ADTECH PAPER

Pierre Damiba

SharpestMindsAI

Notes:

Review cross validation

Precision/Recall stuff

Table of Contents

Introduction

In the online advertising industry, correctly predicting whether an impression will lead to a conversion can lead to massive profits. An impression is an advertisement, and a conversion is a click. The advertisement could be a picture, video, or text. Google alone serves an average of almost 30 billion impressions per day, meaning that a minuscule improvement in predicting click through rate can create billions of revenue for the company over a year. The methods presented in this paper aim to develop a scalable, flexible and robust click through rate prediction algorithm based on user data. Time series features pertaining to a data set of impressions on an eCommerce site with 40 million impressions are collected and assigned to two classes(conversion or no conversion) using logistic regression. The Precision Recall Threshold is then adjusted to find the optimal bidding strategy**.**

Performance Metrics

Due to the nature of click through rate prediction, accuracy is not a good metric for measuring our model. We want to put a higher focus on correctly predicting that impressions lead to conversions. One could build a model that achieves high accuracy scores by correctly predicting no clicks, but with poor performance when predicting clicks. That model would not be very valuable in a click through rate prediction environment because it does not deliver value.

Predicting click through rate is problematic for companies because of the unbalanced distribution between Clicks and No clicks. There is also a lot of noise in the data and a small amount of conversions. Companies can usually capture around 0.05% of total impressions into conversions. The dataset being used in this project has upsampled the conversion rate to 17%.

Instead of accuracy, we will use a confusion matrix and focus on precision score as our metric for model performance. A confusion matrix shows us how our modeled performed compared to the ground truth. The precision score is the number of true positives divided by the number of false positives. Precision score allows us to measure how well our model can avoid predicting a positive when the observation was negative. The recall is the opposite.

Suppose a model designed to predict click through rate predicts 8 clicks in a day that had a total of 12 clicks and some no clicks. Of the 8 predicted clicks, 5 are actually clicks, or

true positives, while the rest are no clicks, or false positives. The model's precision is 5/8 or 0.625. The recall is 5/12, or .42. In this case, the precision tells us how accurate the model can correctly predict a click among actual clicks. The recall tells us how well our model can find all the correct clicks.

Methodology

We will use Sci-kit Learn's implementation of a Stochastic Gradient Descent Classifier to classify our data into clicks or no clicks. Next, we create a GridSearchCV to tune our hyperparameters and find the model with the highest precision score. Then we will find the best decision threshold using the precision-recall curve and the ROC curve. Finally, we will present our results and conclusions.

To begin we will sample the dataset of 10 days of click through rate data from Avazu mobile ads down to 1 million records. A random distribution of the dataset will be imported representing each hour of the total 10 days recorded.

Next, we check the value counts to confirm sampling did not alter the unbalance of the classes. No click represents the vast majority of the data, almost 83%, while click represents 17%.

Target Encoding

The label encoder was used to encode our labels in a one-vs-all method. The label binarizer allows us to assign the class for which the model gave the greatest confidence. The majority class, No Click, will be negative.

Feature Encoding

I found the most useful feature engineered was the historical click through rate. This metric was created by looking at the click through rate for different features by the first chronological 10% of the data. The new features were then inserted into the remaining 90% of the data, and then split again for training and testing. Intuitively, this makes sense, the best predictor for how an advertisement will perform in the future, is how similar ads performed in the past.

We will drop features from our model that we do not expect to have predictive power, and create a train test split. It is important to specify shuffle=False here to ensure that our splits times are separate and do not result in us using the future to predict the past.

Model

For our model we will use a Stochastic Gradient Descent (SGD) Classifier. This estimator implements regularized linear models with SGD learning. This is an online learning model that will help us manage compute time. SGD models sequentially update the model with training samples one at a time, instead of the entire training set at one. Online learning allows our model to be adaptable to new data as it comes in, and updating the model accordingly. A click through rate prediction model must include the most recent user behavior and trends. The gradient of the loss is estimated each sample at a time.

We create our hyperparameter grid and create our scorers. The hyperparameter that I focused on the most was eta0, the initial learning rate value. Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our model with respect the loss gradient. The lower the value, the slower we travel along the downward slope. While using a small learning rate might be effective in terms of making sure that we do not miss any local minima, it could also mean that we'll be taking a long time to converge.

Our model uses five scorers, the precision recall scores, which were covered above. Next we have the accuracy score and the ROC of AUC score. The accuracy scorer will simply look at how many predictions were correct, while the ROC scorer will look at the true positive rate versus the false positive rate. Finally, we will score our model on a custom metric based on the results of the previous models.

Grid Search Wrapper

This Grid Search Wrapper function fits our SGD model using refit_score for optimization and prints the accompanying confusion matrix. In this case, we want to refit_score based on the scorers we created to optimize our model. Our custom model got the best results as you can see in the results section.

Stratified K Fold

K fold is a cross validation technique that divides the dataset into k groups of similar size. One fold is held out for testing and the other folds are used for training. The process is then repeated k times and each time a different fold is used for testing. While K fold is relatively computationally intensive as the model has be rerun from scratch several times, it offers advantages to our use case. K fold reduces bias as every observation is tested and in training multiple times giving us a less optimistic score. Stratification creates the folds in a way that we have a similar percentage of Clicks and No Clicks throughout our folds. This confirms that one fold of data does not have an over or under representation which would impact our unbalanced classes.

Precision Recall Threshold

The lower the Precision Recall Threshold(PRT), the more clicks our model will capture. However, a low PRT will also result in more no-clicks being predicted as clicks. The company's strategy and business teams can decide which model fits best depending on the situation. A PRT of .2 is aggressive enough to capture 66% of the clicks, but predicts a click incorrectly only 41% of the time. The more aggressive our bidding strategy, the more clicks we can confidently capture. This model would work best for high value items, such as luxury goods. The cost of impressions for these items is high, but the revenue we can generate by capturing conversions is also high. That will offset the money spent on impressions that do not lead to conversions.

Custom Metric

Instead of optimizing just on precision, we can create a custom classifier that uses a custom metric to choose a decision threshold. A dummy classifier would perform at 17%, and we got some noise at 99%, so we subtracted 18% from the precision and then added a portion of the recall. We chose custom alpha and beta values for the metric and used a one shot grid search where we incremented by 1% each time depending on the precision.

Results

In this figure we can see that as we increase the precision recall threshold, we also increase the amount of No clicks we predict. At a Precision Recall Threshold of .2, more than half of the clicks are being captured(9,861/16,976) are being captured, but there is not an overwhelming amount of no clicks being predicted clicks(27,819/83,205.

By changing the precision recall threshold of our model, we can very specifically alter how many clicks or no clicks we capture. This allows us to deliver a robust but flexible model that is able to adapt to the company's needs. By tweaking the precision recall threshold of your model and using historical rates, the company can dramatically change the amount of impressions that are converted into clicks.