

# ICS635 Homework 3

Lambert Leong

March 14, 2019

## 1 Introduction

I chose to participate in the Santander Customer Transaction competition. A quick survey of the raw training data revealed that there are 200 features which contain positive or negative float values. My initial thought was to perform a principal components analysis (PCA) to try and reduce the dimensionality of the data. Several visualization kernels showed that the distribution for each class with respect to each field, i.e. 'var\_0', 'var\_1', etc..., is not very different. In fact, there is a significant overlap of the two classes for almost all features and determining a decision boundary appears to be non-trivial. This led me to believe that mapping the data into a new space with PCA would not be useful for the purposes of reducing data dimensionality.

I first looked at an ensembling method and used gradient boosting to classify the data as is, with all 200 features. I then used the same gradient boosting classifier on training data which had the dimensionality reduced via PCA. Comparing the results of the original data, with 200 features, to the PCA transformed data, with less than 200 features, indicated that I should not proceed with PCA and models generated with all 200 features were able to generalize to the validation set better.

The instruction also noted that the transaction amount is not really relevant. Each of the 200 feature fields contained either a positive or negative number which could indicate money coming in and money going out, respectively. Under this assumption, I sought to capture the amount of times money came in and the number of times money went out for a particular individual to see if it had any correlation to a particular class. In other words, I capture the total amount of positive values and the total amount of negative values for each individual in the dataset.

Although it is stated that the amount of the transaction is not relevant I added features that indirectly correlate to the amount and these features include mean, standard deviation, skew, and kurtosis. I applied the previously mentioned models, gradient boosting, LSTM, and 1D CNN, to the new dataset with new features to see if it would lead to better classification.

## 2 Methods

The first thing that was done was to split the training data into a training and validation set. The Scikit-learn `train_test_split` function was used to split the training data. A `random_state` of 18 was used and the test size was 0.2 which resulted in 80% of the original training data to be used for training and the remaining 20% to be used for validation.

## 2.1 PCA

PCA was performed using Scikit-learn's decomposition module. PCA was performed on centered and non-centered data which yielded similar results. I sought to reduce the number of features by keeping those which explain 95% of the variance. It turns out that 118 features explain about 95% of the total variance which is more features than what was initially expected. The top 118 features were kept and used to train a gradient boosting model.

## 2.2 Gradient Boosting

The sklearn xgboost module was used to create the gradient boosting model. The model was trained and evaluated on the original training data, with all 200 features, as well as with the PCA reduced data with 118 features. Area under receiver operation characteristic (AUROC) was used to evaluate how well the model was at generalizing to the validation test set. The non-reduced data, the original data yielded higher AUROC values, at around 0.885, when compared to the models trained on PCA transformed data. Different numbers of features were experimented with for PCA but the full dataset still yield the best results.

## 2.3 Feature Engineering

Functions were written to iterate through each row or record and count the number of positive values and the number of negative values. These two numbers were appended to the end of the dataset as two extra features. Another function was written to capture the longest sequence, within each row, of positive numbers and negative numbers. These features were engineered under the hypothesis that this could be sequence data and under the assumption that the order of the features have not been randomized.

## 2.4 Parameter Tuning

Model parameters, which include `n_estimators`, `max_depth`, and the learning rate were chosen with the help of the `GridSearchCV`. `GridSearchCV` exhaustively searches through a range of chosen values for each of the model parameters and helps you choose the best parameter values. Parameters and their final values are as follows: `n_estimators = 1500`, `max_depth = 3`, and the learning rate `= 0.3`. The `tree_method` was `gpu_hist` and predictor was `gpu_predictor`; these parameters were not tuned and chosen so that GPU acceleration could be enabled.

## 3 Results

Due to the fact that not much was known about the data and the 200 features, I thought we may be able to reduce the dimensionality and build models using fewer features. Various visualization kernels on kaggle showed great similarities between the two classes with respect to a particular feature. No one feature seems to show any significant amount of separability between the two classes however, it appeared that some features were slightly more separable than others. Figure 1 shows the relationship between the number of principal components and the variance explained by those number of components. I had hoped that the majority of the variance, about 95%, was explained by a few components.

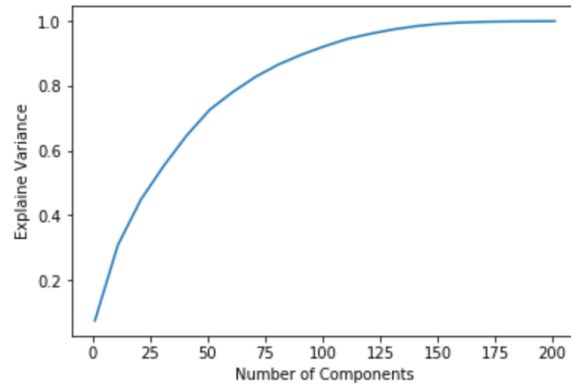


Figure 1: Number of principal components and the explained variance

I experimentally determined if it was worth proceeding with PCA. Using the XGBoost gradient boosting module I defined a set of consistant parameters for comparision which are shown in Table 2. I trained the model using the training data, 80% of the full training dataset, and varied the amount of components used during training. Predictions were made on the validation set, 20% of the original training dataset, and the AUROC were calculated. Figure 2 shows how varrying the number of principal components affects the AUROC score.

Table 1: XGBoost parameters

Parameter	Value
n_estimators	1000
tree_method	gpu_hist
predictor	gpu_predictor
max_depth	3
eval_metric	auc

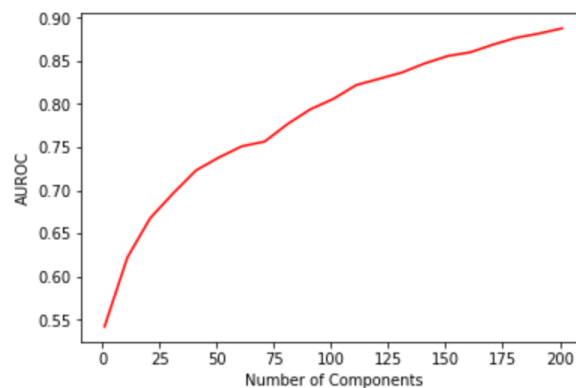
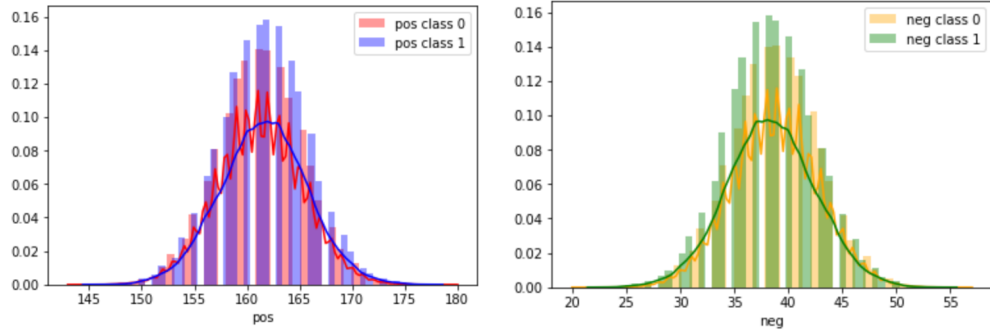


Figure 2: AUROC scores have a postive correlation to the number of principal components

The results in Figure 2 led me away from PCA. Using all the components with XGBoost

resulted in a fairly good AUROC at 0.887. I decided to stick with the XGBoost classifier and proceed with feature engineering. As mentioned in the previous section, I constructed two extra features. Frequency distributions for each of the features are plotted in the following figures. They are colored by class.

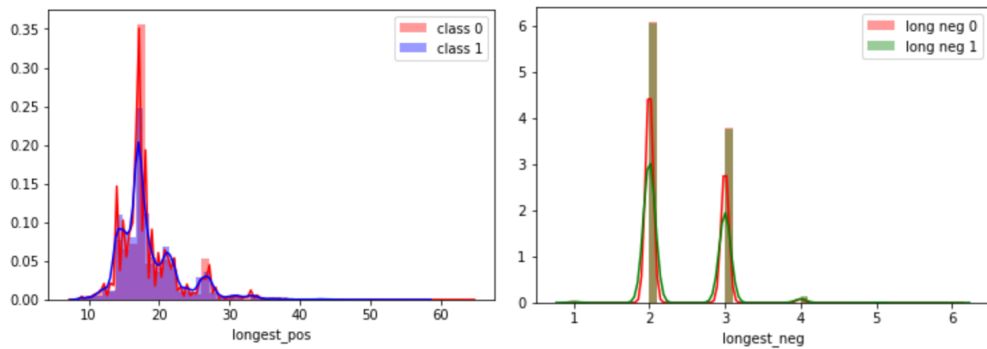


(a) Distribution of positive numbers by class (b) Distribution of negative numbers by class

Figure 3: Frequency distributions for positive and negative values colored by class

Figure 4 indicates that there are some noticeable differences between class 0 and class 1 when looking at the total number of positive feature values and the total number of negative feature values. The separation seen in Figures 4a and 4b display more difference and less overlap than any of the 200 original features. Due to this fact, I hypothesised that adding these features to the model may help with training.

Distributions of the longest positive and negative sequences with respect to each class are shown in Figure 4. These two features were constructed under the assumption that the features correspond to some type of sequence data. Again, I was trying to find features which show more separation between the two classes.



(a) Distribution of positive numbers by class (b) Distribution of negative numbers by class

Figure 4: Frequency distributions for longest positive sequence and longest negative sequence values colored by class

The resulting distributions for the longest positive and negative sequences are different which is

unlike the total count of positive and negative numbers. There are slight differences with respect to each class which may help to increase the models AUROC on the validation set. Using this new feature may come with some risk because it is likely that the feature order may have been randomized before the datasets were released. If that was the case, the longest positive and negative sequences will be different and possibly irrelevant.

The hyper parameters for the XGBoost classifier were tuned using GridSearchCV from sklearn. GridSearchCV does not support GPU tree\_methods and so the search was done on the CPU tree methods. Final models were constructed using the GPU versions of the hyper parameters for speed gains. Finalize parameters are shown in Table 2.

Table 2: XGBoost final parameters

Parameter	Value
n_estimators	3500
tree_method	gpu_exact
predictor	gpu_predictor
max_depth	2
eval_metric	auc

An XGBoost classifier was trained on the full training dataset using the hyperparameters from Table 2. The classifier was then used to make predictions on the test set. The predictions were uploaded and the public score is shown below in Table 3

Table 3: Final public results

Model	Best Public Score
1D CNN	0.833
LSTM	0.829
PCA XGBoost	0.882
XGBoost	0.899

As of 3/14/2019 the top score on the leader board is 0.924. My best score is 0.899 which was the result of the XGBoost classifier with some added features.

## 4 Conclusion