A Market-based Rule Learning System[1]

QingQing Zhou[a] & Martin Purvis[b]

[a]GuangDong Data Communication Bureau
China Telecom
1 Dongyuanheng Rd., Yuexiunan, Guangzhou 510110, China

zqq@gddc.com.cn

[b]Department of Information Science
University of Otago
P O Box 56, Dunedin, New Zealand

mpurvis@infoscience.otago.ac.nz

Abstract

In this paper, a 'market trading' technique is integrated with the techniques of rule discovery and refinement for data mining. A classifier system-inspired model, the market-based rule learning (MBRL) system is proposed and its capability of evolving and refining rules is investigated. Experimental results indicate that the MBRL system is a potentially useful additional tool that can be used to refine neural network extracted rules and possibly discover and add some new, better performance rules. As a result, it can lead to improved performance by increasing the accuracy of the rule inference performance and/or improving the comprehensibility of the rules.

Keywords: market-based, rule learning, data mining.

1 Introduction

People have used markets for thousands of years to get things done. In human markets, shopping centres get built, and new products are designed, all without a global controller or overseer. Any systems that use the concept or certain features found in a market can be called "market-based systems" (Clearwater 1996). In contrast to the use of a centralized controllers, a market-based system does not need any of the agents in the system to know all the parameters of the system in order for the overall system to function smoothly, instead through the simple interactions of trading among individual agents, a global optimization can be achieved. This feature of market-based systems offers promise (Zhou & Purvis 1999) for application to rule extraction and refinement systems. By adopting the concept of economic trading behaviour among individual commercial agents, a rule discovery system can be thought of as an artificial market where individual rule agents are interacting and competing in order to achieve satisfactory behaviour.

In this paper, a 'market trading' technique is integrated with the techniques of rule discovery and refinement for data mining. A classifier-system-inspired model, the market-based rule learning (MBRL) system is proposed. As described in (Zhou 2003), current classifier systems share major weaknesses: difficulties in interpretation, initial classifier chain generation, and initial system parameter setting. This paper addresses how the MBRL system can solve or lessen these difficulties.

In this research, the MBRL system is proposed as a post-processing tool to be used with feed-forward neural networks, and the feed-forward neural network rule extraction technique, NeuroLinear (Setiono & Liu 1997), in order to improve the quality of extracted rules from feed-forward neural networks.

2 The MBRL System Structure

The market-based rule learning (MBRL) system is an adaptive learning system based on the market principle and is used to modify rule sets that have been previously generated by other learning systems to improve the performance of rule sets in terms of predictive accuracy and comprehensibility.

The MBRL system takes the basic structure of a classic classifier system (Holland&Reitman 1978) but introduces some changes in each of the layers. Like a classifier system, an MBRL system consists of three layers (see Figure 1). The first layer sees to it that the system is able to provide answers to the problems it is confronted with. This is the rule and message system. The second layer evaluates the performance of the first layer. It can also adjust that layer's performance by using the payoff provided by the environment. This payoff is high if the behaviour of the system is good and low otherewise. It cannot, however, change the behaviour in a creative way, as it can only adjust things that are already present in the system. The algorithm used for this aspect is called "apportionment of credit". The task of the third layer is to try to find new ways in which the learning system can perform its function. This is accomplished by the genetic algorithm (Holland 1975).

[1] Copyright ©2004, CRPIT Series Volume 32 - ACSW Frontiers 2003 (Edited by James Hogan, Paul Montague, Martin Purvis and Chris Steketee. ISBN 1 920682 14 7) by the Australian Computer Society. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

2.1 Rule and Message System

In general, the rules derived by means of rule extraction strategies are of the following form:

```
IF
<condition1>&<condition2>&...<conditionN>
Then <action>
```

Classifier systems have traditionally used binary strings to represent rules and inputs. In contrast, real-valued representations are adopted by the MBRL system.

As described in (Zhou 2003), the binary string representations are partially responsible for the difficulty of setting initial system parameters. Many binary string related parameters, such as the word length of the messages, the word length for each condition and action, and the probability of selecting a wildcard (#) in a randomly generated population, must be set. If real-valued representations are used, this parameter setting can be omitted. In addition some other parameters, such as the number of conditions in the antecedent, can be set in a straightforward fashion.

The binary-string rule representations also provide a barrier for inspecting rules transparently. In order to exhibit the learning results on the application of the Wisconsin Breast Cancer Database (UCI 1989), Wilson (2000) had to convert the rules with binary string representations into real-valued representations. The MBRL system's real-valued representation has an advantage in terms of the interpretation of learned rules and avoids the additional overhead of conversion across different rule representations that are often involved with classifier systems.

Furthermore, since real-valued variables, such as *temperature* or *age*, are typical in real world problems, it is .natural to use real-valued representations for these applications.

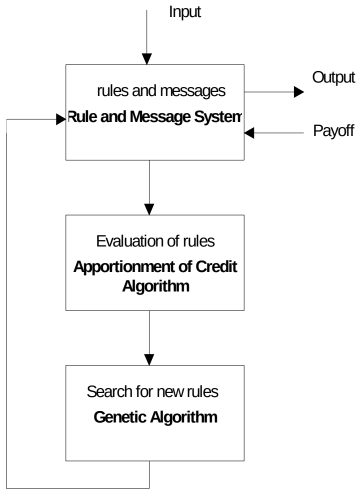

Figure 1: Model of a market-based rule learning system

2.2 Apportionment of Credit Algorithm

A classic classifier system employs the bucket brigade algorithm to modify strength of classifiers. Strength modifications occur via three interrelated mechanisms:

Auction

Reinforcement & punishment

Taxation

In the MBRL system, an algorithm called the *market-based trading algorithm*, is used for modifying the strengths of rules. Strength modification occurs via two mechanisms:

Reinforcement & punishment

Taxation

In the MBRL system when rules are matched against environmental messages or actions of other rules, they do not participate in an activation auction. Instead, all matched rules are allowed to perform their actions. The motivation for elimination of the auction mechanism comes from the MBRL system's principal role as a rule refinement tool. It is expected that the MBRL system only deals with existing rule sets extracted using other learning techniques, and thus the number of initial rules in the MBRL is not large (say, less than 100). Under this assumption, the elimination of the auction mechanism gives each rule an equal opportunity to perform and affect the outside world.

Imagine a message from the external environment flowing to the system as the action of a business startup that is offering shares on the stock market. The individual rules in the rule base can be thought of as stock trader agents that may make an investment (buy shares) in the business in order to gain a profit. If the number of trader agents that are interested in purchasing some shares of the business is not large, these trader agents should be considered to have an equal opportunity to make an investment without unnecessary competition in an auction. So when a rule is activated, it can be imagined that a rule agent has made an investment to purchase some shares of the business: its investment serves as a payment to the business shares provider, which is the incoming environmental message in this case. In the meantime, the activated rule posts its action on the message list. If its action triggers other rules, we can imagine that the shares are sold to other agents, and the receipt is collected and transformed from these buyer agents. However, if its action does not trigger any other rules, we can consider this as due to no trader agents being interested in making investments on it, and no payment is received.

If a rule sequence produces a 'good' (i.e. that which is desired) final output, it can be imagined that the successful chains of trader agents led to the ultimate buyers - a reward is received from the environment. However, if a rule sequence produces a 'bad' final output, we can think that the unsuccessful chains of trader agents have not led to the ultimate buyers - no reward is given by the environment. Apart from Wilson's XCS (1995) model, which is a single-layer classifier system that does not need to use the auction and the bucket brigade algorithm, the author has not seen any other systems in the field that deliberately omit the auction mechanism.

As in classifier systems, each rule in the MBRL system maintains a record of its net worth, called *strength* (wealth). When a rule matches an environmental message or an action of another rule, it becomes active. It then invests some of its wealth (proportional to its strength) called *payment*. Every matched rule has to pay, so every matched rules gets its strength decreased by the amount of its payment. The payment of rule *i* at iteration *t*, $P_i(t)$, is calculated as:

$$P_i(t) = C_{payment} \cdot S_i(t) \quad (1)$$

where,

$C_{payment}$ is a rule payment coefficient that determines what proportion of a rule's strength will be lost on a single step.

$S_i(t)$ is the strength of rule *i* at step *t*.

In the market-based trading algorithm, the activated rules get their strengths decreased by the amount of *payment* and the divided amounts paid to the contributing rules get added to their strengths. Figure 2 shows an example of how the trading algorithm works. Rule C distributes its payment to Rules A and B, which are the parties responsible for its activation. Usually the payment is distributed equally among the contributing rules. In a subsequent time step, activated Rule D makes its payment to the previously active Rule C. Finally, a reward comes into the system and is paid to the last active rule, Rule D, if it produces a good (rewarded) final output.

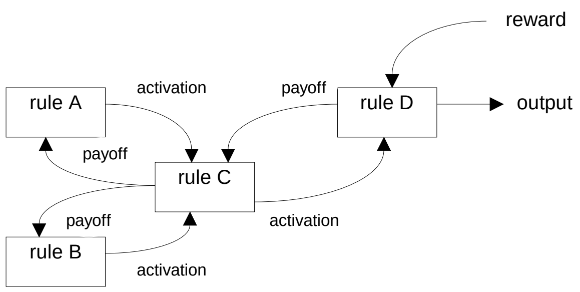

Figure 2: The market-based trading algorithm in action

As in classifier systems, there are also two types of taxes in the MBRL system: *life tax* and *activation tax*. The life tax is applied to every rule on every iteration, just as in the classifier system. Unlike the bid tax in the classifier system, though, the activation tax in the MBRL system is designed to penalize those rules that are active frequently but are not be able to activate other rules.

The rewarding (paying) of rules for being activated is designed to reinforce chains of good rules. Chains of rules will be necessary if the function that is computed by the learning system is complex (that is, when a good result cannot be achieved by a single master rule). So if the answers given by the chain of rules are usually good, the last rule (the one producing the output) will get rewarded, and its strength will increase. The rules that help it to become activated will then get payments, so their strengths will increase as well. This way the good rewards flow down the chain. Bad chains of rules will get no reward (they may even get a negative reward, which can be called a punishment). Rules that activate the rule that produces the bad output will also get very small payments, so over time their strength will decrease, and the strength of all the rules in the chain will, too. Through such a trading system, the rule base can be refined by the genetic algorithm which means that rules with high strength will survive, while rules with low strength will die off. The system may not be able to respond properly to those inputs for which no chain of rules can be established. In order for the system to be able to respond to such input, new rules must be created, and that is where the genetic algorithm comes in.

2.3 The Genetic Algorithm

In Holland's classifier system, rules are represented by bit strings consisting of zeroes, ones, and wildcards. However, in the MBRL system is an integer or floating-point number rather than a single bit. Each chromosome is thus a list of real values. The basic genetic algorithm actions involved in the MBRL system are:

- Selection
- Crossover
- Mutation
- Scanning
- Crowding

The placement of these actions in the overall genetic algorithm is shown in Figure 3.

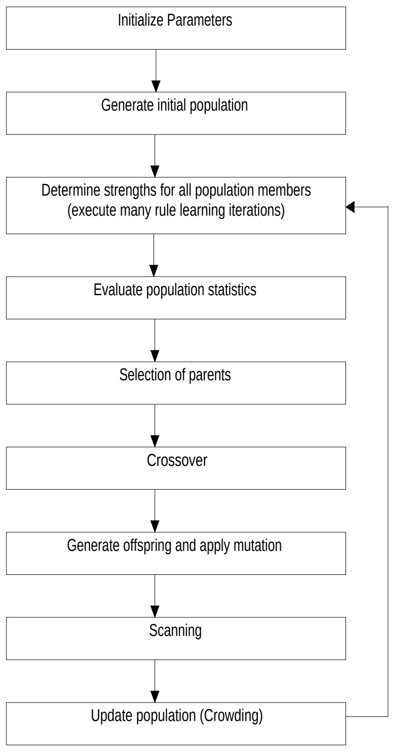
Initialize Parameters
Generate initial population
Determine strengths for all population members
(execute many rule learning iterations)
Evaluate population statistics
Selection of parents
Crossover
Generate offspring and apply mutation
Scanning
Update population (Crowding)

Figure3: The genetic algorithm in an MBRL system

In Figure 3 there is a box that reads “Determine strengths for all population members”; however in an MBRL system this determination does not occur in a single iteration. Instead, an MBRL system determines the strengths and thus the ranking among the population members over multiple iterations with the environment, during which strength changes occur by means of the market-based trading algorithm. Only after multiple interactions with the environment will the rule strengths represent a measure of how well the rule performs in the environment.

After the system parameters have been set, the initial population has been generated, and the strengths of all population members have been determined, the genetic algorithm actions are performed.

The GA in the MBRL system works as follows. The system is cycled many times, with the trading algorithm determining the fitness of the rules. When the strengths of the rules have stabilized, the genetic algorithm is invoked. Individual rules will be selected, crossed over and/or mutated, and scanning and crowding are employed to arrive at a new population. The new population is then used by the system. The system should not forget completely what it has learnt between two steps of the genetic algorithm. Therefore the probability of crossover and mutation must be chosen to be relatively low, so only a small part of the rule set should be supplanted every generation. The set of rules will then change only slowly under the action of the genetic algorithm.

By adding the genetic algorithm as a third layer on top of the basic rule and message system and market-based trading algorithm, the population of rules evolves over time, continually exploring new regions of the space of possibilities. That causes the system not only to learn from experience but also able to create new rules.

3 Market-based Rule Evolution and Refinement Based on Extracted Rules from Feed-forward Neural Networks

In the past few years, there have been several efforts made to find effective algorithms to extract rules from trained feed-forward neural networks. The NeuroLinear approach (Setiono& Liu 1997) is one such effort. It offers some advantages compared to other methods since it works well for problem domains with continuous input attributes and thus does not require discretization of the input data. When extracting rules from a feed-forward neural network, there are two desirable features of the derived rule set:

- Accuracy of classification
- Simplicity (comprehensibility) of the rule set

Unfortunately the complexity of the extracted rule set is often relatively high, due to the complexity of the neural network. In this section, we will see how the MBRL system works on two-level NeuroLinear-generated rules in order to achieve a higher accuracy performance and a smaller rule size.

3.1 NeuroLinear Rule Encoding

A distinguishing feature of the NeuroLinear method is the fact that it generates two levels of rules: the set of rules which describe the relationship between discretized hidden activation values and network inputs are referred to here as *rule base 1*, and the set of rules which describe the network outputs in terms of the discretized activation values of hidden nodes are here called, *rule base 2*. The format of rules in *rule base 1* is as follows:

***R:* If** $a_1I_1 + a_2I_2 + ... + a_nI_n \in (b_1, b_2)$

then the discrete intermediate output **=** $\boldsymbol{c}$

where n is the number of inputs, I_1, I_2, ...I_n are input variables, a_1, a_2, ...a_n are coefficients associated with each of the input variables, (b_1, b_2) is a constraint boundary in which b_1 is a lower boundary and b_2 is a higher boundary, and c is the discrete intermediate output value associated with rule R. The encoding scheme for a rule is as follows: list the coefficients, constraint boundaries and the discrete intermediate output value in order (from left to right). In other words, each individual rule is a list which consists of n coefficients, two boundary values, and one discrete output. This is illustrated by the following example:

Suppose, a rule in *rule base 1* is

```
If -3.54 Altitude - 4.36 Rainfall - 1.37 Temperature + 7.49 Distance ∈ (-2.94, 0.09)
then the discrete intermediate output = 5
```

Then, a list is created as shown in Figure 4.

Coefficient	Boundary	Discrete Intermediate Output
-3.54 -4.36 -1.37 7.49	**-2.94 0.09**	**5**

Figure 4: Illustration of converting a NeuroLinear rule from *rule base 1* into

The format of rules in *rule base 2* is as shown below:

```
If  the discrete intermediate output1 = c1
and  the discrete intermediate output2 = c2
        ...
and the discrete intermediate outputk = ck
then the final output = Y
```

where k is the number of discrete values in the antecedent part of a rule, c_1, c_2, and c_k are the discrete intermediate outputs, and Y is the output.

Encoding a rule from *rule base 2* for the MBRL system is similar: the discrete values and the output are listed in order. For example a rule from *rule base 2* is

```
If the discrete intermediate output1 = 1
  and the discrete intermediate output2 = 4
  and the discrete intermediate output3 = 7
```

```
then the final output = 0
```

Then, a corresponding list is shown in Figure 5.

Discrete Intermediate Outputs	Final Output
1 4 7	**0**

Figure 5: Illustration of converting a NeuroLinear rule from *rule base 2* into a list that serves as a rule representation in the market-based rule learning system

3.2 Evolutionary Learning of NeuroLinear Rules

Neurolinear rules calculate an output indirectly from the input. There are two levels of inference: an output is generated by evaluating an environmental message across two levels of rule bases. So a MBRL system based on NeuroLinear rules is not a single-layer system. As already indicated for multiple-layer learning systems, the rules and the message list are the two major components to form the computational backbone. Information enters from the environment after it has been encoded in the form of a *message*. This environmental message is placed on a *message list*, where the message may then activate the rules in the *rule base 1*. When activated, a rule posts its output action to the message list. These messages may then invoke rules in the *rule base 2*. Whether an active rule actually receives a reward is determined by whether its action ultimately produces the correct output.

In the MBRL system based on the NeuroLinear rules, there are two levels of matching in accordance with the two levels of rules. First, all rules in *rule base 1* are matched against the environmental message and a match list (message list) is constructed, which contains the matched rules. Second, all rules in *rule base 2* are compared to the matched rules from *rule base 1* and a second match list is generated, which contains the active rules from *rule base 2*. With the matching completed, we are ready to distribute payment among the rules by using the market-based trading algorithm.

According to our general description of the market-based trading algorithm in the previous section, the matched rules make their payments to the previously active rules that sent the messages which matched the currently active rules (and thus invoked them). For matched rules from *rule base 1*, the strength of active rules is decreased by a value (proportional to its strength) which can be treated as a payment to the outside world. For every active rule from *rule base 2*, the strength is also decreased by a certain amount which is paid to the active rules from *rule base 1* that helped in activating it. The payment is distributed equally among the contributing rules from *rule base 1*. Thus, every active rule from *rule base 1* gets its strength increased if it is able to help to activate rules from *rule base 2*. Active rules from *rule base 2* that are able to produce correct output are rewarded by the environment. In order to discourage nonproductive rules, two different types of tax are collected from rules: a *life tax* and an *activation tax*. A *life tax* is collected from all rules in both *rule base 1* and *rule base 2* in every cycle of system learning. An *activation tax* is only collected from each active rule in both *rule base 1* and *rule base 2*. The market-based trading algorithm distributes and collects payments and taxes among rules to help assure that good rules achieve high strength and bad rules achieve relatively low strength. Thereafter strength is used as a fitness measure to facilitate a genetic search for new, possibly better rules.

Since there is a representation difference between *rule base 1* and *rule base 2*, the genetic algorithm procedure operates on the two rule sets. While the genetic algorithm is operating on *rule base 1*, all the rules in *rule base 2* remain unmodified. After the GA procedure operates on *rule base 1* for a pre-specified number (for example, 50) epochs, the GA procedure switches to operate on *rule base 2*, while *rule base 1* is held fixed.

The learning procedure consists of the following steps:

1. Input an environmental message to the system.
2. Compare the environmental message to the condition parts of all rules in *rule base 1* and record all matches.
3. Compare the action parts of matched rules in step 2 to the condition parts of all rules in *rule base 2* and record all matches.
4. Process the recorded matches in step 3 through the output interface. If the active rules are able to produce a correct final action, the active rules are rewarded. If the active rules do not lead to a correct final action, then no reward is given.
5. Apply the market-based trading algorithm to reallocate the strengths of all the individual rules in *rule base 1* and *rule base 2*.
6. If a predetermined number of training cycles *N* has been achieved, then go to step 7. Otherwise, go to step 1. The number of training cycles *N* equals the number of training examples in the training data set.
7. Apply the genetic algorithm by using the strengths of each of the individual rules as a fitness measure to discover new rules, while replacing other, lower-strength rules. If both *rule base 1* and *rule base 2* are operated on by the GA procedure, the GA procedure runs initially on *rule base 1* for a predetermined number of generations. Then, the GA operates on *rule base 2* for a pre-specified number of epochs. After each generation, the contents of a rule base are replaced with a new generation produced by the GA.
8. Apply the inference engine to evaluate the performance of the rules. If the inference performance is satisfactory or the pre-specified maximum number of generations of the GA procedure has been reached, end the process. Otherwise, go to step 1. Since the MBRL system allows parallel activation of rules, it is possible that more than one active rule in *rule base 2* can apply their actions to the outside world at the same time. Whether an active rule is actually allowed to

perform a final action is determined by its strength. The stronger a rule, the greater the chance that it can cause a final action when it is activated.

4 Experiment

The Pima Indians Diabetes Data (UCI 1998) is selected for experiments. It consists of 768 samples taken from patients who may show signs of diabetes. Each sample is described using 8 continuous-valued attributes. The class index represents either a positive or negative test for diabetes. For the experiments, 768 samples - 268 samples "tested positive for diabetes" and the remaining 500 samples "tested negative for diabetes" - were used.

Table 1 provides a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values for the NeuroLinear-generated rules before and after using the MBRL system learning. It can be seen that a significant improvement in accuracy was demonstrated on both the training set and the test set. In terms of number of rules, there was a significant reduction on the second-level rules after the MBRL learning. Although an increase on the number of first-level rules was observed, it is not statistically significant.

Table 1: Using the Pima Indians Diabetes data set, a comparison of accuracy, number of rules, number of antecedents per rule, and their associated P-values, for the NeuroLinear-generated rules before and after using the market-based procedure (standard deviations are in parentheses)

		NeuroLinear	**Market-based evolution starting with the NeuroLinear-generated rules (GA operated on *rule base 1* and *rule base 2*)**	**P-value**
Acc. on the training set (%)		77.10(1.4)	79.27(1.4)	0.010
Acc. on the test set (%)		71.10(3.4)	75.29(2.9)	0.006
Number of rules	**I**	3.67(1.580)	6.25(2.060)	0.079
	II	7.33(3.840)	4.78(3.110)	0.000
Number of antecedents per rule	**I**	1.00(0.000)	1.00(0.000)	-
	II	1.56(0.527)	1.56(0.527)	-

Space limitations prevent us from describing a range of experiments that have been conducted, which have included relatively large spatial data sets associated with geographical and climatological data, and have yielded favourable results in comparison with other rule-based data mining techniques, such C4.5 (Quinlan 1993). These details can be found in (Zhou 2003).

5 Conclusion

The development of a novel market-based rule learning (MBRL) system and the investigation of its capability of evolving and refining rules have been key elements of this research. As a classifier system-inspired model, it introduces a novel element by importing existing rule sets generated by other rule extraction techniques into the system. This basic change not only makes the MBRL system begin with pre-established rule sets with a relatively limited complexity, rather than a random set, but also enhances the likelihood of being able to interpret the evolved rules. Moreover, the MBRL system produces various modifications in each of the layers of the structure. With the modifications introduced by the MBRL system, the problems existing in current classifier systems can be solved or lessened. In order to derive information for setting appropriate starting values for system parameters, a mathematical analysis of the MBRL system's steady-state behaviour has also been presented.

The MBRL system has been proposed as a post-processing tool to be used with feed-forward neural networks and the feed-forward neural network rule extraction technique, NeuroLinear, in order to improve the quality of extracted rules from feed-forward neural networks. Experimental results have shown that the MBRL system has a significant capability of simplifying NeuroLinear-generated rule sets as well as improving their predictive accuracy on various computing tasks.

6 References

Clearwater, S. (1996) *Market-based Control: A Paradigm for Distributed Resource Allocation*, World Scientific Publishing, Singapore.

Holland, J. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

Holland, J. & Reitman, J. (1978) Cognitive systems based on adaptive algorithms, In Waterman, D. & Hayess-Roth, F., editors, *Pattern-directed Inference Systems*. Academic Press, New York.

Quinlan, J. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Setiono, R.& Liu, H. (1997) NeuroLinear: from neural networks to oblique decision rules, *Neurocomputing*, Elsevier, Vol. 17, No.1, pp. 1-25.

UCI (1998) UCI Machine Learning Repository, University of California at Irvine Machine Learning Group. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Wilson, S. (1995) Classifier fitness based on accuracy, *Evolutionary Computation*, Vol. 3, No. 2, pp. 149-175.

Wilson, S. (2000) Mining oblique data with XCS, *IlliGAL Report No. 2000028*, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

Zhou, Q. (2003) *Adaptive Knowledge Discovery Techniques for Data Mining*, PhD Thesis, University of Otago, New Zealand.

Zhou, Q. & Purvis, M. (1999) Knowledge extraction using market-based rule evolution, *Proceedings of ICONIP/ANZIIS/ANNES'99 International Workshop*, Dunedin, New Zealand, pp. 203-206.