ARTIFICIAL NEURAL NETWORKS LEARNING IN ROC SPACE

Cristiano Leite Castro
Federal University of Lavras, Department of Computer Science
Campus Universitário - Caixa Postal 3037, 37200-000, Lavras, Brazil

Antônio Padua Braga
Federal University of Minas Gerais, Department of Electronics Engineering
Av. Antônio Carlos, 6.627 - Pampulha, 30161-970, Belo Horizonte, Brazil

Keywords: Multi-Layer Perceptron (MLP), Backpropagation algorithm, ROC analysis, Imbalanced data sets, Cost-sensitive learning.

Abstract: In order to control the trade-off between sensitivity and specificity of MLP binary classifiers, we extended the Backpropagation algorithm, in batch mode, to incorporate different misclassification costs via separation of the global mean squared error between positive and negative classes. By achieving different solutions in ROC space, our algorithm improved the MLP classifier performance on imbalanced training sets. In our experiments, standard MLP and SVM algorithms were compared to our solution using real world imbalanced applications. The results demonstrated the efficiency of our approach to increase the number of correct positive classifications and improve the balance between sensitivity and specificity.

1 INTRODUCTION

Binary classifiers based on Artificial Neural Networks (ANNs) have two objectives related to the performance of each class: one describes the classification accuracy for abnormal or positive examples (*sensitivity*) whereas the other describes the accuracy for normal or negative examples (*specificity*). In general, the simultaneous maximization of both objectives is achieved indirectly, through minimization of a cost function based on global training set error, such as, the mean squared error (Haykin, 1994). This is the case of many learning algorithms designed for the Multi-Layer Perceptron (MLP) topology since the introduction of the standard *Backpropagation* algorithm (Rumelhart and McClelland, 1986). However, it is well explored in the literature that, when the global error is minimized, the balance between *sensitivity* and *specificity* is affected by the difference between the class prior distributions (Provost et al., 1998), (Provost and Fawcett, 2001) and (Cortes and Mohri, 2004).

When training sets are imbalanced, classifiers usually present a good performance for the majority class but their performance for the minority class is poor. This occurs mostly because the global training error considers different errors as equally important assuming that the class prior distributions are relatively balanced (Provost and Fawcett, 2001). In addition, according to (Elkan, 2001), the majority class naturally imposes higher misclassification costs invalidating the uniform cost assumption made by the global error. Considering the case of most real world problems, when the class imbalanced ratio is huge, (Wu and Chang, 2005) observed that the separation surface learned by ANNs is skewed toward the minority class. Consequently, test examples belonging to the small class are more often misclassified than those belonging to the prevalent class.

To overcome this problem, different methods have been proposed. One approach is based on the data preprocessing in input space in order to balance the class distributions (for a good review of these methods see, for instance, (Weiss, 2004)). In the context of ANNs, (Japkowicz, 2000) evaluated *resampling* techniques in MLP classifiers and concluded that they were effective. Similarly, (Zhou and Liu, 2006) investigated the use of *undersampling*, *oversampling*, *threshold-moving* and *ensemble* models for the development of cost sensitive ANNs. Recently, in (Sun

Leite Castro C. and Padua Braga A. (2009).
ARTIFICIAL NEURAL NETWORKS LEARNING IN ROC SPACE.
In *Proceedings of the International Joint Conference on Computational Intelligence*, pages 484-489
DOI: 10.5220/0002324404840489

et al., 2007), the authors compared several cost sensitive *boosting* algorithms to address the learning problem with imbalanced data sets. The main criticism of the data preprocessing approach is the violation of the randomness assumption of the sample (training set) drawn from the target population. From a statistical point of view, as long as the sample is drawn randomly, it can be used to estimate the population distribution. Once the sample distribution is changed, by using resampling techniques, it can no longer be considered random. Nevertheless, these strategies have presented better results than the original methodology.

In the other approach, learning algorithms are adapted to improve performance of the minority class. In particular, in the case of ANNs, most of these strategies are based on the cost function modifications. In (Kupinski and Anastasio, 1999), (Sanchez et al., 2005), (Everson and Fieldsend, 2006) and (Graening et al., 2006), for instance, multi-objective genetic algorithms (MOGA) have been proposed to directly optimize *sensitivity* and *specificity*. In this case, these algorithms produce a set of non-dominated solutions describing the trade-off between these two measures (pareto set). The drawbacks of the multi-objective evolutionary algorithms are the high processing time and the difficulty in setting the parameters. Moreover, a decision algorithm is necessary to select the solution (operation point) in the pareto set. In the works mentioned earlier, the authors observed that the choice criterion is quite dependent on the problem to be solved.

The algorithm proposed in this paper is also based on the cost function modification. Using MLP binary classifiers, we extended the standard *Backpropagation* algorithm (in *batch* mode) to incorporate different missclassification costs via separation of the global mean squared error between the positive and negative classes. The objective is to control the trade-off between *sensitivity* and *specificity* achieving different solutions in ROC space. Our experimental results on both synthetic and real world data sets (from UCI Repository (Asuncion and Newman, 2007)) show that our modified *Backpropagation* algorithm is effective to obtain robust solutions for imbalanced problems.

2 ROC SPACE

According to ROC Analysis (Egan, 1975), the performance of some binary classifier on a particular data set may be summarized by a confusion matrix (see Table 1) (Fawcett, 2004). Each entry E_{kj} of the confusion matrix gives the number of the examples, whose true class was C_k and that were actually classified as C_j. Hence, the entries along the major diagonal represent the correct decisions made: number of true positives (TP) and true negatives (TN); and the entries off this diagonal represent the errors: number of false negatives (FN) and false positives (FP).

Table 1: Confusion Matrix.

	predicted pos	predicted neg
actual pos	TP	FN
actual neg	FP	TN

From this matrix, the metrics *sensitivity* (true positive rate) and *specificity* (true negative rate) can be estimated by the Equations 1 e 2, respectively. The ROC Space is defined as a graph which plots the true positive rate (*sensitivity*) as a function of the false positive rate ($1-$ *specificity*) (Fawcett, 2004). After evaluating a data set, each classifier produces a pair (*sensitivity*, $1-$ *specificity*) corresponding to a single point in ROC space.

$$sensitivity = \frac{TP}{TP+FN} \quad (1)$$

$$specificity = \frac{TN}{TN+FP} \quad (2)$$

3 MODIFIED BACKPROPAGATION ALGORITHM

In this Section, we describe our method to control the trade-off between *sensitivity* and *specificity* of Multi-Layer Perceptron (MLP) binary classifiers. The basic idea behind it is the separation of the global mean squared error and its gradient vector between the positive and negative classes. The theoretical foundations are based on the formulation of the standard *Backpropagation* algorithm (Rumelhart and McClelland, 1986) in *batch* mode, where the weights are updated only after all examples have been presented once for the network.

3.1 MLP Neural Network

Our approach considers *Multi-Layer Percetron* (MLP) neural networks with d inputs, one hidden layer with h nodes (units) and one output layer containing a single node, as shown in Figure 1.

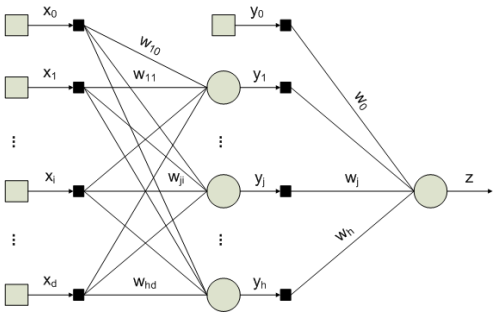

Figure 1: Multi-Layer Perceptron neural network topology considered in this work.

The activation of each hidden node j, due to the presentation of an input example (signal) $\mathbf{x}$, is given by,

$$y_j = f(u_j) = f\left(\sum_{i=0}^{d} x_i w_{ji}\right). \quad (3)$$

where each w_{ji} corresponds to a weight between the hidden node j and the input unit i. Similarly, the activation of the output node is based on the outputs of the hidden nodes,

$$z = f(v) = f\left(\sum_{j=0}^{h} y_j w_j\right). \quad (4)$$

where each w_j represents a weight between the output node and the hidden unit j. For the sake of simplicity, the *bias* was considered as an extra (input/hidden) unit whose value is equal to 1.

Since the scope of the method is limited to binary classification problems, we considered only one single output node and use sigmoid activation functions (hyperbolic tangent) $f(\cdot)$ for all nodes of the network. Thus, the classification of a given example $\mathbf{x}$ is based on the signal of the output z.

Consider also a training set $T = \{(\mathbf{x}_1,t_1),\cdots(\mathbf{x}_p,t_p),\cdots(\mathbf{x}_n,t_n)\}$ containing n examples. The error obtained in the output node due to the presentation of the p-th training example is defined as follows,

$$\varepsilon^{(p)}(\mathbf{w}) = \frac{1}{2}\left(t^{(p)} - z^{(p)}\right)^2 \quad (5)$$

where $z^{(p)}$ and $t^{(p)}$ correspond to the output and target values for the example p, respectively. The vector $\mathbf{w}$ denotes the collection of all weights of the network. From the definition of the error for the p-th training example, the cost function mean squared error can be calculated according to the following equation,

$$E(\mathbf{w}) = \frac{1}{n}\sum_{p=1}^{n} \varepsilon^{(p)}(\mathbf{w}) \quad (6)$$

3.2 Global Mean Square Error Separation

The training set T can be redefined as $T = T^+ \cup T^-$, where $T^k = \left\{(\mathbf{x}_1^k,t_1^k),\cdots(\mathbf{x}_p^k,t_p^k),\cdots,(\mathbf{x}_{n_k}^k,t_{n_k}^k)\right\}$, for $k = \{+,-\}$. The vector $\mathbf{x}_p^k$ corresponds to the p-th example of the class C^k. The target value for a positive example $\mathbf{x}_p^+$ is always $t_p^+ = +1$. The number of positive examples is given by n_+. Equivalent definitions hold for the negative class.

Given the data sets T^+ and T^-, we can describe the functional $E(\mathbf{w})$ as the sum of the cost functions $E^+(\mathbf{w})$ and $E^-(\mathbf{w})$ which represent the mean square error for each class,

$$E(\mathbf{w}) = E^+(\mathbf{w}) + E^-(\mathbf{w}) \quad (7)$$

where $E^k(\mathbf{w})$ is given by,

$$E^k(\mathbf{w}) = \frac{1}{n_k}\sum_{p=1}^{n_k} \varepsilon^{(p,k)}(\mathbf{w}) \;\text{ for } k = \{+,-\} \quad (8)$$

3.3 Gradient Vectors

From the separation of the functional $E(\mathbf{w})$ in $E^+(\mathbf{w})$ and $E^-(\mathbf{w})$, we can calculate the gradient vector for each cost function $E^k(\mathbf{w})$, using the following equation,

$$\nabla\mathbf{E}^k(\mathbf{w}) = \sum_{p=1}^{n_k} \frac{\partial\varepsilon^{(p,k)}}{\partial\mathbf{w}} \;\text{ for } k = \{+,-\} \quad (9)$$

where $\frac{\partial\varepsilon^{(p,k)}}{\partial\mathbf{w}}$ is the gradient vector over all weights of the network $\mathbf{w}$, calculated due to the p-th example of the class C^k.

Each component of the vector $\frac{\partial\varepsilon^{(p,k)}}{\partial\mathbf{w}}$ corresponds to a scalar gradient calculated for a given weight of the network. These values are estimated using the basic formulation established in the standard *Backpropagation* algorithm and proposed by (Rumelhart and McClelland, 1986):

1. for each weight w_j of the output layer, the scalar gradient due to p-th example of the class C^k is obtained using the following chain rule,

$$\frac{\partial\varepsilon^{(p,k)}}{\partial w_j} = \frac{\partial\varepsilon^{(p,k)}}{\partial z^{(p,k)}}\frac{\partial z^{(p,k)}}{\partial v^{(p,k)}}\frac{\partial v^{(p,k)}}{\partial w_j} \quad (10)$$

$$\frac{\partial \varepsilon^{(p,k)}}{\partial w_j} = -\left(t^{(p,k)} - z^{(p,k)}\right) f'\left(v^{(p,k)}\right) y_j^{(p,k)} \tag{11}$$

2. Similarly, the scalar gradient for each weight w_{ji} of the hidden layer is given by,

$$\frac{\partial \varepsilon^{(p,k)}}{\partial w_{ji}} = \frac{\partial \varepsilon^{(p,k)}}{\partial y_j^{(p,k)}} \frac{\partial y_j^{(p,k)}}{\partial u_j^{(p,k)}} \frac{\partial u_j^{(p,k)}}{\partial w_{ji}} \tag{12}$$

$$\frac{\partial \varepsilon^{(p,k)}}{\partial w_{ji}} = -\left(t^{(p,k)} - z^{(p,k)}\right) f'\left(v^{(p,k)}\right) w_j \; f'\left(u_j^{(p,k)}\right) x_i^{(p,k)} \tag{13}$$

3.4 Weight Update

The weight update of the standard *Backpropagation* algorithm (in *batch* mode) at iteration (epoch) *m*, is defined as follows,

$$\mathbf{w}^{(m+1)} = \mathbf{w}^{(m)} - \eta \nabla \mathbf{E}\left(\mathbf{w}^{(m)}\right) \tag{14}$$

where $\mathbf{w}^{(m)}$ is the weight vector at iteration *m* and η is a positive constant (learning rate). The update of $\mathbf{w}^{(m)}$ occurs in opposite direction of the gradient vector.

From the global mean squared error separation, we can describe the gradient vector $\nabla \mathbf{E}(\mathbf{w})$ as a weighted sum of the gradient vectors of $E^+(\mathbf{w})$ and $E^-(\mathbf{w})$,

$$\nabla \mathbf{E}(\mathbf{w}) = \frac{1}{\lambda^+} \nabla \mathbf{E}^+(\mathbf{w}) + \frac{1}{\lambda^-} \nabla \mathbf{E}^-(\mathbf{w}) \tag{15}$$

where the parameters λ^+ and λ^- with values varying from 1 to ∞, are used to constrain (penalize) the gradient vector magnitudes for the positive and negative classes, respectively. These parameters are introduced to assign different misclassification costs for each class and, therefore, to control the trade-off between *sensitivity* and *specificity* of the solutions during the learning process.

Note that, when λ^+ and λ^- assume values equal to 1, the weight update equation from the gradient vector $\nabla \mathbf{E}(\mathbf{w})$ leads to the standard solution which minimizes the global training error. Otherwise, we can try to find solutions in different areas of ROC Space (*sensitivity* x $1-$ *specificity*) according to λ^+ and λ^-.

4 EXPERIMENTS AND RESULTS

In this Section, we conducted experiments that illustrate our approach. Using a two-dimensional synthetic data set, we show that it is possible to control the MLP learning, obtaining different separation surfaces in input space. Moreover, in order to improve the MLP classifier performance on real world imbalanced problems, the parameters λ^+ and λ^- were adjusted to balance the costs imposed by the difference between the number of class examples.

4.1 Synthetic Data

This experiment illustrates the effect caused by the parameters λ^+ and λ^- in the separation surfaces learned by a MLP classifier (topology 2:5:1). A training set was generated from two-dimensional Gaussian distributions with mean vectors $[0,0]^T$ and $[2,2]^T$ and covariance matrices corresponding to the identity matrix. The ratio between the number of negative (circles) and positive (plus) examples is 10 : 1.

Figure 2 shows the separation surfaces obtained in three different situations:

1. Standard solution (dotted line) which minimizes the global error with λ^+ and λ^- equal to 1. Performance on the training set $\Rightarrow$ *sensitivity* = 0.58 and *specificity* = 0.98.
2. Solution (bold solid line) which aims to achieve a balance between the *sensitivity* and the *specificity* with $\lambda^+ = n_+$ and $\lambda^- = n_-$. Performance on the training set $\Rightarrow$ *sensitivity* = 0.90 and *specificity* = 0.89.
3. Solution (solid line) with high *sensitivity* by setting the parameters $\lambda^+ = 1$ and $\lambda^- = 100$. Performance on the training set $\Rightarrow$ *sensitivity* = 1.00 and *specificity* = 0.68.

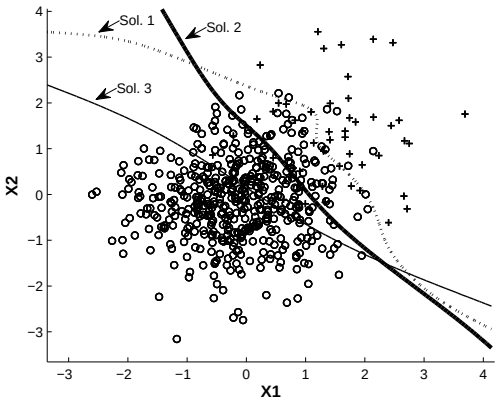

Figure 2: The effect caused by parameters λ^+ and λ^- in the separation surfaces learned by a MLP classifier with topology 2:5:1.

The analysis of these results, led to the following conclusions: the bad performance of the standard solution 1 (dotted line) is due to minimization of the

global mean squared error from an imbalanced training set. By constraining the magnitude of gradient vectors according to the number of class examples, the solution 2 (bold solid line) achieved a better balance between *sensitivity* and *specificity*. Finally, the solution 3 (solid line) obtained maximum sensitivity by having a very high cost for the positive class. However, its performance for the negative class was not good.

4.2 UCI Data Sets

In this Section, we used seven real world datasets from the UCI Repository (Asuncion and Newman, 2007) with different levels of imbalance (see Table 2). In order to have the same negative to positive ratio, stratified 7-fold crossvalidation was used to obtain training and test subsets (ratio 7:3) for each data set.

Table 2: Characteristics of the seven data sets used in experiments: number of attributes, number of positive and negative examples and class ratio: $\frac{n_+}{n_+ + n_-}$. For some data sets, the class label in the parentheses indicates the target class.

Data Set	#attrib	#pos	#neg	ratio
Diabetes	08	268	500	0.35
Breast	33	47	151	0.24
Heart	44	55	212	0.21
Glass(7)	10	29	185	0.14
Car(3)	06	69	1659	0.04
Yeast(5)	08	51	1433	0.035
Abalone(19)	08	32	4145	0.008

To obtain balanced solutions between *sensitivity* and *specificity*, the parameters λ^+ and λ^- were set according to the number of class examples ($\lambda^+ = n_+$ and $\lambda^- = n_-$). We named this strategy as Balanced MLP classifiers (BalMLP) and compared it with Support Vector Machines (SVM) (Cortes and Vapnik, 1995) and standard MLP classifiers (StdMLP) which minimize the global training error. The parameters of these classifiers were selected through grid-based search method (Van Gestel et al., 2004) and were kept equal in all runs for each data set. The optimal choices of these parameters are in Table 3.

Table 4 compares the results obtained for the test set using *sensitivity* and *specificity* measures. For each metric, the mean and standard deviation were calculated from 7 runs with different training and test subsets obtained from stratified 7-fold crossvalidation.

As shown in Table 4, our BalMLP strategy produced better *sensitivity* values for all data sets. Compared to the StdMLP and SVM classifiers, BalMLP performance was superior especially for data sets with higher imbalance degree. The separation surfaces learned in the input space were set to maximize the number of correct positive classifications. A better balance between sensitivity and specificity was achieved.

Table 3: MLP parameters: number of hidden nodes (#nodes) and training epochs (#epochs); and SVM parameters: regularization term (C) and radius of Gaussian function (σ) for RBF kernel.

Data Set	#nodes	#epochs	C	σ
Diabetes	5	7000	1.10	15.50
Breast	1	5000	72.50	19.50
Heart	2	3000	0.30	7.20
Glass	2	5000	3.40	0.40
Car	2	7000	0.60	15.10
Yeast	5	7000	0.1	16.30
Abalone	3	7000	1.00	1.00

However, note that the attempt to achieve a balanced solution based on the number of examples in training set does not necessarily ensure a balanced performance for the test set (see, for instance, Breast and Glass data sets in Table 4).

5 CONCLUSIONS AND FUTURE WORK

By separating the global training error between the positive and negative classes, our method achieved different solutions in ROC Space in order to circumvent the problem of lack of representativeness such as sparse and imbalance of class distributions in training sets.

In the results obtained with real world applications, we demonstrated that it is possible to increase the number of correct positive classifications and improve the balance between *sensitivity* and *specificity*. However, our approach does not consider complexity control techniques such as minimization of the magnitude of parameters, maximization of the separation margin and use of regularization terms. Our future efforts will focus on the relationship between the solutions obtained with our method and those that aim to achieve maximum generalization by controlling the complexity of models. We believe that the union of these strategies will help to direct the search for the optimal solution to the learning problem with imbalanced classes.

Furthermore, it is necessary to establish a precise relation for the adjustment of λ^+ and λ^- and the desired solution in ROC Space. So far, we have ob-

Table 4: Mean and standard deviation of *sensitivity* (in %) and *specificity* (in %) metrics on UCI data sets.

	StdMLP		SVM		BalMLP	
Data Set	sens	spec	sens	spec	sens	spec
Diabetes	61 ± 07	81 ± 06	70 ± 10	77 ± 04	73 ± 09	73 ± 04
Breast	42 ± 15	86 ± 08	60 ± 26	77 ± 08	66 ± 23	74 ± 11
Heart	47 ± 19	83 ± 05	61 ± 10	96 ± 09	73 ± 15	75 ± 09
Glass	84 ± 16	98 ± 02	87 ± 26	100 ± 00	90 ± 13	98 ± 03
Car	00 ± 00	100 ± 00	44 ± 21	98 ± 03	80 ± 15	77 ± 17
Yeast	06 ± 16	100 ± 00	29 ± 27	99 ± 04	82 ± 14	85 ± 03
Abalone	00 ± 00	100 ± 00	00 ± 00	100 ± 00	73 ± 12	79 ± 03

served that it depends on the number of class examples. We have also realized that this dependency can be influenced by the asymptotic boundaries imposed by the reduced size of the training and test data sets and also by the difference in noise level between the classes.

REFERENCES

Asuncion, A. and Newman, D. (2007). UCI machine learning repository.

Cortes, C. and Mohri, M. (2004). Auc optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

Egan, J. P. (1975). *Signal Detection Theory and ROC Analysis*. Academic Press.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI*, pages 973–978.

Everson, R. M. and Fieldsend, J. E. (2006). Multi-class roc analysis from a multi-objective optimisation perspective. *Pattern Recogn. Lett.*, 27(8):918–927.

Fawcett, T. (2004). Roc graphs: Notes and practical considerations for researchers. Technical report.

Graening, L., Jin, Y., and Sendhoff, B. (2006). Generalization improvement in multi-objective learning. In *International Joint Conference on Neural Networks*, pages 9893–9900. IEEE Press.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan, New York.

Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence*, pages 111–117.

Kupinski, M. A. and Anastasio, M. A. (1999). Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characterisitic curves. *IEEE Trans. Med. Imag.*, 18:675–685.

Provost, F. and Fawcett, T. (2001). Robust classification for imprecise environments. *Mach. Learn.*, 42(3):203–231.

Provost, F. J., Fawcett, T., and Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Rumelhart, D. E. and McClelland, J. L. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*, volume 1: Foundations. MIT Press.

Sanchez, M. S., Ortiz, M. C., Sarabia, L. A., and Lleti, R. (2005). On pareto-optimal fronts for deciding about sensitivity and specificity in class-modelling problems. *Analytica Chimica Acta*, 544(1-2):236–245.

Sun, Y., Kamel, M. S., Wong, A. K. C., and Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Van Gestel, T., Suykens, J. A. K., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., De Moor, B., and Vandewalle, J. (2004). Benchmarking least squares support vector machine classifiers. *Mach. Learn.*, 54(1):5–32.

Weiss, G. M. (2004). Mining with rarity: a unifying framework. *SIGKDD Explor. Newsl.*, 6(1):7–19.

Wu, G. and Chang, E. Y. (2005). Kba: Kernel boundary alignment considering imbalanced data distribution. *IEEE Trans. Knowl. Data Eng.*, 17(6):786–795.

Zhou, Z.-H. and Liu, X.-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Trans. Knowl. Data Eng.*, 18(1):63–77.