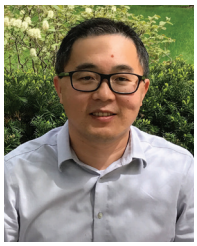
PREDICTING ADVERTISEMENT CLICKS USING DEEP NETWORKS:

Interpreting Deep Learning Models

Student Author

Karan Samel is a May 2017 graduate who studied computer science and applied statistics. Samel's research interests are in the field of machine learning and deep learning predictive models. Outside of research, he has worked with residence hall organizations and the Aerial Robotics Team. Samel plans to enter the job market of data science and machine learning and hopes to pursue a master's degree in the near future.

Mentors



Xiao Wang is an associate professor of statistics whose research focuses on nonparametric statistics, machine learning, and artificial intelligence. He received his PhD from University of Michigan. He has received both the Graduate Student Mentoring Award and the Team Award from the College of Science. He has served as associate editor for many top statistics journals such as Journal of the American Statistical Association and Technometrics.

Qiang Liu is an assistant professor of management at the Krannert School of Management, where he teaches digital and social media marketing. He has been recognized as a Krannert Outstanding and Distinguished Teacher for multiple years. He is an expert in marketing models, pharmaceutical and healthcare industry, digital marketing, and sharing economy. His research has been published in top journals such as *Marketing Science and Management Science*. He holds a BS in Information Management from Peking University and a PhD in Management from Cornell University.

http://dx.doi.org/10.5703/1288284316397

Abstract

Deep learning has achieved state-of-the-art results in a variety of tasks such as classifying images and driverless cars. In this paper, I used deep learning to understand consumer product interests. One of the main goals for advertisement agencies is to develop mathematical models to predict whether consumers will click on their advertisement. Achieving the highest click prediction rate means that these agencies can pay to place their online advertisements effectively to target people most interested in their product. Most existing approaches are based on logistic regression or regression tree models (Trofimov, Kornetova, & Topinskiy, 2012). The model based on deep learning will be discussed to predict the click rate. The data was from the iPinYou competition, where competitors are tasked to build a model that would achieve a high click through rate (CTR). iPinYou provides advertisement data from nine companies. For each instance in the data, various attributes of the person that the electronic advertisement was sent to were provided as well as if the person clicks on the advertisement. I started with exploratory data analysis by splitting data into different seasons, aggregating different advertisers, and cleaning and generating new attributes. I tested my predictive power using a convolutional neural net and a multiple layer perception model. It was shown that the deep learning models have a competitive predictive power and, at the same time, more interpretable for further analysis.

Samel, K. (2017). Predicting advertisement clicks using deep networks: Interpreting deep learning models. *Journal of Purdue Undergraduate Research, 7*, 50–56. https://doi.org/10.5703/1288284316397

Keywords

click through rate, click prediction, advertisement bidding, deep learning, neural networks, convolutional neural networks.

INTRODUCTION

Today, we are constantly sent advertisements and information in forms of e-mail, mail, and online advertisements. Usually, these advertisements are sent to everyone within a company's reach, whether or not users are interested in their content. At the present time, this marketing is getting more selective with developments in targeted advertising and content recommendations that provide advertisements that are more relevant to the user. Models to predict whether a user is going to click on an advertisement are built using regressions and decision trees. We will look into data provided by iPinYou, a demand side platform (DSP) that hosts advertisements, and build models for user advertisement clicks with competitive predictive power to current regression and tree methods. The two models I will introduce are the multilayer perceptron and the convolutional neural network.

Multilayer Perceptron

Multilayer perceptrons (MLP) use a set of weights and transformation functions to convert the data in the input layer to a prediction in the output or softmax layer. Since these models can virtually represent any function (Hornik, Stinchcombe, & White, 1989), I used it to represent a function to transform the features extracted from the data set into a prediction of a user click. The basic structure of this model (see Figure 1) involves taking in inputs from the data attributes and passing them through an activation function using a combination of the inputs and weights to produce an output. Many of these basic perceptron models can be joined to create a MLP by connecting the outputs of one Perceptron to the input of another to create a more expressive model.

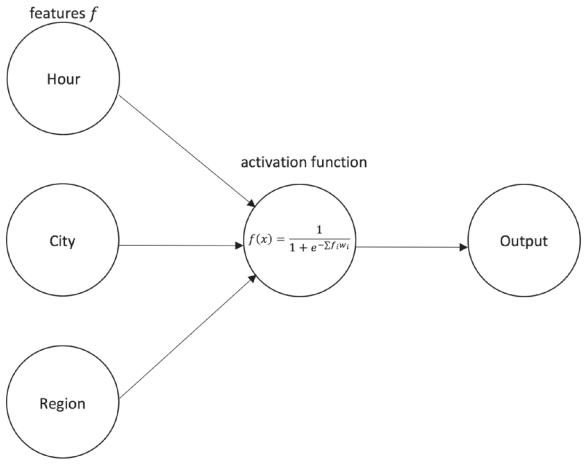


Figure 1. Simple perceptron mode with the input attributes f and weights w that result in the output.

Obtaining the final weights to make the prediction, if a user is clicking the advertisement, was done through backpropogation (Rumelhart, Hinton, & Williams, 1986). With enough examples and iterations through a model training set, the MLP had better predictions over time.

Convolutional Neural Networks

I also wanted to use convolutional neural networks (CNN), as they are known to detect and use features present in the data. This is done by taking the inputs, the data attributes, in a matrix grid arrangement and generating patterns that are applied to each section of the input matrix to calculate new attributes called feature maps (Simard, Steinkraus, & Platt, 2003). These feature maps indicate the presence of the pattern in each part of the input matrix, and sent to the next layer where more patterns are applied to create more feature maps and so on. Eventually, all these values of the resulting matrices are fed into a standard MLP, which uses the new attributes calculated through the multiple feature maps, to make a prediction (see Figure 2).

METHODOLOGY

Data Overview

When obtaining the files from iPinYou, I was provided the training data to train my models and testing data to see how well the models performed on data the models have not seen. I was given data logs for nine different advertisers for which iPinYou hosted advertisements. Each advertiser's log had attributes such as date, time, user IP, city, region, and the dimension of the advertisement of the targeted customer. Each entry also indicated whether or not the advertisement was clicked; the testing data had some extra attributes as well.

When I obtained this data, I had to prepare the data set in such a way that it could be used by my models. Some of the attributes in the logs were combined, hashed, alphanumeric, or simple were not present. The inputs for the models have to be numeric values, so the data had to be transformed and vectorized. Once the data was prepared, I built predictive MLPs and CNNs for each advertiser.

Data Parsing

There are a lot of potential attributes to use, and I started by parsing the useable attributes available. However, some of the values in the data were null or hashed by iPinYou before they put out the data, making the data hard to use. These values were removed from the attribute set.

My second task was the breakdown the remaining attributes into discrete and usable features. Some of these attributes included timestamps, which had to be broken down into individual components, and grouped targeted customer's information, which contained operating system and web browser information of the device where the advertisement was displayed.

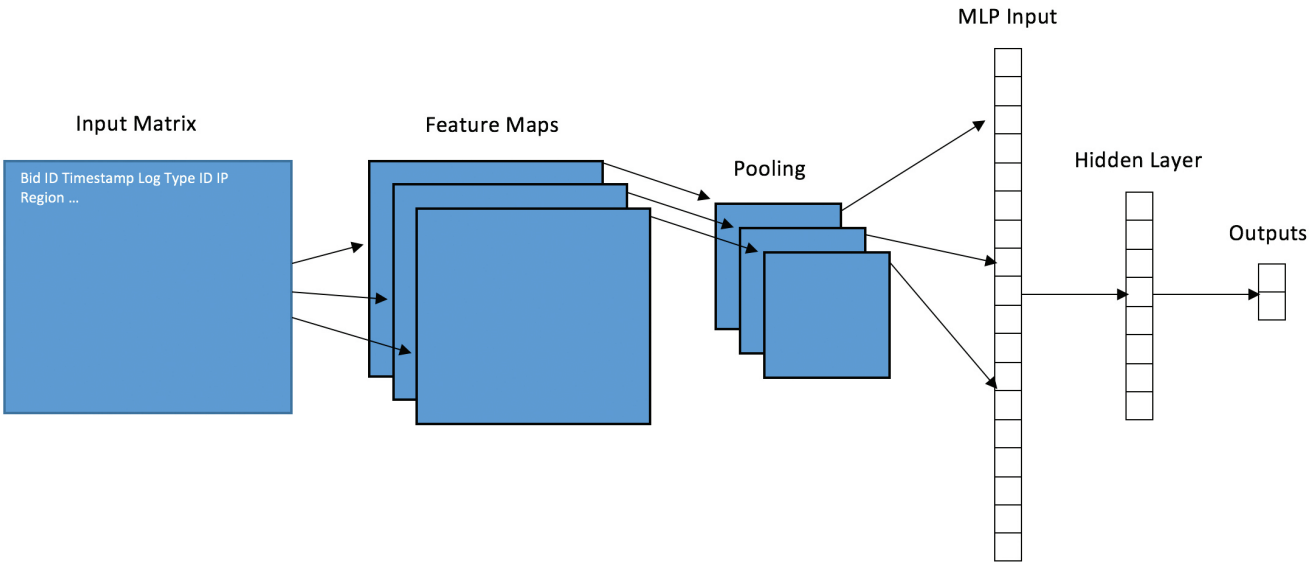


Figure 2. Basic CNN structure, with the input matrix containing the attributes, which filters are applied on to generate feature maps. These feature maps may be pooled to reduce the dimensionality of the model. Then, I fed the resulting matrices to the MLP.

Now that I had some more distinct features, I had to further encode some of the attributes to work with the deep learning models. For the targeted customer's city and region, I performed a one hot encoding, where I had a vector with the length of the attribute's unique values. For this vector, all the values were zero except for the one value that I observed in the current data row, which was indicated as active by a one in that position. This was similarly done for other discrete attributes, such as the operating system and web browser, since there were multiple values for those. In theory, I should have also done this for the different parts of the targeted customer's hashed values, since those are unique. I did not proceed with those due to considerations of the model complexity.

I wanted to change most of these values to a one hot encoding because, when I trained the models, I did not want specific values that represent the data attributes to have additional weight in our models just because they were labeled with a higher number in the original data creation process. Since all values are either zero or one, this mitigated the problem.

I also had to deal with high correlations between the features. Since these models had to take in unique features and attributes to generate weights and predictions, I wanted to minimize the features that may be represented with multiple attributes as it increases how the models would weight those similar attributes if they continued to reoccur. For example, city locations were always correlated with their respective regions in China, or common dimensions between the advertisement height and width were used as standard advertisement sizes (see Figure 3).

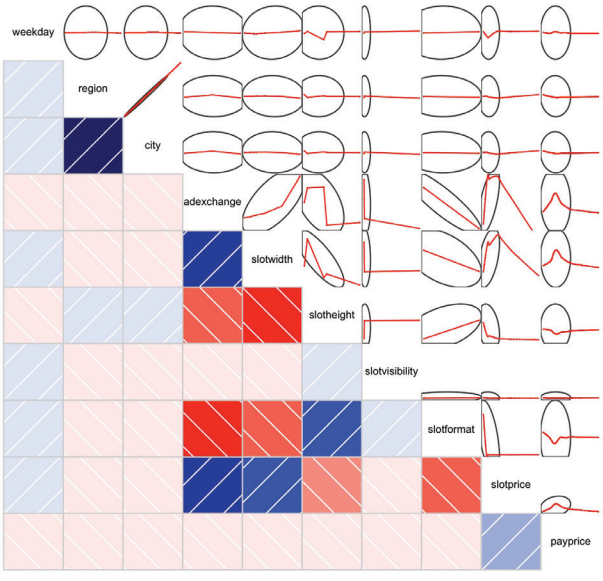


Figure 3. Example correlations for advertiser 1458, with red being positive and blue being negative correlation. To the right are the confidence ellipses.

I used the CNN architecture later on to leverage these correlations, but for the MLP I kept all attributes and parsed features that I made in order to observe performance given the number of attributes that I originally had to drop.

I then had to normalize all my input attributes as I wanted the models to differentiate between the values of the attributes and not consider attributes better if their raw value from the data is higher. This was done by subtracting the attribute with the mean attribute value and dividing the result by the attribute's standard deviation.

Data Selection

When training the models, another issue to overcome was how few positive click examples the training sets had. For example, in the dataset for advertiser 1458, one of the nine advertisers used for initial testing, only 0.08% of the training examples were positive. When training most models under these circumstances, it may take a long time for the model to converge, as these examples rarely come around in order to make updates to the model. Using the training set directly with the models provided poor results in terms of converging time.

To counter the poor performance, I performed a combination of oversampling and undersampling. This involved extracting all positive click data instances from the training data. I then undersampled the negative nonclick examples to about 200,000 instances in order to reduce the time it took to train. I then performed oversampling by repeatedly mixing the positive instances extracted with the undersampled negative instances to create a training data set.

Scoring Selection

With an artificial training set, I also had to determine how I would score a model. While the most common method is model accuracy given the true instance label of click versus nonclick, this was not advisable in the imbalanced data set. Given a base model that always predicted no click, which is the most common occurrence in the data, I would get a model accuracy of 99.92% for advertiser 1458. Any significant performance increases from this model would be hard to distinguish as I would have to look at the narrow window between 99.92% and 100%. A common way to score these models with unbalanced data is to use the area under the curve (AUC) of the receiver

operating characteristic (ROC). This method measures how a model performs by taking into account the true positive rate and the false positive rate. The true positive rate is the proportion of the positive examples correctly estimated as positive over all the positive data. The false positive rate indicates the proportion of the negative examples that were classified as positive over all negative data points. Then, these rates are computed at various thresholds to produce a curve, which I integrate. This gives us a better understanding if the positive examples attempted to be predicted along with the negative, with an AUC ROC score of 0.5 being the baseline for a random predictor (Aidos, Duin, & Fred, 2013).

MLP Trials

For MLP construction, I had to determine the form of my network. In other words, how many individual perceptrons I was going to use in each layer and how many layers would be present between the attribute inputs and the final prediction. A common technique is to keep a nonincreasing number of perceptrons in each subsequent layer (Zhang, Du, & Wang, 2016). Through trials, I observed that moving past one or two layers (called hidden, as they are not directly observed) of perceptrons provides diminishing returns in model predictive power. I settled with two layers to give the model enough expressivity to represent the underlying structure of the true click prediction model. I had a final count of 515 attributes, which I used to generate the first layer of perceptrons and built two more layers before the softmax layer, which output the probability of a click from a training example (see Table 1).

MLP Model	
Input Layer	1×515
Hidden Layer 1	515×400
Hidden Layer 2	400×400
Softmax Layer	400×2

Table 1. MLP model structure that included multiple layers before the final softmax layer, which outputs a probability for click and for no click.

CNN Trials

The CNN trials had to follow a similar procedure as the MLP trials. I had to determine a structure of the CNN and determine various parameters while running it. The CNN architecture was a new challenge regarding how to initialize the patterns and arrange the input data.

Typically, the pattern weights are initialized from the normal distribution, which works well when working with image and sound data to detect important features. Since the patterns were applied to the text data, I used the normal initialization but also experimented with uniformly initializing all pattern weights to 1, indicating that all values calculated in the convolution will be weighted equally. Using these two methods, I saw little difference in performance results and decided to adhere to former normal method for identifying significant combinations of features.

Originally, when I wanted to use CNN for click prediction, I relied on randomly shuffling the attributes in the input matrix. I also tried to subjectively arrange the attributes such that attributes were close to related attributes. Finally, I arranged the final attributes that covaried as I thought that running the patterns over related attributes would generate better attributes to use in the MLP.

When decided the structure of the CNN model, I proceeded with two convolution layers (see Table 2). Since this model was relatively small, when I did convolve, I did not have to pool or subsample the resulting feature maps.

RESULTS

I kept the same model structure for the MLP and the CNN when training and testing on different advertisers. I had high ROC scores for advertisers 1458, 3358, 3427, and 3476, but the rest were close to insignificant (see Table 3). The large discrepancy in performance was mostly due to some advertiser data sets used different segmentation systems when generating the logs (Zhang, Yuan, Wang, & Shen, 2014), thus yielding vastly lower scores given the same models. Another explanation was that the underlying attributes that would be need to predict clicks were not present in the iPinYou data for those advertisers.

Analyzing the two sets of scores, I saw that I got very similar performance using both models. This indicated that convolutions and feature mappings within the CNN contributed little to the overall performance of the model. This meant that the process of going through the convolutions transformed the existing input data but generated little extra feature information that would have improved predictive power at the MLP stage.

Reviewing the resulting weights of the models on the MLP side, I summed over the weights assigned to the different attributes posttraining to get an idea of which attributes were more significant than others.

CNN Model				
Layer	Input Components	Output Components	Height	Width
Convolution Layer 1	1	-	5	103
Transition Pattern Shape	1	7	3	3
Convolution Layer 2	7	-	3	101
Transition Pattern Shape	7	7	3	3
Hidden Layer 1	693	600	-	-
Hidden Layer 2	600	500	-	-
Softmax Layer	500	2	-	-

Table 2. CNN model structure, which includes the different layers and the dimensions on each layer. Input and output components represent number of feature maps in CNN/nodes in MLP while height and width represent sizes of corresponding CNN matrices.

Advertiser	CNN ROC Score	MLP ROC Score
1458	0.935388217853	0.931179830257
2259	0.515805702202	0.528410074345
2261	0.505143481136	0.498935607046
2821	0.514514351067	0.524985237434
2997	0.515771349444	0.503525281223
3358	0.850326610922	0.850720642117
3386	0.566418771409	0.568017099443
3427	0.868989286527	0.853056041602
3476	0.777973272777	0.788934173874

Table 3. Final ROC AUC scores for the CNN and MLP models for the advertisement companies given.

I found that the most influential weights related to region and city for most advertisers. This may be due to the popularity or the need of those products there.

CONCLUSION

Today's digital providers are benefiting from understanding consumer interests in terms of advertisement targeting. From the consumer's perspective, this advancement is also important: We are used to getting spammed with junk or irrelevant advertisements. Having more interesting and relevant advertisements will help us better select our products. Most of the DSPs that host and bid to put advertisements on websites are using more advanced predictive models to predict clicks from targeted users in order to maximize revenue.

The most common models used today are decision trees and regression models. I tried to determine how today's deep learning models would handle similar tasks. A large portion of this task involved data cleaning and parsing. Given all the attributes, I had to remove some that were unusable and split and vectorize the rest to create new features. From there, I had to standardize each of these features to prevent inherent bias in the models. Once I had the data formatted the way I was going to use it, I had to divide it using a combination of oversampling and undersampling to generate a better training set from which my models could better train due to the imbalance in positive data.

Building the models involved decisions for the structure of the model in terms of number of layers, nodes per layer, and pattern structure for the CNN. The models performed about the same, indicating that features generated from the convolutions in the CNN were not contributing any extra information to the model in comparison to using the standard features generated from parsing through the MLP. I attributed some of the prediction power to the region and city feature present in the original data.

Further work in this project would involve implementing other models, such as Radial Basis Neural Networks and Factorization Machines, which have been proven to work in anomaly detection problems and click prediction, respectively. I also could have encoded all the hashed data and used larger training sets to generate the most accurate models given the data. The models performed well on some advertiser data, but having these extra implementations and analytical edge could improve my results further.

ACKNOWLEDGMENTS

I would like to thank Dr. Mark Daniel Ward who headed the Statistics Living Learning Community, which makes this and many other research opportunities possible for undergraduates. This

material is based upon work supported by the National Science Foundation under Grant No. 1246818.

REFERENCES

Aidos, H., Duin, R. P. W., & Fred, A. (2013). Proceedings from the Second International Conference on Pattern Recognition Applications and Methods: *The area under the roc curve as a criterion for clustering evaluation*. https://dx.doi.org/10.5220/0004265502760280

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359–366. https://dx.doi.org/10.1016/0893-6080(89)90020-8

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533–536. https://dx/doi/org/10.1038/323533a0

Simard, P., Steinkraus, D., & Platt, J. (2003). Proceedings from the Seventh International Conference on Document Analysis and Recognition: *Best practices for convolutional neural networks applied to visual document analysis.* https://dx.doi.org/10.1109/icdar.2003.1227801

Trofimov, I., Kornetova, A., & Topinskiy, V. (2012). Proceedings from the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy - ADKDD '12: *Using boosted trees for click-through rate prediction for sponsored search.* New York, NY: ACM. https://dx.doi.org/10.1145/2351356.2351358

Zhang, W., Du, T., & Wang, J. (2016). Deep Learning over Multi-field Categorical Data. In N.Ferro, F. Crestani, M. -F. Moens, J. Mothe, F. Silvestri, G. M. Di Nunzio, . . . G. Silvello (Eds)., *Lecture Notes in Computer Science: Advances in Information Retrieval* (45–57). Switzerland: Springer. https://dx.doi.org/10.1007/978-3-319-30671-1_4

Zhang, W., Yuan, S., Wang, J., & Shen, X. (2014). Real-time bidding benchmarking with iPinYou dataset. *UCL Technical Report, 23*. Retrieved from https://arxiv.org/abs/1407.7073.