Imbalanced Extreme Learning Machine Based on Probability Density Estimation

Jü Yang, Hualong Yu(✉), Xibei Yang, and Xin Zuo

School of Computer Science and Engineering,
Jiangsu University of Science and Technology,
Zhenjiang 212003, Jiangsu, People's Republic of China
yangju_justcs@126.com, yuhualong@just.edu.cn,
yangxibei1980@sina.com, 632343650@qq.com

Abstract. Extreme learning machine (ELM) is a fast algorithm to train single-hidden layer feedforward neural networks (SLFNs). Like the traditional classification algorithms, such as decision tree, Naïve Bayes classifier and support vector machine, ELM also tends to provide biased classification results when the classification tasks are imbalanced. In this article, we first analyze the relationship between ELM and Naïve Bayes classifier, and then take the decision outputs of all training instances in ELM as probability density representation by kernel probability density estimation method. Finally, the optimal classification hyperplane can be determined by finding the intersection point of two probability density distribution curves. Experimental results on thirty-two imbalanced data sets indicate that the proposed algorithm can address class imbalance problem effectively, as well outperform some existing class imbalance learning algorithms in the context of ELM.

Keywords: Extreme learning machine · Class imbalance learning · Probability density estimation · Naïve Bayes classifier

1 Introduction

Extreme learning machine proposed by Huang *et al.*, [1] has become a popular research topic in machine learning in recent years [2]. It is proved that single-hidden layer feedforward neural networks (SLFNs) with arbitrary hidden parameters and continuous activation function can universally approximate to any continuous functions [1]. Some recent research [3–6], however, indicated that the performance of ELM could be destroyed by class imbalance distribution, which is similar with some traditional classifiers, such as support vector machine, Naïve Bayes classifier and decision tree. In class imbalance scenario, the accuracy of the minority class always tends to be underestimated, causing meaningless classification results [7]. Therefore, it is necessary to adopt some strategies to make the classification model provide impartial classification results.

In the context of ELM, some researchers have presented several class imbalance learning algorithms. Weighted extreme learning machine (WELM) appoints different penalty parameters for the training errors belonging to the instances in different categories, decreasing the possibility of misclassifying the minority class samples [3]. The

A. Bikakis and X. Zheng (Eds.): MIWAI 2015, LNAI 9426, pp. 160–167, 2015.
DOI: 10.1007/978-3-319-26181-2_15

penalty parameters, however, can be only allocated empirically. A similar algorithm called Fuzzy ELM (FELM) was proposed in [4], which changes the distributions of penalty parameters by inserting a fuzzy matrix. As two well-known data-layer class imbalance learning algorithms, random oversampling (ROS) and synthetic minority oversampling technology (SMOTE) have also be integrated into ELM to deal with practical class imbalance applications [5, 6].

In this article, we try to present a novel algorithm to deal with class imbalance problem in the context of ELM. First, we analyze the relationship between ELM and Naïve Bayes classifier, and indicate that the decision output in ELM approximately equals to the posterior probability in Naïve Bayes classifier. Then, on the decision output space, we estimate the probability density distributions for two different classes, respectively. Finally, the optimal position of the classification hyperplane can be determined by finding the intersection point of two probability density distribution curves. We compare the proposed algorithm with several popular class imbalance learning algorithms, and the experimental results indicate its superiority.

2 Theories and Methods

2.1 Extreme Learning Machine

Considering a supervised learning problem where we have a training set with N training instances and m classes, $(\mathrm{x}_i, \mathrm{t}_i) \in \mathrm{R}^n \times \mathrm{R}^m$. Here, x_i is an $n \times 1$ input vector and t_i is the corresponding $m \times 1$ target vector. ELM aims to learn a decision rule or an approximation function based on the training data. In other words, ELM is used to create an approximately accurate mapping relationship between x_i and t_i.

Unlike the traditional back-propagation (BP) algorithm [8], ELM provides the hidden parameters randomly to training SLFNs. Suppose there are L hidden layer nodes, then for an instance x, the corresponding hidden layer output can be presented by a row vector $\mathrm{h}(\mathrm{x}) = [\mathrm{h}_1(\mathrm{x}), \ldots, \mathrm{h}_L(\mathrm{x})]$, thus the mathematical model of ELM is:

$$\mathrm{H}\beta = \mathrm{T} \tag{1}$$

where $\mathrm{H} = [\mathrm{h}(\mathrm{x}_1), \ldots, \mathrm{h}(\mathrm{x}_N)]^T$ is the hidden layer output matrix for the whole training set, β is the output weight matrix and T is the target vector. Here, only the output weight matrix β is unknown. Then we can adopt least square method to acquire the solution of β that can be described as follows:

$$\begin{cases} \beta = \hat{H}^{\dagger}\mathrm{T} = \mathrm{H}^T(\frac{\mathrm{I}}{C} + \mathrm{H}\mathrm{H}^T)^{-1}\mathrm{T}, \text{when } N \leq L \\ \beta = \hat{H}^{\dagger}\mathrm{T} = (\frac{\mathrm{I}}{C} + \mathrm{H}^T\mathrm{H})^{-1}\mathrm{H}^T\mathrm{T}, \text{when } N > L \end{cases} \tag{2}$$

Here, $\hat{H}^{\dagger}$ is the Moore-Penrose "generalized" inverse of the hidden layer output matrix H, which can guarantee the solution is least norm least square solution of Eq. (1). C is the penalty parameter to mediate the balance relationship between the training errors and the generalization ability.

2.2 Relationship Between ELM and Naïve Bayes Classifier

According to some previous work, the decision outputs of SLFNs trained by BP algorithm [8] can be regarded as an approximation of posteriori probability functions in Naïve Bayes classifier [9, 10]. Suppose there are lots of training instances, and each of them belongs to one of m classes. We can train an SLFNs to obtain the output weight matrix w. Let $f_k(\mathrm{x}, \mathrm{w})$ be the output of the kth output node of the SLFNs, i.e., the discriminant function corresponding to the kth class w_k, then we can recall Bayes formula,

$$P(w_k|\mathrm{x}) = \frac{P(\mathrm{x}|w_k)P(w_k)}{\sum_{i=1}^{m} P(\mathrm{x}|w_i)P(w_i)} = \frac{p(\mathrm{x}, w_k)}{P(\mathrm{x})}$$

and the Bayes decision for any instance x: choosing the class w_k which has the largest discriminant function $f_k(\mathrm{x}) = P(w_k|\mathrm{x})$. Without loss of generality, suppose the training outputs are restricted as $\{0, 1\}$, where 1 denotes the output of the corresponding class and 0 denotes the outputs of the other classes. The contribution to the criterion function based on a single output unit k for finite number of training samples x is:

$$\begin{aligned} J(\mathrm{w}) &= \sum_{\mathrm{x}} \left(f_k(\mathrm{x}, \mathrm{w}) - t_k\right)^2 \\ &= \sum\nolimits_{\mathrm{x}\in w_k} \left(f_k(\mathrm{x}, \mathrm{w}) - 1\right)^2 + \sum\nolimits_{\mathrm{x}\notin w_k} \left(f_k(\mathrm{x}, \mathrm{w}) - 0\right)^2 \\ &= n\left\{\frac{n_k}{n}\frac{1}{n_k}\sum\nolimits_{\mathrm{x}\in w_k} \left(f_k(\mathrm{x}, \mathrm{w}) - 1\right)^2 + \frac{n - n_k}{n}\frac{1}{n - n_k}\sum\nolimits_{\mathrm{x}\notin w_k} \left(f_k(\mathrm{x}, \mathrm{w}) - 0\right)^2\right\} \end{aligned} \tag{3}$$

where n denotes the number of training instances, while n_k stands for the number of instances belonging to the class w_k. In the limit of infinite data, we can use Bayes formula to express Eq. (3) as:

$$\begin{aligned} &\lim_{n\to\infty} \frac{1}{n} J(\mathrm{w}) = \tilde{J}(\mathrm{w}) \\ &= p(w_k) \int \left(f_k(\mathrm{x}, \mathrm{w}) - 1\right)^2 P(\mathrm{x}|w_k) d\mathrm{x} + p(w_{i\neq k}) \int f_k(\mathrm{x}, \mathrm{w})^2 P\left(\mathrm{x}|w_{i\neq k}\right) d\mathrm{x} \\ &= \int f_k^2(\mathrm{x}, \mathrm{w}) p(\mathrm{x}) d\mathrm{x} - 2\int f_k(\mathrm{x}, \mathrm{w}) p(\mathrm{x}, w_k) d\mathrm{x} + \int p(\mathrm{x}, w_k) d\mathrm{x} \\ &= \int \left(f_k(\mathrm{x}, \mathrm{w}) - p(w_k|\mathrm{x})\right)^2 p(\mathrm{x}) d\mathrm{x} + \int p(w_k|\mathrm{x}) p(w_{i\neq k}|\mathrm{x}) p(\mathrm{x}) d\mathrm{x} \end{aligned} \tag{4}$$

Obviously, the right-hand side in Eq. (4) is irrelevant with the weight w, thus SLFNs is to minimize:

$$\int \left(f_k(\mathrm{x}, \mathrm{w}) - p(w_k|\mathrm{x})\right)^2 p(\mathrm{x}) d\mathrm{x} \tag{5}$$

Because this is true for each class, SLFNs minimizes the sum:

$$\sum_{k=1}^{m} \int (f_k(\mathrm{x}, \mathrm{w}) - p(w_k|\mathrm{x}))^2 p(\mathrm{x}) d\mathrm{x} \tag{6}$$

Therefore, in the limit of infinite data, the outputs of the trained SLFNs will approximate the true posterior probabilities in a least-squares sense, i.e., $f_k(\mathrm{x}, \mathrm{w}) = p(w_k|\mathrm{x})$.

As we know, like BP algorithm, ELM also provides approximate least squares solution of Eq. (1) though it simultaneously minimizes the norm of the weight matrix. Therefore, the decision outputs in ELM reflect the posteriori probabilities of different classes in Naïve Bayes classifier to some extent.

2.3 Probability Density Estimation

As described above, the decision outputs in ELM can reflect posteriori probabilities of different classes, thus for binary-class problem, we can map all instances from original feature space to an one-dimensional space. Here, ELM is used as a feature extraction tool. Then, we regard to estimate the probability density distributions of two different classes on the compressed one-dimensional feature space. Specifically, kernel probability density estimation [11], which is a nonparametric way to estimate the probability function of a random variable, is adopted.

Figure 1 shows a schematic diagram about probability density distributions of a binary-class problem on the one-dimensional decision output space acquired from ELM. From Fig. 1, we observe that after estimating probability density distributions, the prior probabilities for the two classes $p(w_+)$ and $p(w_-)$, and the corresponding conditional distribution probabilities $p(\mathrm{x}|w_+)$ and $p(\mathrm{x}|w_-)$ can be both obtained, then recall Bayes function, the posterior probabilities of two classes can be calculated as:

$$p(w_+|\mathrm{x}) = \frac{p(\mathrm{x}|w_+)p(w_+)}{p(\mathrm{x})}, p(w_-|\mathrm{x}) = \frac{p(\mathrm{x}|w_-)p(w_-)}{p(\mathrm{x})} \tag{7}$$

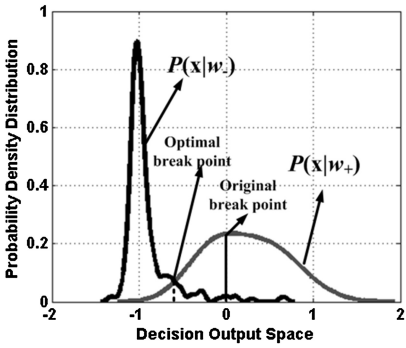

Fig. 1. Probability density distributions, original and optimal break points for a binary-class imbalanced problem, where w_+ denotes the positive class (minority class), and where w_- denotes the negative class (majority class).

It is clear that when $p(w_+|\mathrm{x}) = p(w_-|\mathrm{x})$, i.e., when $p(\mathrm{x}|w_+)p(w_+) = p(\mathrm{x}|w_-)p(w_-)$, the corresponding x value is selected as break point in Bayes decision. For class imbalance data, however, because $p(w_+) \ll p(w_-)$, to guarantee $p(\mathrm{x}|w_+)p(w_+) = p(\mathrm{x}|w_-)p(w_-)$, the actual break point will be pushed towards the minority class. Therefore, Fig. 1 shows that using the original break point acquired from ELM will seriously destroy the performance of the minority class, but when we find the horizon axis corresponding to the intersection point of two density distribution curves, it can be seen as the optimal break point for the classification task.

2.4 Description of the Proposed Algorithm

The detailed computational procedure of the proposed algorithm is described as follows:

Input: A binary-class training set $S = (\mathrm{x}_i, \mathrm{t}_i) \in \mathrm{R}^n \times \mathrm{R}^1$, where i=1,..., N, $\mathrm{t}_i \in \{-1, 1\}$; the number of hidden layer nodes L; the penalty parameter C existing in Eq.(2).

Output: The shift s between the original and optimal break points, and the modified ELM classifier F'.

Process:

a. Adopt the training set S and the penalty parameter C to train an ELM classifier F which has L hidden layer nodes;
b. Acquire the decision output of each training instance, and then map them into an one-dimensional feature space;
c. Obtain probability density distributions of two classes by using kernel probability density estimation approach, respectively;
d. Find the horizon axis x which corresponds to the intersection point of two probability density distribution curves, and then calculate the shift s by using the formula s=0-x;
e. Update ELM classifier from F to F" by adding s on each original decision output.

3 Experiments

3.1 Datasets and Parameters Settings

The experiments are carried out on thirty-two binary-class imbalanced data sets acquired from Keel data repository [12]. The detailed information of these data sets are summarized in Table 1, where IR denotes the class imbalance ratio.

To present the superiority of the proposed algorithm, we compared it with some other class imbalance learning algorithms in the context of ELM, including ELM [1], WELM1 [3], WELM2 [3], ELM-RUS, ELM-ROS [5] and ELM-SMOTE [6]. In addition, to guarantee the impartiality of the comparative results, grid search was adopted to search the optimal parameters, where sigmoid function was used as activation function at the hidden level, and two other parameters L and C were selected from {10, 20, …, 200} and $\{2^{-20}, 2^{-18}, \ldots, 2^{20}\}$, respectively. As for the performance evaluation, G-mean metric was used.

3.2 Results and Discussions

Considering the randomness of ELM, five fold cross-validation was adopted, and each experiment was randomly executed 10 times, finally the average classification results

Table 1. Data sets used in the experiments.

Data set	Number of features	Number of instances	IR
ecoli3	7	336	8.6
glass1	9	214	1.82
haberman	3	306	2.78
new_thyroid1	5	215	5.14
pima	8	768	1.87
vehicle1	18	846	2.9
wisconsin	9	683	1.86
yeast3	8	1484	8.1
abalone9_18	8	731	16.4
ecoli4	7	336	15.8
shuttle_c0_vs_c4	9	1829	13.87
vowel0	13	988	9.98
yeast4	8	1484	28.1
yeast5	8	1484	32.73
page_blocks0	10	5472	8.79
ecoli_0_1_vs_2_3_5	7	244	9.17
ecoli_0_1_vs_5	6	240	11
ecoli_0_3_4_vs_5	7	200	9
ecoli_0_6_7_vs_3_5	7	222	9.09
ecoli_0_6_7_vs_5	6	220	10
led7digit_0_2_4_5_6_7_8_9_vs_1	7	443	10.97
yeast_0_2_5_7_9_vs_3_6_8	8	1004	9.14
yeast_0_3_5_9_vs_7-8	8	506	9.12
cleveland_0_vs_4	13	177	12.62
shuttle_2_vs_5	9	3316	66.67
shuttle_6_vs_2-3	9	230	22
winequality-red_4	11	1599	29.17
winequality-red_3_vs_5	11	691	68.1
iris0	4	150	2
page-blocks_1_3_vs_4	10	472	15.86
vehicle0	18	846	3.25
glass_0_1_5_vs_2	9	172	9.12

were given. Table 2 provides the G-mean values of various algorithms, where in each row, the bold denotes the best result, the underline labels the second best and the italic stands for the worst one.

From Table 2, we observe that nearly all other algorithms outperform the original ELM algorithm, demonstrating each bias correction strategy can effectively alleviate the class imbalance problem. We also note that sampling and weighting technologies have quitely similar classification performance. As we know, WELM1 and WELM2 adopt different weights to punish the training errors, but there seems no a clear winner between them, as they have acquired the highest G-mean values on five data sets,

Table 2. G-mean values of various algorithms on 32 imbalanced data sets.

Data set	ELM	WELM1	WELM2	ELM-RUS	ELM-ROS	ELM-SMOTE	Proposed algorithm
ecoli3	*0.7182*	0.8792	**0.8883**	0.8733	0.8780	0.8602	0.8800
glass1	0.6712	0.6923	*0.6457*	0.6976	**0.7030**	0.6932	0.7020
haberman	*0.4703*	0.6262	0.5300	0.6305	0.6310	0.6103	**0.6419**
new_thyroid1	*0.8540*	0.9683	0.9327	0.9787	**0.9944**	0.9743	0.9831
pima	*0.6936*	0.7442	0.7127	0.7296	0.7314	**0.7483**	0.7474
vehicle1	*0.7709*	0.8248	0.7751	0.8049	0.8285	**0.8327**	0.8293
wisconsin	*0.9570*	**0.9711**	0.9667	0.9615	0.9597	0.9667	0.9707
yeast3	*0.7612*	0.9072	0.8968	0.9199	0.9197	0.9207	**0.9221**
abalone9_18	*0.3536*	0.8498	0.8267	0.7954	0.7851	0.8001	**0.8729**
ecoli4	*0.7966*	**0.9719**	0.8532	0.8891	0.9509	0.8292	0.9661
shuttle_c0_vs_c4	*0.9930*	0.9932	**0.9962**	*0.9930*	0.9932	0.9932	*0.9930*
vowel0	0.9858	0.9860	0.9888	*0.9672*	**0.9981**	0.9978	0.9909
yeast4	*0.2969*	0.8041	0.8088	0.8007	0.8090	0.8133	**0.8167**
yeast5	*0.5668*	0.9623	**0.9684**	0.9414	0.9615	0.9425	0.9680
page_blocks0	*0.7914*	0.8640	0.8510	0.9123	0.9190	0.9199	**0.9265**
ecoli_0_1_vs_2_3_5	0.8624	0.8844	0.8849	*0.8140*	0.8406	**0.8973**	0.8891
ecoli_0_1_vs_5	*0.8737*	0.9072	0.9129	**0.9213**	0.8942	0.8842	0.9181
ecoli_0_3_4_vs_5	*0.8398*	0.8761	0.8765	0.8986	**0.9394**	0.9283	0.9352
ecoli_0_6_7_vs_3_5	*0.7809*	0.8649	**0.8900**	0.7970	0.8112	0.8697	0.8817
ecoli_0_6_7_vs_5	*0.8481*	0.8878	0.8962	0.8629	0.8853	0.8966	**0.8996**
led7digit_0_2_4_5_6_7_8_9_vs_1	**0.8809**	0.8674	0.8653	*0.8223*	0.8331	0.8667	0.8514
yeast_0_2_5_7_9_vs_3_6_8	*0.8497*	0.9072	0.9061	0.8913	0.8987	**0.9082**	0.9063
yeast_0_3_5_9_vs_7-8	*0.4346*	**0.6934**	0.6690	0.6486	0.6526	0.6630	0.6847
cleveland_0_vs_4	0.6549	0.7934	0.7073	**0.8270**	*0.6348*	0.7079	0.8189
shuttle_2_vs_5	*0.9140*	0.9609	0.9389	0.9973	**0.9990**	0.9988	0.9986
shuttle_6_vs_2-3	0.9573	0.9573	0.9573	*0.9357*	0.9573	0.9707	**1.0000**
winequality-red_4	*0.0000*	**0.6936**	0.6405	0.6408	0.6630	0.6329	0.6733
winequality-red_3_vs_5	*0.0000*	0.4057	0.4823	0.6388	0.3562	0.2688	**0.6470**
iris0	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
page-blocks_1_3_vs_4	*0.8186*	0.9770	0.9608	0.9338	0.9549	**0.9841**	0.9782
vehicle0	0.9756	0.9744	*0.9662*	0.9703	0.9716	**0.9757**	0.9719
glass_0_1_5_vs_2	*0.1146*	0.7645	0.7308	0.6829	0.7606	**0.7730**	0.7555

respectively. We consider that the optimal weight settings should be closely related with the practical instance distributions. In sampling series algorithms, oversampling performs better than undersampling on majority data sets, especially on those highly skewed ones. On these data sets, the instances in the minority class are quitely sparse, causing much useful information loss by using RUS algorithm. Moreover, ELM-SMOTE obviously outperforms ELM-ROS as it have acquired two more best results and seven more second best results.

In contrast with six other algorithms, the proposed algorithm performs best, because it has acquired the highest G-mean value on nine data sets and the second highest G-mean value on fifteen ones. The results demonstrate that exploring the prior information about data distribution is helpful for improving classification performance in class imbalance tasks more or less.

4 Conclusions

In this article, a probability density estimation-based ELM classification algorithm is proposed to classify imbalanced data. Unlike other class imbalance learning algorithms, the proposed algorithm does not need to change the original data or weight distributions, but only to estimate the probability density distribution of the decision outputs acquired from ELM and then to find the optimal position to place the classification hyperplane. Experimental results on thirty-two benchmark data sets verified the effectiveness and superiority of the proposed algorithm.

Acknowledgements. This work was supported in part by National Natural Science Foundation of China under grant No. 61305058, Natural Science Foundation of Jiangsu Province of China under grant No. BK20130471, and China Postdoctoral Science Foundation under grant No. 2013M540404 and No. 2015T80481.

References

1. Huang, G.B., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. Part B Cybern. **42**, 513–529 (2012)
2. Huang, G., Huang, G.B., Song, S., You, K.: Trends in extreme learning machine: a review. Neural Netw. **61**, 32–48 (2015)
3. Zong, W., Huang, G.B., Chen, Y.: Weighted extreme learning machine for imbalance learning. Neurocomputing **101**, 229–242 (2013)
4. Zhang, W.B., Ji, H.B.: Fuzzy extreme learning machine for classification. IET Electron. Lett. **49**, 448–450 (2013)
5. Vong, C.M., Ip, W.F., Wong, P.K., Chiu, C.C.: Predicting minority class for suspended particulate matters level by extreme learning machine. Neurocomputing **128**, 136–144 (2014)
6. Sun, S.J., Chang, C., Hsu, M.F.: Multiple extreme learning machines for a two-class imbalance corporate life cycle prediction. Knowl. Based Syst. **39**, 214–223 (2013)
7. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**, 1263–1284 (2009)
8. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagation errors. Nature **323**, 533–536 (1986)
9. Ruck, D.W., Rogers, S.K., Kabrisky, M., Oxley, M.E., Suter, B.W.: The multilayer perceptron as an approximation to a Bayes optimal discriminant function. IEEE Trans. Neural Netw. **1**, 296–298 (1990)
10. Wan, E.A.: Neural network classification: a Bayesian interpretation. IEEE Trans. Neural Netw. **1**, 303–305 (1990)
11. Parzen, E.: On estimation of a probability density function and mode. Ann. Math. Stat. **33**, 1065–1076 (1962)
12. Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J. Multiple-Valued Logic Soft Comput. **17**, 255–287 (2011)