Assignment 3: Example-dependent cost-sensitive regression

Ranveer Sahu ($es21btech11025$)
Aabisheg T ($es21btech11001$)
S Jagadeesh ($es21btech11026$)
Subiksha Gayathiri ($es21btech11031$)
Bhende Adarsh Suresh ($cs21btech11008$)

Indian Institute of Technology, Hyderabad

Abstract

Machine learning algorithms typically assume that all classification errors have equal importance, but this is not always true for imbalanced classification problems. In these scenarios, missing a positive or minority class example is more significant than misclassifying a negative or majority class example. This is relevant in various real-world applications, such as medical diagnosis, spam detection, and fraud identification. A false negative can be more costly than a false positive in such cases, and these cost-sensitive issues occur in different fields like medicine(e.g., disease detection), transport(e.g., traffic-jam detection), engineering(e.g., machine failure detection), and finance.(e.g., fraud detection), etc.
In the context of loan applications where a bank wants to decide upon granting a loan or not, it is better for a bank to deny a loan to a good customer than to approve a loan for a bad customer who may never repay it. In the case of an insurance company determining whether a claim is fraudulent, it is better to identify good claims as fraudulent and follow up with the customer than to approve fraudulent insurance claims.

1. Problem Statement

Logistic Regression is one of the most popular Machine Learning methods that unlike the name suggests, is a classification method. Given the data, it predicts the probability of a sample belonging to class 1. Logistic Regression uses a Binary cross Entropy Loss function that provides an equal cost of misclassification.
Below are some models that we have experimented with:

- Behnsen approach Logistic Regression
- Nikou and Gunnemann's approach Logistic Regression

The loss function for Logistic Regression is as follows:

$$\mathrm{L}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

where ŷ is the probability of the sample being classified to class 1 and is given by:

$$\hat{y} = \sigma(w_1x_1 + w_2x_2 + \ldots + w_nx_n + b)$$

where (.) is the sigmoid function, which converts w1 x1 + w2 x2 + . . . wn xn + b) into a number between 0 and 1 and is given by:

$$\sigma : \mathbb{R} \to [0,1]; \quad \sigma(x) = \frac{1}{1+e^{-x}}$$

Now, we shall now look at Cost Sensitive Logistic Re- gression, which shall provide unequal costs of misclassi- fication, i.e the model shall wish to minimize the error of prediction while also taking care of the cost of misclassifi- cation. The loss function shall now be modified as:

$$L(y, \hat{y}) = \sum_{i=1}^{N}[y_i(C_TP\hat{y} + C_FN(1-\hat{y})) + (1-y_i)(C_FP\hat{y} + C_TN($$

Here, we can see the error is now being weighted by the costs of misclassification, and thus, the model is forced to optimize such that the costs of misclassification are taken into account.

2. Description of the Dataset

The following are the columns in the dataset:

- A : NotCount (Integer)
- B : YesCount (Integer)

- C : ATPM (Float)
- D : PFD (Float)
- E : PFG (Float)
- F : SFD (Float)
- G : SFG (Float)
- H : WP (Float)
- I : WS (Float)
- J : AH (Float)
- K : AN (Float)
- L : Status (Boolean)
- M : FNC (Float)

1. Columns A to L are independent variables.
2. Column L is the dependent variable.
3. Column M is the false negative cost, varying from row to row based on the risk details.
4. True Positive and False Positive cost is constant for all, which is 6
5. True Negative cost is constant for all, which is 0

3. Data Exploration

3.1. Statictics

There is a total 13 columns and 147637 rows entries in the dataset. The dataset is highly-imbalanced, positive labels are (29%) whereas, negative labels are (70%). It is observed that majority (75%) of values in PFD, PFG, SFD, SFG, AH and AN are zero, indicating the label 0 values more likely corresponds to them. The 13th column, 'FNC' is the false negative cost for each row.

Apart from this, the True Positive and False positive costs are 4 for all rows, and the True Negative cost is 0 for all rows.

A snapshot of the dataset is given below:

	NotCount	YesCount	ATPM	PFD	PFG	SFD	SFG	WP	WS	AH	AN	Status	FNC
0	2	21	0.0	0.000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0	0.0
1	23	0	0.0	0.044	0.0	0.0	0.0	0.306179	0.0	0.0	0.0	1	0.0
2	1	22	0.0	0.000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0	0.0
3	5	18	0.0	0.000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	1	0.0
4	1	22	0.0	0.000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0	0.0

Fig 1: A snapshot of the dataset

3.2. Correlation Matrix of Features

It can observed from Fig. 2, that **Status** is highly correlated to **NotCount** & **YesCount** which means they can give good accuracy using logistic regression.

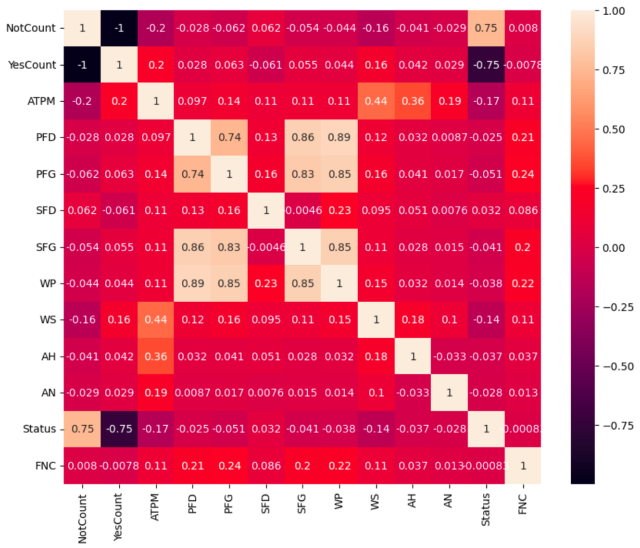

Fig 2: The correlation matrix is a table that displays the correlation coefficients for different variables. The matrix depicts the correlation between all the possible pairs of values in a table.

4. Algorithm and Methodology

We shall implement logistic regression using a one-layer Neural Network (no hidden layers) followed by a sigmoid activation to convert it into a probability.

1. We shall break the dataset into X, which is all the independent features; Y, which is the column 'Status, the dependent variable, and FNC, the False Negative cost for each row.
2. We shall then split the data into training and testing data in the ratio of 80:20 by picking random indices. In this way, we split X, Y, and FNC into training and testing components.
3. We check the statistics of the False Negative Costs and find that it becomes very high for certain rows, as compared to the other costs which are of the order of 100. This results in the model only predicting $\hat{y} = 1$ since that takes the False Negative Cost out of the Equation. Thus, we normalize the False Negative Cost to be constrained between 0 and 5. We also assume that the testing samples will arrive one-by-one; hence, we cannot normalize the False Negative cost for the testing samples using the maximum and minimum of the

testing samples. Thus, we assume the maximum and minimum of the training samples is used in normalizing the False Negative Cost of the testing samples. Finally, we assume the False Negative cost is known to us and can be fed to the model, and hence, can be used as a predictive feature since the False Negative Cost should also have a say during testing.

```
count    1.476360e+05
mean     5.334049e+02
std      8.774011e+03
min      0.000000e+00
25%      2.820820e-01
50%      1.183562e+01
75%      1.069840e+02
max      1.703186e+06
Name: FNC, dtype: float64
```

Fig 3: A snapshot of the statistics of the False Negative Costs Columns.

4. We define the Logistic Regression Model as a one-layer Neural network with no Hidden Layer and a sigmoid activation function at the end. We observed that random initialization of the weights led to a significant change in our results.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Where: z is the input to the function.

Thus, we decided to initialize the weights using Kaiming Initialization. We use an Adam optimizer with a learning rate of 0.005.

5. We train two different models: one with the Bahnsen and one with the Nikou Gunnemann's approach with their loss functions.

6. Finally, we also find the number of misclassification made by both models and compare the two.

4.1. Classification cost matrix

In the table below, the cost matrix is presented, where the cost associated with two types of correct classification, the true positives C_{TP}, and the true negatives C_{TN}; and the two types of misclassification errors, false positives C_{FP}, and false negatives C_{FN}, are shown in the table.

Cost Sensitive Logistic Regression		
	Actual Positive	Actual Negative
Predicted +ve	C_{TP}	C_{FP}
Predicted -ve	C_{FN}	C_{TN}

$$C_{Total} = [C_{TP}, C_{FP}, C_{FN}, C_{TN}] \quad (1)$$

$$C_{Total} = \begin{cases} C_{TPi} \textbf{ if } y_i = 1 \textbf{ and } w_\theta(X_i) \approx 1 \\ C_{TNi} \textbf{ if } y_i = 0 \textbf{ and } w_\theta(X_i) \approx 0 \\ C_{FPi} \textbf{ if } y_i = 0 \textbf{ and } w_\theta(X_i) \approx 1 \\ C_{FNi} \textbf{ if } y_i = 1 \textbf{ and } w_\theta(X_i) \approx 0 \end{cases}$$

5. Results

1. The training and testing curve for the Bahnsen Model is given below

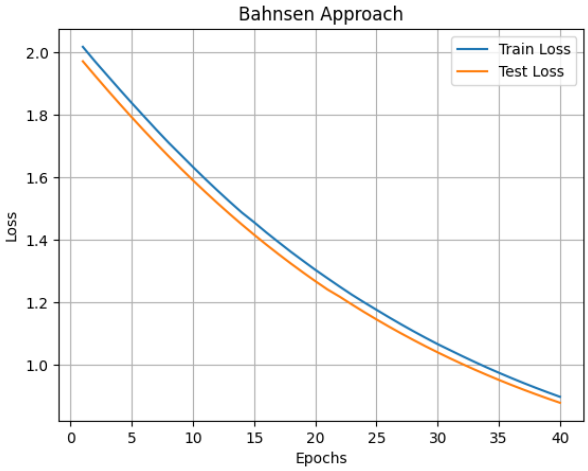

Fig 4: Test Cost Independent Loss for each varient

After applying seeding, an accuracy is 70.64%

2. The training and testing curve for the Nikou and Gunnemann's Model is given below:

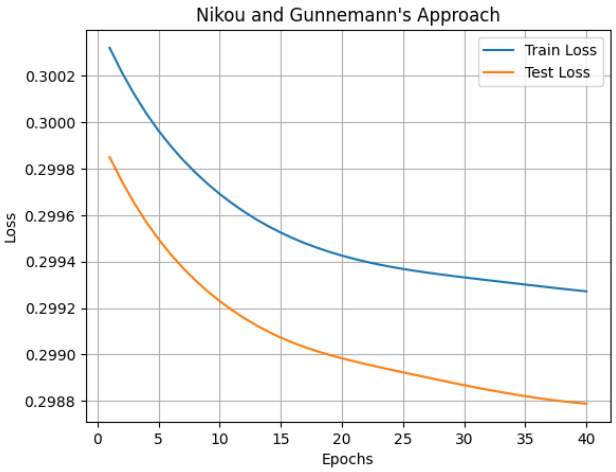

Fig 5: Test Cost Dependent Loss for each varient

After applying seeding, accuracy is 30.1%

3. The accuracy of the Bahnsen Model turns out to be 70.64
4. We compare the number of misclassifications for both models:

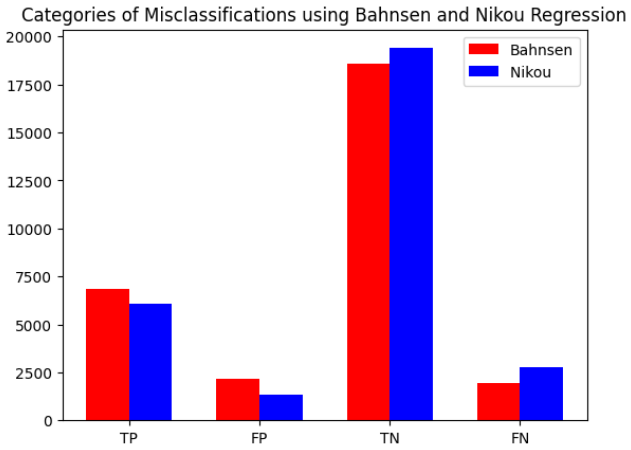

Fig 6: Training and Testing loss for the both Model

5. The accuracy fluctuation is due to our data, as the cost function (or loss function) is non-convex. Consequently, when we run the code, we observe varying accuracies. If we apply seed, we will get a particular accuracy.
6. We find that the number of False Negative Predictions immediately goes down since they have a high cost associated with them. On the other hand, the number of True Negatives goes up since they do not have a cost associated with them. It can also be seen using the act that if the true label y = 0, then if the prediction of $\hat{y}$ goes down, then the prediction of $\hat{y}$ needs to go up. We also find that the Normal model almost never correctly predicts the class y = 1. Thus, we can say that the cost-sensitive loss function has brought about a change in the way the model Learns.