Generalizing from Case Studies: A Case Study[*]

David W. Aha
Research Center, RMI Group
Applied Physics Laboratory
The Johns Hopkins University
Laurel, MD 20723 USA
aha@cs.jhu.edu

Abstract

Most empirical evaluations of machine learning algorithms are case studies – evaluations of multiple algorithms on multiple databases. Authors of case studies implicitly or explicitly hypothesize that the pattern of their results, which often suggests that one algorithm performs significantly better than others, is not limited to the small number of databases investigated, but instead holds for some general class of learning problems. However, these hypotheses are rarely supported with additional evidence, which leaves them suspect. This paper describes an empirical method for generalizing results from case studies and an example application. This method yields rules describing when some algorithms significantly outperform others on some dependent measures. Advantages for generalizing from case studies and limitations of this particular approach are also described.

1 PROBLEM AND OBJECTIVES

A central objective in machine learning research is to determine the conditions describing when one heuristic learning algorithm outperforms others for a given set of dependent variables (e.g., predictive accuracy, speed, storage, etc.). Although formal mathematical analyses are preferred to detail these conditions in the form of average expected computational behavior (Pazzani & Sarrett, 1990), such results are difficult to produce since the algorithms and/or databases are usually complex. Instead, empirical evaluations are conducted to yield *case study results* – measures of some dependent variable(s) obtained from applying a set of algorithms to one or more carefully selected databases (e.g., Kibler & Langley, 1988; Frey & Slate, 1991; Clark & Boswell, 1991; Aha, 1991). Invariably, some algorithms are reported to significantly outperform others in the case study. Although authors usually hypothesize why these performance differences occurred, their explanations are infrequently evaluated and may be inaccurate. More systematic methods are required to accurately generalize case study results.

Few methods for generalizing case studies have been reported in the machine learning literature. However, the approach introduced in this paper has much in common with Rendell and Cho's (1990) investigations. They used artificially-generated databases to examine how the performances of two similar algorithms were affected by several data characteristics, particularly concept size (i.e., the percentage of positive instances) and concept concentration (i.e., the number of prototypes defining the target concept). This paper instead focuses on a general method that characterizes the situations when arbitrarily different learning algorithms have a constant significant performance difference.

More specifically, this paper details a simple empirical method that generalizes case studies. It is independent of the set of dependent and independent variables being investigated, the selected learning task, and the selected learning algorithms. The objective of this generalization method is to derive rules of the form "this algorithm outperforms these other algorithms on these dependent measures for databases with these characteristics." Such rules summarize *when* (i.e., under what conditions) rather than *why* the observed performance differences occurred. However, they should help to focus subsequent mathematical analyses on the task of *explaining* why these performance differences occurred. Providing evidence for this claim remains a goal for future research.

Although this generalization method (Section 2) has many limitations (Section 4), it is a useful framework for generalizing from empirical results. Section 3 details an application of this method to three supervised concept learning algorithms on a large database that produced somewhat surprising case study results. The

[*]To appear in *Proceedings of the Ninth International Conference on Machine Learning* (MLC-92)

rules induced as a result of these case study generalizations are evaluated in Sections 3.6 and 3.7.

2 GENERALIZING CASE STUDIES

The proposed generalization method (Table 1) assumes that one or more of the algorithms significantly outperforms the others on some dependent variable(s) when applied to the selected database. It yields rules detailing when these performance differences occur.

2.1 COLLECT CASE STUDY DETAILS

These include the selected algorithms, the values for the dependent variables, and the characteristics of the database (c.f. Rendell & Cho, 1990). Although often difficult to obtain, these characteristics are required for the subsequent attempt to mimic the case study results on an artificially-generated database. Example parameters for characterizing databases include:[1]

1. number of instances,
2. number of target concepts,
3. number and types of attributes per instance,
4. correlations of attributes to target concept disjuncts,
5. distribution of instances within disjuncts of target concepts,
6. distribution of instances among concepts, and
7. amount and type of noise subjected to the instances, attributes, and target concepts.

Although more parameters are required to perfectly characterize a database (e.g., inter-attribute correlations), these provide a useful start. Each is varied or held constant in Section 3's experiments.

2.2 MODEL THE DATABASE WITH ARTIFICIAL DATA

This proposed generalization method requires an accurate characterization of the database to yield accurate algorithm-preference rules. Artificially-created databases are required for this task since modifying the characteristics of a database with some unknown characteristics investigates only the *relative* rather than absolute behavior of the selected algorithms (e.g., as done in (Quinlan, 1986)). However, every effort must be made to ensure that the artificially created database is highly similar to the original database; they should share many characteristics and yield similar values for the selected dependent variables.

[1]These example characteristics foreshadow that our experiments involve concept learning tasks. Characteristics should be chosen in a task-dependent manner.

2.3 SELECT THE INDEPENDENT VARIABLES AND THEIR SETTINGS

The database generator's parameters are dimensions in the *database-characterization space*. The original database's set of characteristics defines an instance in this space. The next step of this generalization method attempts to locate this instance, or at least one highly similar to it, so that it can be used as a basis for empirically exploring the database-characterization space. This requires locating settings for the database generator's parameters such that the algorithms' significant performance differences recur on the generated databases (i.e., we want to replicate the case study results as closely as possible with artificial data).

Once found, the instance in database-characterization space corresponding to these parameter settings, which we will call the *base* instance, is assumed to lie in the same disjunct of that space as the characterization of the original database, where all instances in this disjunct yield the same set of significant performance differences. The generalization method's objective is to characterize this disjunct. This is done by examining the values of the selected dependent variables obtained when applying the algorithms to similar instances in the database-characterization space (i.e., similar settings for the database generator's parameters). The independent variables are selected from among the database generator's parameters. When available, knowledge concerning the differences between the algorithms' capabilities should be used to guide this selection.

Ideally, every instance in database-characterization space similar to the base instance should be sampled to determine whether it yields similar significant performance differences for the selected algorithms. However, this is not feasible for large database-characterization spaces. Instead, strong continuity assumptions are required (i.e., similar instances in database-characterization space are assumed to yield similar performance differences for the selected algorithms). There exists a tradeoff: larger-sized samples reduce the strength of these assumptions but require additional calls to the database generator and additional applications of the algorithms. Some methods for exploring this space include testing on instances that differ from the base instance on only one of its values for the independent variables, testing on instances randomly sampled according to some distribution centered on the base instance, and using a factorial design, again centered on the base instance, in which the tested instances correspond to all combinations of a few values for each independent variable. All three of these methods are exemplified in the experiments described in Section 3.

Table 1: Outline of the Generalization Method

1.	Collect case study details
2.	Model the application database with artificial data
3.	Select the independent variables and their settings
4.	Evaluate the algorithms on artificially-generated databases
5.	Derive a rule summarizing when the performance differences occur

2.4 EVALUATE THE ALGORITHMS ON THE GENERATED DATABASES

The selected algorithms must be tested repeatedly on each set of parameter settings so that the significant performance differences can arise. The number of repetitions should be chosen carefully; it determines the degrees of freedom in the significance tests.

2.5 DERIVE RULES THAT SUMMARIZE THESE RESULTS

The final step derives rules that describe the conditions under which the significant performance differences hold. They can be manually generated by noting the commonalities of the independent parameters' settings when the significant performance differences recurred. For more complex studies, rules can be generated by an appropriate rule-generating algorithm (e.g., CN2 (Clark & Niblett, 1989)), where instances are points in the database-characterization space classified according to whether the significant performance differences occur. The parameters held constant should be included as conditions to these rules. Both manual and CN2-generated rules are exemplified in Section 3.

As with other learning tasks, the rules resulting from this "meta-learning" task should be evaluated for their accuracy (e.g., on a disjoint set of test instances drawn from the database-characterization space). Example evaluations are described in Sections 3.6 and 3.7.

3 AN APPLICATION

This section describes an application of the method introduced in Section 2. Alas, it is itself a case study, and only for three learning algorithms and one database.

3.1 COLLECTING CASE STUDY DETAILS

Terrence Fogarty (1992) recently discovered some surprising case study results with Frey and Slate's (1991) letter recognition database (Table 2). They applied several variations of Holland-style genetic classifier systems on this database, where each instance represents a typewritten letter in one of twenty fonts and is described by 16 integer-valued attributes representing primitive statistical features of the pixel distributions in the original pixel images of the letters. Training on the first 16,000 instances, Frey and Slate's most accurate variant had an 82.7% predictive accuracy on the final 4000 instances. Fogarty (1992) discovered that the nearest neighbor algorithm's accuracy on this same task was 95.7%, an increase of 13%.

Table 2: Characteristics of the Letter Recognition Database (Frey & Slate, 1991)

1. Number of training instances: 16,000
2. Number of test instances: 4000
3. Number of target classes: 26
4. Number of prototypes per class: 20
5. Number of attributes: 16
6. Type of attributes: Integer-valued
7. Range of each attribute's values: [1,16]
8. Distribution of instances among concepts: Uniform
9. Distribution of instances about prototypes: Normal

Table 3: Average Accuracies and Standard Deviations (10 trials) on Frey and Slate's (1991) Letter Recognition Database when Testing on its last 4000 Instances

Algorithm	Size of training sets	
	16,000	1600
BACKPROPAGATION	–	$81.9 \pm 0.6\%$
IB1	$95.7 \pm 0.4\%$	$81.7 \pm 0.7\%$
CN2	$87.9 \pm 0.8\%$	$68.7 \pm 1.0\%$
C4	$86.4 \pm 0.7\%$	$67.4 \pm 0.8\%$
Classifier systems (Frey & Slate, 1991)	82.7%	–

I tested four learning algorithms on the same training and test sets, and, at Fogarty's suggestion, also tested them on ten smaller-sized training sets whose union is the original training set. The results of these tests are summarized in Table 3. The algorithms are:

1. BACKPROPAGATION (Rumelhart, McClelland, & the PDP Research Group, 1986), the well-known multi-layer connectionist algorithm,
2. IB1 (Aha, Kibler, & Albert, 1991), a minor variant of the nearest neighbor algorithm,
3. CN2[2] (Clark & Niblett, 1989; Clark & Boswell, 1991), a set-covering rule-learner that employs a

[2]CN2 was always evaluated using its ordered-rules option (Clark & Boswell, 1991).

Table 4: Average Accuracies and Standard Deviations (10 trials) When Using only the First Five Letters in Frey and Slate's (1991) Letter Recognition Database

Algorithm	Accuracy
IB1	$91.9 \pm 1.2\%$
CN2	$84.5 \pm 1.6\%$
C4	$86.3 \pm 1.9\%$

noise-tolerant significance test to determine which rules to retain, and

4. C4 (Quinlan, 1986), a decision-tree inducer that prunes trees to tolerate noise.

Backprop[3] and IB1 attained significantly higher predictive accuracies than CN2 and C4.[4] This is somewhat surprising; previous comparisons showed that C4 usually outperformed IB1 or recorded similar accuracies, although IB1's accuracy was significantly higher for a densely-populated tic-tac-toe endgame database whose attributes contained little information for C4's splitting criterion (Aha, 1991). However, the tic-tac-toe endgame database is quite different from the letter recognition database (e.g., its attributes are not numeric-valued, it has only two classes). Furthermore, whereas CN2 performed as well as IB1 on the tic-tac-toe endgame database, its accuracies were significantly lower here.

These three algorithms were selected for the present study,[5] the dependent variable chosen was classification accuracy, and the task chosen was supervised concept learning. The objective was to yield a rule describing the conditions when IB1 yields significantly higher classification accuracies than CN2 and C4.

From the standpoint of computer resource requirements, it is quite fortunate that similar significant differences can be obtained with small subsets of the letter recognition database. For example, Table 4 summarizes the averaged results when using only the first five letters of the alphabet from the ten 1600-instance training sets and the 4000-instance test set. IB1's accuracies are still significantly higher than CN2's ($df = 9, t = 3.1, p < 0.01$) and C4's ($df = 9, t = 2.0, p < 0.05$). This reduces the learning task to be modeled in the experiments from 26 to five classes (i.e., corresponding to the first five letters), 16,000 to 311 training instances, and 4000 to 753 test instances (i.e., these are the average number of training and test instances in the 1600-instance databases that are members of one of these five classes).

[3]Backprop was tested with 96 hidden nodes, a 0.01 learning rate, a 0.6 momentum coefficient, and attribute values were normalized by subtracting their mean and dividing by their standard deviation.

[4]For example, IB1's accuracies were significantly higher than CN2's on both the 16,000 ($df = 9, t = 11.0, p << 0.005$) and 1600-instance training sets ($df = 9, t = 9.9, p << 0.005$). They were also significantly higher than C4's for these same datasets (i.e., ($df = 9, t = 13.2, p << 0.005$) and ($df = 9, t = 10.9, p << 0.005$) respectively).

[5]Backprop and Frey and Slate's classifier system were not included; it would be difficult to properly tune their parameters for each of the databases investigated.

3.2 MODELING THE DATABASE WITH ARTIFICIAL DATA

The data-generating program selected for this study has several parameters required to closely characterize the letter recognition database. It is a minor variant of Benedict's (1990) DGP/2 program:

1. Generate a parameterized number of prototypes with locations randomly chosen according to a uniform distribution within a hyper-cubic subspace of parameterized size centered in the instance space.
2. Generate a parameterized number of instances, uniformly distributed over the prototypes, such that each is located within a parameterized maximum distance from a randomly selected prototype. The frequency of occurrence of instances is normally distributed around prototypes.
3. Randomly assign each prototype to one of a parameterized number of classes. Each instance's class is assigned to that of its nearest prototype.

This program generates instances described by integer-valued attributes, distributes disjuncts uniformly among concepts, distributes instances uniformly among disjuncts, and distributes instances normally around the disjuncts' prototypes.

3.3 SELECTING THE INDEPENDENT VARIABLES AND THEIR SETTINGS

Frey and Slate's (1991) extensive description of their database allowed it to be modeled accurately with this database generator. Nonetheless, many of its characteristics remain unknown (e.g., the relevance of the attributes for predicting class information, the distribution of prototypes in the instance space, and the amount and type of noise) and do not necessarily correspond with the generator's assumptions. However, a brief manual search for artificially-generated databases with similar performance differences located parameter settings for the generator (Table 5) that yield relatively similar performance differences for the three algorithms (Table 6). Although these results are not identical, the same significant performance differences hold between IB1 and the other algorithms (($df = 19, t = 2.5, p < 0.025$) and ($df = 19, t = 3.9, p < 0.005$) for CN2 and C4 respectively).

The first four parameters' settings shown in Table 5

Table 5: Parameter Settings for the Five Letter Recognition Database and the Settings Chosen for the Artificial Database Generator

Parameter	Database	
	Five Letters	Artificial
# Instances (train/test)	311/753	311/753
# Classes	5	5
Each attribute's value range	[1,16]	[1,16]
# Prototypes per class	20	20
# Relevant attributes	? (16 total)	13
# Irrelevant attributes	?	3
Instance distribution range	?	1
Prototype distribution range	?	15

Table 6: Average Accuracies and Standard Deviations (20 trials) for the Five Letter Database and the Selected Parameter Settings for the Data Generator

Algorithm	Database	
	Five Letters	Artificial
IB1	$91.9 \pm 1.2\%$	$93.1 \pm 1.5\%$
CN2	$84.5 \pm 1.6\%$	$85.9 \pm 3.2\%$
C4	$86.3 \pm 1.9\%$	$80.2 \pm 3.4\%$

were selected to match the five letter recognition database's settings. The most salient of these parameters is number of prototypes; IB1 should outperform the other algorithms in highly disjunctive spaces because it retains rather than generalizes over information concerning small disjuncts, although this performance difference will decrease as the number of training instances increase, the number of classes decrease, the distribution range for prototypes increases, and the number of prototypes per class decreases.

The number of relevant and irrelevant attributes were selected as independent variables to examine the claim that the accuracy of nearest neighbor algorithms such as IB1 decreases quickly with the number of irrelevant attributes (Aha, 1989; Kelly & Davis, 1991). Relevant attributes are those used by the database generator to generate instances. Irrelevant attributes are added afterwards; their values are randomly selected according to a uniform distribution on the attribute's value range. Thirteen relevant and three irrelevant attributes provided results that were similar to the case study results.

A small Euclidean distance (i.e., 1) for distributing instances around prototypes provided good fits between performances on the five letter database and the corresponding artificial database; larger values quickly reduced the accuracies of CN2 and C4. Significant performance differences should occur independent of this parameter's setting because it can only be increased and IB1's advantage should increase monotonically with this value (i.e., because this increases concept overlap and delays learning in algorithms that use significance tests to tolerate noise).

Finally, while prototype locations were selected randomly from a uniform distribution, the subset of instance space from which they were drawn was controlled. In particular, prototypes were drawn from within a hyper-cubic subspace centered in the instance space whose size was varied. Smaller prototype-distribution ranges should increase the pruning activities of CN2 and C4 when the number of training instances is low, which will decrease their accuracies in subspaces of the instance space where prototypes of different classes lie closely together.[6] Since IB1 is a primitive algorithm without a noise-tolerating mechanism, its accuracies should not decrease as quickly.

The ranges of the values tested for the independent variables are shown in Table 7, where values corresponding to the five letter recognition database are boldfaced. The other values were selected so that a moderately large sphere is explored in the experiments, where this sphere is nearly centered on the base instance.

3.4 EVALUATING THE ALGORITHMS ON THE ARTIFICIAL DATASETS

The three algorithms were evaluated twenty times on each set of parameter settings for all of the experiments described below. Thus, a factorial exploration using the values in Table 7 is prohibitively expensive; it requires $40,824 \times 3 \times 20 = 2,449,440$ learning trials. One-tailed t-tests were used to compare the average classification accuracies of the algorithms on the twenty trials per instance space (i.e., there were 19 degrees of freedom). Differences were considered significant when $p < 0.1$.

The first experiment conducted placed strong continuity assumptions on the database-characterization space; each independent variable was varied individually while the others were held constant at their boldfaced value in Table 7. This required only $25 \times 3 \times 20 = 1500$ learning trials. The following paragraphs summarize the results of this experiment.

Number of training instances: IB1's accuracies were significantly higher than those for CN2 and C4 for all the values tested (Figure 1). Increasing this number above the maximum value tested would eventually allow all of the algorithms to attain insignificantly different accuracies close to 100%.

Number of classes: IB1 significantly outperformed

[6]Larger instance-distribution ranges cause similar confusions. These two parameters are partially redundant.

Table 7: Values of the Independent Variables that were Tested in the Experiments

Parameter	Settings Tested
# Training Instances	{38,77,155,**311**,466,622,777}
# Classes	{2,**5**,8}
Value range	{[1,8],**[1,16]**,[1,24],[1,32],[1,64],[1,96]}
# Prototypes per class	{5,10,**20**,30}
# Relevant attributes	{6,**13**,20}
# Irrelevant attributes	{0,**3**,6}
Instance distribution range	{**1**,2,3}
Prototype distribution range	{5,10,**15**}

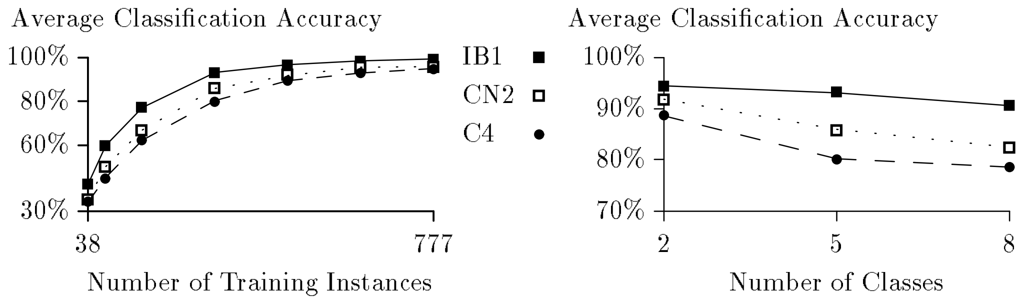

Figure 1: Average Accuracies (20 trials) as the Numbers of Training Instances and Classes are Varied

the other algorithms for the set of values tested for this independent variable (Figure 1). IB1 performed comparatively well when the probability of confusing classifications was high.

Range of values: IB1 significantly outperformed the other algorithms for the values tested, although the difference in performance degraded as this value was increased (Figure 2). IB1 performed comparatively well when the distance between prototypes of different classes was small.

Number of prototypes per class: IB1's significantly higher accuracies became less significant as this value decreased and were insignificant for small numbers of prototypes per class, at which point all the algorithms attained near-perfect accuracies (Figure 2). This agrees with the expectation that IB1 performs well for highly disjunctive target concepts.

Number of relevant and irrelevant attributes: IB1's accuracies were significantly higher only when the percentage of irrelevant attributes was small (Figure 3). This agrees with previous reports that IB1 is extremely sensitive to the presence of even moderate percentages of irrelevant attributes (Aha, 1989; 1991).

Instance distribution range: IB1's accuracies were significantly higher than the other algorithms across the entire range of settings tested for this variable (Figure 4). These differences increased for larger instance-distribution ranges, which decreased the average distance between instances in different classes. This adversely affected the performance of algorithms such as CN2 and C4, which require a significant amount of information regarding class boundaries before they retain sufficiently-detailed abstractions in their concept descriptions. Algorithms without this restriction (e.g., IB1) can learn target concepts more quickly in these situations.

Prototype distribution range: IB1's performance benefit declined as this value was increased. However, its accuracies were significantly higher across the set of values tested (Figure 4). IB1 performed best for small values of this variable because, in such cases, abstraction-based algorithms require that abstractions have significant information content before they are included in the concept description. Thus, they learn more slowly than IB1, which immediately incorporates each training instance's information into its concept description.

An optimal algorithm for these databases would determine the exact location of each prototype and classify instances according to their nearest prototype. CN2 and C4 would perform better in these experiments if they abstracted prototypes rather than hyper-rectangular abstractions. Instead, IB1's learning bias

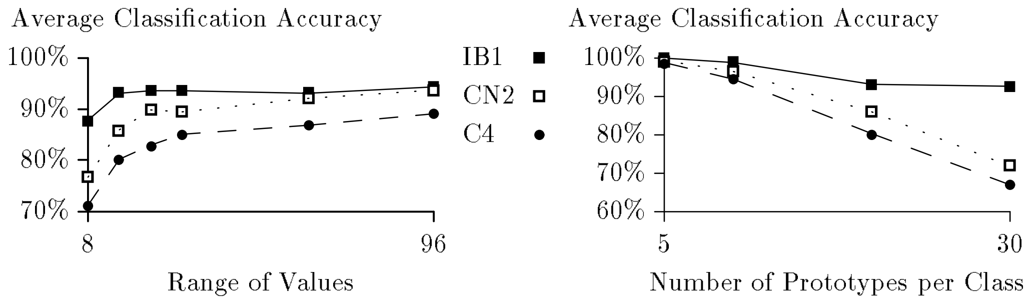

Figure 2: Average Accuracies (20 trials) as the Size of the Attributes' Range and the Number of Prototypes per Class are Varied

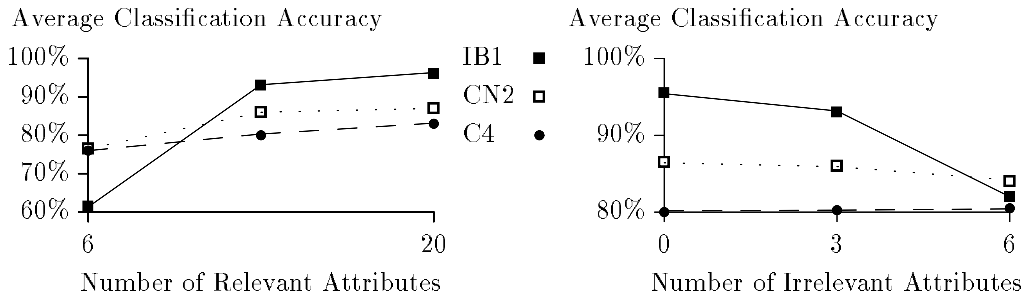

Figure 3: Average Accuracies (20 trials) as the Number of Relevant and Irrelevant Attributes are Varied

is more appropriate here, even though it neither learns prototypes nor uses them in classification predictions.

3.5 DERIVING RULES THAT GENERALIZE THE CASE STUDY

The results described in Section 3.4 can be summarized by the following rule:[7]

> If a database's characteristics match the conditions listed in Table 8, then IB1 will attain significantly higher classification accuracies on that database than CN2 and C4.

Unfortunately, this manually-derived rule is inaccurate. For example, for some database-characterization instances, IB1 performs significantly better than the other algorithms independently of the settings for some of the variables examined. Also, strong continuity assumptions are presumed for the database-characterization space, and several additional variables might also be included among those held constant in the experiments (e.g., the shape of prototypes, which were always hyper-spherical in these experiments, although a brief experiment using prototypes with hyper-cubic shapes produced similar performance differences). Nonetheless, this type of rule is a hypothesis that can be evaluated both empirically and analytically to help understand when some algorithms outperform others.

While only 25 instances in database-characterization space have been sampled, CN2 can still be used to extract rules for predicting whether IB1 will significantly outperform the other algorithms for instances in this space. *Positive* instances are those for which this significant performance difference occurred. The attributes used to describe instances are the dimensions of the database-characterization space (i.e., the parameters of the database generator).

When used to summarize performance differences between IB1 and CN2, CN2 created a rule with three disjuncts. The disjunct corresponding to the predicted location of the original database in database-characterization space is shown in Table 9, along with the complete rule for C4. Their conditions are a subset of those shown in Table 8. This is expected since CN2 was not told which variables were held constant and

[7]No *general* claims are being made here concerning IB1's ability to record higher accuracies than the other, more elaborately designed algorithms. It is a relatively brittle algorithm that, for example, performs comparatively poorly in the presence of many irrelevant attributes.

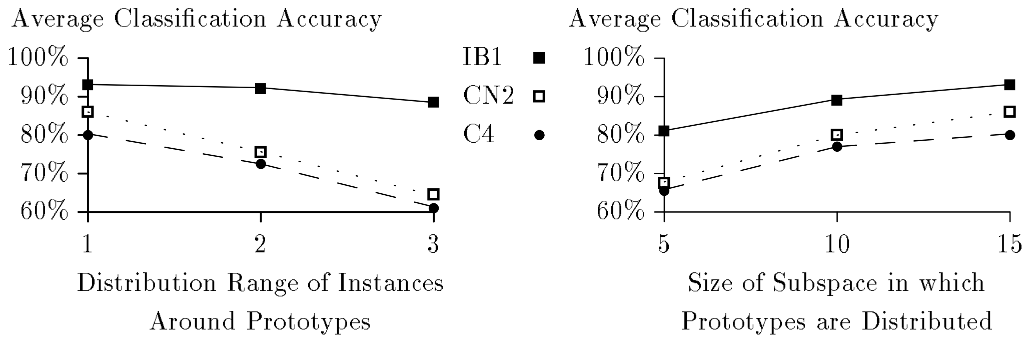

Figure 4: Average Accuracies (20 trials) as the Distribution of Instances Around Prototypes and the Size of the Subspace in Which Prototypes were Located are Varied

Table 8: Conditions When IB1 Significantly Outperforms CN2 and C4

Variables whose values were varied in the experiments:

1. training set represents a small portion of the instance space
2. relatively many classes
3. relatively small attribute-value range
4. large number of prototypes
5. small percentage of irrelevant attributes
6. large distribution range of instances around prototypes
7. distribution range of prototypes is small

Variables whose values were held constant:

1. non-noisy data
2. integer-valued attributes
3. instances are distributed normally around and evenly among prototypes
4. prototypes are distributed randomly in a hypercubic subspace centered in the instance space
5. prototypes are distributed evenly among classes

Table 9: Database-Characterization Disjuncts Induced by CN2 from the First Experiment's Results, where $>>(A_1, A_2)$ means Algorithm A_1 is Predicted to have Significantly Higher Accuracies than Algorithm A_2

```
IF (# training instances < 700) AND
   (# prototypes per class > 7.5) AND
   (# relevants > 9.5) AND
   (# irrelevants < 4.5)
THEN >>(IB1,CN2)

IF (# prototypes per class > 7.5) AND
   (# relevant attributes > 9.5)
THEN >>(IB1,C4)
```

it attempts to generalize over its training set. C4's accuracy was significantly lower than IB1's for a superset of the database-characterization instances when CN2's accuracy was significantly lower. Thus, the conditions for C4's rule are a subset of those in CN2's disjunct. The constants in the rules may seem somewhat arbitrary. Their values depend on which settings were used for the independent variables in the experiments; different training conditions will yield different values for these constants.

3.6 EVALUATING THE DERIVED RULES

These rules' predictions were evaluated on 50 additional instances from the database-characterization space, which were drawn randomly according to a uniform distribution from the ranges shown in Table 10, although the prototype distribution range was forced to be less than the size of the value range. The ranges were chosen so that approximately half of the generated databases satisfied the first disjunct's conditions in Table 9. Tests were conducted on a disjoint set of 500 instances, as was also done later in the factorial study.

The disjunct induced for CN2 performed well; its conditions were satisfied by 21 positive instances and no negative instances. However, an additional 24 positive instances did not satisfy the conditions of CN2's induced disjunct. Likewise, the rule induced for C4 correctly classified 29 positive and 4 negative instances, but incorrectly classified 16 positive instances and 1 negative instance. Thus, the rule induced for C4 is too specific. The complete rule induced for CN2 has three disjuncts; it correctly classified 44 of the 50 test instances. Thus, the disjunct containing the five letter recognition database's location in database-characterization space is one of many in which IB1 performs comparatively well, at least when using this representation – these eight parameters – to define the database-characterization space.

Table 10: Ranges of Parameters Used to Test the Generated Rule

Parameter	Range Examined
# Instances	50-900
# Classes	2-25
Value range	[1,4]-[1,1000]
# Prototypes per class	1-32
# Relevant attributes	8-18
# Irrelevant attributes	0-6
Instance distribution range	1-5
Prototype distribution range	3-30

3.7 A FACTORIAL STUDY

Additional generalization rules, presumably more accurate than the originals, were obtained by re-running CN2 after adding these 50 instances to the original 25. The newly induced rules were then evaluated on database-characterization instances obtained by conducting a small-scale factorial study using both the extreme and boldfaced values shown in Table 7. Extreme values were chosen so as to maximize the size of the database-characterization subspace explored, although only sparsely since at most three values were tested per parameter. This study required applying the three algorithms twenty times each to $2^2 \times 3^6 = 2916$ sets of parameter settings. The value range was constrained to be greater than the prototype distribution range, which reduced this number to 2430. This yields $2430 \times 3 \times 20 = 145,800$ experiments, which required over 70 cpu-days to complete on SUN Sparcstation 2's and ELC's provided for use by the Department of Computer Science of The Johns Hopkins University.

The induced disjuncts containing the base instance (Table 11), which are refinements of the disjuncts shown in Table 9, performed extremely well; the pertinent disjunct induced in the rule for CN2 was satisfied by 480 of the 2430 instances, 460 (95.8%) of which were positive. The pertinent disjunct induced in the rule for C4 was satisfied by 720 of its 2430 instances, 696 (96.7%) of which were positive.

The complete rules each had 4 disjuncts. They performed poorly; the rules induced for CN2 and C4 correctly classified only 50.9% and 64.6% of their 2430 instances respectively. However, their accuracy should improve under more appropriate training conditions.

4 LIMITATIONS

4.1 LIMITATIONS OF THIS CASE STUDY

Only eight parameters were varied in the experiments while many important parameters were held constant.

Table 11: Database-Characterization Disjuncts Induced by CN2 from 75 Instances

```
IF (# training instances < 737) AND
   (# prototypes per class > 5.5) AND
   (# relevants > 8.5) AND
   (# irrelevants < 5.5)
THEN >>(IB1,CN2)

IF (# training instances < 737) AND
   (# prototypes per class > 5.5) AND
   (# relevant attributes > 8.5)
THEN >>(IB1,C4)
```

Thus, the rules are highly constrained; more general rules would be preferred. Also, constructive induction techniques may be required to construct database-characterization spaces that support the induction of more accurate case study generalizations.

Furthermore, among the variables examined, only a few of their values were examined, even though a small factorial analysis of their values was conducted. Instances in the database-characterization space that were not evaluated for performance differences were assumed to be similar to their neighbors. However, these continuity assumptions may not hold; more finely-grained experiments are required to determine whether overgeneralizations were made.

Although the rules describing when IB1 is expected to outperform the other algorithms were tested on 2480 additional sets of parameter settings, they were not tested on other databases whose characteristics satisfied their conditions, which would provide valuable feedback on their accuracy.

4.2 LIMITATIONS OF THIS APPROACH

Formal analyses should be used to reduce the amount of testing required for a factorial design by providing additional insights on continuities in the database-characterization space. Automating the accurate selection of useful independent variables and their settings would greatly improve this approach and simplify comparisons with other approaches that generalize case study results.

Next, this method relies on having a good characterization of the original database, which is often difficult to obtain. For example, although the number of prototypes per concept is known, the database does not describe which instances are in which disjunct. Data analyses must be accompanied by detailed information on how the database was constructed.

The original database may lie in a different disjunct of the database-characterization space than its corresponding artificial database. This can occur when insufficient information is known regarding the database's characteristics and the algorithms coincidentally perform similarly on these two databases. In such cases, the extracted rule could be useless; perhaps no real-world database would satisfy its conditions.

Finally, algorithms with several tunable parameters (e.g., BACKPROP) must be accompanied by an automated tuning procedure to be used with this generalization method due to the large number of – and differences among – the databases examined.

5 CONCLUSION

This paper describes and applies an empirical method for generalizing case study results. It uses these results to focus on a location in database-characterization space in which known performance differences occur and generates rules characterizing when these differences occur. This should help to focus mathematical analyses investigating the causes of the observed performance differences. This generalization method is independent of the selected learning task, dependent variables, and learning algorithms.

Although the rules derived from this method are highly constrained, they are more useful than the single datapoint provided by the case study results; the rules yield valuable characterizations describing when to prefer using specific learning algorithms over others. The difficulty of locating even so constrained a rule highlights the difficulty of finding general rules of this form.

A future research goal is to demonstrate that this method can also be used to help determine *why* algorithms have significant performance differences. For example, the performance of variants of the same algorithm can be compared to determine when the variations improve performance (e.g., as in (Dietterich, Hild, & Bakiri, 1990)).

Acknowledgements

Suggestions by Isaac Bankman, Pat Langley, Dennis Kibler, and the four anonymous reviewers greatly improved the content and presentation of this paper. Vince Sigillito and Fernando Pineda assisted with the design, implementation, and tuning of BACKPROP. Thanks to Terrence Fogarty for sharing his results on the letter recognition database and, thus, for providing the motivation for this paper. Thanks also to Powell Benedict for providing DGP/2, to Robin Boswell and Peter Clark for providing CN2, and to J. Ross Quinlan for providing C4. Finally, thanks to Jane Scott for proofreading a near-final draft. The U.C. Irvine Repository for Machine Learning Databases, which contains the letter recognition database and DGP/2, can be accessed via remote ftp to ics.uci.edu. Computing equipment for this work was provided in part by NSF Grant No. CDA-9015667.

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. (1991). Incremental constructive induction: An instance-based approach. In *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 117–121). Evanston, IL: Morgan Kaufmann.

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

Benedict, P. (1990). The second data generation program – DGP/2. University of Illinois, Urbana-Champaign, Inductive Learning Group, Beckman Institute for Advanced Technology and Sciences. Unpublished.

Clark, P. E., & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In *Proceedings of the Fifth European Working Session on Learning* (pp. 151–163). Porto, Portugal: Springer-Verlag.

Clark, P. E., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*, 261–284.

Dietterich, T. G., Hild, H., & Bakiri, G. (1990). A comparative study of ID3 and Backpropagation for English text-to-speech mapping. In *Proceedings of the Seventh International Conference on Machine Learning* (pp. 24–31). Austin, TX: Morgan Kaufmann.

Fogarty, T. C. (in press). First nearest neighbor classification on Frey and Slate's letter recognition problem. To appear in *Machine Learning*.

Frey, P. W., & Slate, D. J. (1991). Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, *6*, 161–182.

Kelly, J. D., Jr., & Davis, L. (1991). A hybrid genetic algorithm for classification. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence* (pp. 645–650). Sydney, Australia: Morgan Kaufmann.

Kibler, D., & Langley, P. (1988). Machine learning as an experimental science. In *Proceedings of the Third European Working Session on Learning* (pp. 81–92). Glasgow, Scotland: Pitman.

Pazzani, M. J., & Sarrett, W. E. (1990). Integrating empirical and explanation-based learning: Experimental and analytical results. In *Proceedings of the Seventh International Conference On Machine Learning* (pp. 339–347). Austin, TX: Morgan Kaufmann.

Quinlan, J. R. (1986). The effect of noise on concept learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. II). San Mateo, CA: Morgan Kaufmann.

Rendell, L., & Cho, H. H. (1990). The effect of data character on empirical concept learning. In *Machine Learning*, *5*, 267–298.

Rumelhart D. E., McClelland, J. L., & The PDP Research Group (Eds.), (1986). *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1). Cambridge, MA: MIT Press.