Multi-Pass Instance Based Learning

by

Steve G. Romaniuk

Department of Information Systems and Computer Science

National University of Singapore

10 Kent Ridge Crescent

Singapore 0511

e-mail: stever@iscs.nus.sg

Keywords: Instance-based Learning, Complexity Analysis, Empirical Evaluation, Neighborhood Spheres/Boxes, Knowledge Acquisition.

Abstract

This paper introduces a new modified approach to the instance based learning theory. Instance based learning is augmented by neighborhood spheres and multi-pass training to improve both on generalization capabilities and storage requirements. Two models for creating neighborhood spheres are investigated and put in perspective with the IBL instance based learner. The IBL system considered here is based on the proximity algorithm, the growth (additive) algorithm and a noise resistant modification of the growth (additive) algorithm. The herein described experiments will address the similarity of the MPIL and the IBL algorithms, but also point out significant differences in the approach of reducing storage requirements and increasing generalization. A time complexity analysis of the proposed multi-pass instance based learning approach is provided. Several domains are used in this study, which include a real world domain in CMOS wafer fault diagnosis to allow for a comparison of these two approaches. Finally, the task of knowledge extraction in form of rules is addressed.

1 Introduction

A newly emerging category of symbolic learning systems that has been proposed in recent years, is based upon an exemplar representation [1]. In the exemplar-based model, each concept is represented as a set of exemplars, where an exemplar can be viewed as an abstraction of a concept, or it may even be an individual instance of a concept. The following is a short excerpt of the properties normally attributed to the exemplar model [10].

- There is no known set of necessary and sufficient features for representing a concept.
- Representations are explicitly disjunctive, that is, any of the exemplars within a given set can be treated as a valid description of the underlying concept.
- The properties of a concept are a function or blend of its exemplar's properties.

Three approaches to constructing exemplar-based models are identified in [10].

1. **Proximity Model:** This model is based on the well known nearest-neighbor model. Every training instance is considered a valid exemplar of some class and is therefore stored.
2. **Best Example Model:** Here only typical training instances are stored. It is based on a prototypical model as found in many psychological studies in the area of classification and concept formation.
3. **Selected Example Model:** The model is similar to the previous model but also more general in that it does not assume the presence of prototypes within concepts.

Aha and Kibler [10] identified three possible algorithms for exemplar selection. Two of these algorithms have their roots in the Selected Example model. The three algorithms in question are:

1. **Proximity Algorithm:** It is the same as the proximity model, since all training instances are retained as exemplars.
2. **Growth (Additive) Algorithm:** It incrementally stores training instances that are not correctly classified.
3. **Shrink (Subtractive) Algorithm:** All training instances are used for concept representation. Each instance is presented for classification. If an instance has been correctly classified by the remaining instances, it is removed.

In our experiments we will be studying the performance of the Proximity Algorithm (IB1), the Growth Algorithm (IB2) and a noise resistant extension of the latter algorithm (IB3) [1].

2 The Learning Algorithms

2.1 MPIL, Multi-Pass Instance Learner

The instance based learner discussed in this section differs from the traditional model - as outlined earlier -, in that it allows multiple passes over the complete set of instances as opposed to a single pass as performed by the IBL algorithm. Furthermore, associated with every instance is a neighborhood (spanned by the instance), which can be either a sphere or a K-dimensional box. The purpose of the neighborhood is to define a degree of similarity between instances defining the same concept. Storage reduction can be achieved by first, appropriately defining the neighborhood for every instance, and secondly by removing instances that fall within another instance's neighborhood (assuming only instances describing the same concept). Instances associated with different concepts are used to define valid neighborhoods. We consider a neighborhood valid, if the instances enclosed within it describe the same concept as the instance that spans the neighborhood. Next, we discuss two approaches for forming such neighborhood.

2.1.1 The Neighborhood Instance Sphere

The simplest approach to building a neighborhood for an instance is directly based on the nearest neighbor algorithm. We consider every instance I_j of K features (the features of an instance describe a concept) to represent points in $\mathcal{R}^K$. By associating a radius R with every instance we can define a neighborhood sphere around it. The instance (point in K-space) is at the center of the sphere. The radius of the sphere is determined by the instance's nearest neighbor instance I_k, such that, the concept described by I_k and the one by I_j are different and the distance between the two is minimal. Throughout the text we will refer to distance between two points as our similarity measure between two instances. The similarity measure used here is the well known euclidian distance. The above described approach will also guarantee that all training instances, unless contradictory, be correctly classified after training. In this sense the MPIL algorithm is very similar to the nearest neighbor algorithm. In the following empirical study section it will be evident that this guarantee is extremely beneficial in guaranteeing high performance accuracy. The complete multi-pass algorithm based on instance spheres is described in Figure 1.

S_{Train} : Set of all training instances.
$O(I_j)$: Output concept for jth instance.
$R(I_j)$: Radius of jth instance.
$N(I_j)$: Number of instances within instance I_j's sphere.
$F_K(I_j)$: Kth input feature of instance I_j.
$\delta(I_j, I_k)$: Euclidian distance measure between instances I_j and I_k.
ϵ : Some arbitrarily small value > 0 (fixed at 0.001).
δ : Closeness measure for input features (fixed at 0.05).

$CALCULATE\_RADII$:
(1) For all Instances $I_j \epsilon S_{Train}$ do
  (1.1) $ND = min\{d \| d = \delta(I_j, I_k), \forall I_k \epsilon S_{Train} : O(I_j) \neq O(I_k)\}$
  (1.2) $R(I_j) = ND - \epsilon$
  (1.3) $Marked(I_j) = FALSE$

$IN\_SPHERE(\rightarrow I_k)$:
(1) $N(I_k) = 0$
(2) For all Instances $I_j \epsilon S_{Train}$ do
  (2.1) If $\neg Marked(I_j) \wedge O(I_j) = O(I_k) \wedge \delta(I_j, I_k) \leq R(I_k)$ then
    (2.1.1) $N(I_k) = N(I_k) + 1$

$MPIL\_1$:
(1) CALCULATE_RADII
(2) $BI = \max\{c \| c = IN\_SPHERE(I_j), \forall I_j \epsilon S_{Train} : \neg Marked(I_j)\}$
(3) For all Instances $I_j \epsilon S_{Train}$ do
  (3.1) If $\neg Marked(I_j) \wedge O(I_j) = O(I_{BI}) \wedge j \neq BI \wedge \delta(I_j, I_{BI}) \leq R(I_{BI})$
    (3.1.1) $S_{Train} = S_{Train} - \{I_j\}$
(4) $Marked(I_{BI}) = TRUE$

Figure 1. High Level Description of MPIL-1 Algorithm

The time complexity of this algorithm is summarized in the following Theorem.

Theorem 1 *Let $card(S_{Train}) = N$ be the cardinality of the train set and $S_{Train} = S^+_{Train} \cup S^-_{Train}$ and $S^+_{Train} \cap S^-_{Train} = \emptyset$. Furthermore, let us define $card(S^+_{Train}) = p$ and $card(S^-_{Train}) = N - p$. Finally, let K be the number of features (dimensionality of domain) describing every instance. The worst case, best case and average case time complexity of MPIL is then,*

$$\begin{array}{l} \mathcal{O}_{Worst}(N, K) = K(\frac{N}{2})^2 \\ \mathcal{O}_{Best}(N, K) = K(N-1) \\ \mathcal{O}_{Avg}(N, K) = \frac{K}{6}(N^2 + \frac{3N}{2} - 1). \end{array} \tag{1}$$

Proof Theorem 1 *The most time consuming phase of the MPIL algorithm is the calculation of the instance spheres. Calculating the radii for each of the N stored instances requires that all instances in S^+_{Train} are compared with those in S^-_{Train}. Comparing a single pair of instances requires K (number of features) operations. In general,*

$$\mathcal{O}(N, K) = p(N-p)K \tag{2}$$

To consider the worst case requires setting the cardinality of both S^+_{Train} and S^-_{Train} equal. That is, $p = \frac{N}{2}$. From this it follows,

$$\begin{array}{ll} \mathcal{O}_{Worst}(N, K) & = K\frac{N}{2}(N - \frac{N}{2}) \\ & = K(\frac{N^2}{2} - \frac{N^2}{4}) \\ & = \frac{KN^2}{4} \\ & = K(\frac{N}{2})^2 \end{array} \tag{3}$$

The best case results if $p = 1 \vee p = N - 1$.

$$\mathcal{O}_{Best}(N, K) = (N-1)K \tag{4}$$

Finally, to derive the average case we form the average over all sizes of p. Note, that the maximum value is $p = \frac{N}{2}$.

$$\begin{array}{ll} \mathcal{O}_{Avg}(N, K) & = \frac{K\sum_{p=1}^{\frac{N}{2}}(pN - p^2)}{\frac{N}{2}} \\ & = \frac{K(N\sum_{p=1}^{\frac{N}{2}} p - \sum_{p=1}^{\frac{N}{2}} p^2)}{\frac{N}{2}} \end{array} \tag{5}$$

We can state the results for each of the sums as,

$$\sum_{p=1}^{\frac{N}{2}} p = \frac{N^2}{8} + \frac{N}{4} \tag{6}$$

$$\sum_{p=1}^{\frac{N}{2}} p^2 = \frac{N^3}{24} + \frac{3N^2}{24} + \frac{2N}{24} \tag{7}$$

We can now state,

$$\begin{aligned} \mathcal{O}_{Avg}(N,K) &= \frac{K(\frac{N^3}{12} + \frac{3N^2}{24} - \frac{N}{12})}{\frac{N}{2}} \\ &= \frac{K}{6}(N^2 + \frac{3N}{2} - 1) \end{aligned} \tag{8}$$

q.e.d.

2.1.2 The K-Dimensional Neighborhood Box

A logical extension of the K-dimensional neighborhood sphere, is to associate a radius with every feature of an instance. The overhead would involve storing K radii besides the instance itself. The potential gain would be twofold: First, a more dynamic neighborhood could be constructed in the form of an K-dimensional box structure as opposed to the uniformly expanding sphere. This would allow creating fewer and more precise overlaps of neighborhoods to approximate concepts. Secondly, and even more importantly, it would provide for a means to decide if features of an instance may be redundant or not. It would support knowledge extraction in the form of rules, an important feature when applying multi-pass instance based learning to the task of knowledge acquisition in an expert system environment, requiring both consultation and explanation facilities.

The major change to the original MPIL-1 algorithm to accommodate for multiple radii of an instance is in the *CALCULATE_RADII* algorithm. In our version of MPIL-2 initially all radii of a given instance are set to infinity. The motivation for this is to later place constraints on an instance's neighborhood by dynamically adjusting its radii depending only on its nearest neighbors. The dynamic adjustment of the radii is based on a greedy approach, in which as few radii as possible are modified. The rational behind this algorithm is to increase the number of radii with length infinity. After the radii have been determined for every instance, features with radii infinity can be deemed redundant. No constraint has been placed on them, therefore there existence or non-existence does not affect the classification process on instances of S_{Train}.

A feature with infinite radius can not exclude any other instance of the input space from the neighborhood, and may therefore be ignored. Determining if a radius of a feature needs to be constrained, requires examining if any other instance belonging to a different concept has the same feature value (same distance from origin). The algorithm is greedy in the sense that at every iteration the feature selected is the one for which the majority of neighbor instances (belonging to a different concept) can be excluded from the instance's neighborhood. In other words, the more neighbor instances have the same distance from the origin along one of the K dimensions, the higher the precedence of chosing to constraint the radius associated with that dimension. The iterations are continued until either all radii of the current instance have been constrained, or all neighbors have been excluded from the instances neighborhood. A high level description is given in Figure 2.

$InBox(I_l)$: set of instances $I_j \in S_{Train}$ for which $O(I_l) = O(I_j)$ and I_j within neighborhood of I_l.
$UnTouchable(I_j)$: if TRUE instance I_j cannot be modified.
$InstanceUsed(I_j)$: If FALSE instance I_j belongs to neighborhood of some other instance.
$Covered(I_j)$: If TRUE instance I_j belongs to same output class as instance for which the radii are currently calculated.
$InputCnt(K)$: number of instances not yet covered which have an absolute input feature value F_K difference $> \delta$.
$InputUsed(K)$: input feature F_K that cannot be modified, $R_K < \infty$.

$MPIL\_2:$
(1) CALCULATE_RADII
(2) Reset UnTouchable to FALSE and InstanceUsed to TRUE
(2) do
  (2.1) $T_{Max} = max(\{t \| \forall I_l \epsilon S_{Train} \ \ \neg UnTouchable(I_l) \wedge InstanceUsed(I_l) \wedge$
  $t = card(InBox(I_l))\})$
  (2.2) $\forall I_l \in S_{Train} \wedge I_l \in InBox(I_{T_{Max}}) \ \ InstanceUsed(I_l) = FALSE$
  (2.3) $UnTouchable(I_{T_{Max}}) = TRUE$
  (2.4) If $T_{Max} = 0$ then
    (2.4.1) Exit do-loop

$CALCULATE\_RADII:$
(1) For all Instance $I_j \epsilon S_{Train}$ do
  (1.1) $\forall K \ \ R_K(I_j) = \infty$
  (1.2) For all Instances $I_l \epsilon S_{Train}$ do
    (1.2.1) Covered(I_l)=[$O(I_l) = O(I_j)$]
  (1.3) While $\neg(\forall I_l \in S_{Train} \ \ Covered(I_l))$ do
    (1.3.1) Reset InputCnt to 0 and InputUsed to FALSE
    (1.3.2) INPUT_IMPORTANCE(I_j)
    (1.3.3) RETURN_INPUT(I_j)
  (1.4) End-While

$INPUT\_IMPORTANCE(\rightarrow I_j):$
(1) For all input features K do
  (1.1) If $\neg InputUsed(K)$ then
    (1.1.1) For all Instances $I_l \in S_{Train}$ do
      (1.1.1.1) If $\neg Covered(I_l) \wedge \|F_K(I_j) - F_K(I_l)\| > \delta$ then
        (1.1.1.1.1) InputCnt(K) = InputCnt(K) + 1

$RETURN\_INPUT(\rightarrow I_j):$
(1) $C_{Max} = -1$
(2) For all input features K do
  (2.1) If $\neg InputUsed(K) \wedge InputCnt(K) > C_{Max}$ then
    (2.1.1) $C_{Max} = InputCnt(K)$, $I_{Max} = K$, $MV = \infty$
    (2.1.2) For all Instances $I_l \in S_{Train}$ do
      (2.1.2.1) If $\neg Covered(I_l)$ then
        (2.1.2.1.1) $MV = max(min(MV, \|F_K(I_j) - F_K(I_l)\|), \epsilon)$
(3) For all Instances $I_l \in S_{Train}$ do
  (3.1) If $\neg Covered(I_l) \wedge \|F_{I_{Max}}(I_j) - F_{I_{Max}}(I_l)\| > \delta$ then
    (3.1.1) Covered(I_l) = TRUE
(4) $R_{I_{Max}}(I_j) = MV - \epsilon$
(5) InputUsed(I_{Max}) = TRUE

Figure 2. High Level Description of MPIL-2 Algorithm

The time complexity of MPIL_2 is given in Theorem 2.

Theorem 2 *Again, let $card(S_{Train}) = N$ be the cardinality of the train set and $S_{Train} = S^{+}_{Train} \cup S^{-}_{Train}$. Furthermore, let us define $card(S^{+}_{Train}) = p$ and $card(S^{-}_{Train}) = N - p$. Finally, let K be the number of features (dimensionality of domain) describing every instance. The worst case and best case time complexity of MPIL-2 is then given as,*

$$\begin{array}{l} \mathcal{O}_{Worst}(N,K) = N^2K^2 - \frac{N}{6}(2K^3 + K + 3) \\ \mathcal{O}_{Best}(N,K) = (N^2 - N)(2K - 1) \end{array} \tag{9}$$

Proof Theorem 2 *The majority of work is conducted in routine CALCULATE_RADII. We can summarize the actions taken as follows:*

$$\begin{array}{ll} \mathcal{O}_{Worst}(N,K) & = N * \sum_{i=1}^{K}(N-i)[2(K-i)+1] \\ & = N[2\sum_{i=1}^{K}(N-i)(K-i) + \sum_{i=1}^{K}(N-i)] \end{array} \tag{10}$$

The first N is due to the loop in step (1) of CALCULATE_RADII. The expression $(N-i)(K-i)$ represents the number of not yet covered instances and the number of not yet used input features. The factor 2 is a result of the expression occurring both in routines INPUT_IMPORTANCE and RETURN_INPUT. The term $N-i$ is sufficient to describe the remaining work-load for RETURN_INPUT. For the worst case analysis it is assumed that during every iteration of step (1) of CALCULATE_RADII only one instance is covered. At the same time, during every iteration only one input has its status changed from not used to used (Note, the change is permanent once made). We continue by stating,

$$\begin{array}{ll} \sum_{i=1}^{K}(N-i) & = \frac{K}{2}(2N-K-1) \\ \sum_{i=1}^{K}(N-i)(K-i) & = \sum_{i=1}^{K} NK - \sum_{i=1}^{K} i(N+K) + \sum_{i=1}^{K} i^2 \\ & = NK^2 - \frac{(N+K)K(K+1)}{2} + \frac{K(K+1)(2K+1)}{6} \\ & = \frac{3NK^2 - 3NK - K^3 + K}{6} \end{array} \tag{11}$$

It now follows,

$$\begin{array}{ll} O_{Worst}(N,K) & = N[\frac{3NK^2 - 3NK - K^3 + K}{3} + \frac{K}{2}(2N-K-1)] \\ & = \frac{6N^2K^2 - 2NK^3 - NK - 3N}{6} \\ & = N^2K^2 - \frac{N}{6}(2K^3 + K + 3) \end{array} \tag{12}$$

The best case can be simply derived by observing that the initial loop (step (1) in CALCULATE_RADII) is only executed once. That is, all instances are covered during the first iteration. It follows,

$$\begin{aligned} O_{Best}(N,K) &= N(N-1)[2(K-1)+1] \\ &= (N^2-N)(2K-1) \end{aligned} \tag{13}$$

q.e.d.

2.2 IBL, Instance Based Learner

In IBL instance-based learning only specific instances are stored. As opposed to symbolic learning algorithms (decision tree-based and rule-based) all features of an instance enter the necessary calculations when comparing an unseen instance with a stored instance (usually some distance measure is used for determining their similarity). The IBL algorithms discussed in this section are derivable from the well known nearest neighbor pattern classifier [4].

- **IB1** is essentially equivalent to the nearest neighbor algorithm, except it normalizes the attributes' ranges, and has the capability to deal with missing values. It has its roots in the proximity model.
- **IB2** displays a high degree of similarity with the edited nearest neighbor algorithm [9], since it saves and uses only selected instances for making its predictions. To be specific, only misclassified instances are saved. By being selective in the instances it saves, its storage requirements can be significantly smaller than those of IB1 (provided instances are adequately separated in feature space). On the other hand, IB2 appears to be more sensitive to noise than IB1 [1]. IB2 is equivalent to the growth (additive) algorithm.
- **IB3** is a noise resistant extension of the IB2 algorithm that bases its decision whether a saved instance will perform well during classification on a *wait and see* evidence gathering method. With every saved instance a classification record is maintained. This record keeps track of an instance's prediction capability on subsequently presented training instances. It also provides for a future outlook on an instance's performance. A significance test is employed to determine those instances that may be infested by noise and may be poor classifiers (Setting of SA and SD).

It is important to point out that in the subsequently presented experiments the confidence test used a high acceptance value (SA=90%) and a rather low level for dropping instances (SD=75%). Acceptance of an instance is based on its classification accuracy and its class's observed frequency. An instance is removed from the concept description, if its accuracy is lower than SD, which also takes into account class frequency. If other values are chosen for SA and SD this will be indicated throughout the text.

3 The Experiments

3.1 Domains of Study

The choice of the iris domain has been based on its significant past usage [18]. The Harris CMOS semiconductor wafer failure domain was selected because it represents a rather large real world domain for which an extensive study using both symbolic (ID3 based) [12] and neural network (Quickprop) [6] learning algorithms exists. The Soybean and the Tank domain are both examples of binary encoded domains, with one exhibiting a small and the other a large input space. The Thyroid data has both continuous and discrete inputs. Table 1 describes each of the tested domains in terms of its structure. Note, that the column labeled CME represents *Classes Mutually Exclusive.* For the Iris, Soybean, Thyroid, Vote (6 different types used), Mushroom, and Heart domain 10 trial runs were conducted.

Table 1. Domain Features

Domain	#Classes	#Feature	Type	CME	#Ex˙	Train Size	Test Size
Harris	28	25	Nominal	No	1328	828	678
Iris	3	4	Cont.	Yes	150	100	50
Soybean	17	208	Binary	Yes	289	193	96
Thyroid	2-5	29	Cont./Binary	Yes	3772	2800	972
Tank	2	16x16	Binary	Yes	1157	157	1000
Vote	2	16	Binary	Yes	435	290	145
Shuttle	2	20	Binary	Yes	15	7	8
Mushroom	2	22	Nominal	Yes	8124	5415	2708
Heart-Cleveland	5	13	Cont.	Yes	303	202	101
Heart-Hungarian	5	13	Cont.	Yes	294	196	98

The Harris domain was obtained from Harris Semiconductor Corporation at Melbourne. It is the only knowledge base in which domain specific knowledge was used during training. The information made available to both systems consisted of a grouping of the 28 possible classes, such that, 6 groups were obtained. Output classes within each group are considered mutually exclusive. The total number of diagnosis for the train set is 928 and for the test set 678. An extensive study using decision tree, rule, and neural net-based algorithms can be found in [13, 17]. A study comparing machine generated rules with human experts can be found in [11] [13]. The Heart domain consists of 2 similar databases, one which was collected by the Cleveland Clinic Foundation (by Robert Detrano, M.D.) and the other from the Hungarian Institute of Cardiology, Budapest (by Andras Janosi, M.D.).

3.2 Description of Experiments

The first set of experiments attempts to compare the performance accuracy of MPIL with that of IBL. We investigate the IB3 algorithm for 4 different settings of the significance parameters: IB3 (SA=90%, SD=75%), IB3-2 (SA=80%, SD=85%), IB3-3 (SA=70%, SD=95%) and IB3-4 (SA=60%, SD=99%). Additionally, the performance accuracy was split into three parts. First, performance on the train set only (important since neither IB2 nor IB3 guarantee 100% correct classification). Second, performance on the test set to determine the generalization capabilities of the algorithms. Third, performance on the combined train and test set, to obtain an overall quality measure. Finally, the storage requirements of the different approaches are listed. IB1 is used as a base case, since it requires storage of all training instances. For the MPIL-2 algorithm the average number of attributes per generated rule is provided (Recall, MPIL-2 allows extraction of rules from stored instances).

3.3 Experiments

Tables 2-4 give insight into the performance of the two learning approaches. When comparing the IBL learners the most striking result is the poor performance of IB2 and IB3 at high significance parameters (Table 4). Performance is extremely poor for IB3 for the test data (a mere 35% for the Harris domain). In general, as the significance parameters are relaxed for IB3 it does improve, but only slightly. Classification is still significantly lower compared to IB1 and both versions of MPIL. MPIL-1's combined performance is somewhere in between that of IB1 and IB2. Its main gain over IB2 and especially IB3 is due to the fact that it achieves 100% correct diagnosis on the training set (which boosts its overall quality). MPIL-2 shows the most significant improvement in performance of all systems compared to IB1. That is, perfect results on the train set (Table 2) and a very high degree of accuracy on the test set (Table 3). In all cases except the 2 heart domains does MPIL-2 fair as well (or better) as IB1, but only requires a fraction of the storage. Table 5 displays the storage requirements for the various algorithms. As we can clearly see the required storage for MPIL-2 is extremely low, compared to the base case IB1.

Table 2. Performance Accuracy on train set only

	IB1	IB2	IB3	IB3-2	IB3-3	IB3-4	MPIL-1	MPIL-2
Harris	100%	69.2%	56.9%	62.6%	77.6%	75.4%	100%	100%
Iris	100%	97.4%	93.8%	96.6%	97.7%	97.0%	100%	100%
Soybean	100%	84.9%	87.5%	87.2%	88.9%	89.2%	100%	100%
Thyroid-ALLBP	100%	94.5%	86.2%	89.5%	91.2%	92.9%	100%	100%
Thyroid-ALLHYPER	100%	94.5%	86.2%	89.5%	91.2%	92.9%	100%	100%
Thyroid-ALLHYPO	100%	95.4%	81.6%	86.6%	87.8%	90.6%	100%	100%
Thyroid-ALLREP	100%	98.6%	90.3%	91.4%	91.7%	94.3%	100%	100%
Thyroid-DIS	100%	98.6%	93.5%	93.4%	90.9%	94.8%	100%	100%
Thyroid-SICK	100%	95.0%	73.5%	78.8%	88.7%	91.1%	100%	100%
Tank	100%	55.4%	79.6%	91.7%	93%	100%	100%	100%
Vote	100%	92.9%	92.5%	93.4%	94.2%	94.3%	100%	100%
Shuttle	100%	89%	0%	14%	37%	28%	100%	100%
Mushroom	100%	100%	99.8%	99.9%	99.9%	99.9%	100%	100%
Heart-Cleveland	100%	89.4%	48.7%	55%	58.2%	60.7%	100%	100%
Heart-Hungarian	100%	66.8%	71%	85.6%	92%	92.2%	100%	100%

Table 3. Performance Accuracy on test set only

	IB1	IB2	IB3	IB3-2	IB3-3	IB3-4	MPIL-1	MPIL-2
Harris	82.3%	65.0%	35.4%	61.7%	62.1%	63.7%	65.6%	95.6%
Iris	94.2%	94.4%	90.2%	95.2%	95.8%	95.0%	94.8%	93.8%
Soybean	93.7%	78.4%	77.4%	82.6%	83.5%	84.3%	88.2%	92.6%
Thyroid-ALLBP	95.3%	89.3%	84.1%	87.8%	87.3%	90.6%	95.1%	95.8%
Thyroid-ALLHYPER	97.5%	95.1%	94.1%	92.1%	90.8%	94.9%	96.6%	97.6%
Thyroid-ALLHYPO	94.0%	90.2%	82.2%	87.3%	85.5%	88.5%	93.3%	96.2%
Thyroid-ALLREP	96.3%	95.4%	90.3%	90.0%	89.8%	92.1%	95.0%	96.7%
Thyroid-DIS	98.1%	96.4%	93.5%	94.0%	89.3%	93.6%	97.9%	97.6%
Thyroid-SICK	92.5%	88.9%	75.2%	78.1%	88.1%	88.5%	90.6%	96.1%
Tank	99.5%	51.6%	77.9%	91.6%	94.5%	98.4%	97.3%	98.5%
Vote	92.1%	88.3%	92.9%	91.6%	89.0%	91.2%	89.4%	93.7%
Shuttle	58%	54%	0%	20%	46%	34%	56%	54%
Mushroom	100%	100%	99.7%	99.9%	100%	99.9%	99.5%	100%
Heart-Cleveland	51.1%	47.1%	45.9%	47.5%	46.8%	46.2%	41%	41.1%
Heart-Hungarian	85.7%	74%	76.3%	85.8%	86.8%	87.9%	65.4%	55.7%

Table 4. Performance Accuracy on train and test set combined

	IB1	IB2	IB3	IB3-2	IB3-3	IB3-4	MPIL-1	MPIL-2
Harris	92.4%	67.5%	48.1%	62.2%	71.3%	70.6%	86.0%	98.2%
Iris	98.1%	96.4%	92.6%	96.1%	97.1%	96.3%	98.3%	97.9%
Soybean	97.6%	82.4%	83.6%	85.4%	86.9%	87.3%	95.5%	97.2%
Thyroid-ALLBP	98.8%	93.1%	85.7%	89.1%	90.1%	92.3%	98.7%	98.9%
Thyroid-ALLHYPER	99.4%	95.9%	93.8%	92.6%	89.8%	95.3%	99.1%	99.4%
Thyroid-ALLHYPO	98.5%	94.1%	81.8%	86.8%	87.2%	90.1%	98.3%	99%
Thyroid-ALLREP	99.0%	97.8%	90.3%	91.0%	91.2%	93.7%	98.7%	99.2%
Thyroid-DIS	99.5%	98.0%	93.5%	93.6%	90.5%	94.5%	99.5%	99.4%
Thyroid-SICK	98.1%	93.4%	73.9%	78.6%	88.5%	90.4%	97.6%	99.0%
Tank	99.6%	52.1%	78.1%	91.6%	94.3%	98.6%	97.7%	98.7%
Vote	97.4%	91.4%	92.6%	92.8%	92.5%	93.3%	96.5%	97.9%
Shuttle	86%	78%	0%	16%	40%	30%	85.3%	84.7%
Mushroom	100%	100%	99.8%	99.9%	99.9%	99.9%	99.8%	100%
Heart-Cleveland	83.7%	75.3%	47.8%	52.5%	54.4%	55.9%	80.3%	80.4%
Heart-Hungarian	94.3%	69.7%	73.1%	85.7%	89.9%	90.5%	88.5%	85.2%

Table 5. Storage requirements

	IB1	IB2	IB3	IB3-2	IB3-3	IB3-4	MPIL-1	MPIL-2	Attr./Rule
Harris	100%	5.9%	15.4%	13.4%	13.3%	13.4%	10.1%	2.2%	1.8
Iris	100%	12.1%	31.3%	23.2%	18.5%	15.5%	9.7%	10%	1.5
Soybean	100%	24.2%	36.6%	40.9%	44.2%	45.0%	31.1%	26.2%	1.7
ALLBP	100%	10.2%	10.2%	9.1%	9.5%	9.2%	6.4%	7.5%	1.5
ALLHYPER	100%	5.2%	8.9%	9.9%	4.6%	5.0%	4.4%	4.2%	1.3
ALLHYPO	100%	10.2%	10.2%	9.1%	9.5%	9.2%	6.4%	4.3%	1.6
ALLREP	100%	5.4%	16.4%	13.0%	8.4%	5.9%	3.0%	3.1%	1.4
DIS	100%	12.6%	6.8%	9.2%	9.6%	9.3%	10.5%	6.7%	1.6
SICK	100%	1.3%	5.3%	24.8%	55.4%	66.2%	18.5%	10.2%	2.0
Tank	100%	1.3%	10.8%	22.9%	48.4%	57.3%	18.5%	10%	2.0
Vote	100%	11.9%	7.6%	6.9%	6.6%	8.5%	20.2%	7.9%	2.6
Shuttle	100%	45%	0%	3%	9%	11%	46%	46%	1.1
Mushroom	100%	0.5%	1.2%	1.0%	1.0%	0.9%	0.7%	0.2%	1.4
Heart-Clev.	100%	56.3%	5.4%	12.7%	15.2%	17.9%	67.2%	66.2%	1.4
Heart-Hung.	100%	1.3%	4.8%	16.3%	41.9%	45.1%	38.1%	44.7%	1.3

The reduction in number of attributes per rule is important, since it further reduces storage requirements. This is even more evident in case of the IBL algorithms were all K features describing an instance are stored. MPIL-2, not only obtains a high degree of generalization, but also reduces storage requirements drastically. IB3-4 performs best out of the set of IB3 learners, but still remains clearly below the generalization cap marked by IB1 and both versions of MPIL.

3.4 Overall Performance Results

In order to evaluate the performance of IBL and MPIL approach over all domains, a significance test of 5% is introduced to determine if the performance accuracy of one algorithm is significantly higher than that of another. For each of the domains, all algorithms are compared against one another. If the performance accuracy of algorithm A is greater or equal to 5% over algorithm B, we say A outperforms B for the particular domain under consideration. In Table 6 we present the number of comparisons won by algorithm A (horizontal list) over the remaining algorithms with respect to the studied domain (vertical list). The first entry represents the number of direct comparisons won based only on the performance on the train set, whereas the second entry shows the results for the test set only, and finally the third entry displays the results for the combined train and test sets. The overall quality of performance (for all three cases) for each of the algorithms is given under the *Total* heading of Table 6. The results clearly indicate that both versions of MPIL show very similar classification accuracies as IB1. The advantage of MPIL over IB1 is clearly that only a fraction of the necessary training instances and features deemed important need to be stored. IB2 and IB3 (and its variations) display far lower performance. MPIL-2 on the other hand shows the best performance in preserving IBLs classification accuracy and significantly reducing storage requirements.

Table 6. Direct Comparison of Algorithms

	IB1	IB2	IB3	IB3-2	IB3-3	IB3-4	MPIL-1	MPIL-2
Harris	5,6,6	2,1,2	0,0,0	1,1,1	3,1,2	3,1,2	5,1,5	5,7,7
Iris	1,0,1	0,0,0	0,0,0	0,1,0	0,1,0	0,0,0	1,0,1	1,0,1
Soybean	5,0,5	0,0,0	0,0,0	0,1,0	0,2,0	1,2,0	5,4,5	5,5,5
ALLBP	5,4,5	2,1,1	0,0,0	0,0,0	1,0,0	1,1,1	5,4,5	5,5,5
ALLHYPER	5,2,3	2,0,1	0,0,0	0,0,0	1,0,0	1,0,1	5,1,3	5,2,3
ALLHYPO	4,4,4	3,1,3	0,0,0	1,1,1	1,0,1	1,1,1	4,3,4	4,5,4
ALLREP	4,3,4	3,3,3	0,0,0	0,0,0	0,0,0	0,0,0	4,2,4	4,3,4
DIS	4,1,4	3,1,1	0,0,0	0,0,0	0,0,0	0,0,0	4,1,4	4,1,3
SICK	5,3,4	3,2,2	0,0,0	0,0,0	2,2,2	2,2,2	5,2,4	5,6,5
Tank	4,4,4	0,0,0	1,1,1	2,1,2	2,2,2	4,3,3	4,3,3	4,3,3
Vote	5,0,1	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	5,0,1	5,1,4
Shuttle	5,4,5	4,4,4	0,0,0	1,1,1	3,3,3	2,2,2	5,4,5	5,4,5
Mushroom	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0
Heart-Cleveland	5,3,5	4,2,4	0,0,0	1,2,0	1,2,1	2,2,1	5,0,5	5,0,5
Heart-Hungarian	5,4,5	0,2,0	1,2,1	2,4,2	3,4,2	3,4,2	5,1,2	5,0,2
Total	62,43,56	26,17,21	2,3,2	8,12,7	17,17,13	20,18,15	62,26,51	62,42,56

3.5 Comparison with other Learning Approaches

The purpose of this section is to put the empirical results presented earlier into perspective with other approaches to machine learning. The algorithms included in this comparison represent a diversified mix of different methods, such as decision tree based, prototypical, neural networks, and hybrid approaches. The following results have been taken directly from published articles. For explanations of individual learning algorithms the reader is referred to appropriate articles.

For the Iris domain SC-net [13, 8] (hybrid symbolic, connectionist learning system) achieves 100% (train set), 94.6% (test set), 98.2% (combined) accuracy, 22% storage requirement, and 2.58 attributes/rule [15]. In [13] ID3 [12] (decision tree) achieves 94.7%, GID3 [2] (a generalized version of ID3) obtains 92.7%, CN2 [3] (rule inducer) 98% on train set and 92.7% on test set. In [18] 9 algorithms are compared using the leaving one out error rate. The best performance is given by a linear discriminant method yielding 98% recognition, whereas the worst performance was delivered by a Bayes 2nd order method with 84%. Furthermore, in [5] using a split size of 1/2 for train and test set, C4 (symbolic decision tree based algorithm) achieves 92.9%, NTGrowth (instance-based) 92.3%, and PROTO-

TO [5] (prototype building algorithm) reaches 96% accuracy. In [14] Divide & Conquer Networks (dynamic network constructing neural net) attains 95% recognition. Finally, in [16] Cascade Correlation [7] (dynamic neural net) obtains 94.8% and EGP (Hybrid Genetic Neural Perceptron) reaches 94.7%.

For the Harris Semiconductor domain SC-net obtained 100% recognition on the train set, 97.1% on the test set and a combined accuracy of 98.8%. Storage requirements are 5.1% and the average number of attributes per rule are 3.58 [15]. ID3 reaches 96.3%, GID3 95.8% and CN2 92.3% recognition [11, 17].

For the Soybean domain SC-net obtains 100% recognition on the train set, 91.9% on the test set, and 96.4% on the combined set, with 27.6% storage requirement and 2.21 attributes per rule [15]. In [13] ID3 obtains 92.4%, GID3 93.6%, Quickprop [6] (neural network) 93.1%, and CN2 98.9% on train set and 89.8% on test set.

In [13] the following results are described for the tank domain: ID3 94.3%, GID3 92.6%, QuickProp 90.3%, SC-net 93%, and CN2 92.3% on train set and 85.8% on test set.

For the Voting domain Divide & Conquer Networks achieve 93.4% recognition, Cascade Correlation 91.3% [14], C4 95.3%, and PROTO-TO 90.4% [5].

Finally, in [16] Cascade Correlation reaches 97% recognition on the mushroom data and EGP 96.9%.

The earlier obtained results for MPIL compare favorably with a wide variety of different learning methods. Using the 5% significance rule MPIL-2's performance is equivalent to the best reported results in the literature for several of the studied domains.

3.6 Extracted Rules

We mentioned earlier that MPIL-2 allows knowledge extraction in form of rules. To demonstrate this, one complete set of rules for the Iris domain is presented in Figure 3.

RULE 1: IF ($0.000 \leq$ petal_length < 2.999) THEN iris_setosa ;

RULE 2: IF ($2.701 <$ petal_length < 4.499) THEN iris_versicolor ;

RULE 3: IF ($1.701 <$ petal_width < 2.899) THEN iris_viginica ;

RULE 4: IF ($4.801 <$ sepal_length < 7.199) &

($1.501 <$ petal_width < 1.699) THEN iris_versicolor ;

RULE 5: IF ($5.701 <$ sepal_length < 7.699) &

($4.901 <$ petal_length < 5.099) THEN iris_versicolor ;

RULE 6: IF ($4.001 <$ petal_length < 5.599) &

($1.201 <$ petal_width < 1.599) THEN iris_versicolor ;

RULE 7: IF ($3.101 <$ sepal_length < 6.699) &

($1.601 <$ petal_width < 1.799) THEN iris_viginica ;

RULE 8: IF ($5.101 <$ petal_length < 6.499) THEN iris_viginica ;

Figure 3. Extracted Iris Rules

The output class iris_setosa is linearly separable from the other two classes. The remaining two output classes are not. It is especially beneficial for learning systems to exhibit the capability of knowledge extraction in expert system environments. First of all, it allows the implementation of consultation and explanation facilities. Secondly, the extracted rules can be investigated by human experts who may edit or expand on these. Modified or newly created rules can be fed back into MPIL-2 as domain knowledge and new information can be learned. This refinement process of extraction, modification and learning can be repeated as many times as desired by the knowledge engineer.

4 Future Extensions

Several extensions can be added to the MPIL approach to enhance its applicability in the area of knowledge acquisition for expert systems. The majority of these extensions directly deal with the question of how to represent symbolic domain specific knowledge. The importance of domain knowledge can be crucial to the performance of a learner to derive an acceptable representation of knowledge and can support the knowledge refinement process. Refinement of knowledge itself is a vital part of any learning system, since it allows for direct interaction of the domain expert with the learning system. It allows the expert to compose initial assumptions of the underlying domain, which are then incorporated by the learning system. New information is added by means of learning and existing knowledge refined. Rule extraction allows the domain expert to further inspect the newly derived knowledge and augment it if necessary. The process may be repeated as many times as he or she sees fit. We now address some of the types of extensions that can be made to MPIL.

1. Associating individual distance measures with selected features and provide means for their aggregation.
2. Applying weighted distance measures to aggregation process to encode a priori knowledge of importance of features.
3. Allow rules to be encoded prior to learning to support knowledge refinement in conjunction with rule extraction.
4. Implement expert system standards such as consultation sessions.

To address the first point: Different features may give rise to different distance measures by their very nature. For example, ordinal attributes require a distance measure that reflects the actual distance of one value with respect to all other values. Values may have different distances from one another. These distances are encoded under the supervision of the domain expert to reflect his or her knowledge. Nominal attributes on the other hand do not require any ordering of their values. Instead a binary distance measure may be sufficient. Either an attributes' values are the same (distance 0) or they are not (distance 1).

To address the second point, it may be possible to encode the relative importance of some features being more important than others. Importance can be in relation to some class or a group of classes. Simple weighting schemes can be used prior to distance aggregation to represent this fact. For example, assume attribute A has importance p_1 and attribute B has importance p_2 on an absolute scale of $[N, M]$. We can express the actual distance between two instances (based on the attributes A and B) as being $\frac{p_1}{M-N}\delta_M(A) + \frac{p_2}{M-N}\delta_M(B)$.

Another important issue is dealing with missing values. This can be accomplished in MPIL-2 by assigning a feature a radius of infinity.

Encoding rules within MPIL-2 is straightforward. All that is required is to represent a given rule as a collection of instances, where missing (deemed unimportant) attributes have their radius set to infinity. The transformation proceeds by representing rules as trees and converting the leafs into instances.

Finally, expert system like consultation sessions are accomplished by determining the relative importance of features for a given group of stored instances. This can be estimated by determining the relative frequency at which these features occur within a group of instances, or by utilizing a priori knowledge obtained from the domain expert. The features with the highest importance factor are presented first during the consultation process. The response to such query by the user then decides which queries to proceed with next.

5 Summary

This paper introduced a modification to the instance based learning approach. Associating neighborhoods of similarity (either K-dimensional spheres or boxes) with single instances by allowing multi-pass processing of the training set, allows MPIL learning systems to exhibit the following beneficial features:

- Decrease in storage requirements.

- Increase in performance accuracy.
- Ability to extract rules when using K-dimensional neighborhood box.
- Guarantee to correctly abstract from training set, unless contradictions are present.
- No tuning parameters required during learning.

The cost of achieving these improvements lies in the increased complexity of the training phase. The time complexity is of order $O(KN^2)$ for MPIL-1 and $O(K^2N^2)$ for MPIL-2, where K is the size of the input space and N is the number of instances in the train set. The MPIL approach was directly compared to the IBL algorithms with respect to classification accuracy and storage requirements. Ten domains were used in this study, which represent a good mix of structurally different formats. The Harris domain was of importance, since it is a considerably sized real world domain. The issue of accuracy on the training set was pointed out. IB2 and IB3 are not 100% accurate on the training set. This has ramifications in domains in which the examples that make up the training set are commonly seen. For example, in the semiconductor domain, the training set examples will be seen many times over the course of the semiconductor manufacturing process, as they represent common chip faults. To get an idea of the true classification accuracy after the learning phase, it is important to know the accuracy on the set of training examples. Hence, accuracy here is calculated as that on the combined train and test sets. IB3 is sensitive to the settings of SA and SD. Good settings for these parameters can significantly improve the performance of IB3. IB2 and IB3 require less storage of examples than IB1. Storage requirements are significantly lowered for IB2, IB3, and especially the MPIL learners. Finally, the study supports the claim that exemplar based learners can be adequately applied to machine learning tasks and can achieve high storage reduction paired with good overall accuracy across a wide variety of real world domains.

References

[1] Aha, D. W., Kibler, & D., Albert, M. K. (1991). Instance-Based Learning Algorithms, *Machine Learning, 6,* 37-66.

[2] Cheng, J., Fayyad, U.M., Irani, K.B., & Qian, Z. (1988) Improved Decision Trees: A Generalized Version of ID3. *5th International Conference on Machine Learning*, Ann Arbor, Michigan.

[3] Clark, P., & Niblett, T. (1987) Induction to Nosiy Domains. In I. Bratko, & N. Lavarac (Eds.), *Progress in Machine Learning*, Wilslow: Sigma Press.

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

[5] de la Maza, M. (1991) A prototype based symbolic concept learning system. *In Proceedings of the Eight International Workshop on Machine Learning* (pp. 41-45). San Mateo, CA: Morgan Kaufman.

[6] Fahlman, S.E. (1988) Faster-learning Variations on Backpropagation: An empirical study. *Proceedings of the 1988 Connectionist Models Summer School* (pp. 788-793, Vol. 1). Carnegie Mellon.

[7] Fahlman, S.E., & Lebiere, C. (1990) The Cascade Correlation Learning Architecture. In D. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2.* San Mateo, CA. Morgan Kaufman.

[8] Hall, L.O., & Romaniuk, S. G. (1990) A Hybrid Connectionist, Symbolic Learning System. *AAAI-90*, Boston, Ma.

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

[10] Kibler, D. & Aha, D.W. (1990), Learning representative Exemplars of Concepts: An Initial Case Study. In J. Shavlik and T. Dietterich (Eds.) *Readings in Machine Learning.* Los Gatos, Ca: Morgan Kaufman.

[11] Perez, R.A., Hall, L.O., Romaniuk, & S.G., Lilkendey, J,T. (1992) Inductive Learning For Expert Systems In Manufacturing. *HICCS-25, Hawaii International Conference on Systems Sciences,* Koloa, Hawaii.

[12] Quinlan, J. R. (1979), Discovering rules by induction from large collections of examples, In D. Mitchie (Ed.), *Expert systems in the micro electronic age.* Edingurgh University Press.

[13] Romaniuk, S.G. (1991) Extracting Knowledge from a Hybrid Symbolic, Connectionist Network, PhD Dissertation, University of South Florida.

[14] Romaniuk, S.G., & Hall, L.O. (1993) Divide and Conquer Neural Networks. To appear *Neural Networks.*

[15] Romaniuk, S.G., & Hall, L.O. (1993) SC-net: A Hybrid Connectionist, Symbolic System. To appear *Information Sciences.*

[16] Romaniuk, S.G. (1993) Evolutionary Growth Perceptrons, Technical Report, DISCS, National University of Singapore.

[17] Victor, P., Romaniuk, S., Perez, R. & Hall, L.O. (1990) Evaluation of some Inductive Algorithms for Automatic Knowledge Acquisition. *Third Florida Conference on Computer Integrated Eng. and Manufacturing,* (pp. 51-57). Tampa, Fl.

[18] Weiss, S.M. & Kapouleas, I. (1989) An Empirical Comparison of Pattern Recognition. *Neural Nets, and Machine Learning Classification Methods, Proceedings of IJCAI '89*, Detroit, Mi.