Dynamic Refinement of Feature Weights Using Quantitative Introspective Learning

Zhong Zhang and **Qiang Yang**
School of Computing Science
Simon Fraser University
Burnaby, B.C.
Canada V5A 1S6

Abstract

Recently more and more researchers have been supporting the view that learning is a goal-driven process. One of the key properties of a goal-driven learner is introspectiveness – the ability to notice the gaps in its knowledge and to reason about the information required to fill in those gaps. In this paper, we introduce a quantitative introspective learning paradigm into case-based reasoning (CBR). The result is an integrated problem-solving model which will learn introspectively feature weights in a case base in order to be responsive dynamically to its users. In contrast to the existing qualitative methods for introspective learning, our model has the advantage of being able to capture accurate learning information in the interactions with its users. A CBR system equipped with quantitative introspective learning ability can allow the feature weights to be captured automatically and to track its users' changing preferences continuously. In such a system, while the reasoning part is still case-based, the learning part is shouldered by a quantitative introspective learning model. Weight learning and evolution are accomplished in the background. The effectiveness of this integration will be demonstrated through a series of empirical experiments.

1 Introduction

Case-based reasoning (CBR) is a problem-solving strategy which uses stored previous cases to solve current problems [Kolodner, 1993]. It has enjoyed tremendous success for solving problems related to knowledge reuse [Leake and Ram, 1993]. Usually in a case base a case's index is a set of important descriptors of the case. It can be used to distinguish a case from others. In many implementations, these descriptors are represented as feature-value pairs and usually a feature is associated with an importance value called *feature weight* to indicate how important it is in the case retrieval process. When a new problem is presented, its index will be extracted and used to trigger a search in the case base. The cases with the most similar indices will be retrieved for further consideration [Kolodner, 1993].

The performance of a CBR system depends on how to use appropriate features to index cases, and how to obtain an accurate measurement of the similarity between cases in the case retrieval process. Therefore the feature weights play an important role in determining the success of CBR applications. How to choose and maintain an appropriate set of feature weights in a case base is a non-trivial problem in CBR research. In addition, the relative importance of the cases is changing with time, partly due to the uneven and changing distribution of the inherent problem space, also partly due to the changing interests of its users. How to evolve a case base continuously in an automated manner is also becoming an urgent task of the knowledge base industry.

One approach to tackling this problem is to use introspective learning, which has a representation of its own process in order to detect deviations that show when the learning is needed as well as what the learning needs [Leake *et al.*, 1995; Ram and Cox, 1993]. In the past, various introspective learning methods have been employed in feature weighting in CBR systems [Fox and Leake, 1995; Leake *et al.*, 1995; Wettschereck *et al.*, 1997; Bonzano *et al.*, 1997]. A main theme is to learn through *qualitative* introspective learning, whereby the feature weights are adjusted based on a rough estimate of the *direction* for a change: if the weights are too high, then adjust them so that they become lower, and vice versa. But how much has to be changed quantitatively is not sufficiently determined. In this work, we extend qualitative introspective learning to quantitative introspective learning within CBR. With the quantitative learning methods, we can adjust the weights not only in the right *direction*, but also in the right *amount*. We claim that such an extension provides a sound and promising continual weight introspective learning method in CBR.

This paper is organized as follows. In Section 2, we introduce some related work on the application of introspective learning to feature weighting. Section 3 presents a novel quantitative introspective learning model integrated into case-based reasoning. In Section 4 we demonstrate the experimental results for evaluating the perfor-

mance of our integrated model. And also there we cross-validate our work with others'. Section 5 concludes our discussion, where we will also explore our future work.

2 Qualitative Introspective Learning Methods

Leake et al. [Leake and Ram, 1993] summarize in a symposium report the goal-driven learning process from various aspects. They indicate that one of the three key properties of a goal-driven learner is its *introspectiveness*, an ability to notice the gaps in its knowledge and to reason about the information needed to fill in those gaps. They also pinpoint that introspective learning acquires problem-solving knowledge by monitoring its run-time performance, seeking chances in this process to learn by itself.

In [Fox and Leake, 1995], Fox et al. describe their experiences with introspective learning in CBR. The ROBBIE system described is an application of an introspective model to the task of refining indexes used to retrieve cases. Its goal is to improve reasoning process when encountering failures in its reasoning. The introspective learning component in the system monitors its reasoning process by comparing it with a declarative model which is used to describe the system's ideal reasoning process. Once a failure is found, the model is used to create an explanation of the failure in terms of other failed assertions, and to suggest a repair. The authors claim that even under knowledge-poor initial conditions, the introspective learning of new feature indexes improves the success rate of the system. But they still indicate that there exists a problem with the ordering of the presentation of training cases to the system due to the inherent shortcoming of their learning mechanism.

As a variation of a model that is introduced in [Munoz-Avilz and Huellen, 1996], Bonzano et al. [Bonzano *et al.*, 1997] also propose introspective learning as an approach to feature weighting in CBR, demonstrating their system which combines introspective learning with CBR. They first pose the problem with their experience in constructing a CBR system for Air Traffic Control. The problem encountered is that it is difficult to determine the important features and adjust their relative importance. The situation is further complicated by the fact that the features are highly context-sensitive; the predictiveness of a feature depends heavily on the current context. They use so-called **pulling** and **pushing** techniques to adjust the feature weights. Given a target T and two cases A and B, if it is judged that A is a correct solution to T but B is not, the learning method will **push** B away from T, and **pull** A closer to T. As to its weight updating policy, their introspective learning method uses a decaying learning process as shown in the following two formulae.

$$\text{increase: } W_i(t+1) = W_i(t) + \Delta_i \frac{F_c}{K_c} \quad (1)$$

$$\text{decrease: } W_i(t+1) = W_i(t) - \Delta i \frac{F_c}{K_c} \quad (2)$$

where K_c represents the number of times that a case has been *correctly* retrieved, F_c represents the number of times that a case has been *incorrectly* retrieved, and Δ_i determines the initial weight change. The ratio between F_c and K_c is used to reduce the influence of the weight update as the number of successful retrievals increases. We can observe that the timing of triggering the adjustment process is very important; when to trigger the adjustment of the weights using the above two formulae is a crucial issue yet to be further addressed in the work. This limitation makes it necessary to involve a human user in the learning process. In contrast, instead of relying on a domain-independent decaying factor, what we propose in this paper is a continual learning process in the *lifetime* of the case based reasoner. This extension releases the human manager of the decision to explicitly trigger a learning process.

The second limitation of the work by Bonzano et al. is that it is qualitative in nature. While the direction of change in feature weights is indicated in the above two formulae, the amount of change is only influenced by the frequency of successes and failures and the decaying factor. A quantitative change would be needed to reflect the amount of adjustment in proportion to the error.

The third limitation, reported by the authors, is that the learning method does not work well for pivotal cases, as the redundancy in a case base is essential in such a learning process. A pivotal case is the one that provides coverage not provided by the other cases in a case base [Smyth and Keane, 1995]. In contrast, the quantitative introspective learning paradigm that we will present in this paper will allow not only pairs of cases to be compared, but also any number of cases to participate in the learning process. This is achieved through a process in which a user can provide feedback at any time to all top-ranking cases, not just to a few selected. In Section 4, we will provide experimental comparisons between the quantitative and qualitative methods.

3 A Quantitative Introspective Learning Paradigm

3.1 Problem Statement

Using introspective learning, we wish to acquire feature weights in a case base in a changing and multi-user environment. In a changing environment, users and their preferences for what cases are the best for their problems also change with time. For example, in an electronic commerce application using CBR, cases may represent a configuration of a product (say a computer) model. The features then can represent various user specifications on the product, and the weights can indicate the level of interest of a user in a particular feature. Since a user's preferences may change with time, there is a need to acquire and track her/his preferences. Furthermore, in a multi-user environment, there is a strong need for catering to users with different needs. For example, an on-line computer vendor may have different sets of feature weights for students and teachers. It is desirable to adapt a CBR system with its users.

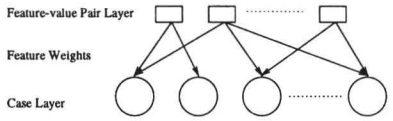


Figure 1: Two-layer Architecture of a Case Base

Our assumptions for the research are as follows. We assume that our desired quantitative introspective learning model is given as an input a set of features where each feature has some values. Some subset of the features and values may be relevant to a particular case at hand at any given time, but there is no prior knowledge on which ones are actually useful to the reasoner currently. Our model monitors its running process through the interactions with its users.

For a case base our learning task is of two folds:

1. *Weight acquisition* to acquire the feature weights after a user has used the system for a certain period of time;
2. *Continual tracking* to adapt the feature weights to a user's preference which may change with time, and to allow different users to have different preferences.

3.2 Strategy for Feature Weighting

The mechanism in our quantitative introspective learning process resembles that of a back-propagation neural network, which is a very popular learning paradigm in AI. For details on the mathematical foundation and applications of back-propagation neural networks, see [Zurada, 1992; Gupta and Ding, 1994].

More specifically, we use a two-layer architecture to model a case base. The front layer consists of a set of feature-value pairs. The back layer consists of a set of cases. A feature-value pair is associated with a case if it may exert influence on that case. Furthermore, there is a weight attached to the association. Although we use a two-layer feature weighting system in this paper, the architecture can be potentially extended to multiple layers which can include hidden layers. We address this situation in [Zhang and Yang, 1998].

Using two-layer architecture to model a case base is conceptually shown in Figure 1

For a case base of J cases, suppose that there are N features. For each feature F_i, there are m_i values, where $i = 1, 2, \cdots, N$. There is a total of $I = \sum_{i=1}^{N} m_i$ feature-value pairs. We label these feature-value pairs as FV_i, where $i = 1, 2, 3, \cdots, I$. We use C_j to represent each case, where $j = 1, 2, 3, \cdots, J$. There is a weight $V_{j,i}$ attached to the connection between case C_j and feature-value pair FV_i if there is an association between them.

Computing A Case's Score

A case's score is computed using the feature-value pairs selected by a user to represent an input query. For each case C_j, its score is computed using the following formula:

$$S_{C_j} = \frac{2}{1 + e^{-\sum_{i=1}^{I}(V_{j,i} * X_i)}} - 1 \qquad (3)$$

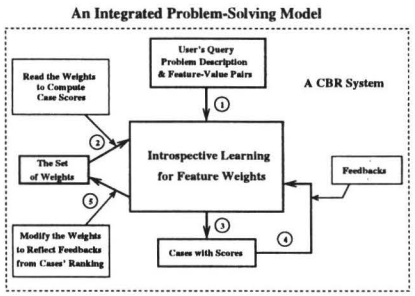


Figure 2: Quantitative Introspective Learning in A CBR System

where $j = 1, 2, 3, \cdots, J$, S_{C_j} is the score of case C_j, and X_i is 1 if there is a connection between case C_j and feature-value pair FV_i *and* FV_i is selected. Otherwise X_i is 0.

Learning Feature Weights

After a query is presented, all the previous cases have been scored according to Formula 3 with the promising ones at higher positions. If the user wants to feed back some information to the system as whether a case is desired or not, we then can employ the following formula to compute a new weight based on her/his feedback information.

$$V_{j,i}^{new} = V_{j,i}^{old} + \frac{1}{2} * \eta * (DS_{C_j} - S_{C_j}) * (1 - S_{C_j}^2) * X_i \qquad (4)$$

where $V_{j,i}^{new}$ is the new weight to be computed, $V_{j,i}^{old}$ is the old weight attached to the connection between case C_j and feature-value pair FV_i, DS_{C_j} is the desired score for C_j, S_{C_j} is the score computed in Formula 3, and η is the learning rate. X_i is 1 if there is a connection between case C_j and feature-value pair FV_i *and* FV_i is selected by the user. Otherwise it is zero.

We have to emphasize that, in the above formula, the term $(DS_{C_j} - S_{C_j}) * (1 - S_{C_j}^2)$ encodes the quantitative information, i.e., the actual gap between the desired and computed behavior. On the other hand, in Formulae 1 and 2 there is no such quantitative information encoded. Although they try to use the number of retrieval success and failure in their weight learning, we consider that such a representation is not sufficiently accurate when compared with ours.

3.3 User's Interaction Model

The introspective learning process in our integrated model is similar to that of a back-propagation neural network. An important difference between them is that our learning process is interactive rather than batching and automatic. In our learning process, there is no training data explicitly defined; the system is continuously being trained by its user throughout its lifetime. A user's responses to the system's behavior form an implicit source of training data.

After our introspective learning model is integrated into a CBR system, the problem-solving process is paradigmed in Figure 2. The CBR system receives a

user's current problem description and a set of selected feature-value pairs (label 1). It will then access the weights (label 2) to compute the case scores, and present the result to the user for her/his judgment (label 3). If the user feeds back some judgment to the system (label 4), the system will compute quantitatively the gap between the computed and the desired score, and if necessary, modify the weights accordingly (label 5).

4 Empirical Tests

The introspective feature-weight learning model is fully implemented and integrated in the framework of the CaseAdvisor™ system [Zhang and Yang, 1998], which is a CBR system implemented by the Case-Based Reasoning Group at Simon Fraser University. Right now the resultant CaseAdvisor™ system from such an integration is able to learn the unpredictable information hidden in an end user's behavior. A user's interactions with the system provide the guidance in determining quantitatively not only what the right direction is for updating weights but also how much the weight should be updated *quantitatively*.

In this section, we will demonstrate that our proposed learning model conforms to our expectations. In particular, we wish to confirm through the experiments that the model could learn a user's queries after sufficient interactions with its users and could scale up to case bases with realistic sizes. This is shown in the first experiment suite which is conducted on different case bases from the Repository of Machine Learning Databases and Domain Theories [1] at the University of California at Irvine (UCI). Furthermore, in the second experiment, we perform a comparison between the quantitative method and the qualitative method proposed by Bonzano et al. We will demonstrate that the quantitative method can achieve better learning accuracy and faster convergence rate through continual updating.

4.1 Experiment with Case Bases from UCI

We take the Dermatology Database and Car Database from the UCI Repository. We test our system on both databases and obtain the similar experimental results. For brevity, in the following we focus on the Dermatology Database. The experiments are conducted on a platform of SUN SparcStation 4 (SunOS 5.6) with 32 MB memory.

The Dermatology Database contains 366 instances (tuples) and 34 attributes. We divide this database into increasingly larger databases, which contain 50, 100, 150, 200, 250, and 300 instances, respectively, and test the performance of our learning model. In our experiment, we first adapt these databases into the case bases that our system can be applied by converting all rows into cases and all columns into features. The values for a feature are contained under each column. After conversion, these case bases contain 50, 100, 150, 200, 250, and 300 cases, respectively. For each case base, we generate a set of queries for testing, the size of which is half the number

[1] http://www.ics.uci.edu/mlearn/MLRepository.html

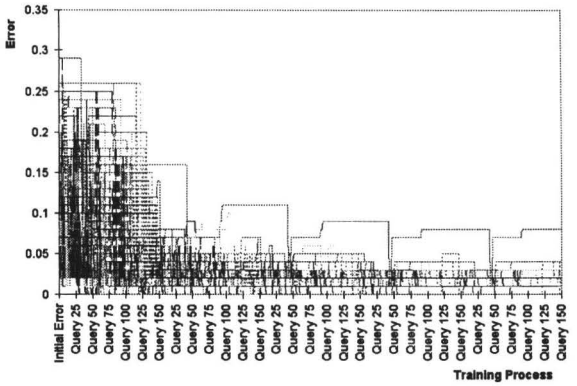


Figure 3: Error Convergence Chart for 150 Highest Cases in 150 Queries in A Case Base of 300 cases

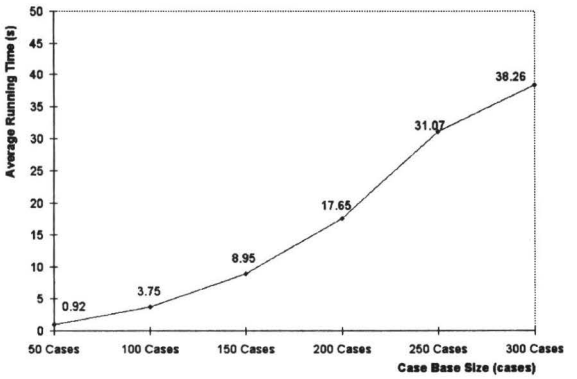


Figure 4: Average Running Time for Training Individual Cases

of cases. For instance, for the case base with 300 cases, the number of queries is 150. Note that in these tests, the score of a case is scaled to between 0.0 and 1.0.

The training process is composed of five rounds for all the case bases. Figure 3 shows the error convergence chart of 150 queries for the case base of 300 cases. The X-axis represents the training process while the Y-axis represents the error ranging from 0.0 to 1.0. It can be found that, almost all the cases, after five training rounds, have their errors falling into an acceptable range (in our experiment this range is set to 0.02). The error convergence for other five case bases also demonstrates the same trends. For brevity, we do not show them here.

We also measure the average CPU time required for the adjustments for individual cases in each of these six case bases. The result is shown in Figure 4, where the X-axis represents the six case bases with different sizes measured in cases, while the Y-axis represents the average running time for each case in CPU seconds. We can see that the increase of the running time is in proportion to the square of the number of cases in a case base. Therefore we can say that our algorithm is fast enough to be used in real-world practice, in which usu-

ally the case base sizes are not very huge. This confirms the scalability conjecture we make about our learning model.

4.2 Comparison with Bonzano et al.'s Approach

A closely related work on introspective feature-weight learning is proposed by Bonzano et al. [Bonzano *et al.*, 1997] (also discussed in Section 2). We implement their algorithm and examine the convergence and performance comparison between our two models. The case base involved is also converted from the Dermatology Database in UCI Repository. It contains 100 cases.

There are three types of case bases in Bonzano et al.'s model. The first is the training case base (It is composed of a user's queries in our learning model), the second one is the case base itself and the third one is a test case base (in our model, there is no explicit test case base; it is implicit in the usage of the system). For the experiment using their method, we set the training case base and the test case base to be the same. This comparison experiment is also conducted on a platform of SUN SparcStation 4 (SunOS 5.6) with 32 MB memory.

Figure 5 (a) shows the comparison on the errors between our two models. In the figure, the X-axis represents all the 50 test cases, while the Y-axis represents the errors of these cases along the training process. From the figure, we can easily see that among the 50 test cases, our model produces smaller errors for 43 cases.

In order to make a further comparison on the convergence trends, we plot the error chart for the first five training rounds in Figure 5 (b), where the X-axis represents the training process, while the Y-axis represents the average error for each case. The figure shows that at the very first five training rounds our model has already produced an optimal training result. Most of the trained cases in our method show the trends to approximate their desired scores.

We now analyze the learning and adjustment Formulae 1 and 2 used in Bonzano et al.'s model. These formulae give an estimation of what should be done when a retrieval success or failure is encountered. However, such an estimation is not precise enough. For example, if the desired case has a higher (similarity) score than expected, the case is over ranked and we have to reduce the weights associated with its feature-value pairs in order for it to be properly ranked, rather than increasing its weights. In the two formulae, there is no quantitative estimate associated with these information. In contrast, our adjustment strategy not only decides when to do the adjustments, but also takes into account at a more detailed level the quantitative gap between the current score and the target score, thus resulting in better learning quality.

However, we have to indicate that a limitation of our learning model, as compared to the qualitative model, is that it might take longer time to learn for each individual case. On average our model takes about four CPU seconds while Bonzano et al.'s uses approximately 1.8 CPU seconds to complete an individual learning task.

4.3 Discussion

Our learning model allows incremental changes to be made to a case base. As shown in the above experiments, for a case base of 100 cases, our model takes about four CPU seconds (running on a Sun SparcStation 4 with 32 MB memory) to train an individual case. Therefore for 50 queries, it takes about 20 CPU minutes to train the whole case base. This shows that it is practically possible to retrain the whole network after introducing a new case into the case base.

In our experiments, we also observe an interesting phenomenon among feature-value pairs. In the above case base of 300 cases, we find that not all the cases converge to their desired scores; in that experiment, five out of 150 cases in the 150 queries oscillate around their desired scores. We also find that no matter how long our training process undertakes, these five cases still cannot converge. We attribute this phenomenon to the *interactions* among feature-value pairs. Definitely, removing such interactions from a case base will help increase the learning quality. How to detect and remove those interactions pose an interesting research problem we wish to address in our future work.

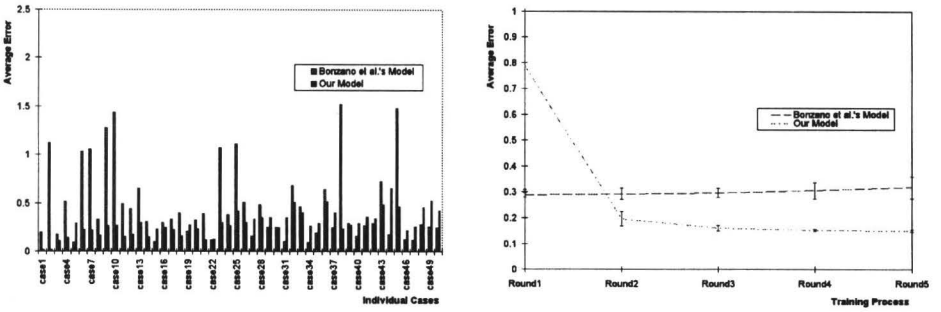
We have to emphasize again that the quantitative introspective learning in our model is a continuous process and can be triggered at any time if necessary. Our system will respond to its user at any time. Every time a user changes her/his behavior, the change will be captured, and reflected in the next work session.

5 Conclusion and Future Work

We have shown the empirical test results of our quantitative introspective feature-weight learning model. Based on the discussions throughout the paper, we can find that our proposed quantitative learning model fulfills our expectations. It captures the interactions between the system and its end user, and seeks chances to evolve itself. In the experiments, the model quickly approximates a user's behavior within a number of iterations.

Our work aims to introspectively learn feature weights in a dynamic context in the case retrieval process of CBR. The needs from practical application of CBR in a fielded diagnosis system motivate us to explore their dynamic nature. The research is also motivated by our desire that a CBR system be a responsive system; its behavior needs to simulate its end user's behavior, incorporating her/his preferences. Furthermore, a user's behavior is changing, requiring that a CBR system keep its pace with the changes. The integration of an introspective learning network into a CBR system makes these expectations possible.

We also note that our learning model has some limitations. Although in our experiments nearly all the cases converge to their desired scores, we actually encounter divergence several times due to the interactions among



(a) Individual Cases (b) Along Training Process

Figure 5: Comparisons between Bonzano et al.'s Model and Our Model (In (a), each case has two bars. The left bar corresponds to Bonzano et al.'s model while the right bar corresponds to ours)

different features. The effects of such interaction could be possibly reduced by introducing stronger bias factors into the system. We are also seeking other efficient and effective techniques to deal with the problem. In addition, one of the assumptions of our learning model is that the user of our system should be consistently one person in a certain period. If a different user comes to use the system, s/he might not satisfy and thus destroy the previous optimal case retrieval result, requiring the whole case base be retrained.

We will further explore, for our model, the more accurate relationship between the average running time and the size of a case base (including the number of feature-value pairs and cases). In the future, we hope to address these problems by introducing more effective learning and feedback control mechanisms and architectures into CBR.

References

[Bonzano *et al.*, 1997] A. Bonzano, P. Cunningham, and B. Smyth. Using introspective learning to improve retrieval in CAR: A case study in air traffic control. In *Proceedings of the Second International Conference on Case-Based Reasoning, ICCBR-97*, pages 291–302, Providence RI, USA, 1997.

[Fox and Leake, 1995] S. Fox and D.B. Leake. Learning to refine indexing by introspective reasoning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada, August 1995.

[Gupta and Ding, 1994] M.M. Gupta and H. Ding. Foundations of fuzzy neural computation. In Fred Aminzadeh and Mohaamad Jamshidi, editors, *Soft Computing, Fuzzy Logic, Neural Networks, and Distributed Artificial Intelligence*, pages 165–200. PTR Prentice Hall, Englewood Cliffs, New Jersey, USA, 1994.

[Kolodner, 1993] J.L. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Publishers, Inc., 1993.

[Leake and Ram, 1993] D.B. Leake and A. Ram. Goal-driven learning: Fundamental issues (a symposium report). *AI Magazine*, 14(4):67–72, 1993.

[Leake *et al.*, 1995] D.B. Leake, A. Kinley, and D. Wilson. Learning to improve case adaptation by introspective reasoning and CBR. In *Proceedings of the First International Conference on Case-Based Reasoning*, pages 229–240, Sesimbra, Portugal, 1995. Springer-Verlag.

[Munoz-Avilz and Huellen, 1996] H. Munoz-Avilz and J. Huellen. Feature weighting by explaining case-based reasoning planning episodes. In *Proceedings of Third European Workshop on Case-Based Reasoning (EWCBR-96)*, 1996.

[Ram and Cox, 1993] A. Ram and M. Cox. Introspective reasoning using meta-explanations for multistrategy learning. In R. Michalski and G. Tecuci, editors, *Machine Learning: A Multistategy Approach.* Morgan Kaufmann, San Mateo, USA, 1993.

[Smyth and Keane, 1995] B. Smyth and M.T. Keane. Remembering to forget. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 377–382, Montreal, Canada, August 1995.

[Wettschereck *et al.*, 1997] D. Wettschereck, D.W. Aha, and T. Mohri. A review and comparative evaluation of feature weighting methods for lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.

[Zhang and Yang, 1998] Z. Zhang and Q. Yang. Towards lifetime maintenance of case based indexes for continual case-based reasoning. In *Proceedings of the Eighth International Conference on Artificial Intelligence: Methodology, Systems, Applications*, Sozopol, Bulgaria, 1998.

[Zurada, 1992] J.M. Zurada. *Introduction to Artificial Neural Systems.* West Publishing Company, 1992.