Case-Based Learning Algorithms

David W. Aha[*]
The Turing Institute
36 North Hanover Street
Glasgow G1 2AD
Scotland

ABSTRACT

Case-based learning (CBL) algorithms are CBR systems that focus on the topic of learning. This paper notes why CBL algorithms are good choices for many supervised learning tasks, describes a framework for CBL algorithms, outlines a progression of CBL algorithms for tackling learning applications characterized by challenging problems (i.e., noisy cases, poor similarity functions, contextual importance of features), and discusses unsolved problems with the case-based learning approach.

Keywords: learning, noise, case retrieval, determining feature importance, determining feature importance in context, evaluation

1 Case-Based Learning

This paper concerns a subset of CBR algorithms called case-based learning (CBL) algorithms, which focus on learning issues but do not perform case adaptation, are limited to feature-value case representations, and do not necessarily employ smart indexing schemes for their case base.[1] Nonetheless, CBL systems are well-suited for supervised learning tasks, which involve processing a set of training cases and using them to predict values for subsequently presented cases' goal features.[2] This is in large part because they can exploit many types of domain-specific knowledge concerning the learning task (Aha, 1990). For example, two CBL algorithms known to the CBR community are Protos (Bareiss, 1989a; 1989b) and MBRtalk (Stanfill, 1987; Stanfill & Waltz, 1988), which have been sucessfully evaluated on clinical audiology and word pronunciation applications respectively. Protos encodes such domain-specific knowledge as typicality information and featural importances for the purposes of determining accurate similarity assessments. In contrast, MBRtalk is a relatively knowledge-lean algorithm that *computes* rather than *encodes* similarity assessments.

[*]To appear in *Proceedings of the 1991 DARPA Case-Based Reasoning Workshop*, distributed by Morgan Kaufmann Publishers, Inc.

[1]I refer to these algorithms as *instance-based learners* elsewhere, but call them CBL algorithms here to focus on their commonalities with case-based reasoners.

[2]Features are assumed to have either binary, numeric, or symbolic values.

Its domain-specific knowledge mainly concerns how to carefully compute similarity for cases with symbolic-valued features.

Many CBL algorithms have not yet been described in the CBL literature. These algorithms are important because they explore other points along the similarity assessment continuum between all-encoding and all-computing. More importantly, they have been successfully applied to a wide variety of exciting learning tasks. Industrial applications include predicting power load levels for the Niagra Mohawk Power Company (Jabbour, Riveros, Landsbergen, & Meyer, 1987) and appraising oil prospecting sites in the North Sea for the Oil Enterprise Company (Clark, 1989). Other applications include pole balancing (Connell & Utgoff, 1987), speech recognition (Bradshaw, 1987), robotic control tasks (Moore, 1990), molecular biology (Cost & Salzberg, 1990), cost-sensitive robotic decision-making (Tan & Schlimmer, 1990), and medical diagnosis (Salzberg, 1990; Aha, Kibler, & Albert, 1991). Other than these case studies, research investigations of CBL systems include mathematical evaluations on what types of categories they can learn (Aha, Kibler, & Albert, 1991), evaluations of their abilities as unsupervised learning algorithms (Stanfill & Waltz, 1988), methods for learning rule-like abstractions (Salzberg, 1990), development of learning apprentices (Bareiss, 1989a; Clark, 1989), and evaluations of cognitively plausible CBL algorithms (Aha & Goldstone, 1990; Aha, 1990). In summary, CBL algorithms have supported many applications while maintaining the interests of several researchers.

However, few CBL investigations have focused on their empirical limitations. What are their limitations? Have they been so tailored that they can only work on a single application? Do CBL algorithms exist that can be taught or learn sufficient domain-specific knowledge so that they can perform well in a large variety of learning tasks? Investigations like Stanfill's (1987) evaluation of MBRtalk's ability to handle noise and irrelevant features and Bareiss's (1989b) lesion study with Protos are a good start towards answering these questions. However, these studies were on a single domain; we cannot be sure that they will perform as robustly on other domains. Thus, the question remains as to what learning tasks they can solve. More precisely, practitioners would greatly benefit if they could be told what types of *domain characteristics* these algorithms can handle in *general*, such as noisy data and ability to effectively learn feature importances.

This problem leaves CBL advocates open to salvos from advocates of alternative learning approaches. For example, Breiman, Friedman, Olshen, and Stone (1984) examined a simple CBL algorithm and argued that CBL algorithms have several general deficiencies:

1. they are computationally expensive because they save and compute similarities to all training cases,
2. they are intolerant of noise,
3. they are intolerant of irrelevant features,
4. they are sensitive to the choice of the algorithm's similarity function,
5. there is no simple way they can process symbolic-valued feature values, and

6. they give little usable information regarding the structure of the data.

However, Breiman and his colleagues didn't investigate potential solutions to these problems. In fact, solutions may already exist for some. For example, MBRtalk (Stanfill, 1987) uses an algorithm for computing similarity over symbolic-valued features that has been successfully applied in several applications (Cost & Salzberg, 1990). Similarly, Salzberg (1990) developed a general method for deriving abstractions from cases and argued that it helps to show the structure of the data. The following sections describe a framework for CBL algorithms, outline general solutions to the first four of these problems, and summarize continuing progress.

2 A Framework for Case-Based Learning Algorithms

Case-based learning algorithms, as defined here, input a sequence of training cases and output a *concept description*, which can be used to generate predictions of goal feature values for subsequently presented cases. The primary component of the concept description is a case base, but almost all CBL algorithms maintain additional related information for the purpose of generating accurate predictions (e.g., settings for feature weights). Current CBL algorithms assume that cases are described using a feature-value representation, where features are either *predictor* or *goal* features. CBL algorithms are distinguished by their processing behavior; they focus on some parts of the CBR paradigm while deemphasizing others. To be explicit, all CBL algorithms have at least the following functions:

1. Pre-processor: This prepares the input for processing (e.g., normalizing the range of numeric-valued features to ensure that they are treated with equal importance by the similarity function, formatting the raw input into a set of cases, etc.).
2. Similarity: This function assesses the similarities of a given case with the previously stored cases in the concept description. Assessment may involve explicit encoding and/or dynamic computation; most practical CBL similarity functions find a compromise along the continuum between these extremes.
3. Prediction: This function inputs the similarity assessments and generates a prediction for the value of the given case's goal feature (i.e., a *classification* when it is symbolic-valued).
4. Memory Updating: This updates the stored case-base, such as by modifying or abstracting previously stored cases, forgetting cases presumed to be noisy, or updating a feature's relevance weight setting.

Missing from this framework are requirements for elaborate indexing schemes, case adaptation techniques, and methods for generating knowledge-intensive justifications of predictions (although CBL algorithms are not precluded from supporting these capabilities).

The simplest CBL algorithm is *CBL1.* Its pre-processor linearly normalizes all numeric feature values. CBL1 defines the similarity of cases C_1 and C_2 as

$$\text{Similarity}(C_1, C_2, P) = \frac{1}{\sqrt{\sum_{i \in P} \text{Feature\_dissimilarity}(C_{1_i}, C_{2_i})}} \tag{1}$$

where P is the set of predictor features and

$$\text{Feature\_dissimilarity}(C_{1_i}, C_{2_i}) \begin{cases} (C_{1_i} - C_{2_i})^2 & \text{if feature } i\text{'s values are numeric} \\ 0 & \text{if } C_{1_i} = C_{2_i} \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

This defines similarity to be the inverse of Euclidean distance for numeric features and uses a simple matching test for symbolic feature values. CBL1's prediction function is the k-nearest neighbor function, which has a long history in pattern recognition (e.g., Fix & Hodges, 1951; Cover & Hart, 1967).[3] This function predicts that the value for a given case's goal feature is the most frequent goal value among its k most similar stored cases in the concept description. Finally, CBL1's memory updating function simply stores all training cases in its concept description. Therefore, CBL1's work during training is trivial: it simply stores normalized cases. It's similarity and prediction functions aren't needed until it is requested to generate predictions for test cases (i.e., those whose goal feature values are missing or withheld).

CBL1 is highly similar to MBRtalk (Stanfill, 1987), which uses a 10-nearest neighbor prediction function and also stores all training cases. MBRtalk's pre-processor differs; it derives a set of cases (one per letter) for a given word. It's similarity function also differs in that it uses the *value-difference* metric rather than the simple metric used in CBL1. Nevertheless, both algorithms are specified by the CBL framework.

3 Capabilities of Case-Based Learners

This section describes a sequence of extensions to CBL1 that each address one or more of the critiques of CBL algorithms listed in Section 1.

3.1 CBL2: Reducing Storage Requirements

CBL1's learning behavior has been extensively evaluated against the performance of other machine learning algorithms (Aha, Kibler, & Albert, 1991; Aha, 1990). It performed relatively well, but computed a needlessly large number of similarity assessments during prediction attempts. Some method is needed to reduce either the time required to find the best k stored case matches or to reduce storage requirements.

[3] k's value is set to 1 for the experiments described in this paper. Aha (1990) describes performance details when $k > 1$ and when the goal feature is numeric-valued.

Several possible solutions exist. First, a massively parallel machine could be used to compute similarity assessments. Waltz (1990) reported several successful applications of storage-intensive CBL algorithms on such machines. Unfortunately, these machines remain expensive, and Waltz noted that alternative methods might still operate more quickly, essentially because they do not store all of the cases. A second alternative is to carefully index the case base. Unfortunately, simple data structures for indexing, such as k-d trees (Moore, 1990) or Voronoi diagrams (Seidel, 1987), work well only when few predictor features are used to describe cases. Otherwise, they cannot guarantee fast retrieval times (Sproull, in press). Smarter indexing methods offer more hope. For example, Protos (Bareiss, 1989a) maintains typicality strengths with stored cases and uses them to guide it to a good match. Difference links can then be traversed to improve the match when needed. This hill-climbing approach worked extremely well with a clinical audiology application when an expert was available to guide Protos away from non-optimal matches. Computational loads were also decreased by removing cases that the teacher agreed were sufficiently similar to previously stored cases. However, other methods for reducing computational loads are worth exploring because this level of expertise may not always be available.

In fact, what Protos showed was that highly similar cases are a form of redundant knowledge that could be eliminated safely. This information doesn't always require a teacher; automated CBL methods for forgetting cases without reducing predictive accuracy have been known for 30 years (Sebestyen, 1962; Hart, 1968). Aha, Kibler, and Albert's (1991) mathematical analysis of CBL1 showed that, by saving only cases that *discriminate* between different goal feature values, large numbers of cases can be safely forgotten. This is the basis of *CBL2*, which is identical to CBL1 except that it retains only incorrectly classified cases in its concept descriptions. The evidence described in the next section shows that the first critique of CBL algorithms on page 2 no longer holds.

3.2 CBL3: Tolerating Noisy Cases

Both algorithms were applied to a large number of database applications to determine how well CBL2 works in practice (Aha, Kibler, & Albert, 1991). As expected, its storage requirements were extremely low but its accuracy was always slightly less than CBL1's. In particular, CBL2 performed relatively poorly when training cases were noisy (i.e., were incorrectly recorded, whether because of faulty feature measurement devices or malicious alterations). This behavior can be explained with the help of Figure 1, which shows that CBL2 tends to save noisy cases. However, these noisy cases are distinguishable from other cases; they tend to more frequently misclassify subsequently presented cases. *CBL3* (Aha, Kibler, & Albert, 1991), an extension of CBL2, keeps track of the frequency with which stored cases, when chosen as one of the current case's most similar stored cases, matched the current cases's goal feature value. It then uses a statistical test of significance to ensure that only stored cases with significantly high frequencies (i.e., significantly better than chance) participate in predictions of goal feature values. CBL3 is otherwise identical to CBL2.

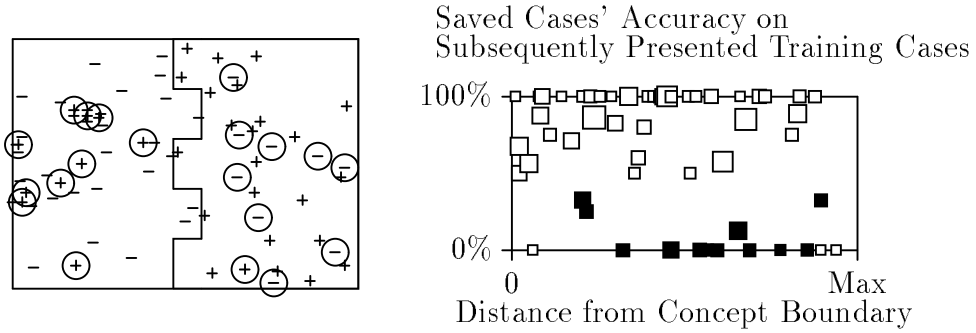

Figure 1: Left: An artificially constructed case space described by two numeric-valued predictor features. The central solid line is the boundary between cases with positive ("+") and negative ("-") goal feature values. This figure shows the cases saved by CBL2 when noise was maliciously added to each of 250 randomly-generated training cases with a probability of 10%. Many noisy cases (circled) were saved because they were misclassified by the stored cases. Right: A plot of each saved case's distance from the boundary line versus their accuracy in classifying subsequently presented cases. Noisy cases (black boxes) typically had poor classification accuracies.

CBL3 works relatively well on the problem shown earlier in Figure 1; it allowed none of the training set's noisy cases to be used in prediction decisions. Furthermore, CBL3 regularly performed well in this simple application. Figure 2 displays results averaged over 50 training/test trials for this application. This figure shows that CBL3's accuracy deteriorates more slowly than the other two algorithms as noise levels increase. Furthermore, its storage requirements (i.e., those stored cases allowed to take part in predictions for goal feature values) were lowest. This occurred because, as shown in the rightmost graph, CBL3 prevented most noisy training cases from participating with its predictions. However, this does not prove that CBL3 will outperform the other algorithms in more practical applications. Therefore, these algorithms were applied to a variety of databases that have often been used in the machine learning literature (Aha, 1990). These results are shown in Table 1. For comparison purposes, results were also obtained for C4, a decision tree algorithm that performed well on a large number of applications (Quinlan, 1987). The first two domains are challenging noisy applications used previously by Breiman et al. (1984) to study decision tree algorithms. CBL3 outperformed both CBL2 and C4 in these applications, where it recorded higher accuracies *and* lower storage requirements than CBL2. Its accuracy was also comparable to CBL1's. This pattern continued with the Cleveland and noisy Hungarian databases, which contain cardiological diagnoses. Thus, the second critique of CBL algorithms on page 2 no longer holds.

However, CBL3 performed relatively poorly in comparison to C4 in the final two applications, which involve large numbers of irrelevant features. Although CBL3 outperformed CBL1 and CBL2, it performs poorly when feature importance varies greatly among predictor features, such as when irrelevant features exist.

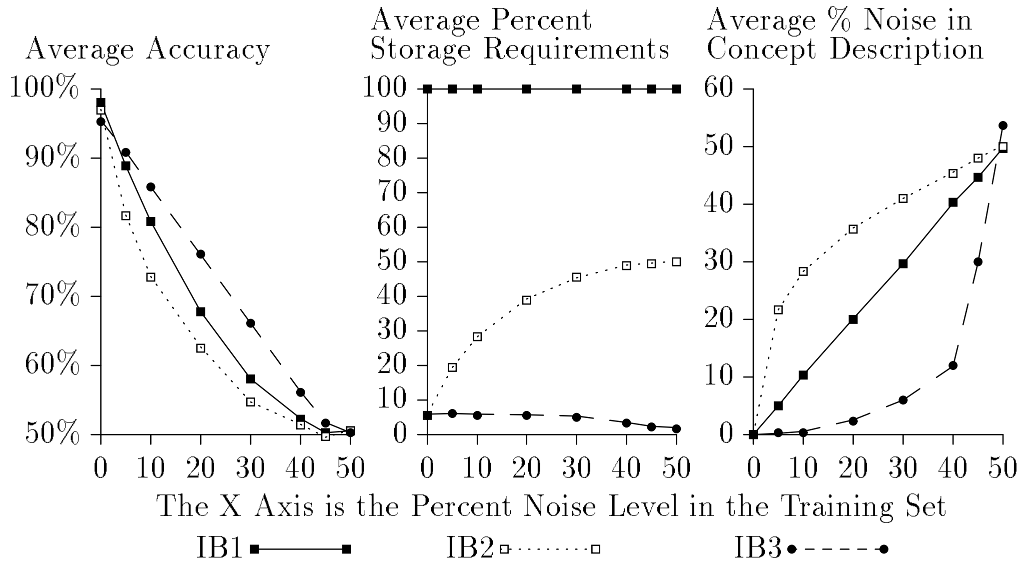

Figure 2: CBL3 filters noise better than either CBL2 or CBL1.

Table 1: Percent accuracy ± standard error and percent storage requirements. CBL3 recorded higher classification accuracies and lower storage requirements than CBL2. In most cases its classification accuracy also compares favorably with CBL1's and C4's.

Database	CBL1	CBL2	CBL3	C4
LED-7 display	71.6±0.4 100	63.0±0.9 41.6	72.5±0.4 20.1	68.9±0.5
Waveform-21	75.5±1.1 100	68.4±1.1 32.3	74.2±1.2 11.1	71.5±1.2
Cleveland	75.1±0.8 100	71.4±0.9 32.0	78.4±0.9 3.9	75.2±1.2
Hungarian	56.1±2.2 100	53.1±2.4 36.9	79.4±0.9 4.3	77.6±0.9
Voting	91.8±0.4 100	90.9±0.5 11.6	90.6±0.6 3.5	96.1±0.6
LED-24 display	47.9±0.6 100	43.7±0.8 60.1	46.6±0.7 25.3	66.9±2.1
Waveform-40	68.6±0.7 100	64.0±0.7 38.3	67.2±1.1 11.8	70.9±1.0

3.3 CBL4: Learning Feature Importance

This lead to the development of *CBL4* (Aha, 1989), which extends CBL3's memory updater; it learns the relative importances of features, represented as feature weight settings, for the purpose of computing accurate similarity assessments.[4] These weights are similar to the ones used by King, Klein, Whitaker, and Wiggins (1988) in their SURVER III program except that CBL4 *learns* rather than is *told* featural importances. Each feature's weight is adjusted after each prediction attempt during the training process by comparing the current training case with its most similar stored cases. CBL4 initially assigns equal weight to each feature, increases settings for features whose values are similar when correct predictions are made (or quite different

[4] CBL4 also differs in that it learns a separate set of feature weight settings for each goal feature. This gives it the freedom to assign different settings for a predictor feature depending on which goal feature's value is being predicted.

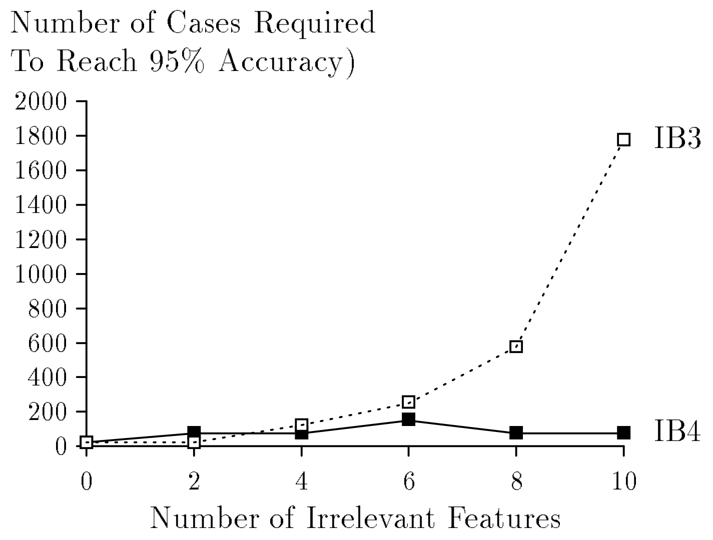

Figure 3: CBL4 requires less training than CBL3 to achieve a high prediction accuracy for an artificially constructed application as the number of irrelevant features increase. The application involves a single relevant binary feature and a varying number of irrelevant binary features. These curves are averages from 25 training/test trials.

Table 2: Average % accuracy ± standard error and % storage requirements (lower lines).

Database	CBL1	CBL2	CBL3	CBL4	C4
LED-24	47.9±0.6 100	43.7±0.8 60.1	46.6±0.7 25.3	66.1±0.6 25.4	66.9±2.1
Waveform-40	68.6±0.7 100	64.0±0.7 38.3	67.2±1.1 11.8	72.1±1.2 12.6	70.9±1.0

when incorrect predictions are made), and otherwise decreases feature settings. The amount of adjustment is determined by the difference between the feature's value for the two case's whose similarity is being assessed.

CBL4 can tolerate irrelevant features better than CBL3, which effectively assigns the same (static) weight setting to each feature. For example, Figure 3 shows how CBL3 requires an exponentially greater number of training cases to combat a linear increase in the number of irrelevant features used to describe cases for a simple application. However, this does not ensure that CBL4 will outperform CBL3 in practical applications. Table 2 gives a better indication; it shows that CBL4 outperforms CBL3 and comparably to C4 on the two applications of interest. (Its results for the other applications were similar to CBL3's, although it tended to have slower learning rates because it must search for good weight settings.) In summary, CBL4 tolerates irrelevant features by learning their relative importances in similarity assessments. This suggests that the third critique of CBL algorithms on page 2 no longer holds.

3.4 Current Research: Learning Feature Importance in Context

To some extent, the evidence described in the last section also shows CBL algorithms do not have to be sensitive to the choice of the similarity function. This is because CBL4 learns a separate set of feature weight settings for each goal feature. Since it is using these weights in its similarity function, it is in effect learning a separate similarity function for each goal feature.

However, CBL4 will not perform particularly well when feature importance is context-sensitive in the sense expressed by Ashley (1989), who noted that experts recognize that featural importance differs depending on the combinations of the other feature-value pairs present describing the case. For example, consider the problem of predicting whether a politician will endorse some proposed legislation on abortion rights. As with most real-world categorization tasks, some features should be given more attention than others to make this prediction. In this situation, the "past voting record" feature would be expected to be important, but this depends on the case's other features' values. For example, if the feature corresponding to the pressure of pro-choice political action groups has a high value, then "past voting record" might have low importance because the legislator might vote according to the pressure group's interests. However, this feature should be assigned higher importance if the legislator's "seek re-election" feature value is "false", which diminishes the influence of political action groups. Context sensitive case weights are required to derive appropriate feature importances in applications where feature importance is context-dependent.

The context-sensitive extension of CBL4 to tackle this problem has not yet been implemented. However, Aha and Goldstone (1990) developed a simple context-sensitive CBL algorithm named *GCM-ISW* for the purpose of testing its psychologically plausibility. It is similar to CBL1 in that it stores all cases, but it also employs an extension of CBL4's weighting scheme: it employs a separate set of feature weights for each $\langle$case,goal feature$\rangle$ pair rather than only one set of weights per goal feature. These additional weights allow GCM-ISW to encode different weights for a feature depending on both the case and on the feature whose value is to be predicted. Thus, GCM-ISW can learn localized, domain-specific contexts.

As in CBL4, these weights are used in GCM-ISW's similarity function. However, localized weighting schemes must be constrainted because, when unimportant features are ignored, dissimilar cases may seem highly similar. This problem is displayed in Figure 4, which shows an example where this might cause misclassifications. The vertical feature is the only important feature for predicting classifications for cases similar to x. However, if x's similarity is computed with newly presented case y, then its context-sensitive feature weights would misleadingly suggest that they are highly similar. Therefore, GCM-ISW learns both context-sensitive and context-independent (i.e., averaged) feature weight settings, which are dynamically combined when similarity assessments are required. The basic idea is that the context-sensitive settings are relied on when the two cases being compared have highly similar feature values. Otherwise, the context-independent settings are used.

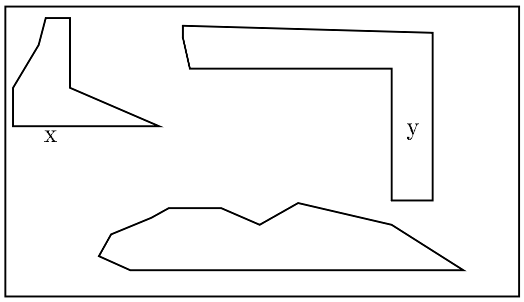

Figure 4: This shows a case space defined by two numeric dimensions. Three disjuncts of a single concept are shown. If the horizontal feature is ignored at case x's location, then x it might seem too similar to seemingly dissimilar cases such as y, whose vertical feature's value is approximately equal to x's. This could cause x to misclassify y.

GCM-ISW was evaluated in a lesion study; the other three algorithms used no weighting scheme, used only one set of feature weights, and used one set of feature weights per goal feature respectively. In simple applications where context-dependent information was unimportant, GCM-ISW's averaged learning rate (i.e., its accuracy on a separate set of test cases versus the number of processed training cases) was as good or better than the other algorithm's rates. However, GCM-ISW's learning rate was *significantly faster* than their learning rates in an application where context-dependent information *was* important (Aha & Goldstone, 1990).

Although GCM-ISW has not been evaluated in practical applications, it has been evaluated against these other algorithms for its ability to fit subject data from psychological experiments (Aha, 1990). Forty subjects completed two experiments that were designed to encourage them to assign different importances to a feature depending on its context (i.e., the feature's other feature values). The results showed that GCM-ISW's correlations to the subject data were *significantly* better than the other algorithms' correlations. Our next step is to compare GCM-ISW versus other algorithms in several database applications to determine whether the context-sensitive learning algorithm will improve learning performance.

4 Unsolved Issues

There are several open issues concerning CBL algorithms. This section highlights three of the more important ones.

Although it appears that relatively robust and general CBL algorithms can be developed, several important issues remain unsolved. For example, the CBLn algorithms described in this paper attempt to learn domain specific information such as which cases should be saved, which cases are noisy, and which features are important in a given context. However, some domain-specific information cannot be learned as easily as these simple types of information, especially when the given features do not properly match the set of features required to support accurate predictions. Automated methods for constructing new features, without the assistance of teachers, should prove useful for extending the capabilities of automated CBL algorithms.

Branting (1989) noted several limitations of the feature-value representation for cases. CBL algorithms cannot yet automate learning in knowledge-rich applications that require more elaborate case representations. Applications involving higher-order feature relationships, such as legal reasoning, are not amenable to current CBL algorithms.

Several CBL algorithms have explored methods for enhancing simple similarity functions with domain-specific information, essentially encoding some aspects of similarity assessments. For example, Optimist (Clark, 1989) uses a set of domain-specific rules to encode geological information to help assess the similarity between two prospecting sites. These rules correspond to particular contexts; if the two cases fit that context, then the rules affect their overall similarity assessment (either negatively or positively). The magnitude of this affect is pre-determined, but the way in which different rules combine their affects is not. Thus, part of the assessment is encoded and part of it is computed dynamically. The similarity assessment continuum between all-encoding and all-computing has been investigated by several other CBL algorithms that combine these two extremes, but methods are not yet available for determining which compromise is best given a particular application's characteristics. This is an interesting topic for future research.

5 Summary

Case-based learning algorithms have been applied to a large range of learning tasks with considerable success. However, most previous investigations on CBL algorithms were of the case study variety, which shows that an algorithm can work for one application but doesn't ensure that it will work for others. This paper showed that some simple and *generally* applicable CBL algorithms can be defined that reduce storage requirements, tolerate noise, and tolerate irrelevant features. Future work will be required to determine the limitations of general CBL algorithms and to more carefully outline how they can encode domain-specific knowledge.

Acknowledgements

Thanks to the Turing Institute for providing excellent computing resources and the SERC for supporting the author as a post-doctoral fellow on research grant GR/G 20424.

References

Aha, D. W. (1989). Incremental, instance-based learning of independent and graded concept descriptions. In *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 387–391). Ithaca, NY: Morgan Kaufmann.

Aha, D. W. (1990). *A framework for instance-based learning algorithms: Mathematical, empirical, and psychological evaluations* (Technical Report 90-42). Irvine, CA: University of California, Department of Information and Computer Science.

Aha, D. W., & Goldstone, R. L. (1990). Learning attribute relevance in context in instance-based learning algorithms. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 141–148). Cambridge, MA: Lawrence Erlbaum.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning, 6*, 37–66.

Ashley, K. D. (1989). Assessing similarities among cases: A position paper. In *Proceedings of a Workshop on Case-Based Reasoning* (pp. 72–76). Pensacola Beach, FL: Morgan Kaufmann.

Bareiss, R. (1989a). *Exemplar-based knowledge acquisition.* San Diego, CA: Academic Press.

Bareiss, R. (1989b). The experimental evaluation of a case-based learning apprentice. In *Proceedings of a Case-Based Reasoning Workshop* (pp. 162–167). Pensacola Beach, FL: Morgan Kaufmann.

Bradshaw, G. (1987). Learning about speech sounds: The NEXUS project. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 1–11). Irvine, CA: Morgan Kaufmann.

Branting, L. K. (1989). Integrating generalizations with exemplar-based reasoning. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 139–146). Ann Arbor, MI: Lawrence Erlbaum.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Belmont, CA: Wadsworth International Group.

Clark, P. E. (1989). *Exemplar-based reasoning in geological prospect appraisal* (Technical Report 89-034). Glasgow, Scotland: University of Strathclyde, Turing Institute.

Connell, M. E., & Utgoff, P. E. (1987). Learning to control a dynamic physical system. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 456–460). Seattle, WA: Morgan Kaufmann.

Cost, S., & Salzberg, S. (1990). *A weighted nearest neighbor algorithm for learning with symbolic features* (Technical Report JHU-90/11). Baltimore, MD: The Johns Hopkins University, Department of Computer Science.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory, 13*, 21–27.

Fix, E., & Hodges, J. L., Jr. (1951). *Discriminatory analysis, nonparametric discrimination, consistency properties* (Technical Report 4). Randolph Field, TX: United States Air Force, School of Aviation Medicine.

Hart, P. E. (1968). The condensed nearest neighbor rule. *Institute of Electrical and Electronics Engineers Transactions on Information Theory, 14*, 515–516.

Jabbour, K., Riveros, J. F. V., Landsbergen, D., & Meyer W. (1987). ALFA: Automated load forecasting assistant. In *Proceedings of the 1987 IEEE Power Engineering Society Summer Meeting.* San Francisco, CA.

King, J. A., Klein, G. A., Whitaker, L., & Wiggins, S. (1988). SURVER III: An application of case-based reasoning. In *Proceedings of the Fourth Annual Applications of Artificial Intelligence.*

Moore, A. W. (1990). Acquisition of dynamic control knowledge for a robotic manipulator. In *Proceedings of the Seventh International Conference on Machine Learning* (pp. 244–252). Austin, TX: Morgan Kaufmann.

Porter, B. W. (1989). Similarity assessment: Computation vs. representation. In *Proceedings of a Workshop on Case-Based Reasoning* (pp. 82-84). Pensacola Beach, FL: Morgan Kaufmann.

Quinlan, J. R. (1987). Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 304–307). Milan, Italy: Morgan Kaufmann.

Salzberg, S. L. (1990). *Learning with nested generalized exemplars.* Boston, MA: Kluwer.

Sebestyen, G. S. (1962). *Decision-making processes in pattern recognition.* New York, NY: Macmillan.

Seidel, R. (1987). On the number of faces in higher-dimensional Voronoi diagrams. In *Proceedings of the Third Annual Symposium on Computational Geometry* (pp. 181–185). Waterloo, Ontario: Association for Computing Machinery.

Sproull, R. F. (in press). Refinements to nearest-neighbor searching in k-d trees. *Algorithmica.*

Stanfill, C. (1987). Memory-based reasoning applied to English pronunciation. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 577–581). Seattle, WA: Morgan Kaufmann.

Stanfill, C. (1988). Learning to read: A memory-based model. In *Proceedings of a Case-Based Reasoning Workshop* (pp. 402–413). Clearwater Beach, FL: Morgan Kaufmann.

Tan, M., & Schlimmer, J. C. (1990). Two case studies in cost-sensitive concept acquisition. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 854–860). Boston, MA: American Association for Artificial Intelligence Press.

Waltz, D. (1990). Massively parallel AI. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 1117-1122). Boston, MA: American Association for Artificial Intelligence Press.