Anirudh Chintaluri, Radin Rezanezhad

Dr. Yilmaz, Period 4

Machine Learn 1 TJ AV

11 December 2024

Quarter 2 Project Proposal

1. **Motivation**

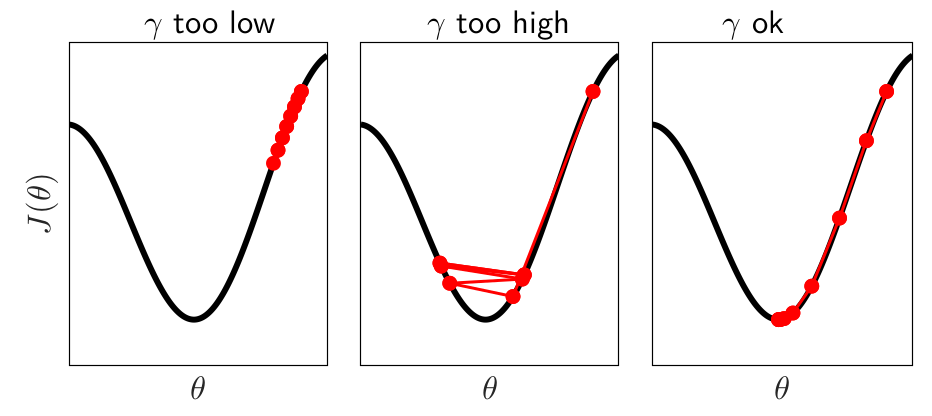
Credit card fraud detection presents a significant challenge in financial security, with fraudulent transactions causing billions in losses annually. This project focuses on detecting fraudulent credit card transactions using machine learning techniques, specifically decision trees, to address this issue. We will leverage the IEEE Credit Card Fraud competition dataset to maximize the accuracy of fraud detection models. [1].

The dataset comprises real-world e-commerce transactions, each characterized by numerous features including temporal information, card identifiers, merchant data, and transaction amounts. What makes this dataset particularly challenging is its high-dimensional nature combined with extreme sparsity. Many features, such as merchant identifiers and card IDs, exhibit high cardinality, meaning they can take on many possible values, while individual merchants or cards appear very rarely in the dataset. This sparsity creates significant challenges for machine learning approaches that rely on dense feature representations. Another challenge in this dataset is the severe class imbalance due to fraudulent transactions constituting a small percentage of transactions. This imbalance has two major implications: first, models tend to bias heavily towards predicting the majority class, and second, traditional accuracy becomes inadequate for evaluation as a model can achieve high accuracy by always predicting the majority class [5].

1. **Method**

We aim to make considerable improvements to the Gradient-Boosted Decision Tree algorithm (GBDT). GBDTs work by initializing a decision tree that will be evaluated on a metric, known as the loss function. After analyzing the loss of the initial decision tree, the model will attempt to minimize the loss by moving in the direction of the negative of the gradient — that is, move in the direction where the loss function is decreasing the fastest so that the loss can be minimized and the model performance can be maximized. In a way, this behaves very similar to neural networks’ backpropagation algorithm with the multivariate correction techniques but instead involves a decision tree that eventually boosts itself to form an ensemble decision tree model.

In order for this GBDT algorithm to apply the direction of the negative of the gradient to the decision tree, it will add another decision tree scaled by a factor called the learning rate, showing how much the tree will go down the loss function. The choice of learning rate is critical. Low learning rates ensure convergence but require more iterations, increasing computational costs. Conversely, high learning rates accelerate convergence but risk overshooting the optimal solution [2]. It can be visualized in the diagram below, with 𝛾 being the learning rate:



We plan to improve upon existing implementations of GBDTs by making this learning rate adaptive, as opposed to constant learning rates utilized by previous literature [5]. According to Brownlee [3], tuning learning rates in GBDTs often requires iterative experimentation, as the optimal 𝛾 depends on both the dataset and the specific loss function. A well-tuned learning rate schedule can reduce the reliance on extensive hyperparameter grid searches, which are computationally expensive. Moreover, Andrej [2] highlights that adaptive strategies can reduce the sensitivity to initial learning rate choices, particularly in datasets with high sparsity, where the gradient signal may vary significantly across iterations.

1. **Intended Experiments**

To evaluate the effectiveness of adaptive learning rates, we plan to conduct the following procedures.

First, we will develop a baseline GBDT model with a fixed learning rate and evaluate the performance using precision, recall, F-1 score, and area under the precision-recall curve (AUC). We will conduct grid search experiments to identify the optimal fixed learning rate.

Following this, we will implement adaptive learning rate mechanisms. We will test decay-based schedules (e.g. exponential or step decay), which are especially effective in the later stages of training when the model approaches an optimal solution and requires finer adjustments [4]. XGBoost supports this function, as it allows dynamic adjustments to the learning rate during model training. Next, we will test feedback-based adjustments to the learning rate. The choice to increase 𝛾 will be made during initial epochs where the gradient magnitudes are large, or when the validation loss decreases significantly over consecutive iterations, indicating that the current learning rate is too small to leverage the residual signal fully [2,3]. The choice to decrease 𝛾 will be made if validation loss begin to plateau, oscillate, or increase, suggesting that the model is overfitting or making adjustments too aggressively [2].

1. **References/Bibliography**

[IEEE-CIS Fraud Detection | Kaggle](https://www.kaggle.com/c/ieee-fraud-detection/overview) [1]

[Selecting Optimal Parameters for XGBoost Model Training | by Andrej Baranovskij | Towards Data Science](https://towardsdatascience.com/selecting-optimal-parameters-for-xgboost-model-training-c7cd9ed5e45e) [2]

[Tune Learning Rate for Gradient Boosting with XGBoost in Python - MachineLearningMastery.com](https://machinelearningmastery.com/tune-learning-rate-for-gradient-boosting-with-xgboost-in-python/) [3]

[Learning rate - Wikipedia](https://en.wikipedia.org/wiki/Learning_rate#:~:text=A%20learning%20rate%20schedule%20changes,%2C%20step%2Dbased%20and%20exponential). [4]

[IEEE-CIS Fraud Detection - Top 5% Solution | by Arun Mohan | Towards Data Science](https://towardsdatascience.com/ieee-cis-fraud-detection-top-5-solution-5488fc66e95f) [5]