Josh Reitz

Professor Carsten Eickhoff

BIOL 1595

3 March 2020

Assignment 1 Report

For the first group assignment of BIOL 1595, my group and I created a machine learning model to predict whether or not a patient will be readmitted to the Intensive Care Unit within 30 days after discharge. We trained and validated our model on a dataset of 68 features for over 28,000 patients from the Beth Israel Deaconess Medical Center in Boston. Our model utilized logistic regression to predict the readmission label of a patient with a total accuracy of over 86 percent. This report will formally detail our model and explain its successes and shortcomings.

Logistic regression is the repetition of regression, sigmoid activation, and gradient descent until the loss is minimized. We found logistic regression to be suitable for this project because for this binary classification problem, logistic regression would calculate a probability of readmission for each patient. This type of output would be intuitive and easy to understand for ourselves while we created it and to any others who viewed our model’s outputs. Logistic regression also uses hyperparameters that we could tune to optimize our model. We decided against other models such as a nearest-neighbors classification due to its large memory requirements. Similarly, we decided against Naive Bayes since it did not iteratively learn from our dataset, which we thought necessary for this project. We also thought that a decision tree classifier would be too prone to overfitting while creating a random forest would be too computationally expensive.

To begin training the model, the weights must first be initialized to 0. We used 69 weights: one weight corresponding to each feature and 1 bias weight. In order to calculate the prediction for a patient, the sum of each weight multiplied by its corresponding feature was calculated. The bias weight was then added to this value to produce continuous outputs with no bounds. But, these values would not be useful for making predictions. To address this, we processed the regression output with a sigmoid activation function. This compresses our unbounded outputs to an S-shaped distribution over the interval [0,1], effectively converting these values into readmission probabilities.

After we have obtained the probabilities, the weights must be modified based on how accurately they predict the readmission label. We accomplished this through gradient descent. The derivative, or gradient, of the sigmoid function was calculated, then multiplied by the difference between the predicted and actual labels, and again multiplied by the feature value. This product measured how accurate our predictions were and how drastically our weights needed to be modified in turn. A unique weight modification was calculated for each weight so that each feature’s contribution to the readmission probability could be modified to account for more or less relevant features. These weight modifications were then normalized and subtracted from the current weights to produce updated weights.

Finally, we optimized our model for cross entropy loss. This is a loss function based on the prediction probability of each patient normalized over the number of data points. As we printed this value every 50 iterations of the model, we would see the loss decrease, then eventually plateau after about 500 iterations, or epochs. We plotted our cross entropy loss over the number of epochs to discover this relationship and determined that the optimal hyperparameter value for the number of epochs was 500. Lastly, we performed this training cycle on batches, or portions, of the training set. This strategy minimized the memory usage and runtime of our model while preserving the convergence speed and ability to escape local minimums. However, it required modifying nearly every function we wrote to iterate over batches rather than the whole training set.

One challenge faced while creating this model was the implementation of principal component analysis. We hoped PCA would prevent overfitting, improve our runtime, and maintain our high accuracy scores. We plotted the explained variance over the number of principal components to determine how many components our model should contain after PCA to preserve the majority of the data’s variance. We chose 50 principal components because the loss of the first 18 components removed an almost negligible amount of variance. However, after simplifying our 68 features into 50 principal components to reduce dimensionality, our accuracy scores decreased slightly. Thus, we abandoned PCA in our final code base. Additionally, we needed to address the class imbalance present in the data since readmitted patients were far less common than the other class. We accomplished this by oversampling the readmitted data points in our training set until the classes were balanced in number. Oversampling resulted in an immediate increase in our accuracy scores, especially for the readmitted class, which made sense since this class was no longer underrepresented in the training set.

Despite these challenges, our logistic regression model employed batching and oversampling, but not PCA, to achieve peak F1 scores of .923 and .402 on the validation set. While these scores are impressive, they leave room for improvement. Potential additions to our model in the future could involve a successful implementation of PCA to reduce dimensionality and prevent overfitting. Lastly, we could implement a method to further prevent overfitting, such as cross-validation, which trains and tests changing folds of the dataset, or regularization, which explicitly discourages overfitting through the use of a penalty to the loss function.