Cost-Sensitive Learning with Neural Networks

Matjaž Kukar [1], **Igor Kononenko**[1]

Abstract. In the usual setting of Machine Learning, classifiers are typically evaluated by estimating their error rate (or equivalently, the classification accuracy) on the test data. However, this makes sense only if all errors have equal (uniform) costs. When the costs of errors differ between each other, the classifiers should be evaluated by comparing the total costs of the errors.

Classifiers are typically designed to minimize the number of errors (incorrect classifications) made. When misclassification costs vary between classes, this approach is not suitable. In this case the total misclassification cost should be minimized.

In Machine Learning, only little work for dealing with non-uniform misclassification costs has been done. This paper presents a few different approaches for cost-sensitive modifications of the backpropagation learning algorithm for multilayered feedforward neural networks . The described approaches are thoroughly tested and evaluated on several standard benchmark domains.

1 Introduction

Artificial neural networks are very popular and useful tools in many fields of Computer Science. They are also well known in Machine Learning, where they are often used for learning classifiers (i.e., "procedures" that attempt to solve classification problems). In the last decade the backpropagation algorithm [9, 4, 3] has emerged as the most popular algorithm for the supervised training of multilayered feedforward networks.

Usually it is presumed that all classification errors have equal costs, but in practice this assumption often fails. There exist some domains where error costs may drastically differ between each other, i.e. in medicine, where overlooking an ill patient can be fatal.

This paper presents different approaches for cost-sensitive modifications of the basic backpropagation learning procedure. The approaches (cost-sensitive classification, adaptive output, adaptive learning rate and minimization of misclassification costs) are described and their results are compared to the original backpropagation procedure and to the simple "least expected costs" algorithm. Our experiments, thoroughly performed on several datasets, and utilizing several different cost situations, show that in most cases the minimization of misclassification costs performs significantly better than any of the other methods.

The paper is organized as follows. In Section 2 we provide some basic definitions which are used throughout the paper. Section 3 describes basic ideas of cost-sensitive learning, briefly describes the original backpropagation procedure and sketches different cost-sensitive modifications of the original backpropagation learning procedure. Section 4 presents experimental work, done to evaluate the different approaches. In Section 5 we show an example, how can the cost sensitive neural learning be used in the ROC analysis of the real-world medical problem. Section 6 gives some conclusions and indicates possibilities for further work.

2 Definitions

For dealing with misclassification costs we need to define the following terms: a *cost matrix*, a *cost vector*, an *uniform* case, and a metric for evaluation of classifers' performance in the non-uniform case.

The cost of misclassifying an example is a function of the predicted class and the actual class. This function, $cost(actual\ class, predicted\ class)$ is represented as a *cost matrix*. It is an additional input to the learning procedure and is also used to evaluate the ability of the trained network to reduce misclassification costs. The *cost matrix* is defined as follows:

- $Cost[i,j]$ = cost of misclassifying an example from "class i" as "class j"
- $Cost[i,i] = 0$ (cost of correct classification).

When all costs are equal, we have the *uniform* cost matrix:

$$\forall i,j : Cost[i,j] = \begin{cases} 1, & i \neq j \\ 0, & i = j \end{cases} \tag{1}$$

The *cost vector* represents the expected cost of misclassifying an example that belongs to the i-th class:

$$CostVector[i] = \frac{1}{1-P(i)} \sum_{j \neq i} P(j) Cost[i,j] \tag{2}$$

$P(i)$ is an estimate of the prior probability that an example belongs to i-th class. In the equal-cost case we have the *uniform cost vector*:

$$\forall i : CostVector[i] = 1 \tag{3}$$

The performance criterion is no longer the error rate (or classification accuracy), but the *average cost per example*:

$$Average\ cost = \frac{1}{N} \sum_{i=1}^{N} Cost[\text{actual class}(i), \text{predicted class}(i)] \tag{4}$$

where N is the number of testing examples. The error rate may be viewed as a special case of the average cost, when the uniform cost matrix is used.

$$\begin{array}{lcl} Error\ rate & = & \#\ of\ incorrectly\ classified\ examples\ /\ N \\ Accuracy & = & 1.0 - Error\ rate \end{array} \tag{5}$$

As a reference point to which all the results are compared, a simple algorithm that predicts the least expected cost class is used [7]. The

[1] Faculty of Computer and Information Science, University of Ljubljana, Tržaška 25, 1001 Ljubljana, Slovenia, {matjaz.kukar,igor.kononenko}@fri.uni-lj.si

© 1998 M. Kukar, I. Kononenko
ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

least expected cost class is found by minimizing the average cost of guessing class c on the training set of size N_T:

$$\textit{Least expected cost} = \min_{c \in Class} \frac{1}{N_T} \sum_{i=1}^{N_T} Cost[\text{actual class}(i), c] \quad (6)$$

Note that in the uniform case the least expected cost class is equivalent to the *default* class (the class that most frequently occurs in the training set).

3 Cost-sensitive learning of neural networks

In the field of Machine Learning, there has been some work done concerning cost-sensitive learning, starting with Breiman et al. [1], Knoll et al. [5], Pazzani et al. [7], Provost [8] and Turney [10]. There exist several articles in which different techniques are suggested [11], but little work has been done to evaluate and compare them. However, none of them deals with the cost-sensitive learning of neural networks.

3.1 Backpropagation learning of neural networks

The multilayered feedforward network consists of many interconnected nodes – neurons. Neurons are organized into one output layer and one or more hidden layers. Usually, the sigmoid activation fuction $f(x) = 1/(1+e^{-\Theta x})$ is used to calculate the output of the neuron.

Backpropagation is a specific technique for implementing gradient descent in weight space for a multilayered feedforward network [3]. The basic idea of this technique is to efficiently compute partial derivatives of an approximating function realized by the network. They are computed with respect to all the elements of the adjustable weight vector $\mathbf{w}$ for a given value of input vector $\mathbf{x}$. The learning procedure basically feeds the input vector to the network, calculates the output vector $\mathbf{o}$ of the network and compares it to the desired output vector $\mathbf{y}$. Based on the difference, the backpropagation procedure performs a gradient descent in the weight space.

Normally, the backpropagation algorithm minimizes the squared error of the network:

$$E = \sum_{Examples} \frac{1}{2} \sum_{i \in Output} (y_i - o_i)^2 \quad (7)$$

where o_i is the actual output of the i-th output neuron and y_i is the desired output.

The backpropagation learning algorithm modifies the synaptic weights w_{ji} with the *delta rule* [9, 3] by computing the δ's (i.e., the local gradients) and proceeding backward, layer by layer, starting with the output layer:

$$\begin{aligned} \Delta w_{ji}^{(n+1)} &= \alpha \Delta w_{ji}^{(n)} + \eta \delta_j y_i \\ w_{ji}^{(n+1)} &= w_{ji}^{(n)} + \Delta w_{ji}^{(n+1)} \end{aligned} \quad (8)$$

where n is the index of training iterations, η is the learning rate parameter that controls the magnitude of the weight changes and α is the momentum constant.

The computations of the δ's differ between output and hidden neurons (the following expression assumes that the sigmoid activation function is used):

$$\delta_j = \begin{cases} (y_j - o_j) \cdot o_j (1 - o_j), & \text{for output neurons} \\ o_j (1 - o_j) \sum_k \delta_k w_{kj}, & \text{for hidden neurons} \end{cases} \quad (9)$$

3.2 Estimating class probabilities

Each neuron from the output layer of the network represents one of the possible classes. An example is classified in the class which corresponds to the neuron with the maximum output. However, the output of the network can also be viewed in the probabilistic sense. We can take outputs of all output neurons and normalize the values with their sum:

$$\forall_{i \in Class} : P(i) = \frac{o_i}{\sum_{j \in Output} o_j} \quad (10)$$

The normalized output can be viewed as an estimate of the probability $P(i)$ that an example belongs to i-th class.

3.3 Modifications of the backpropagation learning algorithm

Clearly, the original backpropagation learning procedure is not suitable for cost-sensitive learning, because it minimizes the squared error (7) and not the costs of the errors made. When conceiving the modifications, one of important goals was that in the uniform case the behaviour of the modified algorithm remains identical to that of the original backpropagation algorithm.

Cost-sensitive classification. The most straightforward way to reduce misclassification costs is to leave the learning procedure intact and modify only the probability estimates of the network during the classification of unseen (testing) examples. The probability $P(i)$ that an example belongs to the class i is replaced with the altered probability, which takes in account the expected costs of misclassification:

$$P'(i) = \frac{CostVector[i] P(i)}{\sum_j CostVector[j] P(j)} \quad (11)$$

This favours the classes with higher expected misclassification costs. Note that only the probability estimate is changed. The actual output of the network is left intact, so the backpropagation learning works as in the original case.

Adapting the output of the network. This approach differs from the previous in that the actual outputs of the network (and not the estimated probabilities) are changed and appropriately scaled:

$$o'_j = \frac{CostVector[j] o_j}{\max_i CostVector[i]} \quad (12)$$

The idea is to correct the output of the network to give the classes with higher expected misclassification costs relatively higher impact on the learning. The corrected output values are also used in the backpropagation step.

Adapting the learning rate. This approach closely follows Breiman's [1] *altered priors* approach. The idea of this approach is that the high cost examples (that is, examples that belong to classes with high expected misclassification costs) can be compensated for by increasing their prevalence in the learning set. The altered priors can be simulated by the *weighted sampling* from the original learning set, that is, giving the high-cost examples higher weight. In neural network this can be implemented by increasing learning rate for high cost examples, thus giving them greater impact on the weight changes. To ensure the convergence of the modified backpropagation procedure, the corrected learning rate should also be accordingly normalized (p is the current training example):

$$\eta(p) = \frac{\eta \cdot CostVector[class(p)]}{\max_i CostVector[i]} \tag{13}$$

Minimization of the misclassification costs The misclassification costs can also be taken in account by changing the error function (7) that is being minimized. Instead of minimizing the squared error, the backpropagation learning procedure should minimize the misclassification costs. The error function is corrected by introducing the factor $K[i,j]$, $i =$ desired class, $j =$ actual class:

$$E = \sum_{p \in Examples} \frac{1}{2} \sum_{i \in Output} ((y_i - o_i) \cdot K[class(p), i])^2 \tag{14}$$

The factor $K[i,j]$ should be defined in such a way, that the behaviour of the backpropagation algorithm in the uniform case remains the same. Depending on the correct class, two cases are to be considered:

1. When dealing with output neuron j that corresponds to the desired class c of the training example p, the difference between desired and actual output of the neuron equals $y_c - o_c = 1 - o_c$, and can be interpreted as a probability of misclassifying the training example p into any of the incorrect classes. In the ideal case the $o_c = 1$ and misclassification probability is 0. The misclassification probability should be weighted with the expected misclassification cost of the class c, that is $CostVector[c]$.
2. For all other output neurons j that do not correspond to the correct class c of the training example p, the difference between the desired and the actual output of the neuron equals $y_j - o_j = 0 - o_j = -o_j$ (actually, this expression is squared, so its sign can be ignored). This can be interpreted as a probability that the example p that belongs to the class c will be incorrectly classified into the class j. This probability should be weighted with the cost of misclassifying class c as a class j, that is $Cost[c,j]$.

The factor $K[i,j]$ should therefore be defined as follows:

$$K[i,j] = \begin{cases} CostVector[i], & i = j \\ Cost[i,j], & i \neq j \end{cases} \tag{15}$$

If we look at the derivation of the backpropagation algorithm, we can see that the $K[i,j]$ behaves as a constant factor in the partial derivatives of the error function [3]. So the delta rule that takes in account the misclassification cost can be written as follows (c is the desired class of the current training example):

$$\delta_j = \begin{cases} (y_j - o_j) \cdot o_j(1 - o_j) \cdot K^2[c,j], & \text{for output neurons} \\ o_j(1 - o_j) \sum_k \delta_k w_{kj}, & \text{for hidden neurons} \end{cases} \tag{16}$$

To ensure the convergence of the modified backpropagation algorithm, the δ factor for output neurons should be normalized with $\max_{i,j} K[i,j]^2$:

$$\delta' = \frac{\delta}{\max_{i,j} K[i,j]^2} \tag{17}$$

4 Experiments

Our experiments were performed on several well-known datasets from the UCI repository (see Table 1).

Table 1. Some basic characteristics of the datasets used in our experiments.

Domain name	# examp.	# attrib.	# classes	Default class
Primary tumor	339	17	22	0.24
Heart disease	270	13	2	0.56
Soybean	630	35	15	0.14
Breast cancer (Ljubljana)	288	10	2	0.80
Pima diabetes	768	8	2	0.65
Hepatitis	155	19	2	0.79
Lymphography	148	18	4	0.54
Iris	150	4	3	0.33
Chess endgame (king–rook vs. king)	1000	18	2	0.67
Voting	435	16	2	0.61
Segmentation	2310	19	7	0.14

4.1 Experimental methodology

For each domain 100 runs of the ten-fold stratified cross validation were performed. In each run the new cost matrix was created from randomly generated numbers with real values between 1.0 and 10.0. Each matrix was required to have at least one non-diagonal element to be equal to 1.0.

The topology of the neural network was fixed to 20 hidden neurons in a single hidden layer. The learning parameters were also fixed. The learning rate η was set to 0.5 and the momentum constant α was set to 0.3. In every cross-validation step 10% of the training set was selected as a validation subset. The learning process was stopped, when the average costs measured on the validation set ceased to decrease.

4.2 Experimental results

The averaged results of 100 stratified cross-validation runs with random cost matrices are shown in Table 2. The focus is on the *minimization of the misclassification costs (MIN)* method.

The two-tailed paired t-tests were performed to test the hypothesis that the mean of the MIN method significantly differs from any of the other method. The bold-faced numbers in Table 2 show where this hypothesis holds.

As can be seen in Table 2, on average the *MIN* method performs best. The differences to all methods are significant ($p < 0.001$), using two-tailed, paired t-test. Also, the *MIN* method performs significantly better than the adaptive rate method on the problems with large number of classes (primary tumor – 22, soybean – 15 and segmentation – 7).

All cost-sensitive modifications as well as the original backpropagation algorithm perform in most cases significantly better than the simple least expected cost class method (Table 2). It is interesting to note that the average costs do not differ significantly between methods in domains where high classification accuracy was reached and the number of classes was small (iris, chess endgame, voting). The reason for this is relatively small magnitude of misclassification costs in our experiments (max. 10). The differences should be more distinct when using the higher cost (in magnitude of 100 or 1000)

The cost-sensitive classification and the adaptive output method perform only slightly better than the unmodified backpropagation procedure. Of them, the cost-sensitive classification should be preferred, because adaptive output method sometimes (breast cancer, diabetes) exceeds the least expected misclassification cost level.

The best methods are definitely *MIN* and adaptive rate. While the *MIN* method performs better, it also seems to be defined more in the

Table 2. Experimental results: bold-faced average costs are significantly ($p < 0.001$) worse than that of the *MIN* method. The "no costs" stands for the original backpropagation algorithm without cost-sensitive modifications.

Domain name	Classif. accuracy (no costs)	Average cost					
		MIN	Adap. lr. rate	Adap. out.	Cost-sens. classif.	No costs	Least exp. cost class
Primary tumor	0.44	2.65	**2.75**	**2.80**	**2.81**	**2.82**	**3.58**
Heart disease	0.84	0.30	0.30	**0.44**	**0.34**	**0.46**	**4.94**
Soybean	0.75	0.76	**1.05**	**1.12**	**1.18**	**1.20**	**3.74**
Breast cancer (Ljubljana)	0.80	0.35	**0.38**	**0.54**	**0.44**	**0.56**	**0.44**
Pima diabetes	0.73	0.35	0.35	**0.70**	**0.44**	**0.74**	**0.51**
Hepatitis	0.77	0.30	**0.31**	**0.39**	**0.39**	**0.63**	**0.44**
Lymphography	0.78	0.91	0.90	**1.03**	**0.96**	**1.09**	**1.73**
Iris	0.97	0.13	**0.14**	0.13	0.13	0.13	**2.10**
Chess endgame (king–rook vs. king)	0.99	0.02	0.02	0.02	0.02	0.02	**0.48**
Voting	0.95	0.09	**0.10**	**0.19**	**0.10**	**0.14**	**0.49**
Segmentation	0.94	0.22	**0.25**	**0.25**	**0.26**	**0.26**	**3.49**
Average over domains	—	0.55	**0.59**	**0.69**	**0.64**	**0.73**	**1.99**

spirit of the original backpropagation algorithm and should therefore be preferred.

4.3 Comparison with the previous work

Pazzani et al. [7] have tested several methods for cost-sensitive learning on four domains: Detrano heart disease, diabetes, mushroom and telephone. Unfortunately, their and our work can only be compared on the diabetes dataset. For this dataset (a two-class problem) mistaking a healthy patient for one with diabetes has a cost of 1 and the cost of mistaking a patient with diabetes for a healthy patient has a cost of 2:

$$\begin{array}{r|cc|} & \textbf{diabetes} & \textbf{healthy} \\ \textbf{diabetes} & 0 & 2 \\ \textbf{healthy} & 1 & 0 \end{array}$$

The testing strategies were also different. While Pazzani et al. [7] trained their methods on 440 examples and tested on the remainder, we performed ten-fold stratified cross-validation.

The results of our experiments are compared in Table 3. The comparison is only informative as the experimetal methodologes are quite different.

Table 3. Comparison of the results on the diabetes dataset

Method name	Average cost
Pazzani [7] – best result	0.37
Minimization of miscl. costs	0.31
Adaptive learn. rate	0.33
Adaptive output	0.41
Cost-sensitive classification	0.34
No costs (backpropagation)	0.44
Least expected cost class	0.65

5 ROC analysis in the ischaemic heart disease diagnostics with cost-sensitive neural networks

ROC (receiver operating characteristic) graphs have long been used in signal detection theory to depict tradeoffs between hit rate (*sensitivity*) and false alarm rate ($1.0 - \mathit{specificity}$). ROC analysis has later been extended for use in visualizing and analyzing the behaviour of diagnostic systems, and is used for visualization in medicine [8], where specificity–sensitivity relations are often analyzed.

$$\mathit{sensitivity} = \frac{\mathit{true\ positive\ test\ results}}{\mathit{total\ positives}} \tag{18}$$

$$\mathit{specificity} = \frac{\mathit{true\ negative\ test\ results}}{\mathit{total\ negatives}} \tag{19}$$

5.1 Ischaemic heart disease

Ischaemic heart disease is one of the world's most important causes of mortality [2], so improvements and rationalization of diagnostic procedures would be very useful. Because the suggestibility is possible, not all available diagnostic data is used in the diagnostic process. Since Machine Learning methods are not prone to suggestibility, they can use all the available data and might therefore be used to increase the classification accuracy, sensitivity and specificity of the diagnostic process.

In the usual setting, the Machine Learning algorithms are designed to maximize the classification accuracy. In our case, the sensitivity and the specificity were more important. The clinicians especially wanted to see if it is possible to increase the specificity of the diagnostic process without affecting the sensitivity too much. The ultimate goal was to reduce the number of patients that must unnecessarily be submitted to further pre-operative examinations (these can be potentially dangerous, unpleasant and are very costly).

The provided dataset consisted of 327 patients with completed diagnostic procedures, described with 77 attributes [6]. In 229 cases the disease was angiographically confirmed (definite proof of IHD presence) and in 98 cases it was excluded. We decided to use the

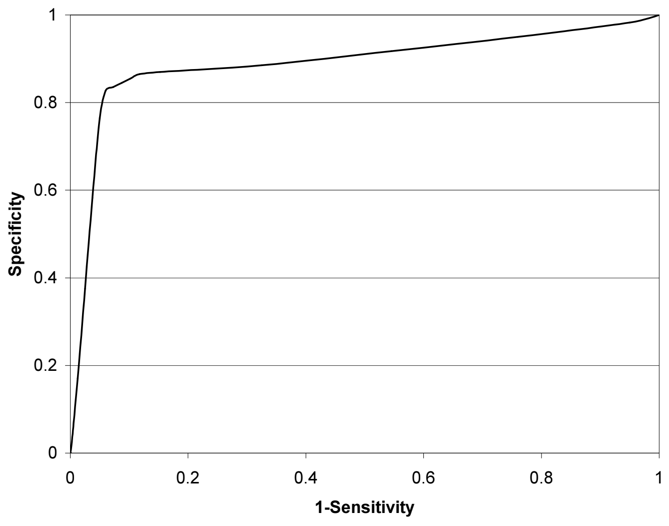

Figure 1. IHD diagnostic: the ROC curve

cost-sensitive neural networks with different misclassification costs in order to create a bias in to favour one or another class. We depicted the obtained results in a ROC graph and analyzed it in order to see whether the above goal can be reached.

Our experiments were conducted with costs ranging from 20 : 1 in favour of the positive class to 1 : 20 in favour of the negative class. The ROC graph (a ratio between sensitivity and specificity) is shown in Figure 1. By changing the misclassification costs, one actually traverses along this curve.

Experimental results showing how accuracy, specificity and sensitivity change with different costs are presented in Figure 2. The results shown are averages of the ten-fold stratified cross validation.

From the Figures 1 and 2 it can be seen how even slight increase of specificity drastically reduces sensitivity and vice versa. However, the most suitable point on the ROC curve shows the 2.5% increase in specificity and 5.5% in sensitivity, when compared with the clinicians' results on the same data. In the clinicians' opinion these results are good enough to be useful in practice.

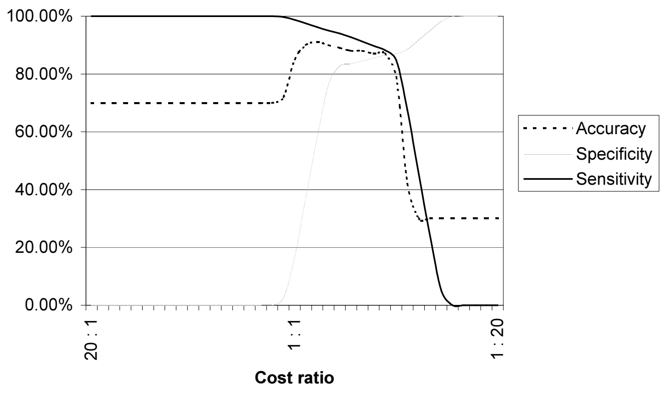

Figure 2. IHD diagnostic: accuracy, sensitivity and specificity

6 Conclusion

We have developed four different methods for cost-sensitive learning of the multilayered feedforward neural networks *(cost-sensitive classification, adaptive output, adaptive learning rate and minimization of misclassification costs)*.

Our experiment show that while all the methods successfuly minimize misclassification costs, it is best to use the *minimization of misclassification costs* method. In most cases it performs best, even significantly better than other methods. Its results also favourably compare to the results achieved by other approaches [7].

One interesting side-effect of the cost-sensitive modifications of the backpropagation learning procedure is that it provides the basis for the ROC analysis of the classifier, learnt by the neural network. This may be of interest in many fields of application, especially in medicine, where this approach was already successfuly validated.

There are a few things that remain to be explored furtherly. More experiments should be made on multi-class problems and with higher magnitudes of misclassification costs.

Overfitting of training data is one of the serious problems of the backpropagation learning procedure. It is caused by an oversized network and results in loss of its generalization abilities. The approach we used was to stop the learning process in the moment when the generalization abilities of the network ceased to increase (measured on the validation set). However, for this problem there exist more advanced solutions that deal with network complexity regularization. Our experiments with existing network complexity regularization procedures such as weight decay or weight elimination procedure [12] did not give satisfactory results. Therefore, these methods also need to be adapted for the cost-sensitive learning.

Acknowledgements

We thank dr. Ciril Grošelj from the University Nuclear Medicine Clinic in Ljubljana for collecting the ischaemic heart disease data and interpreting the results. This work was supported by the Slovenian Ministry of Science and Technology.

REFERENCES

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth International Group, Belmont CA, 1984.

[2] C. M. Gerson, 'Test accuracy, test selection, and test result interpretation in chronic coronary artery disease', in *Cardiac Nuclear Medicine*, ed., C. M. Gerson, 309–347, Mc Graw Hill, New York, (1987).

[3] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, New York, 1994.

[4] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley, 1990.

[5] U. Knoll, G. Nakhaeizadeh, and B. Tausend, 'Cost-sensitive pruning of decision trees', in *Proc. ECML'94*, (1994).

[6] M. Kukar, C. Grošelj, I. Kononenko, and J. Fettich, 'An application of machine learning in the diagnosis of ischaemic heart disease', in *Proc. Sixth European Conference of AI in Medicine Europe (AIME 97)*, Grenoble, France, (1997).

[7] M. Pazzani, C. Merz, P. Murphy, K. Ali ant T. Hume, and C. Brunk, 'Reducing misclassification costs: Knowledge-intensive approaches to learning from noisy data', in *Proc. 11th International Conference on Machine Learning*, pp. 217–225, (1994).

[8] F. J. Provost and T. Fawcett, 'Analysis and visualization of classifier performance: Comparision under imprecise class and cost distributions.', in *Proc 3rd International Conference on Knowledge Discovery and Data Mining (KDD-97)*. AAAI Press, (1997).

[9] D.E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, volume 1: Foundations, MIT Press, Cambridge, 1986.

[10] P. D. Turney, 'Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm', *Journal of Artificial Intelligence Research*, **2**, 369–409, (1995).

[11] P. D. Turney, 'Cost-sensitive learning bibliography'. `http://ai.iit.nrc.ca/bibliographies/cost-sensitive.html`, 1996.

[12] S. Weigand, A. Huberman, and D. E. Rumelhart, 'Predicting the future: a connectionist approach', *International Journal of Neural Systems*, **1(3)**, (1990).