Fraud Analytics-Assignment4

ADEPU VASISHT
cs20btech11002@iith.ac.in

VANGA ARAVIND SHOUNIK
cs20btech11055@iith.ac.in

DIYA GOYAL
cs20btech11014@iith.ac.in

DONTHA AARTHI
cs20btech11015@iith.ac.in

April 2023

1 Problem Statement

Cost-sensitive logistic regression is a machine learning algorithm used to address the issue of varying misclassification costs for different classes. Traditional logistic regression aims to optimize classification accuracy, but this approach may not be practical in situations where the cost of misclassifying a particular class is much higher than others. Cost-sensitive logistic regression takes into account the cost of misclassifying each class and adjusts the decision boundary accordingly. This technique has many applications in areas such as fraud detection, medical diagnosis, and spam filtering, where the cost of false positives or false negatives can have significant consequences.

2 Dataset

The dataset we're working with contains 147636 samples, each having 11 features and a corresponding label of either 0 or 1. Additionally, we've been provided with the false negative cost for each sample. In terms of costs, the true negative cost is 0, while both the false positive and true positive costs are 4. Our goal is to train a classifier that utilizes the sample features to accurately predict the label.

3 Algorithm

3.1 Cost Sensitive Algorithm

The cost sensitive loss is given by the following formula.

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{N} y_i(h_w(x_i)\mathcal{C}_{TP_i} + (1-h_w(x_i))\mathcal{C}_{FN_i}) + (1-y_i)(h_w(x_i)\mathcal{C}_{FP_i} + (1-h_w(x_i))\mathcal{C}_{TN_i})$$

where $h_w = sigmoid(w^T x)$ The cost sensitive loss function is non-convex and hence we cannot use standard gradient descent to optimize the parameters as the probabilty of the model being stuck at a local minima is high.

Hence to optimize these type of non-convex functions as mentioned in the paper we are going to use genetic algorithm to optimize the loss

3.2 Genetic Algorithm

Genetic algorithm is a powerful optimization technique inspired by the principles of natural selection and genetics. It is widely used in many fields to solve complex optimization problems, where traditional methods may not be effective. The algorithm starts with a population of potential solutions and applies selection, crossover, and mutation operations to evolve and improve the population over generations. By simulating the process of natural selection and genetic variation, genetic algorithm efficiently searches for the optimal solution within a large search space. This technique has numerous applications, including machine learning, engineering, finance, and bioinformatics, among others. The steps involved in the algorithm are.

1. Initialization: A population of potential solutions is randomly generated.
2. Evaluation: Each individual in the population is evaluated based on a fitness function, which determines how well it solves the problem.
3. Selection: The most fit individuals are selected for reproduction based on their fitness scores.
4. Crossover: The selected individuals undergo crossover, which combines their genetic information to create new offspring.
5. Mutation: Random changes are introduced to the offspring to introduce genetic diversity.
6. Replacement: The new offspring replace the least fit individuals in the population.
7. Termination: The algorithm stops when a satisfactory solution is found or when a maximum number of generations or evaluations is reached.

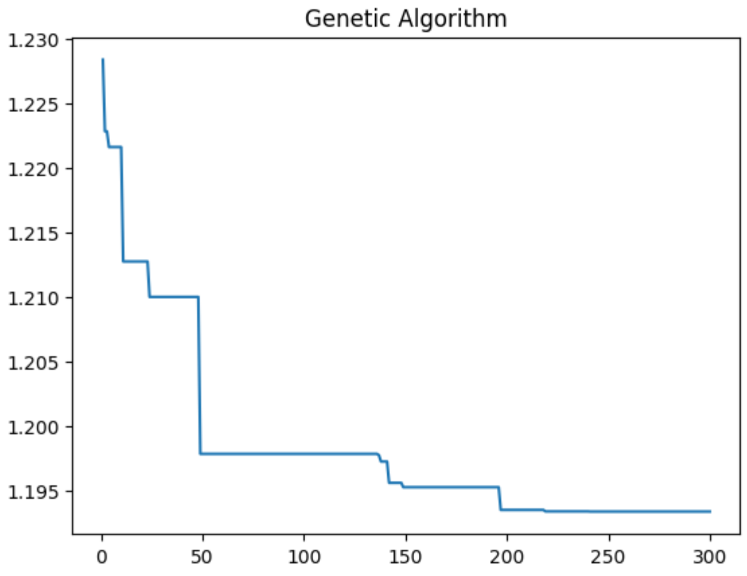
4 Results

Since the function is not convex we cannot use loss.backward() or our standard optimization techniques and hence we used genetic algorithm. If we would have used standard optimization technique the loss function would have reached a saddle point halting at an accuracy of 0.298 but with our genetic algorithm loss continues to decrease albeit at a slower rate than usual.

Our parameters for genetic algorithm are.

```
Number of iterations = 500
Population size = 50
mutation probability = 0.01
```

The Graph will look something like this with Y-axis as our **loss function** and X-axis as our **number of iterations.**



4.1 Conclusion

We have sucessfully implemented cost sensitive logistic regression which learns using genetic algorithm implemented from scratch.