Report on

Example Dependent Cost Sensitive Regression

Arnab Ghosh, Kritik Agarwal, Raghavendra Kulkarni, Shagun Sharma, and Trishita Saha

Instructor: Dr. Sobhan Babu Chintapalli
Fraud Analytics using Predictive and Social Network Techniques (CS6890)
Dept. of Computer Science And Engineering
Indian Institute of Technology, Hyderabad

ABSTRACT

Logistic Regression is a classical machine-learning problem where the machine learns to classify the data points into two or more classes. A standard Logistic Regression problem defines the log loss cost function as a function of the given output class labels and the class labels predicted by the model. However, this log loss function gives an equal cost to all the combinations of class label pairs, namely True Positive, True Negative, False Positive, and False Negative predictions. But in some applications like the Cancer Prediction task, the cost of False Negative predictions can be higher than that of False Positive predictions. Hence, the cost function has to be modified to assign different costs for different types of class label predictions. The Cost-Sensitive Regression problem is a candidate solution where we slightly modify the cost function to incorporate these costs. This work discusses two different approaches: the Bahnsen's Approach and the Gunnemann's Approach for a Cost-Sensitive Regression problem. The results compare the Misclassification Cost of the Cost-Sensitive Regression models with the standard Logistic Regression model.

Keywords: Cost-Sensitive, Example-dependent, Misclassification, Regression

PROBLEM STATEMENT

Given a Classification Dataset with 11 features, a Class Label, and the various classification costs, train a Cost-Sensitive Logistic Regression model to predict the class label for the data points while minimizing the overall classification cost.

DATASET DESCRIPTION

The dataset contains one .csv file, named as *costsensitiveregression.csv*. Table 1 below shows the first 5 entries of the *costsensitiveregression.csv* file.

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
1	22	0	0	0	0	0	0	0	0	0	0	0

Table 1. The first 5 entries of *costsensitiveregression.csv*

The .csv file contains a total of 13 columns:

- The first 11 columns represent the features of the data point
- The 12th column represents the class label of the data point
- The last column contains the False Negative Cost of the data point

The other three classification costs, namely the True Positive cost, the True Negative cost and the False Positive cost are constant for each data point and are 6, 0 and 6 respectively.

THE COST SENSITIVE REGRESSION MODEL

The Cost Sensitive Regression model differs from a standard Logistic Regression model only in terms of the cost function and the optimization approach of this cost function. In a standard Logistic Regression problem, the cost function to minimize is given by:

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

In this section we discuss two approaches for modifying this cost function and incorporate the individual classification costs of the data points into the cost function.

Bahnsen's Approach

In this approach we eliminate the log of the predicted values and bring in all four combinations of the classification. With the classification costs of the i^{th} data point as $C_{TP_i}, C_{TN_i}, C_{FP_i}$ and C_{FN_i}, we re-write the classification cost function as:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i(C_{TP_i} h_\theta(x_i) + C_{FN_i}(1-h_\theta(x_i))) + (1-y_i)(C_{FP_i} h_\theta(x_i) + C_{TN_i}(1-h_\theta(x_i)))$$

This cost function is then minimized over θ to get the optimal solution model.

Gunnemann's Approach

In this approach we eliminate the log of the predicted values and bring in all four combinations of the classification. With the classification costs of the i^{th} data point as $C_{TP_i}, C_{TN_i}, C_{FP_i}$ and C_{FN_i}, we re-write the classification cost function as:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i(C_{TP_i} \log h_\theta(x_i) + C_{FN_i}\log(1-h_\theta(x_i))) + (1-y_i)(C_{FP_i} \log h_\theta(x_i) + C_{TN_i}\log(1-h_\theta(x_i)))$$

This cost function is then minimized over θ to get the optimal solution model.

IMPLEMENTATION

We make use of the ***scipy.optimize*** package in Python for optimizing the cost function since the cost function is now not guaranteed to be differentiable or convex. The ***scipy.optimize*** is a Python package that provides advanced function optimizing methods using genetic algorithms which work well for even complex and non-differentiable or non-convex functions. We also use the ***scikit-learn*** package in Python to evaluate the classification costs from a standard Logistic Regression model to compare and show the improvements.

Preprocessing

- We examine the min-max range of all the features in the dataset and check for any missing values.
- Then, we normalize all the features except the False Negative Cost column using a MinMaxScaler. This is essential for accurate results.
- We finally split the dataset into training and testing data with 10% of the dataset for testing.

scipy.optimize

The main functionality of scipy.optimize used for implementation are:

- ***Differential Evolution*:** This function is used the optimal point which minimizes the cost function. The function evaluates global minimum of any multivariate function. Since the function does not make use of function gradients, the function need not be differentiable always. The function can handle larger area of candidate solution space. However, the number of function evaluations is quite large when compared to gradient based optimization methods.

RESULTS

We first trained the standard Logistic Regression model and computed the classification costs. Then we fit the data for a Bahnsen's Cost Sensitive Logistic Regression model and computed the classification costs. Finally, we fit the data for a Gunnemann's Cost Sensitive Logistic Regression model and computed the classification costs. The Table 2 below tabulates the classification costs of training and testing data for all the three models.

Models	Training Accuracy	Training Cost	Testing Accuracy	Testing Cost
Standard Logistic Regression	0.8667	4136710.5680	0.8622	394769.0613
Bahnsen's Logistic Regression	0.7699	446740.8989	0.7649	50819.3214
Gunnemann's Logistic Regression	0.2982	797232	0.3017	88584

Table 2. The classification costs of training and testing data for all the three models.

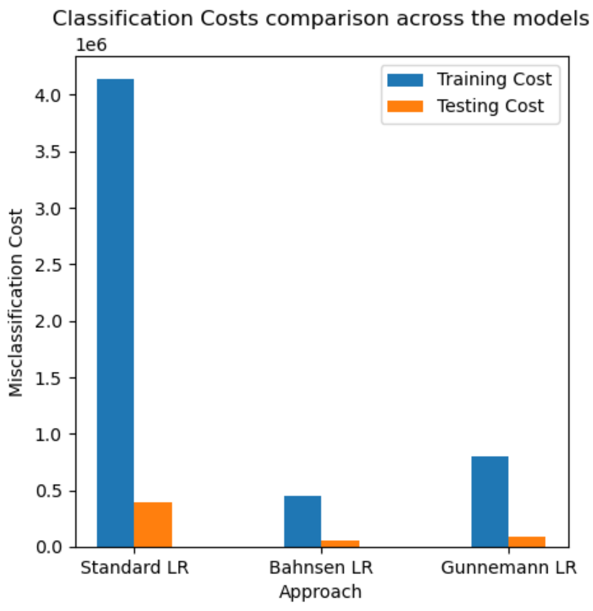

Figure 1. Classification Costs comparison across the models

Inference

Figure 1 above shows the Classification Costs comparison across the models.

- From Table 2 and Fig. 1 above, we can infer that the Cost Sensitive Regression models are successful in reducing the overall classification cost.
- Specifically, the Bahnsen approach has turned out to be the efficient model with an acceptable accuracy and optimized overall classification cost.

CONCLUSION

In this work, we implemented two variants of the Cost Sensitive Regression model, namely the Bahnsen approach and the Gunnemann approach

REFERENCES

[1] N. Günnemann and J. Pfeffer, "Cost matters: A new example-dependent cost-sensitive logistic regression model," in *Advances in Knowledge Discovery and Data Mining* (J. Kim, K. Shim, L. Cao, J.-G. Lee, X. Lin, and Y.-S. Moon, eds.), (Cham), pp. 210–222, Springer International Publishing, 2017.