 \) \( \backslash t \ \text{value}\_\text{pair} \) | nonconversion
end if
end for
end for

Algorithm 19: Training (Mapper)

examples in the training set. For Campaign \( C_1 \) (Table 11.1), Associative Classification-based Look-alike Modeling has a lift (conversion) of 82% as compared to Random Targeting, 301% as compared to GBDT-based Look-alike Modeling, and 100% as compared to SVM-based Look-alike Modeling. Moreover, the lifts in AUC of our approach as compared to GBDT-based and SVM-based Look-alike Modeling techniques are 2% and 11% respectively. For Campaign \( C_2 \) (Table 11.2), our approach as compared to Random Targeting and SVM-based Look-alike Modeling has positive lifts (conversion) of 48% and 12% and lift of -40% as compared to GBDT-based Look-alike Modeling. The AUC lifts of our approach as compared to GBDT-based and SVM-based Look-alike Modeling techniques are -14% and -6% respectively.

1: \textbf{for} each feature or feature pair \( f \) \textbf{do}
2: calculate \( \text{freq}\_\text{conv}_f, \text{freq}\_\text{nonconv}_f \), and F-LLR
3: print \( f \rightarrow \text{conversion}, \text{F-LLR} \)
4: \textbf{end for}

Algorithm 20: Training (Reducer)
Algorithm 21: Scoring

From these results we see that our approach performs very well with respect to all the three baselines for \( C_1 \) (a tail campaign) according to both metrics, lift in conversion rate and lift in AUC. But, for \( C_2 \) (a head campaign), the performance of our approach with respect to the three baselines is comparable. Our approach to Look-alike Modeling works very well in case of tail campaigns, because it leverages the affinity the Associative Classification rules have towards the positive class, even though there are very few positive class training examples. SVM and GBDT expect more balanced train sets, which makes modeling hard on tail campaigns. We are also conducting experiments on a more extensive set of campaigns.

**Table 11.1** Results for Campaign \( C_1 \)

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Lift (Conversion Rate)</th>
<th>Lift (AUC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Targeting</td>
<td>82%</td>
<td>–</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>301%</td>
<td>11%</td>
</tr>
<tr>
<td>GBDT</td>
<td>100%</td>
<td>2%</td>
</tr>
</tbody>
</table>

**Table 11.2** Results for Campaign \( C_2 \)

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Lift (Conversion Rate)</th>
<th>Lift (AUC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Targeting</td>
<td>48%</td>
<td>–</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>-12%</td>
<td>-6%</td>
</tr>
<tr>
<td>GBDT</td>
<td>-40%</td>
<td>-14%</td>
</tr>
</tbody>
</table>

11.4 Summary of Associative Classification-based approach to Look-alike Modeling

Online advertising world is constantly becoming more competitive, with advertisers now wanting not only to optimize for CTR, but also for conversions, which is their ultimate goal. Custom campaign-based Look-alike models, that we present in this thesis can be used to target users with conversion as the goal, not just clicks. Training custom Look-alike models for advertisers is a challenging problem. Advertisers
want to optimize for conversions which typically occur at a rate on the order of one conversion for every 1M impressions. With beaconing by advertisers, we are able to collect training data while the ad campaign is running, automatically train models, and score all users to find the best users to target.

We present an Associative Classification-based approach to Look-alike Modeling. This approach helps a lot in dealing with the highly skewed and sparse datasets that one faces when training targeting models for tail campaigns based on conversions, which are extremely rare events. The characteristics of our approach are reflected in our results. We obtained very good results for tail campaigns. Our approach seems particularly effective for datasets which have very few positive examples in the training set (tail campaigns).