A Feature-Pair-based Associative Classification Approach to Look-alike Modeling for Conversion-Oriented User-Targeting in Tail Campaigns

Ashish Mangalampalli
IIIT, Hyderabad, India
ashish_m@research.iiit.ac.in

Adwait Ratnaparkhi
33Across Inc., Sunnyvale, CA
adwait.ratnaparkhi@33across.com

Andrew O. Hatch
Yahoo! Labs, Sunnyvale, CA
aohatch@yahoo-inc.com

Abraham Bagherjeiran
Yahoo! Labs, Sunnyvale, CA
abagher@yahoo-inc.com

Rajesh Parekh
Groupon Inc., Palo Alto, CA
rparekh@ieee.org

Vikram Pudi
IIIT, Hyderabad, India
vikram@iiit.ac.in

ABSTRACT

Online advertising offers significantly finer granularity, which has been leveraged in state-of-the-art targeting methods, like Behavioral Targeting (BT). Such methods have been further complemented by recent work in Look-alike Modeling (LAM) which helps in creating models which are customized according to each advertiser's requirements and each campaign's characteristics, and which show ads to users who are most likely to convert on them, not just click them. 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 who are similar to the set of converters the advertiser provides. The advertisers get more freedom to define their preferred sets of users which should be used as a basis to build custom targeting models.

In behavioral data, the number of conversions (positive class) per campaign is very small (conversions per impression for the advertisers in our data set are much less than 10^{-4}), giving rise to a highly skewed training dataset, which has most records pertaining to the negative class. Campaigns with very few conversions are called as tail campaigns, and those with many conversions are called head campaigns. Creation of Look-alike Models for tail campaigns is very challenging and tricky using popular classifiers like Linear SVM and GBDT, because of the very few number of positive class examples such campaigns contain. In this paper, we present an Associative Classification (AC) approach to LAM for tail campaigns. Pairs of features are used to derive rules to build a Rule-based Associative Classifier, with the rules being sorted by frequency-weighted log-likelihood ratio (F-LLR). The top k rules, sorted by F-LLR, are then applied to any test record to score it. Individual features can also form rules by themselves, though the number of such rules in the top k rules and the whole rule-set is very small. Our algorithm is based on Hadoop, and is thus very efficient in terms of speed.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Applications—*Data mining*

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0637-9/11/03.

General Terms

Algorithms, Experimentation

1. 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.

Training a traditional Associative Classifier [2] (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 1 – $O(|D| \times n^2)$ complexity). The pairs of features are modeled as AC 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 $F\text{-}LLR = P(f) \times log\left(\frac{P(f|conversion)}{P(f|non-conversion)}\right)$. $P(f_{ij}) = \frac{\sum_D v^r_{ij}}{|D|}$ gives the probability (in D) and $v^r_{ij} = min(v^r_i, v^r_j)$ gives the value (in record r) of a feature-pair f_{ij} (v^r_i and v^r_j are values of features f_i and f_j in r). The definition for

normal LLR is $LLR = log\left(\frac{P(f|conversion)}{P(f|non-conversion)}\right)$. The F-LLR for f_{ij} is then calculated in the same manner as that for an individual feature (Reducer–Algorithm 2 – $O(n)$ 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 3 – $O(n^2)$ 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 AC rules model the affinity (association) of each feature/feature-pair towards conversion. This affinity, in the form of F-LLR, is used for classification.

2. EXPERIMENTAL RESULTS

We use three baseline models, namely Random Targeting, Linear SVM-based [1] and GBDT-based LAM. We have used lifts ($lift = \frac{new\_model\_metric - baseline\_metric}{baseline\_metric}$) 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 AC-based LAM 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 examples in the training set. For Campaign C_1 (Table 1), AC-based LAM has a lift (conversion) of 82% as compared to Random Targeting, 301% as compared to GBDT-based LAM, and 100% as compared to SVM-based LAM. Moreover, the lifts in AUC of our approach as compared to GBDT-based and SVM-based LAM techniques are 2% and 11% respectively. For Campaign C_2 (Table 2), our approach as compared to Random Targeting and SVM-based LAM has positive lifts (conversion) of 48% and 12% and lift of -40% as compared to GBDT-based LAM. The AUC lifts of our approach as compared to GBDT-based and SVM-based LAM techniques are -14% and -6% respectively.

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 AC 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.

3. REFERENCES

[1] A. Bagherjeiran, A. O. Hatch, A. Ratnaparkhi, and R. Parekh. Large-scale customized models for advertisers. In *ICDM Workshops*, pages 1029–1036, . 2010.

[2] F. A. Thabtah. A review of associative classification mining. *Knowledge Eng. Review*, 22(1):37–65, 2007.

Algorithm 1 Training (Mapper)

```
for each record r ∈ D do
  conv = target (class) of r
  for each feature f_i ∈ r do
    v_i = value of f_i
    if conv = 1 then
      print f_i \t v_i|conversion
    else
      print f_i \t v_i|non-conversion
    end if
    feature[i] = f_i
    value[f_i] = v_i
  end for
  for {i = 0; i < feature[].sizeof() − 1; i++} do
    feat_1 = feature[i]
    value_1 = value[feat_1]
    for {j = j + 1; j < feature[].sizeof(); j++} do
      feat_2 = feature[j]
      value_2 = value[feat_2]
      if value_2 < value_1 then
        value_pair = value_2
      else
        value_pair = value_1
      end if
      if conv = 1 then
        print feat_1, feat_2 \t value_pair|conversion
      else
        print feat_1, feat_2 \t value_pair|nonconversion
      end if
    end for
  end for
end for
```

Algorithm 2 Training (Reducer)

```
for each feature or feature pair f do
  calculate freq_conv_f, freq_nonconv_f, and F-LLR
  print f →conversion, F-LLR
end for
```

Algorithm 3 Scoring

```
score = 0
for each rule r ∈ top k rules do
  F-LLR_r = F-LLR of r
  min_value = ∞
  for each feature f_i ∈ precedent of r do
    v_i = value of f_i
    if v_i < min_value then
      min_value = v_i
    end if
  end for
  score+ = min_value × F-LLR_r
end for
```

Table 1: Results for Campaign C_1

Baseline	Lift (Conversion Rate)	Lift (AUC)
Random Targeting	82%	–
Linear SVM	301%	11%
GBDT	100%	2%

Table 2: Results for Campaign C_2

Baseline	Lift (Conversion Rate)	Lift (AUC)
Random Targeting	48%	–
Linear SVM	-12%	-6%
GBDT	-40%	-14%