Improved Logistic Regression Algorithm Based on Kernel Density Estimation for Multi-Classification with Non-Equilibrium Samples

Yang Yu[1], Zeyu Xiong[1, *], Yueshan Xiong[1] and Weizi Li[2]

Abstract: Logistic regression is often used to solve linear binary classification problems such as machine vision, speech recognition, and handwriting recognition. However, it usually fails to solve certain nonlinear multi-classification problem, such as problem with non-equilibrium samples. Many scholars have proposed some methods, such as neural network, least square support vector machine, AdaBoost meta-algorithm, etc. These methods essentially belong to machine learning categories. In this work, based on the probability theory and statistical principle, we propose an improved logistic regression algorithm based on kernel density estimation for solving nonlinear multi-classification. We have compared our approach with other methods using non-equilibrium samples, the results show that our approach guarantees sample integrity and achieves superior classification.

Keywords: Logistic regression, multi-classification, kernel function, density estimation, non-equilibrium.

1 Introduction

Machine Learning has become one of the most popular fields in recent years. There are two main tasks of Machine Learning: 1) classification, which goal is to divide instances into the appropriate categories, and 2) regression, which goal is to study relationship between samples. The most basic classification problem is binary classification. which can be solved using algorithms such as Naive Bayes (NB), support vector machine (SVM), decision tree, logistic regression, KNN, neural network, etc. More generally, multi-classification problems such as identifying handwritten digits $0 \sim 9$, and and labeling document topics have gained much attention recently. To provide few examples, Liu et al. [Liu, Liang and Xue (2008)] proposed a multi-classification algorithm based on fuzzy support vector machines, which provides better classification accuracy and generalization ability compared with traditional One-*vs.*-Rest methods. Tang et al. [Tang, Wang and Chen (2005)] proposed

[1] HPCL, School of Computer Science, National University of Defense Technology, Changsha, 410073, China.
[2] Department of Computer Science, University of North Carolina at Chapel Hill, North Carolina 27599, USA.
[*] Corresponding Author: Zeyu Xiong. Email: xiongzeyu08@nudt.edu.cn.

a new multi-classification algorithm based on support vector machine and binary tree structure to solve the problem of non-separable regions.

In the existing regression algorithm, support vector machines are mostly used for multi-classification problem, but there are some limitations in algorithm. The logistic regression algorithm can only solve the problem of dichotomy and linear classification. Support vector machines typically support only small training samples and are equally difficult to deal with multiple classification problems. Naive Bayes is based on the assumption that the characteristic conditions are independent. Once the dataset does not satisfy this assumption, its classification accuracy will be greatly affected.

In order to solve the problem above, towards difficult for implement large scale samples, not applicable to multi-classification and uncertainty to constraint conditions, Chen et al. [Chen, Chen, Mao et al. (2013)] proposed a model of Density-based Logistic Regression (DLR), which has a good result in practical application. Our model is based on kernel density-based logistic regression and we construct a new kernel function for multi-classification problems. This has three advantages: 1) It makes better improvements to classification effect. 2) It is an extension of DLR model to multi-classification problems. 3) It shows good generalization performance on nonlinear and unbalanced data. We will describe the theoretical rationality and check classifying quality according to practical application for our new model.

The rest of the paper is organized as the following. In Section 2, we explain background knowledge including logistic regression binary classification, multi-classification, SoftMax and DLR model. In Section 3, we introduce several solutions for multi-classification problems with imbalanced samples. In Section 4, we explain our approach in details. In Section 5, we compare our approach to other methods and analyze the performances. Finally, we conclude in Section 6.

2 Logistic regression and related knowledge

2.1 Logistic regression

Logistic regression is based on linear regression, and a sigmoid logic function is applied, which is a logarithmic probability function. Logistic regression is represented as follows,

$$y = \frac{1}{1+e^{-z}}. \tag{1}$$

In the model of sigmoid function, z values are distributed within the range of [0,1]. When the independent variable is taken near 0, the z-value change curve is very steep, while the z value is relatively stable at other values. Therefore, the binary classification tasks can be handled well if taking 0 as the boundary. However, it is sometimes difficult to make the representation model approximate to the expected model. By adding a constant term b to the function,

$$z = w^T x + b. \tag{2}$$

By substituting Eq. (2) into Eq. (1), we have

$$y = \frac{1}{1 + e^{-(w^T x + b)}}. \tag{3}$$

Based on these formulae, assuming a given dataset $\mathcal{D} = \{x_i, y_i\}, i = 1, \cdots, N, x_i \in R^D$, D is the dimension of samples, and $y_i \in \{0, 1\}$, logistic regression is described as follows:

$$p(y = 1|x) = \frac{1}{1 + e^{-w^T \phi}}, \tag{4}$$

where w stands for feature weight, which is a parameter to be learned. ϕ is the characteristic transformation function.

In LR model ϕ is usually defined to be equal to x. The key step is to learn unknown parameters w and b. If y in Eq. (3) is regarded as posterior probability estimation $p(y = 1|x)$, Eq. (4) can be rewritten as:

$$\ln \frac{p(y = 1|x)}{p(y = 0|x)} = w^T x + b, \tag{5}$$

$$p(y = 1|x) = \frac{e^{w^T x + b}}{1 + e^{w^T x + b}}, \tag{6}$$

$$p(y = 0|x) = \frac{1}{1 + e^{w^T x + b}}. \tag{7}$$

Then w can be obtained by the maximum likelihood estimate. With the definition of $b_i = p(y_i = 1|x_i)$, y=0 or 1, for a single sample, the posterior probability is,

$$p(y|x, w) = (b_i(x))^y (1 - b_i(x))^{1-y}. \tag{8}$$

Then, the maximum likelihood function is represented as follows,

$$\begin{aligned} L(w|x, y) &= \textstyle\prod_{i=1}^m p(y_i|x_i, w) \\ &= \textstyle\prod_{i=1}^m (b_i(x))^{y_i} (1 - b_i(x))^{1-y_i}. \end{aligned} \tag{9}$$

For the convenience of calculation, the negative log of the maximum likelihood function is used as the objective function to be optimized,

$$\begin{aligned} E(w) &= -\ln(L(w|x, y)) \\ &= -\textstyle\prod_{i=1}^m (y_i \ln(b_i(x)) + (1 - y_i) \ln(1 - b_i(x))). \end{aligned} \tag{10}$$

Since the maximizing likelihood probability is equivalent to minimizing negative likelihood probability, the last step is to minimize the Loss function.

2.2 Density-based logistic regression

In the DLR model, ϕ is a function that maps x to the eigenspace,

$$\phi_d(x) = \ln \frac{p(y=1|x_d)}{p(y=0|x_d)} - \frac{D-1}{D} \ln \frac{p(y=1)}{p(y=0)}, \tag{11}$$

where D is the dimension of the input data, $\ln \frac{p(y=1|x_d)}{p(y=0|x_d)}$ measures the contribution of x_d to the probability of $y = 1$, and $\frac{D-1}{D} \ln \frac{p(y=1)}{p(y=0)}$ measures the degree of imbalance for datasets. $p(y = 1)$ is the proportion of data in the training set, whose label is $y = 1$.
Nadaraya-Watson is usually used to estimate $p(y = k|x_d)$ where $k = 0, 1$.

$$p(y=k|x_d) = \frac{\sum_{i \in \mathcal{D}_k} K(x_d, x_{id})}{\sum_{i=1}^{N} K(x_d, x_{id})}. \tag{12}$$

where $D_k \subseteq D$ is the subset of data in class k, and $K(x, y)$ is a Gaussian kernel function defined as follows,

$$K(x,y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-y)^2}{2h_d^2}}, \tag{13}$$

where h_d is the bandwidth of the kernel density function. The h_d is usually set using the Silverman's rule of thumb [Silverman and Green (1986)],

$$h_d = 1.06\sigma N^{-\frac{1}{5}}, \tag{14}$$

where N is the total number of samples and σ is the standard deviation of x_d.
Next we need to train w through the learning algorithm until w converges. Given $b_i = p(y_i = 1|x_i)$, the loss function based on likelihood probability is calculated as follows,

$$\begin{aligned} E(w,h) &= -\ln(p(y|w,h)) \\ &= -\textstyle\sum_{i=1}^{N}(y_i \ln b_i + (1-y_i)\ln(1-b_i)). \end{aligned} \tag{15}$$

2.3 Extension of logistic regression to multiple classification

Since the logistic classification is a binary classification model, it is necessary to extend it for multiple classification, common extensions include multiple binary classification models or SoftMax models.

2.3.1 N-logistic model

The N-logistic model generally adopts One-vs-Rest or One-*vs.*-One. When classifying a sample, we first classify the two classifiers, then vote, and select the category with the highest score. At the same time, to prevent the same vote, we also add the probability of the class to each classifier in the voting. The predictive accuracy of these two approaches is usually very similar, so unless there is a specific need for data characteristics, it is generally arbitrary to choose one approach to calculate.

2.3.2 SoftMax model

SoftMax regression is a generalization of logistic regression to multiple classification problems. Its basic form is described as follows,

$$p(y = c|x) = \frac{e^{w_c^T x}}{\sum_{i=1}^{C}(e^{w_i^T x})}. \tag{16}$$

When in the test, to sample x, if there is a category c, for all the other category $c^*(c^* \neq c)$ meet the $p(y = c|x) > p(y = c^*|x)$, then x belongs to the category c.

On the question of choosing N-logistic model or SoftMax model, many scholars have conducted in-depth exploration. Currently, it is accepted that it is necessary to investigate whether the various categories are mutually exclusive. If there is a mutual exclusion relationship between the categories to be classified, we'd better choose SoftMax classifier. While if there is no mutual exclusion between categories, and the categories are intersecting, it is best suited to the N-logistic classifier. We verify this conclusion according to corresponding datasets in Section 5.

3 Analysis of the classification results with unbalanced sample proportion

In our actual classification tasks, there are often needs to deal with problems with unbalanced data sample proportions. For example, the ratio of positive and negative samples in a dataset is 10:1, including 100 positive classes and 10 negative classes. If using this kind of data to train a classifier, it is very likely that the test data will be divided into positive classes. Obviously, this classifier is invalid.

For this kind of data, traditional logistic regression method usually fails to work. In recent years, studies on the problem of unbalanced classification have been very active [Ye, Wen and Lv (2009)]. In this section we introduce several common approaches to solve the problem of sample imbalance classification.

3.1 Obtain more samples

For unbalanced classification, the first solution is to obtain more samples and expand a few samples to balance the sample proportion. However, in most cases, the sampling procedure needs specific conditions. Thus, it is generally difficult to obtain more samples under the same conditions.

3.2 Sampling methods

The general sampling method is mainly based on modifying the number of unbalanced samples. The research of Estabrooks et al. [Estabrooks, Jo and Japkowicz (2004)] show that the general sampling method has a better effect on solving unbalanced classification problems.

3.2.1 Under-sampling method

Under-sampling method is also called down-sampling [Gao, Ding and Han (2008)], which is to eliminate some samples from majority class samples, so that the number of samples in the whole group tends to be balanced. The commonly used method is random under-sampling downward method. The method is based on N_{min}, the number of minority class samples. We randomly sample from the majority class samples and eliminate N samples, and then $N_{max} - N = N_{min}$, so the samples are balanced.

3.2.2 Over-sampling method

Over-sampling method is also called up-sampling, which refers to increase the number of minority class samples. The method of adding a small number of minority class samples (random over-sampling method) or re-fitting some new data in accordance with some law can be used to make the number of samples balanced. One commonly used method is Synthetic Minority Over-sampling Technique (SMOTE) [Chawla, Bowyer, Hall et al. (2002)]. The method analyzes the distribution of the characteristic space of a few samples and proposes new samples. Compared to the random over-sampling method, the data added by SMOTE sampling method is completely new, which can follow the regular pattern in the original sample. The main idea of SMOTE is shown in Fig. 1.

For each sample x in a minority class, the Euclidean distance of each sample point of a minority sample is calculated, and its k neighbors are obtained. A suitable sampling ratio is set according to the sample proportion to determine the sampling rate N. For each of the minority sample x, select several samples randomly from its k neighbors. For each random nearest neighbor x_n, a new sample is constructed with the original sample according to the following equation,

$$x_{new} = x + rand(0,1) * (x^* - x). \tag{17}$$

3.3 Modify evaluation index

For unbalanced classification, using accuracy to evaluate classifiers may biases. For example, assuming ratio of positive and negative samples in a dataset is 9:1, and all samples are labelled be positive. Although the accuracy rate is up to 90%, the classifier is useless.

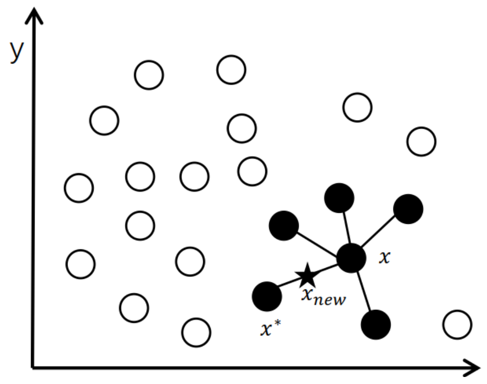

Figure 1: The main idea of SMOTE method

Table 1: A hybrid matrix of binary classification

		GT category Positive	GT category Negative
Pred.category	Positive	TP(True Positives)	FP(False Positives)
Pred.category	Negative	FN(False Negatives)	TN(True Negatives)

Therefore, accuracy can serve as a biased indicator. Davis et al. [Davis and Goadrich (2006)] proposed a new evaluation index named Precision and Recall, some factors are listed in Tab. 1.

$$Accuracy = (TP + TN)/(P + N). \tag{18}$$

$$Error = 1 - Accuracy. \tag{19}$$

$$Precision = TP/(TP + FP). \tag{20}$$

$$Recall = TP/(TP + FP). \tag{21}$$

$Precision$ refers to the proportion of positive samples in all predicted positive samples, and $Recall$ refers to the proportion of all actual positive samples that are being correctly predicted.

3.4 Use penalty items to modify the weights

If samples are difficult to sample directly, the method of modifying sample weights can be used. It increases the weight of minority class samples and reduces the weight of the majority class samples. Because the weights of minority class samples are high, they can lead to better classification results. The commonly used method is to add a penalty item to the majority class samples each time when training the sample weight. In general, we use the regularization method to add a penalty parameter to a objective function, this reduces the chance of the overfitting [Goodfellow, Bengio and Courville (2017)]. The regularized objective function is shown below,

$$J^*(\theta; X, y) = J(\theta; X, y) + \alpha\omega(\theta), \tag{22}$$

where α is a parameter which represents the contribution of the penalty item and the objective function. The penalty can be adjusted by controlling α. If $\alpha = 0$, there is no penalty, otherwise the larger the α, the greater the penalty.

After we chose an appropriate penalty, the training regularize the objective function. In this way, the data error and the parameter scale can be reduced, the computation efficiency can be improved. But in practice, how to select the optimal penalty item is a complicated problem, which needs more tests.

3.5 Kernel-based methods

Towards general classification problem, we can assume that the sample data can be classified directly by linear model. In other words, there is a hyperplane that can separate the samples and ensure that the classification is correct. However, in practice, there is usually no such a hyperplane to partition the original data correctly, which means that the data are not linearly separable. For such a problem, we can consider preprocessing data. Using the principle of support vector machine, data in the low-dimensional space are transformed into the high dimensional space through nonlinear transformation, so that they can be linearly separable [Zhou (2016)]. Using this method, the relationship between data samples can be written as dot product. For example, the linear regression function can be rewritten as follows,

$$w^T x + b = b + \textstyle\sum_{i=1}^{m}(\alpha_i x^T x^{(i)}), \tag{23}$$

where $x^{(i)}$ is the training data. α is the coefficient vector. Replacing the dot product with a function of the kernel $k(x, x^{(i)}) = \phi(x) \cdot \phi(x^{(i)})$, we can get,

$$f(x) = b + \textstyle\sum_i(\alpha_i k(x, x^{(i)})). \tag{24}$$

This function is nonlinear with respect to x, while it is linear with respect to $\phi(x)$.

Kernel function can deal with nonlinear unbalanced classification well. It uses a convex optimization technique to address nonlinear problems in a linear manner. At the same time, this method can guarantee convergence and improve the accuracy of classification. And there is some simplification in parameter determination. In addition, it is much more efficient to use the kernel function to transform data into a transformation function [Goodfellow, Bengio and Courville (2017)].

SVM can convert sample data into high dimensional feature space through a kernel function. According to the principle of maximum spacing of SVM, the hyperplane of the optimal classification can be constructed in the characteristic space of high dimension to realize the classification. If the interval of classification can be extended, especially between minority class samples and the optimal classification hyperplane, the generalization performance of the classifier and the accuracy of classes with small samples can be effectively improved. This enables the correct classification of unbalanced data [Liu, Huang, Zhu et al. (2009)].

4 Improved method of kernel density estimation model for multi-classification

We extend the DLR model to solve the multi-classification problem and design an improved multi-classification algorithm. Assuming there are C classes, for $k = 1, 2, \dots, C$, the DLR model is defined as follows,

$$p(y = k|x) = \frac{e^{w_k^T \phi_k(x)}}{\sum_{j=1}^{C}(e^{w_j^T \phi_j(x)})}, \tag{25}$$

where $w_k = (w_{k1}, w_{k2}, \dots, w_{kD})$ is the feature weighting parameter of class k, and $\phi_k = (\phi_{k1}, \phi_{k2}, \dots, \phi_{kD})$ is the characteristic transformation function of class k.

$$\phi_{kd}(x) = \ln p(y = k|x_d) - \frac{D-1}{D} \ln p(y = k). \tag{26}$$

According to the Nadaraya-Watson estimator, the probability formula of class k is obtained as follows:

$$p(y = k|x_d) = \ln \frac{\sum_{i \in D_k} e^{-\frac{(x_d - x_{id})^2}{h_d^2}}}{\sum_{i=1}^{N} e^{-\frac{(x_d - x_{id})^2}{h_d^2}}}. \tag{27}$$

Finally, we need to minimize the loss function,

$$\begin{aligned} E(w,h) &= -\textstyle\sum_{i=1}^{N} \sum_{k=1}^{C} (1_{y_i=k} \ln p(y_i = k|x_i)) \\ &= -\textstyle\sum_{i=1}^{N} \sum_{k=1}^{C} (1_{y_i=k} \ln \frac{e^{w_k^T \phi_k(x_i)}}{\sum_{j=1}^{C}(e^{w_j^T \phi_j(x_i)})}) \\ &= -\textstyle\sum_{i=1}^{N} \sum_{k=1}^{C} (1_{y_i=k} (w_k^T \phi_k(x_i) - \ln \sum_{j=1}^{C}(e^{w_j^T \phi_j(x_i)}))), \end{aligned} \tag{28}$$

where, $1_{y_i=k}$ is 1 if and only if $y_i = k$, otherwise it takes value 0.

Algorithm 1 LR training

```
1: Load data $x$;
2: Initialize $w$;
3: repeat
4:     for $i = 1$ to $N$ do
5:         $b_i = \frac{1}{1+e^{-w^T x_i}}$;
6:         $E_i = -(y_i \ln b_i + (1 - y_i)\ln(1 - b_i))$;
7:     end for
8:     $w \leftarrow w - \bigtriangledown_w E$;
9: until $w$ converges
```

Now we present the process of evaluating the gradient of the Loss function with respect to w_k,

$$\begin{aligned} \bigtriangledown_w E &= -\textstyle\sum_{i=1}^{N}\sum_{k=1}^{C}(1_{y_i=k}(x_i - \frac{e^{w_k^T \phi_k(x_i)}}{\sum_{j=1}^{C}(e^{w_j^T \phi_j(x_i)})} x_i)) \\ &= -\textstyle\sum_{i=1}^{N}\sum_{k=1}^{C}(x_i(1_{y_i=k} - \frac{e^{w_k^T \phi_k(x_i)}}{\sum_{j=1}^{C}(e^{w_j^T \phi_j(x_i)})})) \\ &= -\textstyle\sum_{i=1}^{N}\sum_{k=1}^{C}(x_i(1_{y_i=k} - p(y_i = k|x_i))). \end{aligned} \tag{29}$$

We adjust the weight w_k according to the direction of the gradient descent, until the w_k converges and w_k in the model is well trained. During the testing, the same kernel function transformation is performed on the testing data. The transformed $\phi(x)$ and trained w_k are substituted into Eq. (25). Then we compare the probability of the different classes and choose the class with the largest probability as the result category. At this point, we have completed the generalization of the logistic regression to multi-classification based on kernel density function.

To show the difference between kernel density estimation logistic regression and classical logistic regression, we will compare the corresponding algorithms later.

In the DLR algorithm, the input x is given a feature transformation to get ϕ before calculating the probability in Eq. (25). And then substitute ϕ for x as the input to the probability formula. At the same time, the probability formula is changed from the Sigmoid function to the SoftMax function.

After conducting experiments, we have found that the differences of ϕ among different labels obtained using the DLR algorithm are small. There is a large error in the final classification result. And the minority class samples cannot be discriminated at all. And the value of loss function is not reduced by training. Therefore, in the process of constructing the bandwidth of kernel function and preprocessing the data, we improve it by the following scheme.

Algorithm 2 DLR training

```
1: Load data $x$;
2: Initialize $w$;
3: Initialize $h = 1.06\sigma N^{-\frac{1}{5}}$;
4: Transform $x$ to the feature matrix $\phi$ under $h$;
5: for $j = 1$ to $C$ do
6:     repeat
7:         for $i = 1$ to $N$ do
8:             $b_{ij} = \frac{e^{w_j^T \phi_{ij}}}{\sum_{k=1}^{C}(e^{w_k^T \phi_{ik}})}$;
9:             $E_{ij} = -1_{y_i=j} \ln b_{ij}$;
10:        end for
11:        $w_j \leftarrow w_j - \nabla_w E_j$;
12:    until $w$ converges;
13: end for
```

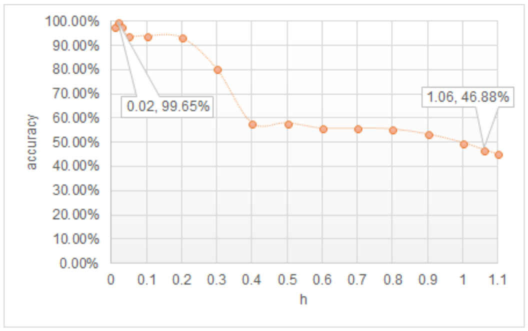

Figure 2: The process of searching for the optimal coefficient

First, We try to train the parameters of the kernel function by modifying the weight values on the basis of Eq. (14). We conducted 16 groups of experiments, as shown in Fig. 2. In the previous experiment, since the value of w was too large, the characteristics of the input data X itself were difficult to distinguish. Properly reducing $\|w\|$ can limit the complexity of the model, thereby improving the generalization performance of the model. Through comparison experiments, we found that changing 1.06 in Eq. (14) to 0.02 can significantly improve the accuracy of the model. According to Fig. 2, we reduce the bandwidth of kernel function in Eq. (14).

$$h_d = 0.02\sigma N^{-\frac{1}{5}}. \tag{30}$$

In this way, the difference of h_d has been improved. However, it may cause the value of y become too large and overflow in subsequent c alculations. Feature scaling is a crucial step in the data preprocessing process. For most machine learning and optimization algorithms, scaling the values of features to the same interval can make their performance even better. In order to accelerate loss function convergence rate, we normalize ϕ using the min-max method.

Algorithm 3 DLR++ training

```
1: Load data $x$;
2: Initialize $w$;
3: Initialize $h = 0.02\sigma N^{-\frac{1}{5}}$;
4: Transform $x$ to the feature matrix $\phi$ under $h$;
5: for $j = 1$ to $D$ do
6:     $\phi_{min} = min(\phi_j)$;
7:     $\phi_{max} = max(\phi_j)$;
8:     for $i = 1$ to $N$ do
9:         $\phi_{ij} = \frac{\phi_{ij} - \phi_{min}}{\phi_{max} - \phi_{min}}$;
10:    end for
11: end for
12: for $j = 1$ to $C$ do
13:    repeat
14:        for $i = 1$ to $N$ do
15:            $b_{ij} = \frac{e^{w_j^T \phi_{ij}}}{\sum_{k=1}^{C}(e^{w_k^T \phi_{ik}})}$;
16:            $E_{ij} = -1_{y_i = j} \ln b_{ij}$;
17:        end for
18:        $w_j \leftarrow w_j - \nabla_w E_j$;
19:    until $w$ converges;
20: end for
```

$$x_{norm}^{(i)} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{min}}. \tag{31}$$

The training process of the improved model is established in Algorithm 3.

In the next section, we will conduct a comparative test to analyze the relationship between test results and training results after using Algorithm 3.

5 Application of improved algorithm: datasets and verification analysis

In particular, we have implemented the following methods for testing.

1) N-logistic model, One-vs-Rest methods, abbreviated as NLR.

2) N-logistic model, One-vs-Rest methods, combined with the oversampling method, abbreviated as NLR_Sample.

3) N-logistic model, One-vs-Rest methods, combined with the Smote method, abbreviated as NLR_Smote.

4) SoftMax model.

5) SoftMax model combined with Algorithm 3, abbreviated as DLR++.

We choose three datasets for testing. The first one is the fitting dataset $Numb$ constructed by us. In this dataset, each data element contains 10 floating point values, ranging from 0 to 5. The data distribution is divided into three categories: $GroupA$, $GroupB$ and $GroupC$. The second dataset is the $Iris$ from UCI. There are four features, including calyx length, calyx width and petal width, and the eigenvalue is floating-point number. The target value is the classification result of irises, including $virginica$, $versicolor$, and $setosa$. The third dataset is the $Wine$ from UCI, which uses the various parameters of the $Wine$ to predict the quality of the Wine. There are 11 characteristic values, including volatile

Table 2: Accuracy (%) of different methods on three datasets

Dataset	NLR	NLRSample	NLRSmote	SoftMax	DLR++
Numb	91.58	92.63	97.24	96.84	98.95
Iris	93.33	94.44	95.73	95.56	99.47
Wine	90.74	92.59	94.44	83.33	98.36

Table 3: Time(s) for different methods on three datasets

Dataset	NLR	NLRSample	NLRSmote	SoftMax	DLR++
Numb	0.63	0.58	0.69	24.76	5.12
Iris	0.57	0.56	0.59	6.66	3.39
Wine	1.81	1.69	1.81	23.75	3.41

Table 4: The number of iterations of training Loss convergence on three datasets

Dataset	NLR	NLRSample	NLRSmote	SoftMax	DLR++
Numb	1000	1000	1000	1000	40
Iris	1000	1000	1000	500	200
Wine	5000	5000	5000	1500	100

acidity, non-volatile acid, citric acid, residual sugar, chlorine, total sulfur dioxide, free of sulfur dioxide, sulfate, concentration, PH and alcohol. There are three quality classes:1, 2, or 3.

In order to keep the data more versatile, and the classification results more persuasive, we use k-fold cross validation and assign the dataset to the training set and testing set according to the ratio 7:3. The test results are given as follows.

From Tab. 2 to Tab. 4, we can see that the DLR++ algorithm shows better prediction accuracy. In the three datasets, $Numb$ is linear, while $Iris$ and $Wine$ are non-linear. We can see from the results that both N-logistic and SoftMax models can solve the multi-classification problem well. Both oversampling and smote sampling method can be used to improve the classification results of the sample imbalance problem with the accuracy rate increased by 1.34% and 3.92% respectively. The improved DLR++ model based on kernel density is the best among all these methods, and it has an advantage in solving nonlinear multi-classification problems. From Tab. 2 to Tab. 4, we can see that the improved DLR++ model converges faster than the original logistic model, using only 1/20 of the training times. At the same time, the accuracy rate has been increased 7.04%, at the cost of a higher operation time.

From Tab. 5 to Tab. 6, we can see that the improved DLR++ model has a better performance on datasets of large scales and multiple categories. It offers an accuracy of 93.0% while LR

Table 5: Performance of DLR++ on different scales of datasets

Scale	100	200	500	1000
Size of Train dataset	89	164	413	834
Size of Test dataset	84	84	84	84
Accuracy	98.81%	98.81%	100.0%	100.0%

Table 6: Performance of DLR++ on different number of categories

Num of categories	3	5	7	10
Size of Train dataset	196	191	193	200
Size of Test dataset	95	94	93	100
Accuracy	98.95%	98.94%	98.92%	93.0%
Comparation(Best of the others)	91.58%	84.0%	79.57%	47.0%

offers an accuracy of 47.0% for 10-classification problems.

6 Conclusion

In this paper, we propose an improved logistic regression model based on kernel density estimation, and it can be applied to solve nonlinear multi-classification problems. We have compared and tested several common algorithms for logistic regression. For the experimental results, we found that the sampling method [Gao, Ding and Han (2008); Chawla, Bowyer, Hall et al. (2002)] can improve the classification accuracy, but the training samples obtained are very different from the original samples, which destroys the data characteristics inherently in the original sample. However in contrast, our improved model guarantees the integrity of the samples, it has obvious advantages in classification accuracy, and has good generalization ability with an ideal training speed. But there is still room for optimization in training, especially in the matrix operation stage. In the future, we will reduce the size of the matrix and block calculation, expected to decline training time and improve efficiency. Combining application to document retrieval [Xiong and Wang (2018); Xiong, Shen, Wang et al. (2018)], we will also expect to check the improved method in this paper is effect to document classification which is interested by us.

Acknowledgement: The authors would like to thank all anonymous reviewers for their suggestions and feedback. This work was supported by National Natural Science Foundation of China (Grant No. 61379103).

References

Chawla, N.; Bowyer, K.; Hall, L.; Kegelmeyer, W. (2002): Smote: synthetic minority over-sampling technique. *Artificial Intelligence Research*, vol. 16, no. 2002, pp. 321-357.

Chen, W.; Chen, Y.; Mao, Y.; Guo, B. (2013): Density-based logistic regression. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 140-148.

Davis, J.; Goadrich, M. (2006): The relationship between precision-recall and roc curves. *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233-240.

Estabrooks, A.; Jo, T.; Japkowicz, N. (2004): A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, vol. 20, no. 1, pp. 18-36.

Gao, J.; Ding, B.; Han, J. (2008): Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing*, vol. 12, no. 6, pp. 37-49.

Goodfellow, I.; Bengio, Y.; Courville, A. (2017): *Deep Learning*. Beijing: Posts and Telecom Press (In Chinese).

Liu, H.; Huang, M.; Zhu, Q.; Wang, C. (2009): Research on the unbalance data classification algorithm based on support vector machine. *Computer Application Research*, vol. 26, no. 8, pp. 2874-2875.

Liu, T.; Liang, Y.; Xue, X. (2008): A new fuzzy support vector machine multi-classification algorithm. *Computer Application Research*, vol. 25, no. 7, pp. 2041-2042 (In Chinese).

Silverman, B.; Green, P. (1986): *Density Estimation for Statistics and Data Analysis*. Chapman and Hall Press.

Tang, F.; Wang, Z.; Chen, M. (2005): Support vector machine multi-classfication algorithm research. *Control and Decision*, vol. 20, no. 7, pp. 746-749 (In Chinese).

Xiong, Z.; Shen, Q.; Wang, Y.; Zhu, C. (2018): Paragraph vector representation based on word to vector and cnn learning. *Computers, Materials & Continua*, vol. 55, no. 2, pp. 213-227.

Xiong, Z.; Wang, Y. (2018): New document scoring model based on interval tree. *Journal of Visual Languages and Computing*, vol. 45, no. 2018, pp. 39-43.

Ye, Z.; Wen, Y.; Lv, B. (2009): Research review on unbalanced classification. *Journal of Intelligent Systems*, vol. 4, no. 2, pp. 148-156 (In Chinese).

Zhou, Z. (2016): *Machine Leaning*. Beijing: Tsinghua University Press.