Tutorial - Project 2

Aytunc Sahin, Djordje Miladinovic

1 Multiclass Classification

In this tutorial we discuss ways of doing multiclass classification.

1.1 One-vs-All

Suppose we have C classes. In this case, we are going to train C classifiers. We construct classifier 1 by using 1 as a label for class 1 and 0 for the remaining classes. Classifier i is constructed by using 1 as a class label for class i and 0 for the remaining classes. After training C classifiers, we evaluate them for each new test point and choose the class which gives the maximum value. One potential problem with this approach is that for each classifier we might have class imbalance depending on the number of class and training data.

1.2 One-vs-One

In this case, we are going to train $\binom{C}{2}$ classifiers. We construct classifier c_{ij} between class i and j by only using training data from class i and j. For each new test point, we evaluate all $\binom{C}{2}$ classifiers and decide about the outcome using a voting scheme (e.g use the most frequent class label). However, in some cases you might get equal votes for some classes and you need to come up with a tie-breaking rule.

1.3 Multinomial Logistic Regression

We start with a summary of logistic regression. Suppose you have n data points and you are given $\mathscr{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ where each $x_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$. The logistic function is defined

$$p_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)}$$

Then we can define the cost function as

$$L(\theta) = -\sum_{i=1}^{n} y_i \log(p_\theta(x_i)) + (1 - y_i)\log(1 - p_\theta(x_i))$$

and optimize it. Now we are going to extend this framework for multiclass classification. The crucial difference is now $y_i \in \{1, \ldots, C\}$, there are C different classes. In this case you can write the cost function as

$$L(\theta) = -\sum_{i=1}^{n} \sum_{c=1}^{C} \mathbb{1}_{\{y_i = c\}} \log\left(\frac{\exp(\theta_c^T x_i)}{\sum_{k=1}^{C} \exp(\theta_k^T x_i)}\right)$$

2 Introduction to Project 2

In this project, you have 4800 training examples and 3 classes. However you also have class imbalance, namely you have 600 examples for class 1, 3600 examples for class 2 and 600 examples for class 3. Because of the class imbalance we are going to use balanced multiclass accuracy as a metric. Suppose we have the following confusion matrix:

$$M = \begin{bmatrix} 6 & 4 & 0 \\ 4 & 56 & 10 \\ 0 & 3 & 7 \end{bmatrix}$$

From the matrix, we can see that we have 10 samples coming from class 1, 70 samples from class 2 and 10 samples from class 3. For instance, we classified 4 of the class 2 instance as class 1, 56 of the class 2 instance as class 2 and 10 of the class 2 instance as class 3. To calculate the balanced multiclass accuracy, we calculate the true positive rate (sensitivity, recall) of each class and take the mean. For example, using the confusion matrix above, we calculate our metric:

$$\frac{\frac{6}{10} + \frac{56}{70} + \frac{7}{10}}{3} = 0.7$$

2.1 How to deal with class imbalance?

You can try to oversample (adding some instances from underrepresented class) or undersample (remove some instances from overrepresented class). As a second option you can also try to change your loss function using a cost sensitive classifier. Your original loss function is

$$L(\theta) = \sum_{i=1}^{n} l_i(\theta)$$

As you can see, this cost function puts equal weight to every data point. However, in our case, you might want to put more emphasis on the underrepresented classes. So, you can change the loss function by adding weights:

$$L_w(\theta) = \sum_{i=1}^{n} w_i l_i(\theta)$$

Please note that if you use sklearn library in Python, most of the models have a parameter called *class weight* and you can try different weights.