Cost Sensitive Logistic Regression

Anirudh Srinivasan
CS20BTECH11059

Haritha R
AI20BTECH11010

Nelakuditi Rahul Naga
AI20BTECH11029

Rahul V
AI20BTECH11030

Saketh Vaddamani
CS20BTECH11054

1. Problem Statement

We will first discuss about the disadvantages of Vanilla Logistic Regression and how Cost-Sensitive Logistic Regression alleviates them.

1.1. Introduction

A potent method for classification issues where the costs of various types of errors are not equal is cost-sensitive logistic regression. By reducing the projected total cost of classification errors rather than just the misclassification rate, it can help the classification model's overall accuracy.

A variation to the standard logistic regression algorithm called cost-sensitive logistic regression accounts for the expenses related to various classification errors. Traditional logistic regression does not explicitly take into consideration the costs of false positives and false negatives; instead, the goal is to minimise the negative log-likelihood of the training data.

The goal of cost-sensitive logistic regression is to reduce the weighted sum of the negative log-likelihood and classification error costs. The costs of false positives and false negatives, which vary based on the particular application, are what decide the weights.

A false negative, or a patient who is ill but is diagnosed as healthy, may have substantially higher expenses than a false positive, or a patient who is ill but is classified as healthy, for instance, in a medical diagnosis problem. The weight given to false negatives should therefore be greater than the weight given to false positives.

Using an iterative optimisation approach like gradient descent, the cost-sensitive logistic regression algorithm estimates the weights and the logistic regression parameters simultaneously. The class labels of additional data points can then be predicted using the resulting model, which accounts for the costs of various classification failures.

1.2. Motivation

The goal of cost-sensitive logistic regression is to create a classification model that accounts for the expenses incurred in the actual world as a result of various classification failures. False positives and false negatives have different costs in many applications, and misclassifying a data piece might have harmful effects. For instance:

1. Misdiagnosing a patient as healthy when they are in fact ill can have serious repercussions, including delayed care or even death. On the other side, misdiagnosing a healthy patient as ill might result in anxiety, pointless testing, and therapies.
2. Misclassifying a fraudulent transaction as legal in fraud detection can result in considerable financial losses, while misclassifying a normal transaction as fraudulent can result in inconvenience and mistrust.
3. While misclassifying a legitimate email as spam can result in missed opportunities and lost productivity, doing the opposite can result in security breaches and privacy violations.

Cost-sensitive logistic regression can assist reduce the overall cost of classification errors rather than just the misclassification rate by factoring the costs of various types of classification errors into the classification model. This can result in decision-making that is more precise and economical across a range of applications.

2. Description of the dataset

The dataset contains 13 columns labelled as A, B, C, D, ..., L, M as shown in format table 1. The columns A to K contains all independent variables. They can be thought of as to be used as features for training and testing purposes of our Cost-Sensitive Logistic Regression Model. The Column L contains dependent variable. So, can be thought of as a label corresponding to all the features from the column A

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
NotCount	**YesCount**	**ATPM**	**PFD**	**PFG**	**SFD**	**SFG**	**WP**	**WS**	**AH**	**AN**	**Status**	**FNC**
2	21	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0.044	0	0	0	0.306179	0	0	0	1	0
1	22	0	0	0	0	0	0	0	0	0	0	0
5	18	0	0	0	0	0	0	0	0	0	1	0

Table 1. Format of the dataset

to K. Then the last column M contains false negative cost, which is varying from row to row based on Business details. We also were given the True Positive and False Positive Cost, which are constant for all rows and is 4. We see that the traditional logistic regression doesn't explicitly take care of how false negative cost is varying from row to row. So, Cost-Sensitive Logistic Regression is used on this dataset.

3. Algorithms used

Cost-sensitive logistic regression (CSLR) algorithm has similar architecture as a Logistic regression model. The main difference is that CSLR model consists a **new cost function**. The new cost function takes into account the real costs due to misclassification and correct classification. We will first briefly discuss about Logistic regression and then provide the algorithm for Cost-sensitive logistic regression.

3.1. Logistic Regression

Logistic regression is a binary classification model that estimates the probability of the class $y = 1$ as the logistic sigmoid of a linear function of the feature vector. The estimated probability is evaluated as follows :

$$\hat{p}_i = P(y = 1|x_i) = \frac{1}{1 + e^{-\sum_j \theta^j x_i^j}} \tag{1}$$

Let us denote $P(y = 1|x_i)$ as $h_\theta(x_i)$ for notational brevity. We need to find the parameters θ that **minimize** the cost function $J(\theta)$ defined as follows :

$$J(\theta) = \frac{\sum_{i=1}^{N} -y_i \log(h_\theta(x_i)) - (1 - y_i)(1 - \log(h_\theta(x_i)))}{N} \tag{2}$$

There are several methods used to estimate the optimal parameters θ as the logistic regression cost function $\mathbf{J}(\theta)$ **is convex**.

3.2. Cost Sensitive Logistic Regression

The logistic regression cost function described above implicitly assumes that false positives and false negatives have the same costs, i.e, $\mathrm{C}_{\mathrm{FP}_i} = \mathrm{C}_{\mathrm{FN}_i}\ \forall i \in \{1, \ldots, N\}$. This need not be true for real world applications such as Fraud detection.

We provide a new cost function that incorporates the different misclassification costs associated with false positives and false negatives as follows :

$$J^c(\theta) = \frac{1}{N} \sum_{i=1}^{N} y_i (h_\theta(x_i)\mathrm{C}_{\mathrm{TP}_i} + (1 - h_\theta(x_i))\mathrm{C}_{\mathrm{FN}_i}) \\ + (1 - y_i)(h_\theta(x_i)\mathrm{C}_{\mathrm{FP}_i} + (1 - h_\theta(x_i))\mathrm{C}_{\mathrm{TN}_i}) \tag{3}$$

Note that the above cost function is not convex, hence we need to estimate the optimal parameters θ using other cost optimization algorithms.

3.3. Implementation of CSLR using PyTorch

We will use PyTorch to implement the CSLR algorithm since the loss function involved in this problem is Non-convex and Pytorch provides inbuilt modules for loss backpropagation and parameter updates.

Before defining the functions for CSLR implementation, we divided the given dataset into train and test datasets (in the ratio 7 : 3) using sklearn's **train_test_split** module.

Next we defined the Logistic Regression module that performs the forward propapgation as discussed earlier.

We then defined the Cost sensitive loss function using the defintion mentioned in equation (3) and appropriate True Positive, True Negative, False Positive and False Negative costs. Note that True Positive, True Negative, False Positive costs are constant for all the datapoints - 4,0,4 respectively. The False negative cost is unique to each datapoint.

Then, we trained the model on training data and tested it on test data. The training is done for 20,000 epochs and testing is done every 200 epochs. The loss backpropagation and parameter updates are done using the functions **train_loss.backward()** and **optimizer.step()** functions respectively. The optimizer we have used for training the model is the inbuilt Adam module provided by PyTorch.

4. Final Results and Conclusions

The plots pertaining to training loss, training accuracy, test loss and test accuracy as the epochs progress are as follows :

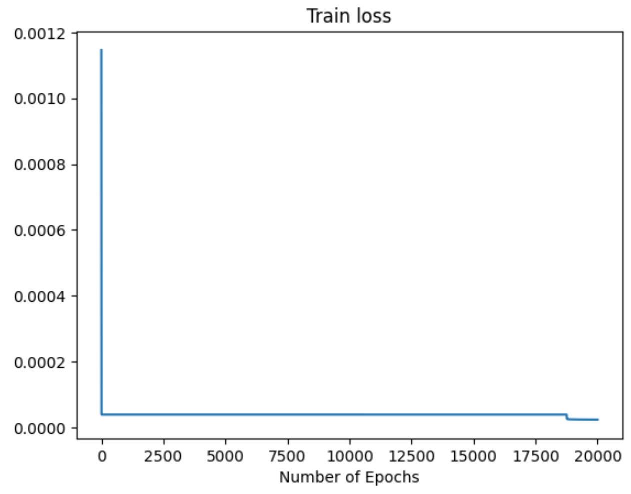

Figure 1. Train Loss

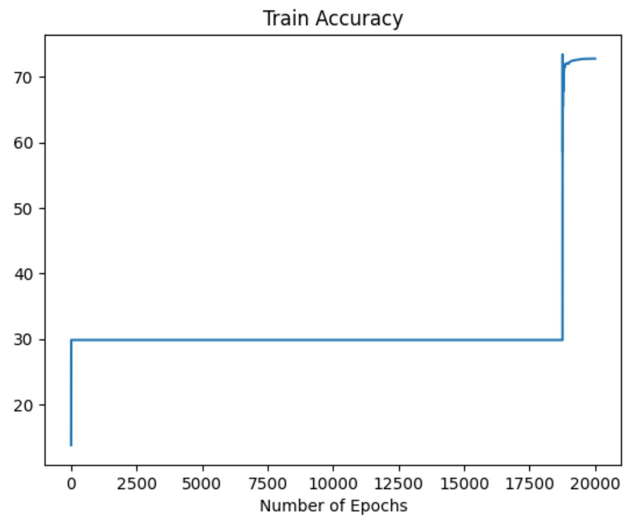

Figure 2. Train Accuracy

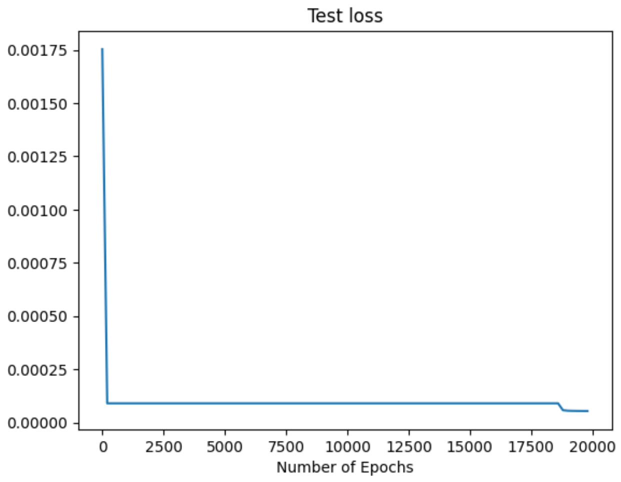

Figure 3. Test Loss

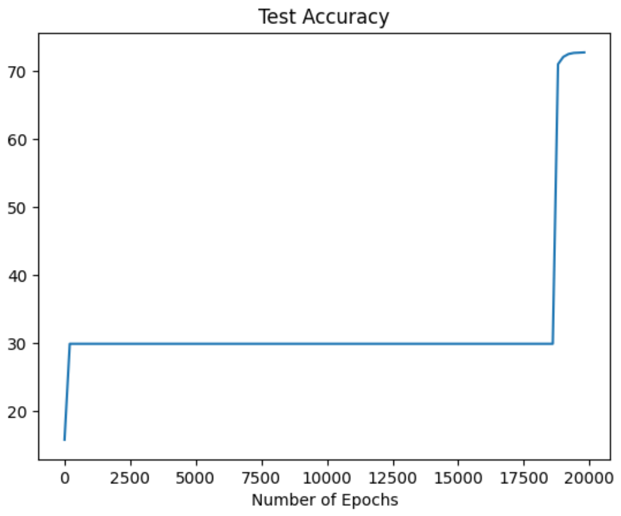

Figure 4. Test Accuracy

References

[1] Logistic Regression with PyTorch - Link

[2] CSLR slides shared through AIMS