Example-dependent Cost-Sensitive Logistic Regression

This assignment is a requisite for the 'Fraud Analytics' course, CS6890

1st Anirudh Joshi
Dept. Computer Science, IITH
CS23MTEHC11002

2nd Malsawmsanga Sailo
Dept. Computer Science, IITH
CS23MTECH11010

3rd Koushik Maji
Dept. Artificial Intelligence, IITH
AI23MTECH11004

***Abstract*—In this study, we delve into the realm of example-dependent cost-sensitive classification problems, where the costs associated with misclassification vary across examples. A prime instance of such a classification problem is credit scoring. Traditional methods often overlook the actual financial costs tied to the lending business while addressing these problems.**

This assignment includes a novel example-dependent cost matrix based classification specifically designed for credit scoring. The findings underscore the significance of incorporating real financial costs into the models. The application of our proposed cost-sensitive logistic regression leads to substantial improvements, particularly in terms of increased savings. The efficiency of our proposed method is tested against leading example-dependent cost-sensitive algorithms using two given dataset.

We implement two approaches for this task: the Bahnsen approach and the Nikou Gunnemann's approach. The grading for this task is divided into two parts: 1. Implementation using the Bahnsen approach, worth 10 marks (5 for coding and 5 for results) 2. Implementation using the Nikou Gunnemann's approach, also worth 10 marks (5 for coding and 5 for results)

I. Problem Statement

When comes to Classification of class imbalanced data is a challenging task to handle in machine learning. In general, the cost function of a logistic regression problem is convex, and it is typically optimized using either the maximum likelihood estimator or gradient descent. However, this cost function assigns the same weight to different types of errors, both false positives and false negatives.

In many real-world applications, this is not the case. Specifically, we have:

$$J_i(\theta) \approx \begin{cases} 0 & \text{if } y_i \approx h_\theta(X_i) \\ \infty & \text{if } y_i \approx (1 - h_\theta(X_i)) \end{cases}$$

In the context of cost-sensitive classification, this implies that $C_{TP_i} = C_{TN_i} \approx 0$ and $C_{FP_i} = C_{FN_i} \approx \infty$. However, in real-world data such as fraud detection, medical decision-making like cancer detection, credit scoring, and spam filtering, the costs associated with false positives and false negatives do not carry the same weight. For instance, in medical diagnosis, a false negative (failing to detect a disease when it is present) could have severe or even fatal consequences, whereas a false positive (erroneously diagnosing a disease when it is not present) might lead to unnecessary stress and medical procedures. Therefore, it is crucial to incorporate these different costs into the model to improve its performance in such real-world applications.

II. Description of the Dataset

The dataset provided for this problem is named `costsensitiveregression.csv`. It consists of 147636 entries with 13 columns. Each entry in the dataset represents a unique instance. The columns represent the following features:

- NotCount, YesCount, ATPM, PFD, PFG, SFD, SFG, WP, WS, AH, AN: These are the independent variables in the dataset.
- Status: This is the dependent variable. It represents the label where 1 indicates fraud and 0 indicates not fraud.
- FNC: This represents the false negative cost, which varies from row to row based on the risk parameter details. This cost indicates the penalty when the model predicts the output as Not fraud but the original label was fraud.

The dataset also has some constant costs associated with each instance:

- True Positive and False Positive cost is constant for all instances, which is 6.
- True Negative cost is constant for all instances, which is 0.

The dataset has a significant imbalance in the number of data rows for fraud and non-fraud classes. It is observed that the Non fraud cases are more than 2 times in number, making it challenging for a normal logistic regression model to perform well, as it assumes the I.I.D property of the dataset.

III. Algorithm Used

Class imbalance in data is a well-known challenge in machine learning. There are several strategies to address this issue, such as assigning a higher loss weight to the minority class, synthetic data generation, cost-proportionate rejection sampling, and oversampling. However, in this study, we focus on two specific methods for modeling the cost function in the context of class imbalance.

The first method is the approach proposed by Alejandro Correa Bahnsen, which incorporates the cost of misclassification into the learning process. The second method is

the approach suggested by Nikou Günnemann, which also models the cost function but with a different perspective. Both methods aim to improve the performance of the model on imbalanced data by taking into account the different costs associated with misclassifying different classes.

A. 1st method : Bahnsen's Approach

Bahnsen's approach is a well-known algorithm used in vector optimization. It is particularly useful for solving linear vector optimization problems. The algorithm is based on the concept of outer approximation, and it is used to compute approximate solution gamuts.

In the context of binary classification, the risk associated with each decision is calculated. For a given example i, the risk of predicting the example as negative is defined as

$$R(c=0|X_i) = C_{TNi}(1-\hat{p}_i) + C_{FNi}\hat{p}_i$$

and the risk of predicting the example as positive is defined as

$$R(c=1|X_i) = C_{TPi}\hat{p}_i + C_{FPi}(1-\hat{p}_i)$$

where $\hat{p}_i$ is the estimated positive probability for example i. We will model $\hat{p}_i$ like logistic regression as follows.

$$\hat{p}_i = P(y=1|X_i) = h_\theta(X_i) = g\left(\sum_{j=1}^{k}\theta_j x_{ji}\right),$$

where $h_\theta(X_i)$ refers to the hypothesis of i given the parameters θ, and $g(\cdot)$ is the logistic sigmoid function.

In the context of cost-sensitive classification, we modify the cost function to incorporate the different costs associated with each example. The problem then translates to finding the right parameters that minimize a given cost function. Usually, in the case of logistic regression, the cost function $J_c(\theta)$ refers to the negative logarithm of the likelihood, such that:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} J_i(\theta),$$

where The new cost function $J_c(\theta)$ is defined as:

$$J_c(\theta) = \frac{1}{N}\sum_{i=1}^{N}[y_i(h_\theta(X_i)CT_{Pi} + (1-h_\theta(X_i))CF_{Ni}) + (1-y_i)(h_\theta(X_i)CF_{Pi} + (1-h_\theta(X_i))CT_{Ni})]. \quad (1)$$

The cost function $J_c(\theta)$ in our problem is non-differentiable and non-convex. Therefore, traditional optimization algorithms like Maximum Likelihood Estimation (MLE) and backpropagation, which rely on gradient information, cannot guarantee finding its global minimum. To overcome this challenge, we resort to a class of optimization algorithms known as Genetic Algorithms (GAs) that are capable of finding global minima in complex search spaces.

B. Differential Evolution

One such GA is the Differential Evolution (DE) algorithm. DE is a population-based, stochastic search algorithm that is characterized by its simplicity, robustness, and power, particularly in handling non-differentiable, non-linear, and multimodal optimization problems. It generates new candidate solutions by adding a weighted difference between two population members to a third member.

For this specific problem, we have used the DE algorithm implemented in the SciPy library. The algorithm is as follows:

Algorithm 1: Differential Evolution Algorithm

Data: Population size NP, Mutation factor F, Crossover rate CR
Result: Optimized parameters
Initialization: Generate initial population;
while *termination criteria not met* **do**
  for $i = 0$ *to* $NP-1$ **do**
    Select three distinct parents X_a, X_b, X_c differing from each other and i;
    Compute the difference vector: $X_{\text{diff}} = X_b - X_a$;
    Mutate the target vector: $X_{\text{mutant}} = X_c + F \times X_{\text{diff}}$;
    Initialize trial vector: $X_{\text{trial}} = X_i$;
    Select a random index R;
    for $j = 0$ *to* $D-1$ **do**
      if $j == R$ *or* *random*() $< CR$ **then**
        $X_{\text{trial}}[j] = X_{\text{mutant}}[j]$;
      else
        $X_{\text{trial}}[j] = X_i[j]$;
      end
    end
    if $f(X_{trial}) < f(X_i)$ **then**
      $X_i = X_{\text{trial}}$;
    end
  end
end

C. 2nd Method: Nikou Günnemann's Approach

Nikou Günnemann's approach addresses the challenge of binary classification in the presence of example-dependent costs. In many real-world applications, the cost associated with misclassification errors can vary among examples. Traditional binary classification methods, such as logistic regression, often assume a constant misclassification cost across all examples, which may not be suitable in these scenarios.

The **general framework** of the cost-sensitive logistic regression model according to Nikou Günnemann's is to minimize the loss function $l(Y, X, \beta)$ defined as:

$$l(Y, X, \beta) = \sum_{i=1}^{m} a_i \cdot y_i \cdot (-\log f(g(x_i, \beta))^{b_i}) + a_i \cdot (1 - y_i) \cdot (-\log(1 - f(g(x_i, \beta)))^{b_i}) \quad (2)$$

Where a_i and b_i depend on c_i. That is, $a_i = a(c_i)$ and $b_i = b(c_i)$ based on functions $a : R^+ \to R^+$ and $b : R^+ \to R^+$. In this equation, a_i and b_i are parameters that depend on the cost associated with each instance. $f(g(x_i, \beta))$ is the predicted probability of the positive class for the ith instance, obtained using the sigmoid function applied to a linear combination of the features x_i and the model coefficients β. y_i is the actual class of the ith instance.

In **Variant A**, the logistic loss function is weighted depending on the cost value c_i by setting the area under the curve equal to the cost, i.e. our assumption is,

$$\int_0^1 a_i \cdot (y_i \cdot (-\log f(g(x_i, \beta)))) \, df = c_i \quad (3)$$

This is achieved by setting $a_i = c_i$ and $b_i = 1$. This means that instances with a higher cost will get a higher average loss value. However, this solution might not well represent the intuition that the cost of misclassification will be higher because it also increases the loss for correctly classified instances.

About optimization, this cost function is non-differentiable and non-convex, making it challenging for traditional optimization algorithms to find its global minimum. To overcome this, we will use a Genetic Algorithm, specifically the Differential Evolution (DE) algorithm the one which we have already mentioned above, to find the global minimum of the cost function.

IV. Results

In this study, we use a specific cost function to evaluate three different models like Logistic Regression model using Neural Networks, Bahnsen's Method, Nikou Günnemann's Method. The cost function used for evaluation is defined as follows:

$$J_cost(\theta) = \frac{1}{N} \sum_{i=1}^{N} [y_i(c_i * 6 + (1 - c_i)CFN_i) + (1 - y_i)(c_i * 6 + (1 - c_i) * 0) \quad (4)$$

where y_i is the true label of the i-th example, c_i is the predicted label of the i-th example generated by the classifier, CFN_i is the cost of false negatives for the i-th example, N is the total number of examples. This cost function is used to calculate the total cost of predictions made by the models on the dataset.

A. Logistic Regression results

Our logistic regression model has been train on pytorch and using binary cross entropy as loss function has given use the best possible accuracy on our test set but in term of total cost it didn't worked well the results are as follows:

TABLE I
Model Evaluation

Model	Accuracy	Total Cost
Logistic Regression	86.27%	337088.46

B. Bahnsen's Method Results

The Bahnsen's method provided different results compared to the logistic regression model. While it may not have achieved the highest accuracy, it performed significantly better in terms of total cost. The results are as follows:

TABLE II
Model Evaluation

Model	Accuracy	Total Cost
Bahnsen's Method	76.56%	50508.05

C. Nikou Günnemann's Method Results

The Nikou Günnemann's method provided different results compared to both the logistic regression model and Bahnsen's method. This method had a lower accuracy but performed differently in terms of total cost. The results are as follows:

TABLE III
Model Evaluation

Model	Accuracy	Total Cost
Nikou Günnemann's Method	30.31%	88458.00

V. Conclusion

In conclusion, for our dataset Bahnson's method works best, this study highlights the importance of considering both accuracy and cost when evaluating models for binary classification problems. While accuracy is an important metric, it does not take into account the potential costs associated with misclassification. By using a cost function that incorporates these costs, we can more effectively evaluate and compare different models.

References

[1] N. G unnemann and J. Pfeffer, "Cost Matters: A New Example-Dependent Cost-Sensitive Logistic Regression Model," Technical University of Munich, Munich, Germany.

[2] A. C. Bahnsen, D. Aouada and B. Ottersten, "Example-Dependent Cost-Sensitive Logistic Regression for Credit Scoring," 2014 13th International Conference on Machine Learning and Applications, Detroit, MI, USA, 2014, pp. 263-269, doi: 10.1109/ICMLA.2014.48.