Information Graph Model and Application to Online Advertising

Marina Danilevsky[*]
Department of Computer Science
University of Illinois, Urbana-Champaign
Urbana, IL, United States
danilev1@illinois.edu

Eunyee Koh
Adobe Research
321 Park Ave
San Jose, CA, United States
eunyee@adobe.com

ABSTRACT

We present an algorithm which adapts a graph-based ranking model to the context of the problem of improving the process of serving advertisements to users. We transform the ad-based clickstream data into a heterogeneous graph model which respects differences in feature types (e.g. geolocation features, or browser-history features). The heterogeneous network model generates meaningful rankings of features which are predictive for each ad, as demonstrated by our classifier's performance. We also discuss how, in addition to serving as the basis for a classifier, this model may also provide an informative view of the data, which is not possible with black-box approaches, and which therefore makes it very suitable to the problem space of targeted ad serving.

Categories and Subject Descriptors

H.4.M [**Information Systems Application**]: Miscellaneous

Keywords

Targeted Advertising, Information Network, Feature Ranking

1. INTRODUCTION

Digital marketers run many marketing campaigns in order to maximize their website revenue, which is sometimes measured in terms of click-through-rate (CTR). Each campaign consists of many different kinds of offers that may be shown to website visitors. The goal of the campaign is to optimize ad offer selection by targeting the visitors based on any available information, to maximize both user experience and CTR. Figure 1 shows an example of an offer displayed to visitors to Adobe.com, the homepage of a computer software company.

A variety of such information as visitors' geolocation, time-related attributes, browsing history, etc, are collected for targeted ad serving. There are many approaches to learning user behavior models from this data, treating each visitor record as one single feature vector. However, most previous work has not attempted to preserve the implicit structures and relationships of different data types, which may carry additional useful information. Recent work by Xu et al. [10] exploits the interaction of different data types by developing a multiview hierarchical bayesian regression model to improve CTR prediction accuracy. However, this approach can only handle two different data types. Our framework uses a graph-based classifier which works on a heterogeneous information network [5, 6, 9] and is therefore flexible enough to handle multiple data types.

[*]This work was developed while the author was at Adobe Research.

UEO'13 November 1, 2013, San Francisco, CA, USA.
Copyright 2013 ACM 978-1-4503-2263-8/13/10
http://dx.doi.org/10.1145/2512875.2512878 ...$15.00.

Adobe® Creative Cloud™ Student and Teacher Edition
Get Creative Cloud for US$19.99/mo.
That's 60% off.
Learn more
Join by 6/25 to lock in this price for a year.
See all special offers ›

Figure 1: An example of a targeted ad offered on Adobe.com

In this paper, we present our Information Graph Model (IGM), a work-in-progress which adapts a graph-based ranking method to the problem of improving the process of serving advertisements to users. We train a classification model to predict user response, treating ads as class labels and leveraging different feature types in a graph structure. We show that our model can generate meaningful rankings of features which are predictive for each class label, as demonstrated by the classifier's performance. Additionally, this model can provide an informative view of the data, which is not possible with black-box approaches, and which therefore makes it very suitable to the problem space of targeted ad serving.

2. IGM ALGORITHM

The IGM algorithm transforms an ad-based clickstream dataset into a heterogeneous information network with a star schema [9]. Once in this form, we apply a graph-based ranking method [5] to construct a model of the dataset. Finally, we construct a classifier to be used for future records.

2.1 Network Construction and Ranking

2.1.1 Node Definition

Within this work, we assume that we are working with an ad-based clickstream dataset, consisting of a set of records, where each record represents a user and contains a number of feature variables, information about an ad that was served, and binary clickthrough information, which labels the record as being positive or negative depending on whether or not the user clicked on the ad. We further assume that during some preprocessing stage, each feature variable has been categorized into *B* categories.

The first step is to transform this dataset into an information network structure. A information network is a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathcal{W} \rangle$ containing m different types of nodes where $\mathcal{V} = \bigcup_{i=1}^{m} \mathcal{X}_i$, $\mathcal{E}$ is the set of links between any two data objects of $\mathcal{V}$, and $\mathcal{W}$ is the set of weight values on the links. If $m = 1$, we refer to $\mathcal{G}$ as a *homogeneous information network*; otherwise, if $m \geq 2$, $\mathcal{G}$ is known as a *heterogeneous information network*.

The clickstream dataset contains m different types of features (e.g., geolocation features). We therefore transform each feature F into *B* binary features $F_1 \ldots F_B$, where the value of a categorized feature F_b is 1 for a given record if that record originally exhibited the value *b* for the feature F. Each feature has also been labeled as belonging to a specific feature type (e.g. geolocation feature type, URL parameter values feature type, etc.).

We then transform these binary features into nodes in our network, where each node thus represents one category F_n of a particular feature F. Thus, each original feature F is transformed into *B* nodes. However, if a particular binary feature does not occur in any records with a positive label (user clicked on the ad), we cannot learn much from this feature, and so we do not construct this node. Two nodes can then be thought of as co-occurring with respect to a given record if the binary features from which the nodes were constructed both had a value of 1 (click) for that record.

We construct our network to follow a *star schema* [9], where one type of node serves as the central (star) objects, and the rest of the nodes are attributes (a link may exist only between a central node and an attribute node, but never between central objects, or between aattribute nodes). Considered as a classification problem, the central object of a star schema can be thought of as the label object type, and every attribute object as a feature. We treat the ads served for each record as proxies for the latent class label for that record. The set of potential ads is transformed into a set of central object nodes, and each feature is transformed into an attribute node. Although each central object node theoretically represents a class, we do not place this as a strict constraint on the network, and instead allow each central object to have a distribution over all classes, with a bias towards the class it *should* represent.

Figure 2 illustrates a sample schema of one central object node - the ad - and 7 attribute object nodes - the features - with different feature types denoted by the node shape and color.

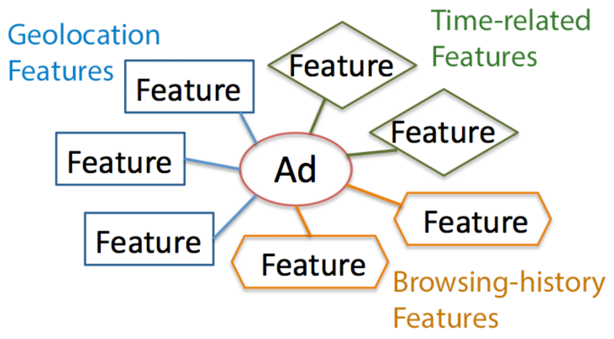

Figure 2: IGM Sample Schema: Three kinds of example feature types presented with different node shape and color

2.1.2 *Edge Weight Definition*

The next step is to specify the edges (and edge weights) of the network. In a heterogeneous information network $\mathcal{G}$, with m object types, the links between two data objects of types $\mathcal{X}_i$ and $\mathcal{X}_j$ may be represented by a relationship graph $\mathcal{G}_{ij}, i, j \in \{1, \ldots, m\}$. This allows us to differentiate among the relationships between different pairs of object types. Let $\mathbf{R}_{ij}$ be an $n_i \times n_j$ relation matrix corresponding to graph $\mathcal{G}_{ij}$, i.e., $\mathbf{R}_{ij}$ represents the relationship between objects of type $\mathcal{X}_i$ and objects of type $\mathcal{X}_j$ (in a general network schema, it is possible that $i = j$). In this way, each heterogeneous network $\mathcal{G}$ can be mathematically represented by a set of relation matrices $\mathcal{G} = \{\mathbf{R}_{ij}\}_{i,j=1}^{m}$. Note that since our network is a star schema, the only edges that exist are between a central object and an attribute object, as shown in Figure 2.

The weight on the link $\langle x_{ip}, x_{jq} \rangle$ is thus defined by the value of the p-th row and q-th column of $\mathbf{R}_{ij}$, denoted as $R_{ij,pq}$. The general definition for $R_{ij,pq}$ is:

$$R_{ij,pq} = \begin{cases} f(x_{ip}, x_{jq}) & \text{if link } \langle x_{ip}, x_{jq} \rangle \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

The simplest definition of *f*, which represents the similarity between any two objects is $f(x_{ip}, x_{jq}) = 1$, resulting in a binary weighted, undirected graph. Classification algorithms use a variety of similarity metrics, such as co-occurrence, lift, cosine similarity, correlation, and others. We define *f* in our framework to be:

$$f(x_{ip}, x_{jq}) = \begin{cases} Cor(x_{ip}, x_{jq}) & \text{if p < 0.05} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Here, p is the p-value for the hypothesis that the association between x_{ip} and x_{jq} is not equal to 0. Filtering using the p value is necessary, since a weak correlation would only introduce noise into our network, without contributing much additional information. We also keep the relation graph undirected, so that $\mathbf{R}_{ij} = \mathbf{R}_{ji}^{T}$.

2.1.3 *Graph-Based Ranking*

Having appropriately defined the nodes, edges, and edge weights for our heterogeneous information network $\mathcal{G}$, we are able to directly follow the graph-based ranking method in [5]. The input to the method is a heterogeneous network with a star schema, such as we have defined. The algorithm iterates over classifying the ads and features, and ranking features within each class, while respecting the differences between feature types (i.e., geolocation features and browsing-history features would not be directly compared). The output is a class distribution for each node, representing the probability of a node's membership within each class, and a ranking distribution, representing the node's rank, with respect to other nodes of the same type, within each class.

For our purposes, it is important to note that the graph-based ranking method also defines a set of weights $\lambda_i, \ i \in [1, m]$, where each λ_i represents the relative importance of node type i. This is valuable for our problem space, since it allows for different types of features to naturally contribute more or less to the model (e.g., customer-defined features may be of a high quality, and should be trusted more than automatically acquired features.) The values of λ may be defined by a user, or learned using some information criterion such as BIC.

2.2 Classifier Construction

The final step is to use the converged ranking distributions to construct a classifier. In this heterogeneous data framework, it is not clear how to combine ranking scores from feature objects of different types. For instance, if feature type $\mathcal{X}_i$ has many more features than feature type $\mathcal{X}_j$, the highest ranked feature object of type

$\mathcal{X}_i$ in class k is almost certain to have a much lower ranking score than the highest ranked feature object of type $\mathcal{X}_j$ in class k. Similarly, the differences in rank scores cannot be compared between the different feature type, since the difference in rank score between the top two objects of type $\mathcal{X}_j$ may be larger than the difference in rank score between the top 10 objects of type $\mathcal{X}_i$. It is therefore not reasonable to use each object's ranking score as its classifier score, (and a similar problem exists with using each object's posterior probability of which class it belongs to).

A good alternative is to transform the ranking distributions into inverse ranks, with respect to feature type and k. The inverse rank of each object x_{ip} of feature type $\mathcal{X}_i$ within class k is simply calculated as $\frac{1}{Rank(x_{ip}|\mathcal{X}_i,k)}$ where $\text{Rank}(x_{ip}|\mathcal{X}_i, k) = 1$ if x_{ip} is the top-ranked object among the objects of feature type $\mathcal{X}_i$ within class k; $\text{Rank}(x_{ip}|\mathcal{X}_i, k) = 2$ if x_{ip} is the second-ranked object; etc.

Recall that the weight λ_{i1} represents the information value of the feature type $\mathcal{X}_i$ (where $i \neq 1$). Therefore, the classifier score for object x_{ip} of feature type $\mathcal{X}_i$ within class k is calculated as:

$$Score(x_{ip}) = \frac{1}{Rank(x_{ip}|\mathcal{X}_i, k)} * \lambda_{i1} \tag{2}$$

This scoring schema does not require explicit feature object filtering, which could involve tuning multiple parameters for each feature type, and is therefore undesirable. Instead, the few top-ranked feature objects within each feature type generally determine the scoring for each class k in a natural way, resulting in an ensemble approach to scoring and classifying a new record. As this scoring schema is a 'positional' method [4] it has the advantage of being scalable, since it may be implemented in linear time.

In order to score a record, the classifier simply treats all of the classifier scores of the feature objects for a given class k as a vector, and performs element-wise multiplication between this scoring vector and the new record (lining up feature names, of course). The sum of the result is then treated as the score for the record for class k. This process is repeated K times until the record has a score vector of length K, representing the classifier's recommendation for each label. The recommended label is thus the one which corresponds to the highest score.

3. EXPERIMENTAL RESULTS

We implement our framework in R and experiment using a real world digital marketing dataset. We compare with several well-known approaches: Random Forest (*RF*) [1], Generalized Linear Model (*GLM*)[8], and Support Vector Machine (*SVM*) [3].[1]

3.1 Dataset Preprocessing

We obtained a marketing campaign dataset from a large, consumer-oriented manufacturing company, in order to conduct our experiments. The dataset consists of records from a ten-week period, from Jul 2^{nd} to Aug 17^{th} of 2012, with a marketing campaign consisting of 7 different ads and no positional bias [2], where every ad was shown in the same location. Each record in the dataset consists of user feature variables, the ad serving selection (which one of seven ads was served), and binary clickthrough information, which labels the record as being positive of negative.

An existing proprietary ad serving mechanism served 90% of the records. The mechanism also incorporates a random bucket to collect exploration clicks from 10% of traffic. We use this 10% randomly served dataset as testing data. We sampled records from the other 90% of the data to construct a subset with a uniform distribution of ad recommendations, which we use as our training dataset. As a result, our dataset consists of 84,035 training records and 13,434 testing records.

The dataset contains two types of features: continuous features, which are numeric and may be any value, and nominal, or categorical features, which are non-numeric (e.g. a geolocation variable.) We apply simple pre-processing on the dataset in order to prepare it for the classifiers. The pre-processing is based purely on feature frequency, instead of other more sophisticated information-theoretic measures, as simple, frequency-based metrics tend to empirically work better for sparse, noisy data. First, we discard all features which have only missing or null values, or which take on only one value. Next, we categorize each feature into 10 categories ($B = 10$), a commonly used technique. For continuous features, we run k-means clustering to find 10 buckets for the continuous values. For nominal features, we identify the 10 most common values, assign an id to each value, and drop the infrequent values.

The features in this dataset naturally split into 11 different categories. Some features represent automatically gathered information, such as geolocation. Other features represent various types of targeting or segmentation information constructed by the client. Incorporating this additional semantic knowledge results in the identification of reasonable feature types, and this information bolsters both the performance of IGM, and the interpretability of the constructed model. Due to the nature of our dataset, there were very few users who appeared multiple times. Therefore, we make the simplifying assumption that each record corresponds to a unique user, and do not consider user information in this work.

3.2 Evaluation Methodology

We next wish to evaluate whether our selected features are high quality, and would be useful in choosing which ad to serve in order to maximize CTR. We follow the offline evaluation process described in [7], since we do not have the kind of supervised setting common in traditional machine learning problems.

For each algorithm we choose a subset of records from the evaluation set, as follows: for each record, if the served advertisement is precisely the one that the given algorithm would choose (because that label had the highest score), we retain that record; otherwise we discard it. Given K possible ads to choose from, each record - which made a random choice of ad - is therefore retained with a probability of 1/K, independent of everything else. Therefore, for a sufficiently large initial evaluation set we are able to extract a subset that can act as a fairly reliable proxy for an evaluation of the algorithm performance on live data.

When evaluating each algorithm's ability to label the testing dataset, it is also necessary to allow the label option of "0", in addition to the existing options of $1 : K$. The "0" label is interpreted as the algorithm's claim that since there is no label for this record with a high enough score, there will be no click, regardless of which label is recommended (this user will not click on anything).

3.3 Evaluation

It is important to note that datasets derived from clickstreams will nearly always be very unbalanced, exhibiting mostly negative labels, with few positive labels. The goals of classifiers in this problem space must also be carefully considered, weighing the importance of correctly predicting many labels versus minimizing the number of prediction errors. We would argue that of the four base confusion matrix metrics, it is most important to a) increase the

[1] We use the R package *randomForest* (available at http://cran.r-project.org/web/packages/randomForest/index.html), the base R system function *glm()*, and the R package *e1071* (available at http://cran.r-project.org/web/packages/e1071/index.html) which interfaces with *libsvm* (available at http://www.csie.ntu.edu.tw/ cjlin/libsvm/.)

Table 1: Performance metrics: GLM, RF, SVM, IGM

Performance Metric	GLM	RF	SVM	IGM
Precision	0.514	0.471	0.318	0.304
Recall	0.247	0.546	**0.798**	**0.704**
Specificity	0.930	0.821	0.491	0.553
F_1 *Measure*	0.334	0.506	0.455	0.424
F_2 *Measure*	0.276	0.529	**0.613**	**0.557**

Table 2: Case Study Scenarios

Performance Metric	Original	CC	AA
Precision	0.304	0.300	0.311
Recall (Sensitivity)	**0.704**	**0.703**	0.684
Specificity	0.553	0.550	0.559

ability to correctly identify true positives, implying that the classifier interpreted the new record; and b) to decrease the predictions of false negatives, which imply that the classifier incorrectly thought the user would not click any ad, and therefore missed some important information.

On the other hand, every user must be served some ad, even if the classifier believes the user will not click any ad. Therefore, if the classifier predicts a false positive, there is no real problem since an ad is served, and the model will learn something for next time. And of course, if the classifier correctly predicts a true negative, it is likely that some other ad serving mechanism will be used (serve a random ad, serve an ad proportional to its CTR in the past, etc.), and the classifier will be validated as not having missed some important information. The conclusion to draw from these examples is that we should particularly value the Recall metric (TP/P), since this metric improves both when true positives are increased, and when false negatives are decreased. Thus, we are interested in the F_2 measure as a proxy for the intuition that Recall is somewhat more important than Precision.

Table 1 suggests that IGM performs somewhat comparably to SVM, on the presented metrics - IGM is somewhat better for Specificity, and slightly worse for everything else. GLM and RF perform well, especially on Precision and Specificity, but are definitely below SVM and IGM on Recall, and consequently on the F_2 measure.

For example, we note that GLM has a very low Recall value of 0.247. Roughly, this means that for every positively labeled record that GLM correctly predicts as positive, there will be 3 positively labeled records that GLM incorrectly predicts to be negative. GLM's Specificity value also reflects that GLM is very good at predicting negatives, but we do not value this ability as highly. Random Forest also suffers from a low Recall value, though not to the same degree as GLM.

We would argue that SVM and IGM are the best performing algorithms for the task of targeted ad serving compared to RF and GLM, specially for this marketing campaign. However, IGM has the capacity to provide the transparency to explore and interpret the reasons behind the classification. The ranking distributions contain all of the information ruling the classifier, and are easily interpretable. If a given feature is highly ranked within its feature type and for a given class, then this means that seeing that feature is a strong indicator for that class. We could therefore observe, for example, the rule that if a record's city is identified to be Seattle, the user is more likely to respond to ad 3 - which turns out to be an ad for umbrellas. Such detailed observations are a little complicated to identify in black-box algorithms such as SVM, where parameters of a solved model are difficult to interpret.

3.4 Case Study

As a toy simulation of incorporating real-world knowledge, we present two cases of explicitly choosing the values of λ for specific feature types. Using just two values, $\lambda_{high} = 0.2$ and $\lambda_{low} = 0.02$, we use our knowledge of the dataset split the feature types into automatically acquired (*AA*) features , and customer-created (*CC*) features. We then consider two scenarios. In the *CC* scenario, we assume that the customer is very savvy, and that the *CC* features should do better than the *AA* features. We set $\lambda = \lambda_{high}$ for all *CC* features, and $\lambda = \lambda_{low}$ for all *AA* features. In the *AA* scenario, we instead assume the customer constructs very useless features, and reverse the λ assignments. As seen Table 2 if our goal is to maximize recall, the *CC* scenario performs just as well as the default scenario, whereas the *AA* scenario yields a slightly poorer performance. We might therefore suggest that in the case of this dataset, if the goal is to maximize recall, the customer-created features should be considered to be more informative.

4. CONCLUSION

The goal of marketing professionals is to run successful online marketing campaigns. An important aspect of this goal is being able to interpret why marketing campaigns were (or were not) successful, in order to better understand customers' behavior. The IGM model provides both a classifier function, and a feature ranking for each feature type, which can be used to discover the features that the model found to be most informative (e.g., which geolocation features were informative? What about features related to browser history?). It is therefore possible to generate insight reports about online visitors' behavior information and targeting rules.

Going further in this direction, we would like to study IGM's effectiveness on additional real world datasets. We also want to study more about how the flexibility of model can be helpful with real-world use cases as IGM can easily incorporate domain knowledge such as the relative importance of different feature types.

5. REFERENCES

[1] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. 10.1023/A:1010933404324.

[2] Y. Chen and T. W. Yan. Position-normalized click prediction in search advertising. In *KDD*, pages 795–803, New York, NY, USA, 2012.

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. 10.1007/BF00994018.

[4] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, New York, NY, USA, 2001.

[5] M. Ji, J. Han, and M. Danilevsky. Ranking-based classification of heterogeneous information networks. In *KDD*, pages 1298–1306, New York, NY, USA, 2011.

[6] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In *ECML/PKDD (1)*, pages 570–586, 2010.

[7] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual- bandit approach to personalized news article recommendation. In *WWW*, pages 661–670, New York, NY, USA, 2010.

[8] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society, Series A, General*, 135:370–384, 1972.

[9] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD*, pages 797–806, 2009.

[10] T. Xu, R. Zhang, and Z. Guo. Multiview hierarchical bayesian regression model and application to online advertising. In *CIKM*, New York, NY, USA, 2012.