Cost-Effective Classification for Credit Decision-Making

Karakoulas, G.

Publisher's version / Version de l'éditeur:

Proceedings of the Third International Conference on Artificial Intelligence Applications on Wall Street (AIAW'95), 1995

In Proceedings of the Third International Conference of AI Applications on Wall Street, June 1995.

Cost-Effective Classification for Credit Decision Making

Grigoris J. Karakoulas

Knowledge Systems Laboratory,
Institute for Information Technology,
National Research Council Canada,
Ottawa, Ontario, Canada K1A 0R6

grigoris@ai.iit.nrc.ca

Abstract

There is an increasing need for credit decision making systems that can dynamically analyze historical data and learn complex relations among the most important attributes for loan evaluation. In this paper we propose the application of a new machine learning algorithm, QLC, to the credit analysis of consumer loans. The algorithm learns how to classify a loan by minimizing the expected cost due to both credit investigation expenses and possible misclassification. QLC is built upon reinforcement learning. A dataset of actual consumer loans is used for evaluating the algorithm. The experiments reported show that QLC performs better than other cost-sensitive algorithms on this dataset.

1. Introduction

According to a recent U.S. Banker survey amongst the 113 top U.S. banks [15], the most popular approaches for automated decision-making for all types of credit products are application scoring and on-line credit bureau scoring. These credit-scoring procedures refer to the evaluation of each applicant by models that are derived from statistical discriminant analysis of the credit history of past applicants [12]. The main drawback of this type of evaluation stems from the reliance of discriminant analysis on a subjective assignment of scores to the credit attributes of an applicant's profile.

As also came out from this survey, more than 60% of the surveyed banks used judgemental — i.e. non-automated — scoring. The most important factors in the adoption of credit decision-making software by banks are: understanding system requirements and understanding credit management needs. In addition, a hindering factor in the deployment of current credit decision systems is their limitation in generating explanations when credit decisions are made. In contrast, the generation of explanations is a relatively easy task when judgemental scoring is used.

Artificial intelligence technologies have been employed for the development of credit-scoring software systems that can meet the emerging needs and requirements [6, 12]. On the one hand, expert systems have the advantage of representing and reasoning about relations amongst symbolic objects. This facilitates the task of generating explanations about objects and about inferences on the relations amongst objects. The disadvantage of expert systems is that the relations embedded in their knowledge base are pre-defined and their maintenance can become a tedious task. The increasing complexity of loan instruments, the volatility of the economic conditions and the importance of risk management in minimizing losses of loan portfolios impose the need for software with learning capabilities for dynamically analyzing various sources of historical data and capturing complex relations amongst the most important attributes for loan evaluation.

On the other hand, neural networks are good for learning complex relations by using non-parametric modeling. However, neural networks are usually considered as black boxes because it is

difficult to understand how learning occurs within their architecture and it is hard to explain how particular decisions are made through the networks once they are trained. Furthermore, neural networks may suffer from slow learning rates. The limitation of neural networks in explanatory capabilities is critical to their adoption for credit scoring. This is because there are regulatory constraints that require explanations to be given to consumers whose applications for a credit product have been rejected [4].

In this paper we propose the application of a new machine learning method for the credit analysis of consumer loans. Most classifiers in machine learning are built with the aim of minimizing errors made when predicting the classification of unseen examples. In contrast, our method is based on the general idea that it is worse to classify a bad customer as good than it is to classify a good customer as bad. Thus, classification errors may ensue different costs depending on the type of error. These costs can be in nominal values or if they are not known they can reflect constraints on the percentage of cases erroneously identified to belong to a particular class. This asymmetry in costs is of particular importance to applications like credit analysis where one class is comparatively rare and of special interest like loan defaults. Asymmetric misclassification costs may act as a focus mechanism for exploring the areas of the attribute space where the rare class is comparatively more common.

In a classification process, in addition to the costs of classification errors there are also the costs of tests which are incurred as information about the attributes of an object is acquired for making a classification decision. For example, credit investigation expenses are involved in the acquisition of information about the credit attributes[1] regarding an applicant. When both types of costs are considered the problem of cost-effective classification amounts to identifying for each state of the classification process an optimal sequence of tests (i.e. an optimal plan) for deciding among competing alternatives (i.e. classifications or additional tests).

Our approach to cost-effective classification is built upon reinforcement learning. The latter is a dynamic optimization paradigm within machine learning [13]. It is used for learning optimal policies in state-space problem-solving tasks. A policy specifies for each state what action to perform next. During learning, the system receives a reinforcement signal (a penalty or reward) after each action. The goal of the system is to find a policy that minimizes/maximizes the expected reinforcement over all future actions. Although reinforcement learning is quite different from typical concept learning, when test and misclassification costs are taken into account credit analysis becomes a stochastic optimization task. The goal of the task is to minimize the total cost of classification of each applicant.

The remainder of this paper is organized as follows. Section 2 proposes a problem formulation that makes reinforcement learning applicable to the cost-effective classification task. Section 3 develops a clustering technique for enhancing the scaleability of reinforcement learning for this complex task. The whole algorithm is presented in Section 4. Section 5 reports on experiments for evaluating the performance of the proposed algorithm. A sample of 1000 actual consumer loans granted is used for the experiments. Related and future work are discussed in Section 6. Conclusions are given in Section 7.

2. Problem Formulation

Consider a task where a case k is to be classified among m classes. There are n tests available each of which can be selected at any time but only once during a trial. The latter is defined as the sequence of tests ended by a classification. At each time t the set of possible actions A_t contains the m classifications and the tests not yet selected in the current trial. At the start of each trial this set has size $m + n$. When the agent selects a test it pays a cost which may be a function not only of the selected test but of prior tests as well. In medical diagnosis

[1] In the sequel, the term *test* will be used for denoting a *credit attribute*.

for example, a set of blood tests shares the common cost of collecting blood from the patient. This common cost is charged only once, when the decision is made to do the first blood test. The result of each test i is denoted by x^i. Having selected test i for case k, the agent observes the value of the test, $x_k^i \in X^i$, which has a distribution conditional on the history of observations prior to time t. The agent must then decide which action to perform next. It may choose to stop further testing and make a classification of the case to class j, $j \in [1, \ldots, m]$. However, if the predicted class is not equal to the actual class of the case, the agent is penalized by the cost of the error made. This cost is defined in the classification cost matrix. Each element $c_{i,j}$ of that $m \times m$ matrix gives the cost of predicting that a case belongs in class j, when it actually belongs in class i. The agent repeatedly goes through cases in order to learn a policy that minimizes in the long run the cumulative cost over all cases.

This problem is characterized by imperfect state information since the state variables referring to the actual classes cannot be observed directly. Instead, the agent gets information about them through the process of testing. For each case k we define the vector of observable history at time t as

$$h_t = \left(x_k^i, x_k^j, \ldots \right) \tag{1}$$

The vector consists of the observed values of the tests performed prior to time t for case k. At the start of each trial (i.e. new case) the dimension of the vector is initialized to zero. As a new observation is added at each stage of a trial, the dimension of the vector increases accordingly. The probability distribution of the history vector can serve as a sufficient statistic that can reformulate the original problem with imperfect state information into a problem with perfect state information. Thus, the state of the reformulated control problem is defined as

$$z_t = P\{h_t | \theta\} \tag{2}$$

where θ is the unknown parameter of the distribution. The performance criterion of the control problem is:

$$V_\pi = E^\pi \left\{ \sum_{t=0}^{\infty} \gamma^t c(h_t, a_t) \right\} \tag{3}$$

where γ with $0 < \gamma < 1$ is the discount factor. The cost function in (3) shows the dependence of the cost on prior tests as well as on the currently selected test. The policy is defined as $\pi: H_t \rightarrow P(A_t)$ mapping the space of observable histories into probability distributions of actions. The stochastic nature of the policy enables exploration of the state and action space for overcoming the problem of simultaneous identification of θ and control via π. We defer further analysis on how the policy probabilities $P^\pi\{a|h_t\}$ are calculated until the next section where a generalization scheme is developed. The probabilities will then be defined upon the generalization space.

The agent's objective is to choose a policy π^* such that:

$$\pi^* = \arg min_\pi \left[E^\pi \left\{ \sum_{t=0}^{\infty} \gamma^t c(h_t, a_t) \right\} \right] \tag{4}$$

Although, Dynamic Programming (DP) equations can theoretically be written for the optimization problem in (4), the assumptions for prior knowledge and the computational intractability of a DP algorithm, leads us to examine Q-learning as an alternative for this problem.

Q-learning is a reinforcement learning algorithm that is based on an asynchronous, stochastic approximation version of the DP equations [16]. Thus, in our problem the Q-learning equation can be written as:

$$Q_{t+1}(z_t, a_t) = (1 - \beta_t) Q_t(z_t, a_t) + \beta_t [c(h_t, a_t) + \gamma V_t(z_{t+1})] \tag{5}$$

where

$$V_t(z_{t+1}) = \min_b Q_t(z_{t+1}, b) \quad (6)$$

It should be noted that almost all of the theory of Q-learning assumes look-up table representations of the Q-value function. Such representation is not suitable for our problem for two reasons. First, the state of the system in (2) is a vector of real-valued variables. The learning algorithm should be able to generalize over the continuous state-action space and over the training dataset in order to perform well on previous unseen cases in the testing dataset. Second, the policy rules of our problem represent a mapping more complicated than the one of the policy rules in typical Q-learning. The generalization scheme should be able to accommodate such mapping. In the next section we develop a clustering technique suitable for tackling the issues associated with generalization in our problem.

3. A Clustering Technique for Generalization

The technique is based on the idea that as the agent explores the input $(Z_t \times A_t)$ and output $(\Re)$ spaces of the task, clusters are formed for each action from instances of points on the Q-surface. Each time a new instance is created from a history vector h_t the clusters of each action $a \in A_t$ are searched in order to estimate the conditional probabilities of selecting each of the clusters of action a given h_t. The Q-value of (z_t, a) can then be estimated from the Q-values of the clusters of action a using the conditional probabilities as weights. The action with the minimum Q-value is selected for the instance. After updating the Q-value of the instance via the Q-learning equations (5) and (6), the agent should also update its memory with the instance accordingly. We next give definitions of cluster and instance and then formalize the above procedures.

3.1 Definition of Cluster and Instance

A cluster i of action a denoted as c_i^a is represented as a 3-tuple:

$$c_i^a = \langle Z_i, Q_i, n_i \rangle \quad (7)$$

where Z_i is a vector $Z_i = [z_{i1}, z_{i2}, \ldots, z_{in}]$ with each z_{ij}, $j \in [1, \ldots, n]$, defined as

$$z_{ij} = P\{x^j = x_i^j \,|\, a, \theta_j\} \quad (8)$$

That is, z_{ij} is the probability of test j displaying the value x_i^j given action a and the parameter of the distribution θ_j. In (7) Q_i is the Q-value of the cluster and n_i is the number of instances that have been aggregated in the cluster.

An instance at time t denoted as s_t, is represented as

$$s_t = \langle h_t, Z_t, a_t, Q_t \rangle \quad (9)$$

where h_t is the history vector at t, a_t is the action that is selected for the instance and Q_t is the Q-value of the instance. Z_t is a vector of probabilities defined similarly to (8), i.e.

$$z_{tj} = P\{x^j = x_t^j \,|\, a_t, \theta_j\} \quad (10)$$

Suppose for the moment that each x^j is a discrete variable with r_j number of values. Also, assume that the agent has beliefs in the form of a prior probability density on $\Theta = [\theta_1, \ldots \theta_n]$. A prior density that is usually assumed in Bayesian analysis is the Dirichlet density [1,2]. The posterior distribution of the probability z_{tj} in (10) is also a Dirichlet density. Omitting some theoretical details we can estimate the distribution in (10) from

$$P\{x^j = x_t^j \,|\, a_t, \theta_j\} = \frac{N_{tj}^{a_t} + 1}{N_j^{a_t} + r_j} \quad (11)$$

where $N_{tj}^{a_t}$ is the number of times that when action $a_{t_{a_t}}$ is selected, x^j has the particular value x_t^j and $N_j^{a_t}$ is the number of times that when action a_t is selected, x^j has a value.

In the case that x^j is a continuous variable, it can either be discretized and treated as above, or one can apply Bayesian analysis for continuous distributions (for example, see [1]).

3.2 Q-value Estimation, Matching and Merging

Suppose that instance s_t is created from the current history vector h_t. Selecting an action for s_t requires estimating for each possible action $a \in A_t$ the value of $Q_t(s_t, a)$ from the Q-values of the clusters of action a. Since averaging over the Q-values of all clusters of an action involves a considerable amount of computation without necessarily a payoff in learning, we choose to average only over the k-nearest neighbors. The latter are determined according to the Euclidean distance between the vector Z_t of s_t and the vector Z_i of cluster c_i^a, i.e.

$$d(Z_t, Z_i) = \sqrt{\sum_{\tau=1}^{n} (z_{t\tau} - z_{i\tau})^2} \qquad (12)$$

where the $z_{t\tau}$ are estimated from (11) and $z_{i\tau}$ are the values stored in the cluster c_i^a. The formula for Q-value estimation is:

$$Q_t(s_t, a) = \sum_{i=1}^{k} P\{c_i^a \mid s_t\} \cdot Q_t(c_i^a, a) \qquad (13)$$

where $Q_t(c_i^a, a)$ is the value stored in the Q_i field of cluster c_i^a. The first term in the sum is the probability that cluster c_i^a is selected given instance s_t. This probability denotes the policy for selecting an action in the space of clusters C^{A_t}, namely $\pi_t: H_t \to P\{C^{A_t}\}$. It is given by

$$P\{c_i^a \mid s_t\} = \frac{P\{s_t \mid c_i^a\} \cdot P\{c_i^a\}}{\sum_j P\{s_t \mid c_j^a\} \cdot P\{c_j^a\}} \qquad (14)$$

The first term in the numerator of (14) is the probability of s_t having the particular history vector h_t given cluster c_i^a. This probability can be considered as a measure of how probable the values of h_t in s_t are, given cluster c_i^a. It is approximated by

$$P\{s_t \mid c_i^a\} = \frac{\sqrt{n} - d(Z_t, Z_i)}{\sqrt{n}} \qquad (15)$$

The second term in the numerator of (14) is the prior probability of any instance coming from cluster c_i^a. This probability is estimated by using a formula suggested by Anderson and Matessa in their work on Bayesian analysis of categorization [1]. Thus, we have

$$P\{c_i^a\} = \frac{c \cdot n_i}{(1-c) + c n_a} \qquad (16)$$

where c is the fixed probability that an instance comes from a cluster, n_i is the number of instances aggregated in cluster c_i^a and n_a is the number of instances aggregated in all clusters of action a.

Merging of an instance with a cluster requires the following two conditions to be satisfied: (i) $d(Z_t, Z_i) < \varepsilon_1$ and (ii) $|Q_t - Q_i| < \varepsilon_2$. If the two conditions are met then the instance is aggregated in the cluster by updating the fields of the cluster: $z_{ij} = z_{ij} \cdot n_i / (n_i + 1) + z_{tj} / (n_i + 1)$, $Q_i = Q_i \cdot n_i / (n_i + 1) + Q_t / (n_i + 1)$ and $n_i = n_i + 1$.

Similar conditions and operations apply when merging two clusters together.

4. The Proposed Algorithm

We assume that the dataset of the classification task has been split into a training set and a testing set. During learning the agent picks a case from the training set randomly without replacement and initiates a sequential decision process for the case, i.e a trial. During the trial the agent selects actions for making new estimates of the probabilities in (11) and updating the Q-values of its generalization space accordingly. When the agent selects a classi-

fication the current trial ends and a new one starts for the next case. Whenever all the cases of the training dataset have been processed this marks the end of an epoch. A new epoch is created by repeating the above procedure for the whole training set. Learning stops when the average cost of classification in the training set — total cost for the set divided by the number of cases in the set — is within ε between two consecutive epochs. The steps of the Q-learning with clustering (QLC) algorithm for one trial are shown in Figure 1.

Do:

(i) Create an instance s_t from the current history h_t;

(ii) For each possible action $a \in A_t$ estimate $Q_t(s_t, a)$ from its clusters;

(iii)Choose with probability ξ the action $a_t = \arg min_b [Q_t(s_t, b)]$;

(iv) Apply action a_t and pay the cost $c(s_t, a_t)$;

(v) If a_t is a test, update the history and probability vectors to h_{t+1} and z_{t+1} respectively;

(vi) Update the Q-value of s_t by

$$Q_{t+1}(s_t, a_t) = (1-\beta_t) Q_t(s_t, a_t) + \beta_t [c(s_t, a) + \gamma V_{t+e}(s_{t+1})]$$

where $V_{t+e} = min_b Q_{t+e}(s_{t+e}, b)$ is the e-step lookahead value of s_{t+1};

(vii) Update the memory either by merging s_t with a cluster of a_t or by creating a new cluster with only one instance s_t; check whether any clusters of a_t can be merged;

Until a_t is a classification action.

Figure 1: The steps of the QLC algorithm for one learning trial.

Step (iii) defines the exploration scheme of the algorithm. A value is randomly sampled from a uniform distribution in (0,1). If this value is less than ξ then the action with the minimum Q-value is chosen. Otherwise, any action is randomly selected. This scheme enables the algorithm to sufficiently explore the state and action space before converging to a good local optimum. In step (vi) the lookahead value $V_{t+e}(s_{t+1})$ is calculated by iterating over steps (i)-(vi) e times. We introduced this lookahead scheme due to empirical evidence from our experiments that this scheme improves the efficiency of the above algorithm.

5. Experiments

The experiments reported in this section were performed on a sample of 1000 actual consumer loans granted by a German bank. There are 20 attributes in the dataset that take symbolic or real values. There are also two classes of loans: good loans (70% of the dataset) and bad loans (30% of the dataset). The dataset was retrieved from the University of California at Irvine collection of datasets [7]. It was donated to the Irvine collection by Hans Hofmann[2].

Two experiments were performed. The purpose of the first experiment was to compare the performance of the QLC algorithm against the performance of other statistical and neural network algorithms on this dataset as reported in [11]. No test costs were assumed in this experiment. The purpose of the second experiment was to demonstrate the performance of QLC when both test and classification error costs are considered. Due to lack of information about actual credit investigation expenses we assumed a cost of one unit for each test. QLC is compared with Nunez's cost-sensitive algorithm EG2 [8]. This algorithm takes into account only the cost of testing. In both experiments the misclassification cost matrix had the form of Table 1.

It should be mentioned that part of the implementation of the QLC algorithm involves a discretization procedure. In both experiments each real-valued attribute of the dataset was discretized by dividing its range of values in the training set into five inter-

[2] The dataset has the URL ftp://ftp.ics.uci.edu/pub/machine-learning-databases/statlog/german/german.data.

vals of approximately equal size. We also used 4-nearest neighbor for estimating the Q-values by (13). The coupling probability in (16) was set to 0.3. The exploration probability of the QLC algorithm was set to $\xi = 0.9$. The learning rate β_t in the Q-value equation (5) had initial value 0.3 and was decayed as a function of learning experience. For each action the Q-value of state-action pairs was initialized to zero. The threshold ε for stopping training was set to 0.001.

Actual Class	Guess Class	Classification Error Cost
class 1	class 1	$0.0
class 1	class 2	positive error cost
class 2	class 1	negative error cost
class 2	class 2	$0.0

Table 1: The matrix of classification error costs.

In the first experiment we used the same procedure as in [11] for splitting the dataset into a training and a testing set. The training set consisted of 200 good and 200 bad loans randomly chosen from the initial dataset. The testing set consisted of the remaining cases, i.e. 500 good loans and 100 bad loans. We adopted this splitting procedure in order to ensure comparability of our results with those in [11]. For the same reason, the positive error cost was set to 1.0 and the negative error cost to 13.3.

The results are shown in Table 2. LDA is linear discriminant analysis; QDA is quadratic discriminant analysis; CART is a statistical method for building decision trees [3]; NN1 is a neural network with two hidden layers, 45 nodes in the first and 5 nodes in the second layer; and NN2 is a neural network with two hidden layers, 40 nodes in the first and 5 nodes in the second layer. The results of these five methods are taken from [11]. %N.E. denotes the rate of negative errors in the testing set, i.e. the fraction of bad loans that the classifier judges positive. %P.E. denotes the rate of positive errors in the testing set, i.e. the fraction of good loans that the classifier judges negative. The average cost is computed as the total cost of classifying the cases in the testing set divided by the number of cases.

Algorithms	No. Attr.	%N.E.	%P.E.	Avg Cost
LDA	20	28.7	29.1	0.88
QDA	20	28.3	34.0	0.91
CART	15	27.7	28.9	0.85
NN1	20	38.0	24.0	1.04
NN2	20	24.0	31.2	0.79
QLC	20	15.7	25.2	0.56

Table 2: Performance with cost ratio=13.3.

The results of LDA and QDA were derived by leave-one-out cross-validation. The results of CART, NN1 and NN2 were computed by using only one testing set. For the training of the CART algorithm 15 attributes were selected from the 20 attributes of the dataset. The QLC algorithm was run on 10 pairs of training and testing sets. Each pair was formed by randomly splitting the initial dataset according to the aforementioned procedure. The results reported on QLC are averages over the 10 testing sets. Although the algorithms have not been evaluated in exactly the same way, QLC shows a better performance than the other algorithms in terms of both average cost and error rates.

The above splitting procedure creates a training set with equally sized classes in order to enhance learning of the rare class of bad loans. In the respective testing set, however, the ratio of the size of the two classes is different from the ratio in the initial dataset. This disparity may be biasing the results of Table 2. In the second experiment we used a different splitting procedure. The initial dataset was randomly split into 10 pairs of training and testing sets. Each training set consisted of two thirds of the dataset and each testing set consisted of the remaining one third. A cost of one unit was assumed for each test. To enable sufficient testing we set the positive error cost to 40.0, i.e. a value greater than the total test cost. The negative error cost was set according to the negative-to-positive error cost ratio. We experimented with two values

of the error cost ratio: 5.0 and 13.3. In [11] these two values are suggested as the lower and upper limits of the error cost ratio.

The results of this experiment are shown in Tables 3 and 4. QLC performs better than EG2. It should be noted that because EG2 considers only test costs the different values of the error cost ratio do not affect the performance of the algorithm in terms of accuracy. QLC has better performance with cost ratio equal to 5.0 than with cost ratio equal to 13.3.

Algorithms	%N.E.	%P.E.	Avg Cost
QLC	18.2	22.6	32.84
EG2	60.9	14.9	42.56

Table 3: Performance with cost ratio=5.0.

Algorithms	%N.E.	%P.E.	Avg Cost
QLC	16.4	27.5	54.67
EG2	60.9	14.9	102.38

Table 4: Performance with cost ratio=13.3.

6. Discussion

There has been an increasing interest within the machine learning community for devising classification algorithms that are sensitive to either the costs of tests, e.g. [8], or to the costs of classification errors, e.g. [9] (see [5] for an extensive list of references). Turney [14] has recently proposed the ICET algorithm that takes both types of costs into account. The aforementioned research has focused on extending typical decision-tree and rule induction algorithms by either incorporating heuristic cost-sensitive attribute selection metrics or by building a two-tiered method for selecting among decision trees or rule-sets based on their cost-effectiveness.

In the statistics field, the CART algorithm [3] allows misclassification costs to be incorporated into the test selection process of a decision tree. A limitation of the CART algorithm is that it requires converting a cost matrix to a cost vector. This conversion results in having a single quantity to represent the importance of avoiding a particular type of error. The accuracy of the conversion depends on the accuracy of two estimates: (i) the frequency of examples of each class and (ii) the frequency that an example of one class might be mistaken for another.

In this paper we have introduced a new strategy for test selection given the goal of minimizing the expected cost due to both testing and classification errors. The strategy is realized through a single incremental learning algorithm. A particular advantage of our approach is that since the algorithm is incremental, after the learning system is deployed new cases of customers' loans can be incorporated in the system's memory depending on how informative these cases are with respect to the classification model already learned. In other work [5], we have empirically shown using three datasets from the domain of medical diagnosis that QLC performs better than related cost-sensitive classification algorithms. In that work actual costs were used for the medical tests. Future work should, therefore, examine the performance of QLC on credit decision making when actual credit investigation expenses are considered for the test costs.

Due to its stochastic optimization context, our algorithm can be extended for developing more sophisticated credit decision making models that take into account additional pragmatic considerations of credit granting decisions such as the risk of cash flows from credit sales [10].

7. Conclusion

This paper examined the problem of minimizing the expected classification cost due to both tests and classification errors in credit decision making. We presented a new cost-effective classification

strategy that is realized through the QLC algorithm. The latter is a single incremental learning algorithm which is based on a stochastic optimization framework. QLC scales up Q-learning for dealing with the intrinsic issues of imperfect state information and of generalization over continuous spaces and over training data.

We empirically evaluated the performance of QLC using a dataset of actual consumer loans granted. Previous work using this dataset focused only on misclassification costs. QLC performed better than the algorithms reported in that work. When test costs are assumed QLC performs better than both the EG2 algorithm that takes only test costs into account.

Further experimentation is needed to analyze the performance of the QLC algorithm especially when actual credit investigation expenses are considered. Other pragmatic considerations of credit decision making should also be investigated.

8. References

[1] Anderson, J.R. & Matessa, M. Explorations of an incremental, bayesian algorithm for categorization. *Machine Learning*, 9, 1992, 275-308.

[2] Berger, J.O. *Statistical decision theory and bayesian analysis*. New York: Springer, 1985.

[3] Breiman, L., Friedman, J., Olshen, R., & Stone, C. *Classification and regression trees*. Wadsworth, 1984.

[4] Borowsky, M. Looking for a Net Gain. *Credit Card Management Europe,* Vol. 2, 1993, 40-42.

[5] Karakoulas, G.J. Q-Learning for Cost-Effective Classification. KSL-IIT Technical Report, National Research Council Canada, May 1995.

[6] Keyes, J. Winning Back Investor's Confidence. Information Strategy: The Executive's Journal, Vol.1, 1992, 42-44.

[7] Murphy, P.M. & Aha, D.W. *UCI Repository of Machine Learning Databases*. University of California at Irvine, Department of Information and Computer Science, 1994.

[8] Nunez, M. The use of background knowledge in decision tree induction. *Machine Learning*, Vol. 6, 1991, 231-250.

[9] Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. Reducing Misclassification Costs: Knowledge-Intensive Approaches to Learning from Noisy Data. *Proceedings of the Eleventh International Conference on Machine Learning, ML-94,* 1994.

[10] Scherr, F.C. Credit-Granting Decisions Under Risk. *The Engineering Economist*, Vol. 37, 1992, 245-262.

[11] Seitz, J. and Stickel, E. Consumer Loan Analysis Using Neural Networks. In: *Adaptive Intelligent Systems,* S.W.I.F.T. (ed.), pp.177-192. Elsevier Science Publishers, 1993.

[12] Srinivasan, V., Kim, Y.H. Credit Granting: A Comparative Analysis of Classification Procedures. *Journal of Finance*. Vol. XLII, 1987, 665-681.

[13] Sutton, R. Introduction: The Challenge of Reinforcement Learning. Special Issue on Reinforcement Learning, *Machine Learning,* Vol. 8, 1992, 225-227.

[14] Turney, P. Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research* Vol. 2, 1995, 369-409.

[15] U.S. Banker. Lots of Ways to Make Credit Decisions. *U.S. Banker*, Vol. 102, 1992, 57-59.

[16] Watkins, C.J.C.H. *Learning from delayed rewards*. Ph.D. thesis. King's College, cambridge University, UK, 1989.