Classification of Unbalanced Data with Unequal Cost

Gan Luan

December 11, 2019

1 Abstract

For binary classification, unbalanced data refer to the data with one class being dominant. With common classification methods, such as logistic regression, Support Vector Machine (SVM), the miss classification rate for class with smaller proportion tends to be high. However, there are some scenarios that correct classification of the class with smaller proportion is crucial, more crucial than correct classification of the class with larger proportion. It is that case that cost of miss classification of one class is greater than that of the other class (Unequal cost). Classic classification methods do not work well in these cases. Here in this course project, I promoted a new group of classification methods based on SVM. In SVM, training error from both cases are assigned with equal cost. Here in new methods, different cost is assigned for training error from each class. Two different form of cost function were promoted in this study. Simulation study were conducted to compare SVM and the new model based on the total overall loss. New models outperform SVM in simulation study. Thus the new model works better than SVM in classification of unbalanced data with unequal cost.

2 Introduction

2.1 Unbalanced Data

In binary classification, both balanced data and unbalanced are common. Balanced data refers to the data in which two classes have same or relatively similar proportion. While unbalanced data refers to data in which two classes have relative different proportions. To illustrate the possible problem associated with classification of unbalanced data, I generated and plotted the following two set of data, balanced data and unbalanced data (see Fig. 1). Both data sets containing data from two classes (red and blue dots in the figure). Coordinates for each color of dots, $\boldsymbol{x} = (x_1, x_2)$, are generated from the same distribution. The only difference is the number of dots in each data set. Also SVM were applied in both cases and decision boundary from both cases were plotted together in the unbalanced data plot. It is clear that compared to the balanced data case, decision boundary in unbalanced case moves towards red dots, the class with smaller proportion. This suggests that it is more likely to miss classify for class with smaller proportion.

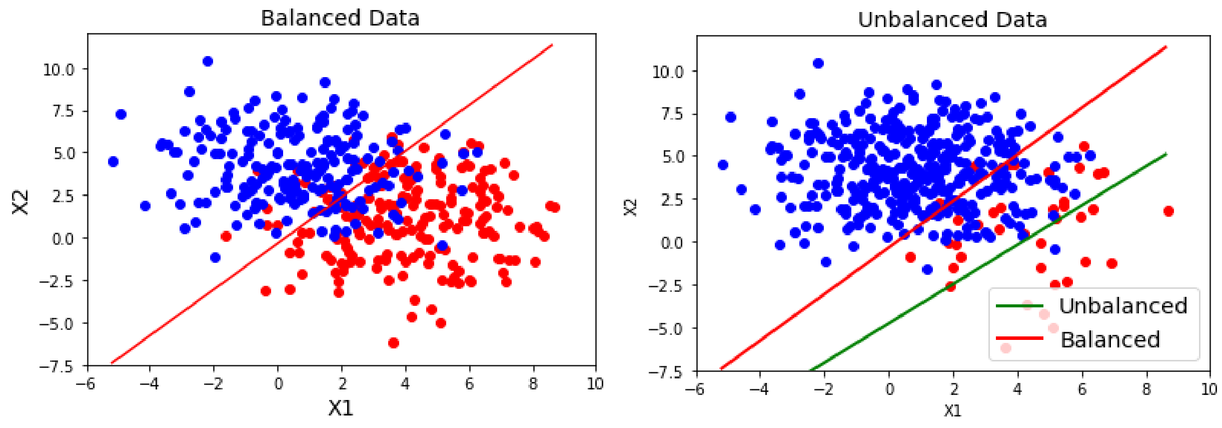

Figure 1: Comparison of SVM in balanced data and unbalanced data. Coordinates for blue dots were generated from $\boldsymbol{N}(\begin{bmatrix}1\\4\end{bmatrix}, \begin{bmatrix}5 & 0\\0 & 5\end{bmatrix})$; coordinates for red dots were generated from $\boldsymbol{N}(\begin{bmatrix}4\\1\end{bmatrix}, \begin{bmatrix}5 & 0\\0 & 5\end{bmatrix})$. 200 blue dots and 200 red dots in left panel; 360 blue dots and 40 red dots in right panel. Straight line are decision boundary from SVM.

To further study the relationship between overall prediction accuracy and prediction accuracy of one class and proportion of that class in the data, a simulation study were performed. In this simulation study, a series of data containing two classes, class A and class B, were generated. The proportion of class A varies from 0.1 to 0.9 in these data. SVM were applied for each data. Overall prediction accuracy and prediction accuracy for class A were calculated and plotted against proportion of class A (Fig. 2). This plot suggests that when we have one dominant class, the overall prediction accuracy is high. Also the prediction accuracy for one class is positive correlated with its proportion in the data.

This is easy to understand, since the classification algorithm is developed to minimize the overall classification error. Dominant class contributes more to the overall classification error than the minor class. Thus dominant class tends to have a high prediction accuracy while the minor class tends to have a low prediction accuracy.

However, there are many scenarios that correct classification of the minor class of unbalanced data is crucial. One good example is the detection of fraudulent credit card transaction. Clearly fraudulent transaction only takes a very small part of the total transactions. However, correctly identifying these transactions can prevent a lot loss for credit card company. In terms of loss, miss classification of one fraudulent transaction may have bigger loss than miss classification of one normal transaction. Thus we need to make sure the prediction accuracy of fraudulent transaction is high enough. However, we do not want to be too conservative that classifying normal transaction to be fraudulent. That will greatly affect the user experience. Other examples include detection of rare disease from the population, identification of genes associated with cancer development.

One way to solve problem in above scenarios is to use unequal cost for two types of classification. Still with the fraudulent transaction detection example, we can assign bigger cost for miss classification of fraudulent transaction to normal transaction than that of miss classification of the other direction. Then we try to develop a method that can minimize the overall cost. This is the problem studied in this course project.

2.2 Problem Statement

The problem to study can be defined as following:

- Sample size n, for this course project, we assume data only has two covariates $\boldsymbol{x} = (x_1, x_2)$
- Let y denotes the two classes: $+1$ and -1, each with proportion of p_1 and p_2
- **Unbalanced data:** $p_1 < p_2$
- **Unequal cost:** $C(+1, -1)$: cost of miss-classifying class $+1$ to class -1; $C(-1, +1)$: cost of miss-classifying class -1 to class $+1$. Let $C(+1, -1) = \alpha C(-1, +1)$, $\alpha \geq 1$, w.o.l.g. assume $C(-1, +1) = 1$

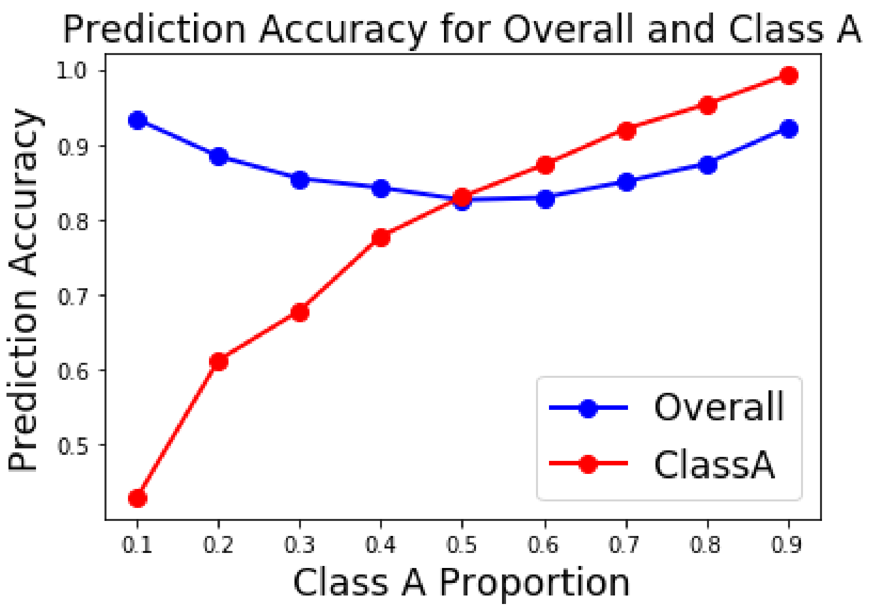

Figure 2: Relation between prediction accuracy and proportion of one class. Sample size for this simulation is 2000. SVM were applied on each data with its best cost got from cross validation.

- In this course project, only search for linear decision boundary
- **Goal:** develop new methods with small overall cost

A Mathematical statement for this problem is:

- We try to find a decision function

$$f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 = \beta_0 + \boldsymbol{\beta}^T \boldsymbol{x}, \text{where } \boldsymbol{\beta} = (\beta_1, \beta_2)^T$$

- Data with coordinates $\boldsymbol{x}$ is classified as following:

$$\hat{y} = \begin{cases} +1, & \text{if } f(\boldsymbol{x}) \geq 0 \\ -1, & \text{if } f(\boldsymbol{x}) < 0 \end{cases}$$

- Problem to solve:

$$argmin_{\beta_0,\beta_1,\beta_2} \sum_i \{\alpha I[y = 1] I[f(\boldsymbol{x}) < 0] + I[y = -1] I[f(\boldsymbol{x}) \geq 0]\} \tag{1}$$

3 Possible Solutions

The objective function to optimize in eq. 1 is not a convex function, thus it is not easy to be solved directly. In this project, I tried two types of methods: grid search and a tractable approximation of the objective function.

3.1 Grid Search

One way to directly solve the optimization problem without any techniques is to use grid search. First we set the possible values for β_0, β_1, and β_2, then for each possible combinations of β_0, β_1, and β_2 training loss is calculated. We take the combination that provides the lowest training loss as the final parameter estimate. Clearly this method has several disadvantages. First, there is no guideline for setting the possible values for our parameters. The true parameter values may be outside these set values. Also this is computational heavy. This method can quickly become infeasible when sample size increases or the dimension increases. We tried this method in this project as a control.

3.2 Tractable Approximation of the Objective Function

Another way to solve the problem is to use a tractable approximation of the original objective function. Then we try to optimize the approximation. If the approximation is close enough to the original objective function, the estimate from optimizing the approximation should be close enough to estimate that optimizes the original objective function. Here we build our approximation based on SVM.

The objective function of SVM is:

$$\begin{aligned} & min_{\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}} && \frac{1}{2}||\boldsymbol{\beta}||^2 + C(\sum_i \xi_i) \\ & \text{subject to} && y_i(\boldsymbol{x}^T\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \ \forall i \\ & && \xi_i \geq 0, \ \forall i \end{aligned} \tag{2}$$

Here ξ_i is the number we 'lend' to the *ith* data point to make the constrain $y_i(\boldsymbol{x}^T\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i$ hold. The value of ξ_i is positive correlated to the training error. Thus we can generate the cost as a function of ξ_i, and use different cost for class $+1$ and class -1. The new objective function can be defined as:

$$\begin{aligned} min_{\beta_0,\boldsymbol{\beta},\boldsymbol{\xi}} \quad & \frac{1}{2}||\boldsymbol{\beta}||^2 + \sum_i \{I[y_i = +1]h(\xi_i) + I[y_i = -1]g(\xi_i)\} \\ \text{subject to} \quad & y_i(\boldsymbol{x}^T\boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \ \forall i \\ & \xi_i \geq 0, \ h(\xi_i) \geq 0, \ g(\xi_i) \geq 0, \ \forall i \end{aligned} \tag{3}$$

Here $h(\xi)$ is the loss for miss classifying class $+1$ to class -1 and $g(\xi)$ is the loss for miss classifying class -1 to class $+1$. And if we let $h(\xi) > g(\xi), \forall \xi \geq 0$, by setting optimal h, g we can approximate the relationship between $C(+1, -1)$ and $C(-1, +1)$. Also by selecting h and g with good conditions, the optimization function in eq. 3 can be easily solved. Two different groups of functions were tried for h and g for this project.

- Model 1
  - $h(\xi) = C_1\xi$
  - $g(\xi) = C_2\xi$, with $C_1 > C_2$
- Model 2
  - $h(\xi) = C_1\xi^2$
  - $g(\xi) = C_2\xi^2$, with $C_1 > C_2$

In model one we let the cost be the linear function of ξ, while in model 2, the cost function is the square of ξ. Clearly model 2 put more cost on ξ that is greater than 1. For h and g in these two models, the optimization problem in eq. 3 is a quadratic programming problem. Thus it can be solved easily, much easier than the original optimization problem in eq. 1. C_1 and C_2 are hyperparameters, their values can be found by cross validation. Grid search method and these two models were compared by a simulation study.

4 Simulation Result

For the simulation study a training data and testing data were created. Both data containing two classes, class $+1$ and class -1. Covariates $\boldsymbol{x}$ of class $+1$ were generated from $\boldsymbol{N}(\begin{bmatrix}4\\1\end{bmatrix}, \begin{bmatrix}5 & 0\\0 & 5\end{bmatrix})$, while covariates of class -1 were generated from $\boldsymbol{N}(\begin{bmatrix}1\\4\end{bmatrix}, \begin{bmatrix}5 & 0\\0 & 5\end{bmatrix})$. In both training and testing data, there are 80 pieces of data from class $+1$ and 320 pieces of data from class -1. Also we assign $\alpha = 3$, i.e. $C(+1,-1) = 3$, and $C(-1,+1) = 1$. I built the model by training data and calculate the loss of eq. 1 from testing data.

First, we apply SVM on training data and search for the best cost by cross validation, denote the best cost as C_{best}. For model 1 and model 2, we use cross validation to search for the best C_1 and C_2. For each model, introduce t as the tuning parameter for cross validation. Let N_1 denotes the number of Class $+1$ data and N_2 denotes the number of Class -1. The relation of t, C_1, C_2, C_{best} is:

$$\begin{cases} N_1C_1 + N_2C_2 = (N_1 + N_2)C_{best} \\ C_1 = tC_2 \end{cases}$$

For Model 1 and Model 2, we searching for the best C_1 and C_2 for each model by searching t from $1, 1.02, 1.04, \ldots, 4.98, 5$. Also as a control, we also tried grid search. For grid search, the values set for β_0, β_1, and β_2 are same, which is $0.02, 0.04, \ldots, 0.98, 1$. The hyperparameters used for the final model, training and testinging loss are summarized in Table. 1. Clearly we can see that SVM preforms worst in the unbalanced and unequal cost setting. This is mainly because, SVM does not consider the unequal cost for two types of miss classifications. Model 1 and Model 2 have the same and lowest testing loss. Grid search has the lowest training loss but higher testing loss than Model 1 and Model 2. I did not perform cross validation for grid search due to high computation burden. Thus the parameters estimates from grid search may be over fitting the training data thus has a larger testing loss. Thus overall model 1 and model 2 performs best. Also for comparison, I plotted the decision boundary from all these four models together with the training data (See Fig. 3). Here red dots represent class $+1$, while blue dots represent -1. Compared to SVM, grid search, model 1 and model 2 move the decision boundary toward blue dots. This is the effect of higher cost for miss classifying red dots than the blue dots. Also decision boundary of model 1 and model 2 are very close and crossed. This suggests that for this training data, these two models provide similar result.

5 Thoughts and Future Plans

Several things learnt from this course project. First is about the idea of using approximation to solve intractable problems. We want the approximation be easier to solve. If

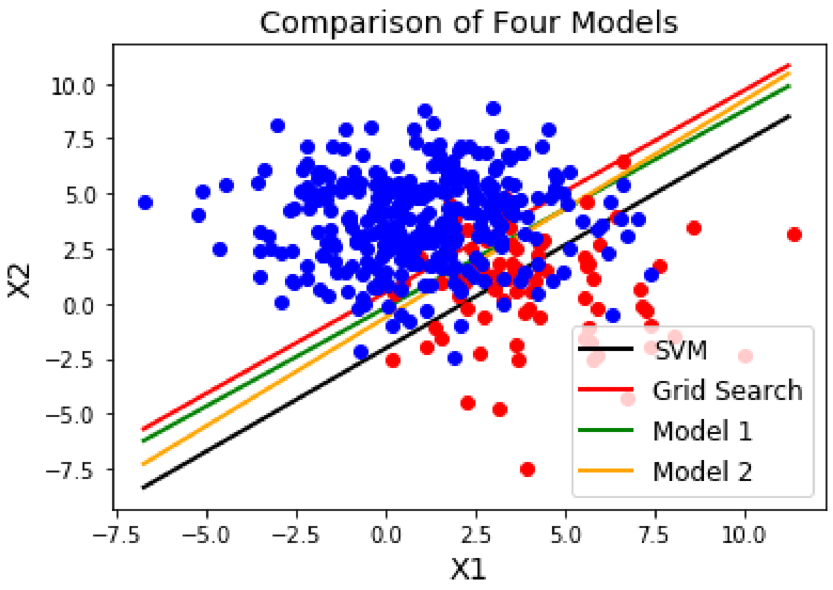

Figure 3: Comparison of four models

Table 1: Loss for each Model

Model	SVM	Grid Search	Model1	Model2
Hyperpara	$C = 0.2$	$\boldsymbol{\beta} \in [-1, 1]$	$C_1 = 0.385$, $C_2 = 0.154$	$C_1 = 0.405$, $C_2 = 0.150$
Training Loss	114	69	78	77
testing Loss	116	98	89	89

the approximation is close to the original problem, the solution from approximation should also be close to the true solution. Second, I learnt how to perform quadratic programming in python. This part takes longest time for this project. The library used is 'cvxopt'. There are several libraries available for quadratic programming. But for this project, this library works best. However, I realized that when $p_1 \ll p_2$, there are some problems regarding quadratic programming with this library. The details need more investigation. Last, though the whole discussion is based on classification of unbalanced data with unequal cost, the idea and the model is not limited to unbalanced data. This idea can also be applied to balanced data with unequal cost.

There are several directions worth trying for future research. First is about finding best C_1 and C_2. Here we use cross validation, which is not very efficient. It is reasonable to assume that best C_1 and C_2 in model 1 and model 2 depends on α and sample size: N_1, N_2. Thus we can try to write C_1, C_2 as the function of α, N_1, and N_2. Second, there are other forms of g and h we can use for cost. Two more examples are:

$$\begin{cases} h(\xi) = C_1(\xi + \xi^2) \\ g(\xi) = C_2(\xi + \xi^2), \ \ C_1 > C_2 \end{cases}$$

OR

$$h(\xi) = \begin{cases} 0, & \text{if } 0 \leq \xi \leq 1 \\ C_1(\xi - 1), & \text{if } \xi > 1 \end{cases}$$

$$g(\xi) = \begin{cases} 0, & \text{if } 0 \leq \xi \leq 1 \\ C_2(\xi - 1), & \text{if } \xi > 1, \ \ C_1 > C_2 \end{cases}$$

The first case combines the cost from model 1 and model2. This optimization problem is still a quadratic programming problem. The second model gives 0 loss for $\xi < 1$. $\xi < 1$ means the data point is correctly classified but crossing the boundary. However for the second case it is not a quadratic programming problem. We need to find new ways to solve the optimization problem in the second case. Last, we can extend our discussion here. Here we limit our discussion to the case that the covariates are of 2 dimensions. We can extend our discussion to higher dimensions. Also here we limit our research to linear decision boundary. Extension to non-linear decision boundary would be another direction for future research.