In: Machine Learning, Meta-reasoning and Logics, pp159-186,
Eds: P. B. Brazdil and K. Konolige, (1990), Boston: Kluwer.
http://www.cs.utexas.edu/users/pclark/papers

A Comparison of Rule and Exemplar-Based Learning Systems

Peter Clark (pete@turing.ac.uk)
The Turing Institute
36 N.Hanover St
Glasgow, UK

1990

Abstract

Recently, there has been renewed interest in the use of exemplar-based schemes for concept representation and learning. In this paper, we compare systems learning concepts represented in this form with those which learn concepts represented by decision rules, such as the ID3 and AQ11 rule induction systems. We aim to clarify the distinction between the two representational schemes, and compare how systems based on the different schemes address the problem of learning within finite resources. Our conclusions are that the schemes differ in two important ways: in the different 'biases' with which they select between alternative concepts during search, and in the different computational approaches of generalising before or during a run-time task. We also show that in addressing the problem of finite resources important commonalities between implementations based on both representational schemes arise, and by highlighting them aim to help enable the transfer of techniques between the two paradigms.
Keywords: bias, exemplar-based, case-based, concept, rule learning, resources.

1 Introduction

Recently, there has been renewed interest in the use of exemplar-based schemes for concept representation (e.g. [4, 6, 16]), marking a notable shift in parts of the machine learning community [21]. Exemplar systems are based on a representation in which selected examples (termed exemplars) are stored and a matching algorithm retrieves that which best matches a new case. However, other methods of concept representation, in particular the production rule paradigm of expert systems, have already proved successful in many applications (e.g. Mycin [44], R1 [23]) and much work in machine learning has been devoted to the automatic formation of generalised statements from examples (e.g. AQ11 [28], C4 [37], Lex-2 [31]). This paper compares and contrasts learning systems based on exemplar representations with such rule learning systems.

Any concept learning system must address the problem of how to transfer knowledge from previously observed to new examples within limited resources, the two most important of which are speed and memory usage. We compare how the issue of dealing with speed and memory constraints are addressed by systems of both kind within the context of the representational methods they support. A principal aim of this paper is to show that, in addressing these issues, important commonalities between the two schemes arise. Systems using either scheme typically compromise, to a degree, their original assumptions

in order to deal with these issues effectively and in so doing have become more similar to each other. We aim to highlight this and the resultant partial convergence of systems of both kinds. It should be noted that we are *not* simply trying to argue, for example, over the merits of storing all the training examples or not, as might on first glance be assumed to correspond to exemplar-based or rule learning systems respectively – rather we wish to indicate that, for example, the issue of what information to store is common to *both* schemes and in addressing the issue the distinction between the two is considerably narrowed. By highlighting these similarities we hope to encourage the transfer of techniques between the two paradigms.

2 The Representations: Descriptions

2.1 Exemplar-Based Systems

Unfortunately, a clear and agreed-on definition of 'exemplar-based system' is difficult to find in the literature. Many systems which store descriptions of examples and perform a classification of some sort based on comparison with the descriptions are referred to as exemplar-based (although almost any learning system can be described in this form). For the purposes of this paper we adopt a narrower definition of 'exemplar-based' in order to allow useful statements to be made about the systems.

Exemplar-based systems are characterised by storing the full description of (ie. all the information provided to the system about) selected training examples ('exemplars') in memory. New examples are classified by means of a matching algorithm which retrieves the exemplar best matching the new example and assigning the exemplar's class to that example. This method sometimes referred to as the 'nearest neighbour' classification algorithm. The one extreme of exemplar representation where *all* examples are stored as exemplars (e.g. Cyrus [18] and Julia [17]) is sometimes referred to as an 'instance-based' or 'case-based' representation. The other extreme, where a *single* exemplar represents a concept, is sometimes referred to as a 'prototype-based' representation (e.g. Taxman-II [22], and by Sowa [46]). Other examples of exemplar-based systems between these extremes include Protos [4, 33] Nexus [6] and those studied by Kibler and Aha [16].

Although matching algorithms vary from system to system, some general statement can be made about the models of the concept they assume. Essentially the matcher assumes a form of 'm out of n' model, whereby (possibly weighted) matches between elements of data are summed to yield an overall match strength. The assumptions in the model which such a matcher implicitly represents are:

- Concept membership can be assessed by a 'summing' (i.e. combining) of evidence
- No single set of conditions is necessary for concept membership
- The boundaries of one concept are partially dependent on the proximity of other nearby concepts
- Concept membership includes a notion of 'degree of membership'
- Concepts may by *polymorphous*, i.e. their examples may occupy several disconnected regions in example space

2.2 Rule-Based Representations

Rather than representing concepts by a set of exemplars and a matching algorithm, it is more common to represent a concept by a *general statement*, usually in the form of a set of descriptors combined into conjuncts and disjuncts. A simple interpreter deems a new example to be a member of the concept if the general description holds true for it. Much work has been devoted to the empirical learning of such statements from examples [9] and two families of rule induction system, based on the AQ [27] and ID3 [36] algorithms, have been particularly successful. Examples in the AQ family are AQ11 [28], Gem [39] and AQ15 [24], and in the ID3 family are Assistant-86 [8] and C4 [37]. In addition, there has been work on the analytic learning of rules [5] sometimes referred to 'explanation-based learning'. Examples include Strips [11], Lex-2 [31] and Soar [20].

As for exemplar-based representations, such systems make assumptions about the world which they observe, the validity of which will affect the performance of the systems. These assumptions are:

- Concept membership can be assessed by the passing of one of a set of tests, each test consisting of the requirement for a set of features to all be present (represented by a disjunctive set of conjuncts)
- Concept membership is categorical - there is no notion of 'degree of membership'
- Concepts may by *polymorphous*, i.e. their examples may occupy several regions in example space[1]

2.3 Comparisons

Some important points of comparison should be made. Firstly, the two representational schemes have equivalent power for representing the extension of a concept; in the extreme case (with a finite example space) an exemplar-based system would simply store all the examples and a rule learning system would have a rule for every example, testing on all its attributes. The areas shown in Figure 1 bound the concepts learned by an exemplar-based and rule learning system respectively after being provided with a large number of training examples. Despite the different representational schemes, the extension of the concept is very similar.

The boundaries of a concept's extension, as found by both rule and exemplar-based learning methods, can be drawn as one or more hyper-polygons in example space. However, each representational method has a different preference or 'bias' about which concept extension is the best one to learn when the density of training examples in example space is low. This arises because although systems in both paradigm prefer syntactically simple to complex concept descriptions (given no other discriminating factor), the extensions of 'simple' concepts in both schemes differ markedly where few training examples are available. Rule learning systems are 'biased' towards, ie. will prefer, hyper-polygons shaped as hyper-rectangles with edges parallel to the feature axes of example space. This is because it is precisely concepts shaped in this way which will produce short decision rules. In the exemplar-based system, the bias is towards hyper-polygons whose edges maintain equal 'distance' from nearest examples of different classes, where distance is measured by

[1] Induced rules differ from the non-polymorphous 'set of singularly necessary and jointly sufficient conditions' with which exemplars are sometimes compared as disjuncts as well as conjuncts are almost invariably allowed

Figure 1: Concept boundaries as learned by exemplar-based (left) and rule induction (right) systems differ less when the density of training examples in example space high. The dotted lines indicate the target concept. (Diagram compiled by and reproduced with kind permission of David Aha, UCI, Ca.)

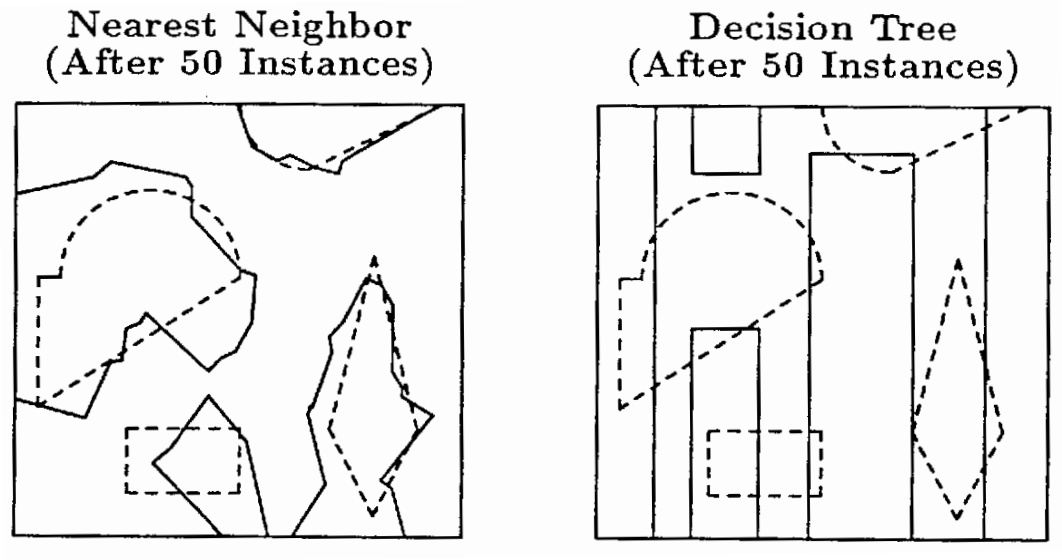
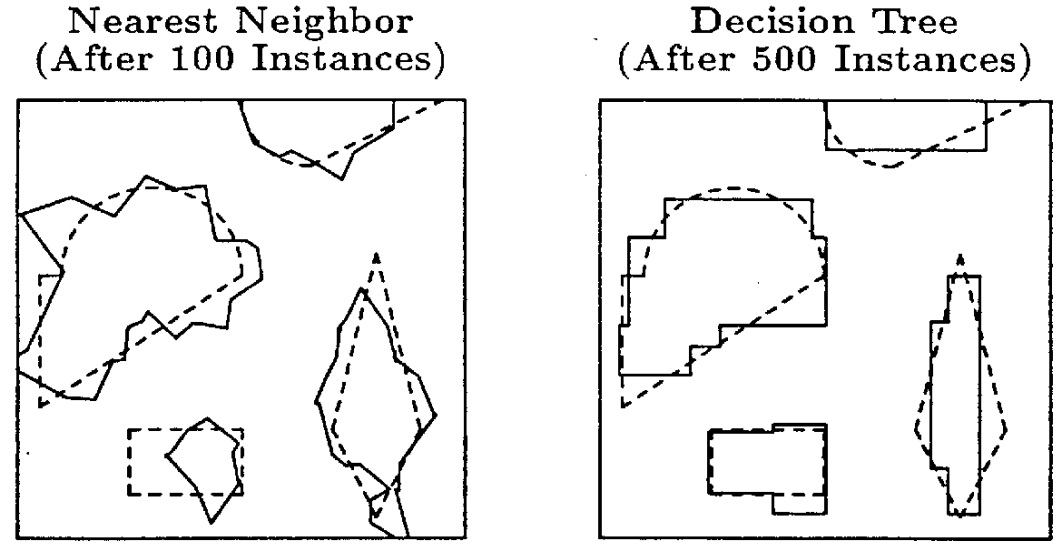


Figure 2: Concept boundaries as learned by exemplar-based (left) and rule induction (right) systems differ significantly when the density of training examples in example space is low. The dotted lines indicate the target concept. (Diagram compiled by and reproduced with kind permission of David Aha, UCI, Ca.)



the system's similarity metric. These boundaries are somewhat analogous to contours of zero potential between positive and negative electric charges in a physical space. There is no preference in exemplar systems for polygon edges to be parallel to the example space axes. Figure 2 shows that the same concept as learned by both types of system with fewer (fifty) examples provided. Although the density of examples in example space used here is still quite high (compared with many real-world applications of concept learning systems) the concepts learned still differ significantly. Which one of these biases is more appropriate for a given problem is a characteristic of the application domain itself.

Another comparison can be drawn concerning the computational processes involved in the two paradigms. Generalisation from examples is neccessary in both paradigms in order for learning to occur. In exemplar systems, the use of a matching algorithm for classifying new examples means that the bulk of the work for knowledge transfer to new examples occurs at run-time. Exemplar systems can thus be loosely viewed as representing a concept by examples plus the means for generalising from them, rather than as generalisations themselves. This is an important feature of such representations, and contrasts sharply with the pre-formed set of generalisations stored by rule-based systems. We discuss the implications of this in sections 4 and 5.

3 Representational issues

3.1 Degree of Concept Membership

One characteristic of rule learning systems has been that they do not provide a measure of 'degree of membership' of a new example with a class. Instead the new example either categorically is or is not a member. Although this is satisfactory for some artificial domains ([4] gives the examples of "a forked position in chess" and "a prime in number theory"), in many domains it would be useful to have a degree of associated confidence.

The problem of incorporating a degree of match in applying rules has already been partially addressed as a result of a common problem with some rule induction systems. This problem is that some induction systems (e.g. those based on the AQ family) may induce logically inconsistent rules where, although it is known that each example must belong to one and only one class, sometimes more or less than one class is predicted (i.e. contradictory or zero rules fire respectively). In order to resolve such conflicts, matching procedures are used to assess which rule the example matches 'best' (e.g. AQ11 [28]).

Recently, the use of matching to augment logical instantiation in rule application has become emphasised as a way in which the notion of degree of match *can* be introduced in rule application. This marks a move towards the exemplar-like matching algorithms by rule induction systems, using similar methods of weighting features and weighting rules according to their coverage (a kind of 'prototypicality') of a concept. In AQ15, the degree to which the 'base concept representation' of rules can be mixed with matching procedures can be varied [25]. Quinlan recently proposed an adaptation of the decision tree interpreter to introduce a measure of probability of class membership [34].

A final point is that using matching processes can act as a useful source of domain knowledge, and can be used to resolve ambiguities in data. If a system is able to recognise near matches (i.e. 'near misses' [49]) as well as when an example matches an exemplar/generalisation perfectly, then this can act as a source of new domain knowledge by inferring the 'missing inference links' and selecting the appropriate interpretation required to make the match perfect [48, 12].

In summary, the representational requirement of providing a degree of concept membership is often important in many applications. As a consequence, some recent work in inductive learning has weakened the assumption that concept membership is either 'all or nothing' thus making a step towards the exemplar-based concept model of membership based on a summing of evidence. The result is a blend of exemplar-based and generalisation-based schemes, with generalisations still being made (which can be viewed as 'generalised exemplars') and the interpreter for their application incorporating a matching algorithm.

3.2 Comprehensibility and Explanation

3.2.1 Rule Learning Systems

One of the principal aims of rule induction systems is that of producing a concept representation which is easily comprehensible - this is the motivation behind searching for the simplest rules to describe the data, and the reluctance to use matching procedures during rule application.

However, one factor limiting comprehensibility of rules is their size – if a rule has many 'condition' parts, or a decision tree is deep, it can become difficult to follow. In order to overcome this, one approach has been to allow the user to structure the task manually into a hierarchy of problems and sub-problems [43, 38], and another, in the case of the learned rule being a decision tree, to constrain the tree to be linear[2] [1].

3.2.2 Exemplar-Based Systems

One of the notable consequences of the exemplar model is that, in assessing concept membership, there is in general a *single* previously encountered example which best matches a new case[3]. This is in contrast to rule learning systems where there may be several examples which equally match the new case (e.g. the examples at the leaf node of a decision tree).

Having a single best matching example can help in explanation. In classification tasks, exemplars offer explanations of its decisions in the form "The new case is similar to X, and X was a Y, hence I predict the new case is also a Y". This form of explanation has the advantage of being in the language of examples – sometimes preferred by experts. However, should the user query why the similarity was found, or ask "but isn't the new case more similar to Z?" the explanation produced is in terms of numerical summations and potentially difficult to understand.

One exemplar system, Protos, has sought to hide the numerical reasoning from the user by exploiting extra domain knowledge. Numerical parameters used by Protos are (mainly) derived from an analysis of explanations provided by the user. Should the user query a decision, Protos can present the inferences giving rise to the numerical parameters rather than the parameters themselves. This is intended to make the reasoning more comprehensible.

[2] i.e. at most one non-leaf node allowed beneath any node in the tree

[3] This is not to say that a single example does all the work of classification; all previous examples may have contributed to finding optimal parameter values for the matcher

3.2.3 Summary

There are several different modes of explanation which can aid comprehensibility. One is to present general summary information in the form of short statements, generally considered comprehensible as partly evidenced by the increased use of rule-based programming. A second method is to converse with the expert in the language of examples, as exemplar-based systems do.

In order to meet the representational requirement of comprehensibility, both schemes make a claim to have achieved acceptability. The modes of explanation they use can be viewed as complementary to each other; there is no reason, given the strong existing overlap between the schemes anyway, why either should not adopt the explanation techniques of the other. Exemplar schemes do generally represent summary information which could be presented to the user, and it would not be difficult to extend rule learning systems to select a previously encountered example and present it to the user in support of a rule used to classify a new case.

4 Efficiency and Compilation

In many problem-solving systems, there is a trade-off between how much computational effort should be expended before a performance task is at hand, i.e. 'compiling' knowledge, and how much should be performed at run-time. Compilation risks performing redundant work and makes incremental learning more difficult, but also may offer large increases in efficiency where the compiled knowledge is repeatedly used (rather than having to be recomputed each time it is required).

Rule and exemplar-based learning methods have markedly different implications for where computational effort will be expended. In rule learning systems, generalisations are sought for and stored explicitly, prior to a performance task. In exemplar-based schemes, however, the generalisation machinery is embodied in the matcher to be used only when needed for classification, and any generalisations which the matcher forms are discarded rather than stored. It is important to note that doing 'matching' can also be viewed as 'doing generalisation at run-time' (rather than some alternative process to generalisation).

4.1 Rule Learning Systems

The pre-compiling of generalisations can make the system faster – on the face of it the discarding of generalisations found by matching may seem wasteful. However, there are three counter-efficiency effects which pre-compilation causes and must be born in mind:

- The system can be swamped with generalisations if it is not selective about which to store. This issue has been examined by workers in explanation-based learning (see [30, 29, 15]).

- There is a computational overhead in forming the generalisations in the first place – it may be that there are many valid generalisations which can be formed, and to form them all beforehand is wasteful. With careful control of search, an exemplar system may never need to fully explore all generalisations which would have been formed by a rule learning system.

  This is especially true in case-based planning systems such as Chef [13] and by Carbonell [7]. Given an example plan, these systems *could* form all the (potentially

numerous) generalised variants of it, then select and instantiate one when a new problem is encountered. However, by instead performing matching at run-time, only a few such variants ever needs to be explored.

- In situations where there is changing domain knowledge, sometimes previously formed generalisations must be amended or removed if the knowledge used to form them becomes invalidated. There is a book-keeping overhead incurred here, plus the discarding of unused generalisations makes the work expended in forming them wasted.

To summarise, it is not always advantageous to compile knowledge before a run-time task is at hand; there is a risk of expended effort being wasteful and can make incremental learning more complex. This issue has not been addressed thoroughly in work on rule learning, where it is generally assumed that speed costs during classification of new examples are much more important to minimise than speed costs during the learning phase. However for the development of incremental learning systems with large databases this assumption may not hold, and as a result it may become preferable to sometimes generalise from examples only as necessary ('lazy generalisation') rather than exhaustively beforehand. As a result, the computational mechanisms for rule learning become more similar to exemplar-based approaches by acknowledging the need for run-time generalisation.

4.2 Exemplar-Based Systems

Despite claims to the contrary, most exemplar-based systems do in fact perform some kind of pre-run-time summarisation of data, and thus also compile information in order to classify new examples. Indeed, without any summary information about examples there is no representation of which features of an example are important for classification and which are not. For instance, it may be that all our example elephants have trunks, but without this commonality being identified there is no reason to consider that a featural match of 'trunk' between a new example and an elephant exemplar should be given higher weight than, say, a featural match of 'male' with other exemplars. This is the problem of 'category cohesiveness' discussed by Bareiss and Porter [3].

In order to represent and use such commonalities, various methods are used in exemplar-based systems. Some such methods are:

1. **Weighting features:** One method for representing important commonalities is by weighting some features as more important than others. In some exemplar-based systems the weights are found by applying statistical methods to the previously observed examples. In others, such as the case-based systems Chef [13] and Judge [2] the weights of importance are user-supplied. In the exemplar-based system Protos, they are found by analysis of the user's explanation of the relevance of an exemplar's feature to its category.

2. **Use of Generalised Structures:** Another method for representing important commonalities is through the use of additional generalised structures to organise memory [42] typically organised hierarchically. Location of a best matching case involves two processes – firstly descending the hierarchy to locate previous cases and secondly applying a matching process to select among those cases encountered (as performed by Mediator [19]). This has the effect of considering some (possibly

generalised) features as essential prerequisites for selecting an old case, and others as optional.

3. **Prototypicality:** In addition to exemplars, the exemplar-based system Protos contains a representation of the categories exemplars belong to. Each category has a set of weighted features connected to it, and represents the importance of features to the general category rather than to a specific exemplar. During classification, sometimes a new case may match a general category better than any single exemplar in which case the most *prototypical*[4] exemplar of that category is selected as a candidate for matching with the new case.

In summary it is false to say that "rules summarise whereas exemplars do not" because, from the representational requirement to model category cohesiveness, exemplar systems have also had to include some form of summary information. This has been recognised in research on exemplar systems, and in practice incorporated into implementations. As a consequence, a large degree of overlap with rule induction systems has been introduced – rule induction systems perform a similar process of searching for overall commonalities in data. In the case where exemplars' summary information is represented by generalised structures such as rules or decision trees, the overlap with rule learning systems is particularly noticeable and it perhaps would be more accurate to describe such systems as a blend of exemplar-based and generalisation-based approaches.

5 Memory Usage and Incremental Learning

Both exemplar systems and rule learning systems have a second major computational issues to address, namely that of how much data to store in memory. The problem can be stated thus:

- Storing all previous examples (the exemplar extreme) is wasteful and makes matching inefficient
- Throwing them all away (the rule-based extreme) makes incremental learning impossible

As a result, some intermediate level of storage can be arrived at by systems using either scheme, again increasing their similarity.

5.1 Exemplar Systems

Exemplar systems rarely store all their training examples – instead only a selected subset are stored. Incremental learning in exemplar systems typically occurs by variations on the 'additive algorithm', described in its simplest form in [16]. If a new example matches an exemplar well and the example is a member of the exemplar's category, the new example is discarded. Otherwise, the example forms a new exemplar on its own. More sophisticated systems will incorporate information about examples before they are discarded, for example by updating weights on features in the best-matching exemplar's description or updating parameters in the matching algorithm. This incremental method is used in Protos and Nexus [6].

[4]The authors of Protos refer the reader to [41] for the definition of prototypicality used

In storing only selected exemplars, some systems adopt a means of annotating the exemplar with extra information to summarise those examples which the exemplar represents (e.g. measures of how representative each exemplar feature is of the discarded examples, how many examples were discarded). This provides a useful mechanism for saving memory and computational cost while simultaneously minimising the information loss to the system.

5.2 Rule Induction Systems

In order for rule systems to learn incrementally, one approach is to maintain a 'full memory' of training examples. With this, an incremental learning algorithm can be applied to cope with new training examples [26], as has been applied in AQ15 [24] and the system by Mozetic [32]. Here

1. rules contradicted by a new example are either specialised to exclude it (as in AQ15) or simply discarded (as in Mozetic's system) and

2. new rules are induced for any examples now without rules 'covering' them.

Similar mechanisms have been applied to incrementally forming decision trees (eg. ID5 [47]).

An alternative to using 'full memory' is simply to retain a set or 'window' of *most representative* training examples, as performed by various implementations of the ID3 induction system [35]. If any new examples arrive which are incorrectly classified, they are added to the window and other examples removed. Following this, the decision tree is re-induced from scratch. The key task here is to identify in the window parcels of instances that had a similar influence in the formation of the rule, and ensure at least one representative is carried forward into the new window.

Usually, the addition of new examples to systems using these algorithms will have little effect – for example the attribute test at the root node of a decision tree is unlikely to change due its choice being based on a large number of examples (hence a single example has small effect). However, sometimes the addition of a single new example *can* cause a large change in the rule-base. Whether this is a good or a bad property is debatable – on one hand it can be argued it gives the rule-base the undesirable property of 'brittleness' [14], on the other hand it enables the occasional major re-organisation of knowledge to occur allowing the system to escape from some local but not global optimum of organisation, enabling it to make 'paradigm shifts' [10] as well as small refinements to knowledge.

5.3 Summary

In order to meet the computational requirement of limited memory usage and the ability to learn incrementally, both representational schemes have converged to a degree on the mechanisms involved. Exemplar schemes typically abandon a 'full memory' approach and instead store only selected examples. Rule learning systems abandon the notion that a concept can be represented purely by a general set of conditions, and have had to extend it to include reference to example cases from which it was derived.

The principal issue here, which both exemplar and rule induction systems need to address, is that of how to minimise information loss without seriously damaging learning or performance. In addressing this issues, the two representational schemes have

taken a large step towards each other. Typically exemplar schemes have addressed this issue in more detail (whereas many rule learning systems still rely on 'full memory') and consequently have techniques to offer to the rule learning methodologies.

6 Appropriateness of Representation

Rule learning systems can be described as compiling a set of generalisations for use at run-time, whereas exemplar-based systems store specific examples plus a mechanism for generalising from them at run-time as necessary. Thus to a first approximation exemplar systems can be said to contain only the *means for generalising* rather than the generalisations themselves; generalisation is deferred until a performance task is known[5].

We have argued throughout this paper that in order to meet constraints on speed and memory usage (and also for rule learning systems to represent degree of concept membership), implementations of these learning paradigms often involve adopting techniques of each other for success. However it is still worthwhile to ask what features of a problem domain make one or other representational scheme appropriate as a base to build on.

6.1 Appropriateness of 'Bias'

We have illustrated in section 2.3 that exemplar-based and rule learning systems will learn substantially different concepts when provided with training examples in numbers such that their density in example space is low. This phenomenon is sometimes described as being due to a difference in pre-programmed preference or 'bias' of the learning methods. Many applications of concept learning systems use training sets where the density of training examples in example space is low, and hence this difference may have important implications for applications of concept learning systems.

Exemplar-based approaches will be more appropriate for domains where concept membership is well modelled by the 'summing' of evidence rather than by a requirement for certain specific properties to be present. The question of which those domains are, however, still evokes considerable debate. It has been claimed that most natural domains are best modelled by exemplar-based approaches (e.g. [45, 4]), although there has been little research conducted into ascertaining the appropriate learning 'bias' by analysis of the domain itself (though see [40] for work in this area).

6.2 Concept Polymorphy and Volatility

In section 4 the merits of pre-compiling generalisations (rule learning systems) compared with those of generating them at run-time only as needed (exemplar-based systems) were compared. The appropriateness of these approaches again depends on the domain of application.

When a concept has a *high polymorphy*, we mean that its extension in example space occupies a large number of disconnected regions. As a result a rule learning system would require a large number of rules, each covering only a small area of example space, in order to fully describe the concept.

In such domains, there are only limited gains to be made by storing generalisations due to the high computational expense of generating them all beforehand and the possibility

[5]This view usefully approximates the property that making statements about a new example is generally more computationally expensive in exemplar-based systems

that many may simply not be required later in the run-time tasks which the system is to perform.

Additionally, as new examples are observed and domain knowledge in the system changes, the concept description may need to be re-assessed. This again may make work in generating previous rules redundant, and is a factor against generating a set of rules before a performance task is at hand. This will particularly be a problem where generalisations are based on few examples, where a single addition of an example may cause a large change in the learned concept. We refer to this as *concept volatility.*

Thus, in domains where there is high concept polymorphy and the concept description may be subject to frequent revision, the exemplar-based approach of generalising only where and when required for a performance task will be preferable.

7 Summary

Exemplar-based and rule learning systems differ in two important respects. Firstly, they have different biases (built-in preferences) about which generalisations are most appropriate to draw from a set of training examples. Exemplar-based systems can be viewed as being 'biased' towards concepts whose extension boundaries follow contours of equal 'similarity distance' from nearest examples of different classes, whereas rule learning systems prefer concepts with extensions enclosed by hyper-rectangles. We have shown in section 2.3 that this difference in bias can produce markedly different concept descriptions when the density of training examples in example space is low. Many applications use training sets which only sparsely populate the example space, and thus this difference between the learning paradigms is a significant point of comparison. The question of which bias is more appropriate is dependent on the particular target application domain; future work is required to enable a system to automatically judge the appropriateness of particular biases from examples and other knowledge about a domain.

Secondly, the computational mechanisms involved in exemplar-based and rule learning systems differ. In the rule learning paradigm, generalisations describing training data are enumerated before a run-time task is at hand, thus performing the bulk of the computation in a learning phase. In the exemplar-based paradigm, computation is deferred until a specific task is known, and can thus target effort for generalising towards the specific task at hand. Problem domains in which the exemplar-based approach may be advantageous are those where the concept to be learned has high polymorphy, and where the concept is sensitive to the addition of new training examples (see section 6).

Finally, throughout this paper we have argued that in order to meet the requirement of learning within finite resources and also some representational requirements (degree of concept membership, explanation), implementations of learning systems based on both paradigms have adopted features of the other and in so doing there is a partial convergence of such systems. By indicating these similarities, we hope to have encouraged the transfer of techniques for addressing computational issues between the two paradigms.

To summarise the resulting similarities we have identified:

- Exemplar systems use matching algorithms which provide a measure of concept membership of an example. Some current work on rule learning is examining the incorporation of a matcher producing measures of concept membership in the rule interpreter.

- Both general statements and examples can be effective for explanation purposes, and

both these explanation modalities are open for adoption by either representational scheme.

- Both schemes include summary information about examples, compiled before a performance task is at hand.
- Both schemes typically retain previously seen examples in memory, and both need to address the issue of which and how much information to discard. This issues has been particularly addressed by exemplar-based systems, and techniques developed.

Acknowledgements

I thank Claude Sammut, Dennis Kibler, Ray Bareiss, Tim Niblett, Spencer Star and David Aha who have given comments on earlier drafts of this paper. I particularly thank David Aha for the several discussions about and extensive comments on earlier drafts of this document.

References

[1] B. Arbab and D. Michie. Generating rules from examples. In *IJCAI-85*, pages 631–633, 1985.

[2] W. M. Bain. A case-based reasoning system for subjective assessment. In *AAAI-86*, pages 523–527, 1986.

[3] E. R. Bareiss and B. W. Porter. *A survey of psychological models of concept representation.* Technical Report AI86-50, The University of Texas at Austin, Computer Sciences Department, Austin, TX, 1987.

[4] E. R. Bareiss, B. W. Porter, and C. C. Wier. PROTOS: an exemplar-based learning apprentice. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, Kaufmann, Ca, 1987.

[5] R. A. Boswell. *An analytic survey of analytic concept-learning programs.* Working Paper 181, Dept. of AI, Edinburgh Univ., UK, 1985.

[6] G. Bradshaw. Learning about speech sounds: the nexus project. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, pages 1–11, Kaufmann, Ca, 1987.

[7] J. G. Carbonell. Learning by analogy : formulating and generalizing plans from past experience. In J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, editors, *Machine Learning, vol. 1*, Tioga, Palo Alto, Ca, 1983.

[8] B. Cestnik, I. Kononenko, and I. Bratko. Assistant 86: a knowledge-elicitation tool for sophisticated users. In I. Bratko and N. Lavrač, editors, *Progress in Machine Learning (proceedings of the 2nd European Working Session on Learning)*, pages 31–45, Sigma, Wilmslow, UK, 1987.

[9] T. G. Dietterich and R. S. Michalski. A comparative review of selected methods for learning from examples. In J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, editors, *Machine Learning, vol. 1*, pages 41–81, Tioga, Palo Alto, Ca, 1983.

[10] W. Emde. Big flood in the blocks world; or non-cumulative learning. In *ECAI-86*, pages 569–575, 1986.

[11] R. Fikes, P. Hart, and N. Nilsson. Learning and executing generalized robot plans. In B. Webber and N. Nilsson, editors, *Readings in AI*, pages 231–249, Tioga, Palo Alto, CA, 1981.

[12] R. Hall. *Learning by Failing to Explain.* Technical Report 906 (AI-TR-906), MIT AI Laboratory, 1986.

[13] K. Hammond. CHEF: a model of case-based planning. In *AAAI-86*, 1986.

[14] J. Holland. Escaping brittleness. In J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, editors, *Machine Learning, vol. 2*, Tioga, Tioga, 1986.

[15] R. Holte and R. Zimmer. Estimating the cost effectiveness of macro-operators. 1987. Brunel Univ., UK (to be published).

[16] D. Kibler and D. W. Aha. Learning representative exemplars of concepts: an initial case study. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, Kaufmann, Ca, 1987.

[17] J. J. Kolodner. Extending problem solver capabilities through case-based inference. In Ca, editor, *Proc. 4th International Workshop on Machine Learning*, pages 167–178, Kaufmann, Ca, 1987.

[18] J. J. Kolodner. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7:281–328, 1983.

[19] J. L. Kolodner, R. L. Simpson, and K. Sycara-Cyranski. A process model of case-based reasoning in problem solving. In *IJCAI-85*, 1985.

[20] J. E. Laird, P. S. Rosenbloom, and A. Newell. Chunking in soar: the anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1986.

[21] P. Langley. The emerging science of machine learning. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, pages v–vi, Kaufmann, Ca, 1987. (Preface to Proceedings).

[22] L. T. McCarty and N. S. Sridharan. *A Computational Theory of Legal Argument.* Technical Report LRP-TR-13, Rutgers Univ., Lab. for C.S. Research, NJ, 1982.

[23] J. McDermott. R1: a rule-based configurer of computer systems. *Artificial Intelligence*, 19(1):39–88, 1982.

[24] R. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system aq15 and its testing application to three medical domains. In *AAAI-86*, pages 1041–1045, Kaufmann, Ca, 1986.

[25] R. S. Michalski. How to learn imprecise concepts: a method for employing a two-tiered knowledge representation in learning. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, Kaufmann, Ca, 1987.

[26] R. S. Michalski. *Knowledge repair mechanisms: evolution vs. revolution.* ISG 85-15, Univ. of Illinois at Urbana-Champaign, Dept. of Computer Science, Urbana, 1985.

[27] R. S. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the 5th international symposium on Information Processing (FCIP 69), Vol. A3 (Switching circuits), Bled, Yugoslavia*, pages 125–128, 1969.

[28] R. S. Michalski and J. Larson. *Incremental Generation of VL_1 Hypotheses: the underlying Methodology and the Description of Program AQ11.* ISG 83-5, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Urbana, 1983.

[29] S. Minton. Quantitative results concerning the utility of explanation-based learning. In *AAAI-88*, pages 564–569, Kaufman, Ca, August 1988.

[30] S. Minton. Selectively generalizing plans for problem-solving. In *IJCAI-85*, pages 596–599, 1985.

[31] T. M. Mitchell. *Towards Combining Empirical and Analytic Methods for inferring Heuristics.* Technical Report LCSR-TR-27, Rutgers Univ., Lab. for Computer Science, 1982.

[32] I. Mozetic. The role of abstractions in learning qualitative models. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, Kaufmann, Ca, 1987.

[33] B. W. Porter and R. E. Bareiss. PROTOS: an experiment in knowledge acquisition for heuristic classification tasks. In *Proceedings of IMAL 1986*, Université de Paris-Sud, Orsay, France, 1986.

[34] J. R. Quinlan. Decision trees as probabilistic classifiers. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, Kaufmann, Ca, 1987.

[35] J. R. Quinlan. Discovering rules by induction from large collections of examples. In D. Michie, editor, *Expert Systems in the Micro-Electronic Age*, pages 168–201, Edinburgh Univ. Press, UK, 1979.

[36] J. R. Quinlan. Learning efficient classification procedures and their application to chess endgames. In J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, editors, *Machine Learning, vol. 1*, Tioga, Palo Alto, Ca, 1983.

[37] J. R. Quinlan, P. J. Compton, K. A. Horn, and L. Lazarus. Inductive knowledge acquisition: a case study. In *Applications of Expert Systems*, pages 157–173, Addison-Wesley, Wokingham, UK, 1987.

[38] A. Razzak, D. Michie, T. Hansan, and A. Ahmad. Case studies of building expert systems using extran. In *Artificial Intelligence and Advanced Computer Technology*, 1986.

[39] R. E. Reinke. *Knowledge acquisition and refinement tools for the ADVISE META-EXPERT system.* M. S. Thesis ISG 84-4, UIUCDCS-F-84-921, Univ. of Illinois at Urbana-Champaign, Dept. of Computer Science, Urbana, 1984.

[40] L. Rendell, R. Seshu, and D. Tcheng. More robust concept learning using dynamically-variable bias. In P. Langley, editor, *Proc. 4th International Workshop on Machine Learning*, pages 66–78, Kaufmann, Ca, 1987.

[41] E. Rosch and C. B. Mervis. Family resemblance studies in the internal structure of categories. *Cognitive Psychology*, 7:573–605, 1975.

[42] R. Schank. *Dynamic Memory*. Cambridge Univ. Press, 1982.

[43] A. D. Shapiro. *Structured Induction in Expert Systems*. Turing Inst. Press, in association with Addison-Wesley, Wokingham, UK, 1987.

[44] E. H. Shortliffe. *Computer-Based Medical Consultations: MYCIN*. American Elsevier, NY, 1976.

[45] E. E. Smith and D. L. Medin. *Categories and Concepts*. Harvard Univ., Cambridge, Ma, 1981.

[46] J. F. Sowa. *Conceptual structures : Information processing in mind and machine*. Addison Wesley, 1984.

[47] P. E. Utgoff. Id5: an incremental id3. In J. Laird, editor, *Proc. 5th Int. Conf. on Machine Learning*, pages 107–120, Kaufmann, Ca, June 1988.

[48] K. VanLehn. Learning a domain theory by completing explanations. In T. M. Mitchell, J. G. Carbonell, and R. S. Michalski, editors, *Machine Learning: A Guide to Current Research*, Kluwer, Lancaster, UK, 1986.

[49] P. H. Winston. Learning structural descriptions from examples. In P. H. Winston, editor, *The Psychology of Computer Vision*, McGraw-Hill, NY, 1975.