CS230

Predicting mortgage loan delinquency status

Ksenia Ponomareva
Department of Computer Science
Stanford University
kp260@stanford.edu

Abstract

Monitoring of loan performance and early identification of high risk consumers aids prevention of loan defaults and is of interest to many banks and investors. For banks especially, timely monitoring ensures regulatory compliance as well as adequately quantifying its risk, accurately calculating its capital and setting aside proper reserves. With this goal in mind, four multi-classification models have been built that take as input characteristics of a loan at inception as well as information about the first twelve monthly payments and predict the status of these payments over the next twelve-month period.

1 Introduction

There is an extensive literature on application of machine learning algorithms to credit scoring, where a model determines relationship between default and loan characteristics. Once built, this model is used to predict the probability of debt being repaid by the obligor. Review and comparison of these types of models are provided in [1]-[3], with [4] focusing on assessing mortgage risk using a deep net and [5] using a convolutional neural network.

As can be seen from these papers, most publications in consumer lending applications cover binary classification, where a loan either defaults, or alternatively becomes severely delinquent, defined as having payment delayed by six months and more, or it does not. There are several reasons for considering multi-classification problem instead. Firstly, a lot of information is lost if only default and non-default states are taken into account. For example, a loan with no missed payments has a different creditworthiness profile from a loan where payments are late by one month only, but occurring several times during the life of the loan. Differentiating between these profiles would allow more accurate tailoring of credit terms and conditions as well as enable early warning signal detection, since borrowers are likely to transition through different credit profiles throughout the term of the loan. In this paper residential mortgage loan data has been used, but model architectures described here can be applied to other type of loans as well, provided a suitable dataset is available for training.

2 Dataset and Features

Publicly available Fannie Mae single family loan performance data [6] has been used for this project. The population considered contains a subset of Fannie Mae's 30-year, fully amortizing, full documentation, single-family, conventional fixed-rate mortgages. This data provides information about mortgage loans issued between 2000 and 2016 and their performance, e.g. late payments in months, up to the fourth quarter of 2017. Original mortgage rate for each example has been scaled by the average 30-year federal funds rate for the corresponding quarter to enable comparison of

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

mortgage loans issued over this 17-year period.

The following eight features have been selected from the available 24 fields characterizing the loan: the original rate, the original amount, the original loan-to-value (LTV), the number of borrowers, the debt-to-income (DTI) ratio, the credit score of the borrower, the first three digits of the zip code and the mortgage insurance percentage. These features have been normalized to speed up training. Please note that this information is not updated throughout the considered two-year period of the loan term. Additionally, loan monthly payment performance information over the first twelve months of the loan term is utilized. Altogether, this makes up twenty input features.

The ground truth output has seven classes with values ranging from zero to six and is based on the actual loan performance over the month 13 to 24. Loan payment performance information indicates the number of days, represented in months, the obligor is delinquent as determined by the governing mortgage documents. Here 0, for example, represents loan that is current or is less than 30 days past due and sequence continues thereafter for every thirty-day period. Class 6 indicates that payment is delayed by six or more months and the underlying loan is considered to be severely delinquent.

In the original dataset with ≈ 4 million entries roughly 96% of entries belong to class 0, as is expected, since most people would make payments on time in the first two years of their mortgage. However, this dominance of class 0 poses some challenges for training, such as model would tend to predict this class all the time to minimize the loss. In order to ensure that training dataset is more balanced, class 0 examples were sampled from the original dataset to represent $\approx 50\%$ of the whole training dataset. Histogram of frequency of each class in the balanced training dataset is shown in figure 1. Proposed solution to deal with training an imbalanced dtaset is discussed further in section 3.5.

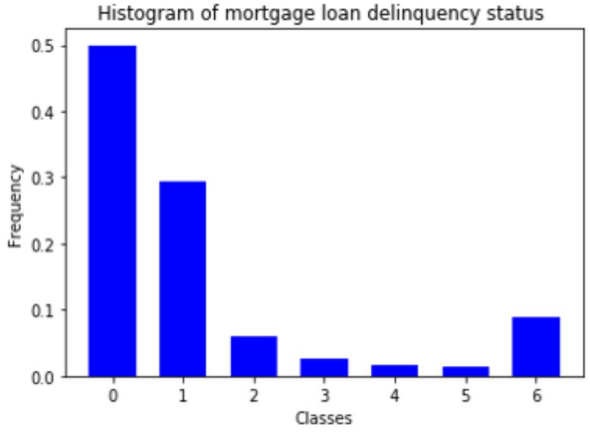

Figure 1: Class frequency distribution in train set

Only entries with complete information were used and records where loans were terminated prior to month 24 were discarded. Finally, 730112 data entries were split into 90% used for train set, 5% for dev set and 5% for test set.

It is worth noting that in order to predict loan payment delinquency status it is possible to use only the eight features discussed above. However, adding information about the payments for the first twelve months of the loan term allows comparison between very different model types and architectures.

3 Methods

Different model architectures are considered for mortgage payment delinquency status prediction problem. The definition of the problem and the dataset available enables comparison of four different multi-classification model architectures, which have been implemented using TensorFlow, [7], with code stored in a github repository, [8]. These are a basic baseline model, a deep neural network, a one-directional LSTM recurrent neural network and a one dimensional convolutional neural network. Accuracy metric, $\sum \mathbb{I}_{argmax(Y)=argmax(\hat{Y})}$, which is averaged for all the examples in the dataset, is used for comparison of models' performance. Here $\hat{Y}$ is the prediction, Y is the true label and $\mathbb{I}$ is an indicator function. Figures 2-5 show high-level architecture for each model and further details are provided below.

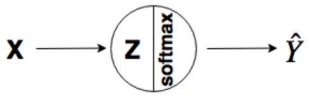

Figure 2: Baseline model architecture

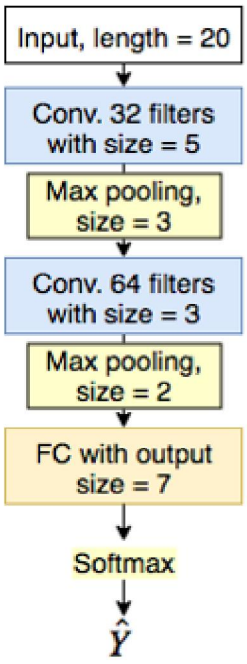

Figure 3: CNN model architecture

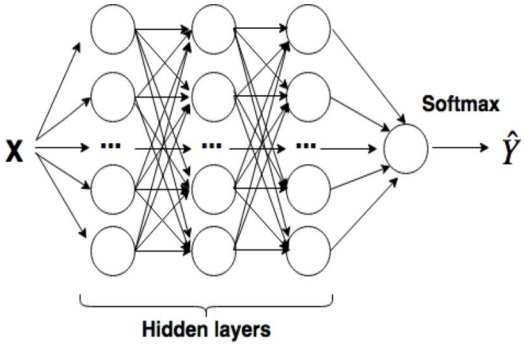

Figure 4: DeepNN model architecture

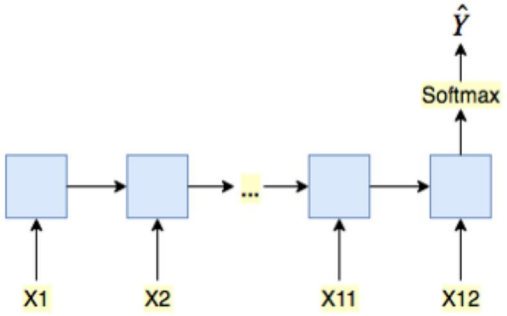

Figure 5: RNN model architecture

3.1 Baseline

Baseline is the simplest architecture considered here and has a single node, as shown in figure 2. Model uses input with 20 features (eight loan characteristics at inception and performance of the first twelve monthly payments), a linear transformation and a softmax activation function to produce a prediction, $\hat{Y}$, probability distribution over seven classes:

$$Z = WX + b,$$
$$\hat{Y} = softmax(Z).$$

3.2 Deep neural network

Deep neural network (DeepNN) model has three hidden layers, as shown in figure 4, with 100, 50 and 10 nodes respectively. All activations are ReLU apart from the final one.This model has the same input and output, final activation function and accuracy metric as the baseline model. By introducing several hidden layers, the model is expected to capture relationships between features and classes better and hence provide some improvement to the baseline results.

3.3 Convolutional neural network

Convolutional neural network (CNN) model uses a one dimensional structure inspired by [5] and has two blocks of convolutional and max pooling layers, followed by a fully connected layer, as shown in figure 4. Activation functions for layers other than final are ReLU. Input is 20x1 (8 loan application features and 12 monthly payments) and output is 7x1 based on the softmax final activation function. The first block has one convolutional layer with 32 filters with size 1x5, stride 1 and same padding, then max pooling layer with filter size 1x3, stride 1x2 and same padding. The second block has a convolutional layer with 64 filters with size 1x3, stride 1 and same padding, then max pooling layer with filter size 1x2, stride 2 and same padding.

3.4 Recurrent neural network

Recurrent neural network (RNN) model is a many-to-one LSTM with a softmax activation applied to the last output only, as shown on figure 5. The input sequence has twelve items, one for each month of the observable monthly mortgage payment performance in the first year of the loan. Each of the inputs has nine features, eight loan characteristics, which are the same for each input, and one monthly payment performance for the relevant month. LSTM hidden layer has 200 units. This model is one directional since the order of loan monthly payment statuses is very important. Many-to-one architecture is used since prediction is needed for the payment status over the second year of the loan term and not for each consecutive month in the first year.

3.5 Weighted loss function

Given class imbalances in the training data, as shown in figure 1, model would tend to predict everything as class 0 to minimize the loss. In order to address this issue, loss function needs to be re-weighted to apply higher penalties for getting wrong predictions for classes 1 to 6 compared to the dominant class 0. Median-frequency re-weighting, introduced in [9], is applied here and is calculated as follows:

$$\alpha_c = median\_freq/freq(c).$$

Here $freq(c)$ is the number of examples of class c present in the mini batch divided by the total number of examples in the mini batch and $median\_freq$ is the median of all class frequencies in the mini batch. In the training procedure, each example's softmax cross-entropy loss is multiplied by the coefficient, α_c, according to the label's class, c. Therefore the dominant labels will be assigned the lowest weight which balances out the training process.

4 Experiments

Models have been iterated over various hyperparameters and network structures to provide the best test accuracy. The following configurations have been run for all models:

- A set of learning rate values, $[0.01, 0.001, 0.0001]$.
- Batch size in the range of $[32, 1024]$.
- Number of units in the LSTM cell, $[10, 50, 100, 200]$, for RNN model.
- Number of hidden layers in the range $[3, 5]$, and units in the range $[10, 200]$, for the DeepNN model.
- Number of different filter and stride sizes for the CNN model.
- ReLU and tanh activation functions for intermediate layers.

- Gradient descent, Adam and Adagrad optimizers.

The best results have been achieved achieved with $learning\_rate = 0.001$, $batch\ size = 128$ and model architectures as described in sections 3.1-3.4.

It is worth noting, that since variance turned out to be small for all models, application of regularization methods was not considered necessary.

5 Results

All models have been trained over ten epochs and results for training and test datasets are provided in the table below.

Accuracy	**Baseline**	**DeepNN**	**CNN**	**RNN**
Training Accuracy	53.3%	60.8%	60.5%	60.8%
Test Accuracy	53.8%	61.1%	61.0%	61.3%

Examining results, a number of observations can be made. First of all, all models tend to perform better on the test set than the training set. This can be explained by the fact that proportion of class 0 examples is higher in the test set, 50.2% vs 48%, and all models tend to have a higher accuracy in predicting class 0 compared to other classes, due to class imbalance in the training dataset. The second observation is that accuracy for all models exceeds 48% which means that some of the classes 1-6 are predicted correctly. Finally, DeepNN, CNN and RNN model show an improvement over the baseline, with a small variation of results between them.

6 Conclusion

Although DeepNN, CNN and RNN models perform better than the baseline, they can be further improved in the future by further fine tuning the hyperparameters (number of layers, hidden units, filter number and size, learning rate) and using F1 score for each class prediction to find the best combination. Additionally, considering more than the first twelve payments as features or supplementing with current/savings/credit account data as in [2] might help improve models' performance. Finally, using an even more balanced sample from the data available might be worth consideration.

References

[1] García, V., Marqués, A. I., & Sánchez, J. S. (2014). An insight into the experimental design for credit risk and corporate bankruptcy prediction systems. *Journal of Intelligent Information Systems*, **44** (1), 159–189.

[2] Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state- of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, **247** (1), 124–136.

[3] Yeh, I. & Lien, C., (2009) The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients, *Elsevier Expert Systems with Applications*, **36**, 2473–2480.

[4] Sirignano, J., Sadhwani, A., & Giesecke, K. (2016). Deep Learning for Mortgage Risk. ArXiv e-prints http://adsabs.harvard.edu/abs/2016arXiv160702470S.

[5] Kvamme, H., Sellereite, N., Aas, K. & Sjursen, S. (2018) Predicting mortgage default using convolutional neural networks, *Elsevier Expert Systems with Applications*, **102**, 207-217.

[6] https://loanperformancedata.fanniemae.com/lppub/index.html.

[7] https://www.tensorflow.org

[8] https://github.com/ksenp260/mortgage_payment_status

[9] Eigen, D. & Fergus, R., (2015) Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, *arXiv:1411.4734v4*.