Selection of Relevant Features and Examples in Machine Learning

Avrim L. Blum (avrim@cs.cmu.edu)
School of Computer Science, Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3891

Pat Langley[◇] (langley@isle.org)
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, California 94306

Abstract

In this survey, we review work in machine learning on methods for handling data sets containing large amounts of irrelevant information. We focus on two key issues: the problem of selecting relevant features, and the problem of selecting relevant examples. We describe the advances that have been made on these topics in both empirical and theoretical work in machine learning, and we present a general framework that we use to compare different methods. We close with some challenges for future work in this area.

To appear in the special issue of *Artificial Intelligence* on 'Relevance', R. Greiner and D. Subramanian (Eds.).

[◇] Also affiliated with the Intelligent Systems Laboratory, Daimler-Benz Research and Technology Center, 1510 Page Mill Road, Palo Alto, CA 94304.

1. Introduction

As Machine Learning aims to address larger, more complex tasks, the problem of focusing on the most relevant information in a potentially overwhelming quantity of data has become increasingly important. For instance, data mining of corporate or scientific records often involves dealing with both many features and many examples, and the internet and World Wide Web have put a huge volume of low-quality information at the easy access of a learning system. Similar issues arise in the personalization of filtering systems for information retrieval, electronic mail, netnews, and the like.

In this paper, we address two specific aspects of this "focusing" task that have received significant attention in the AI literature: the problem of focusing on the most relevant features for use in representing the data, and the problem of selecting the most relevant examples to drive the learning process. We review recent work on these topics, presenting general frameworks that we use to compare and contrast different approaches.

We begin with the problem of focusing on relevant features. In Section 2 we present and relate several important notions of 'relevance' for this task and describe some general goals of feature selection algorithms. We report on methods that have been developed for this problem, characterizing them as 'embedded', 'filter', or 'wrapper' approaches, and we compare explicit feature selection techniques to those based on weighting schemes. We then turn (in Section 3) to the problem of focusing on relevant examples, describing methods for filtering both labeled and unlabeled data. We conclude (in Section 4) with open problems and challenges for future work, on both the empirical and theoretical fronts.

Before proceeding, we should clarify the scope of our survey, which focuses on methods and results from computational learning theory and experimental machine learning. There has been substantial work on feature selection in other fields such as pattern recognition and statistics, and on data selection in fields such as statistics, information theory, and the philosophy of science. Although we do not have the space to cover the work in these areas, readers should be aware that there are many similarities to the approaches that we will discuss.

2. The Problem of Irrelevant Features

At a conceptual level, one can divide the task of concept learning into two subtasks: deciding which features to use in describing the concept and deciding how to combine those features. In this view, the selection of relevant features, and the elimination of irrelevant ones, is one of the central problems in machine learning, and many induction algorithms incorporate some approach to addressing it.

At a practical level, we would like induction algorithms that scale well to domains with many irrelevant features. More specifically, as one goal we would like the number of training examples needed to reach a desired level of accuracy, often called the *sample complexity*, to grow slowly with the number of features present, if indeed not all these are needed to achieve good performance. For instance, it is not uncommon in a text classification task to represent examples using 10^4 to 10^7 attributes, with the expectation that only a small fraction of these are crucial (Lewis, 1992a; Lewis, 1992b). In recent years, a growing amount of work in machine learning – both experimental and theoretical in nature – has focused on developing algorithms with such desirable properties.

Induction algorithms differ considerably in their emphasis on focusing on relevant features. At one extreme lies the simple nearest neighbor method, which classifies test instances by retrieving the nearest stored training example, using all available attributes in its distance computations. Although Cover and Hart (1967) showed that this approach has excellent asymptotic accuracy, a little thought reveals that the presence of irrelevant attributes should considerably slow the rate of learning. In fact, Langley and Iba's (1993) average-case analysis of simple nearest neighbor indicates that number of training examples needed to reach a given accuracy (similar to the PAC notion of sample complexity) grows exponentially with the number of irrelevant attributes, even for conjunctive target concepts. Experimental studies of nearest neighbor (Aha, 1990; Langley & Sage, 1997) are consistent with this discouraging conclusion.

At the other extreme lie induction methods that explicitly attempt to select relevant features and reject irrelevant ones. Techniques for learning logical descriptions constitute the simplest example of this approach, and there are more sophisticated methods for identifying relevant attributes that can augment and improve any induction method, including nearest neighbor. Theoretical and experimental results for these methods are much more encouraging. For instance, theoretical results show that if, by focusing on only a small subset of features, an algorithm can significantly reduce the number of hypotheses under consideration, then there is a corresponding reduction in the sample size sufficient to guarantee good generalization (Blumer et al., 1987). Somewhat in the middle of the above two extremes are feature-weighting methods that do not explicitly select subsets of features, but still aim to achieve good scaling behavior.

We structure the remainder of this section as follows. We begin by describing several important formal notions of 'relevance' in the context of supervised learning. In addition to introducing terminology, these definitions help to illustrate some of the general goals of feature selection algorithms. We then turn to discussing some of the methods that have been developed for this problem, characterizing them as either 'embedded', 'filter', or 'wrapper' approaches, based on the relation between the selection scheme and the basic induction algorithm. This decomposition in part reflects historical trends, but it also helps for comparing approaches that may seem to be very different, but can be seen to belong to the same category and therefore in certain ways have similar motivations. We also compare explicit feature selection techniques to those based on weighting schemes, which tackle the same problem from a somewhat different perspective.

2.1 Definitions of 'Relevance'

There are a number of different definitions in the machine learning literature for what it means for features to be 'relevant'. The reason for this variety is that it generally depends on the question: "relevant to what?" More to the point, different definitions may be more appropriate depending on one's goals. Here, we describe several important definitions of relevance, and discuss their significance. In doing so, we hope to illustrate some of the issues involved and some of the variety of motivations and approaches taken in the literature.

For concreteness, let us consider a setting in which there are n *features* or *attributes* used to describe examples and each feature i has some domain F_i. For instance, a feature may be Boolean (`is_red?`), discrete with multiple values (`what_color?`), or continuous (`what_wavelength?`). An *example* is a point in the *instance space* $F_1 \times F_2 \times \ldots \times F_n$. The learning algorithm is given a set S of training data, where each data point is an example paired with an associated *label* or *classification* (which might also be Boolean, multiple valued, or continuous).

Although the learning algorithm sees only the fixed sample S, it is often helpful to postulate two additional quantities, as is done in the PAC learning model (e.g., see Kearns & Vazirani, 1994): a probability distribution D over the instance space, and a target function c from examples to labels. We then model the sample S as having been produced by repeatedly selecting examples from D and then labeling them according to the function c. The target function c may be deterministic or probabilistic: in the latter case, for some example A, $c(A)$ would be a probability distribution over labels rather than just a single label. Note that we can use the distribution D to model "integrity constraints" in the data. For instance, suppose we are representing a decimal digit by nine boolean features such that feature i is 1 if the digit is greater than or equal to i. We can model this by having D assign examples such as 101010101 the probability zero (even though the target function c is still defined on such examples).

Given this setup, perhaps the simplest notion of relevance is a notion of being "relevant to the target concept".

Definition 1 (Relevant to the target) *A feature x_i is relevant to a target concept c if there exists a pair of examples A and B in the instance space such that A and B differ only in their assignment to x_i and $c(A) \neq c(B)$.*

Another way of stating this definition is that feature x_i is relevant if there exists some example in the instance space for which twiddling the value of x_i affects the classification given by the target concept.

Notice that this notion has the drawback that the learning algorithm, given access to only the sample S, cannot necessarily determine whether or not some feature x_i is relevant. Even worse, if the encoding of features is redundant (say every feature is repeated twice), it may not even be possible to see two examples that differ in only one feature, since at least one of those examples would have probability zero under D. On the other hand, this is often the definition of choice for theoretical analyses of learning algorithms, where the notion of relevance is used to prove some convergence properties of an algorithm, rather than in the algorithm itself. The definition also is useful in situations where the target function c is a real object that the learning algorithm can actively query at inputs of its own choosing (e.g., if the learning algorithm is trying to reverse engineer some piece of hardware) rather than just a convenient fiction.

To remedy some of the drawbacks of the above definition, John, Kohavi, and Pfleger (1994) define two notions of what might be termed "relevance with respect to a distribution," which also has a nice interpretation as a notion of "relevance with respect to a sample".

Definition 2 (Strongly Relevant to the sample/distribution) *A feature x_i is strongly relevant to sample S if there exist examples A and B* in *S that differ only in their assignment to x_i and have different labels (or have different distributions of labels if they appear in S multiple times). Similarly, x_i is strongly relevant to target c and distribution D if there exist examples A and B having non-zero probability over D that differ only in their assignment to x_i and satisfy $c(A) \neq c(B)$.*

In other words, this is just like Definition 1 except A and B are now required to be in S (or have non-zero probability).

Definition 3 (Weakly Relevant to the sample/distribution) *A feature x_i is weakly relevant to sample S (or to target c and distribution D) if it is possible to remove a subset of the features so that x_i becomes strongly relevant.*

These notions of relevance are useful from the viewpoint of a learning algorithm attempting to decide which features to keep and which to ignore. Features that are strongly relevant are generally important to keep no matter what, at least in the sense that removing a strongly relevant feature adds ambiguity to the sample. Features that are weakly relevant may or may not be important to keep depending on which other features are ignored. In practice, one may wish to adjust these definitions to account for statistical variations. For instance, a special case of Definition 3 is that feature x_i is weakly relevant if it is correlated with the target function (i.e., x_i is strongly relevant when all other features are removed), so given a finite sample, one would want to account for variance and statistical significance.

In a somewhat different vein than the above definitions, in many cases rather than caring about exactly which features are relevant, we simply want to use relevance as a *measure of complexity*. That is, we want to use relevance to say how "complicated" a function is, and rather than requiring our algorithm to explicitly select a subset of features, we just want it to perform well when this quantity is low. For this purpose, another notion of relevance as a complexity measure with respect to a sample of data S and a set of concepts C is useful:

Definition 4 (Relevance as a complexity measure) *Given a sample of data S and a set of concepts C, let $r(S, C)$ be the number of features relevant using Definition 1 to a concept in C that, out of all those whose error over S is least, has the fewest relevant features.*

In other words, we are asking for the smallest number of features needed to achieve optimal performance over S via a concept in C. The reason for specifying the concept class C is that there may be a feature, such as a person's social-security number, that is highly relevant from the point of view of the *information* contained, but that is useless with respect to the sorts of concepts under consideration. For additional robustness, this definition is sometimes modified to allow concepts in C with "nearly" minimal error over S, if this produces a smaller relevant set.

The above notions of relevance are independent of the specific learning algorithm being used. There is no guarantee that just because a feature is relevant, it will necessarily be useful to an algorithm (or vice versa). Caruana and Freitag (1994b) make this explicit with a notion of what we might term "incremental usefulness" (and which they simply call "usefulness"):

Definition 5 (Incremental usefulness) *Given a sample of data S, a learning algorithm L, and a feature set $\mathcal{A}$, feature x_i is incrementally useful to L with respect to $\mathcal{A}$ if the accuracy of the hypothesis that L produces using the feature set $\{x_i\} \cup \mathcal{A}$ is better than the accuracy achieved using just the feature set $\mathcal{A}$.*

This notion is especially natural for feature-selection algorithms that search the space of feature subsets by incrementally adding or removing features to their current set — for instance, many that follow the general framework described in Section 2.2 below.

To make these definitions more clear, consider concepts that can be expressed as disjunctions of features (e.g., $x_1 \vee x_3 \vee x_7$), and suppose that the learning algorithm sees these five examples:

```
100000000000000000000000000000 +
111111111100000000000000000000 +
000000000011111111110000000000 +
000000000000000000001111111111 +
000000000000000000000000000000 -
```

The relevant features using Definition 1 would depend on the true target concept (though any consistent target disjunction c must include the first feature). Using Definitions 2 and 3, we would say that x_1 is strongly relevant and the rest are weakly relevant (note that x_2 is weakly relevant because it can be made strongly relevant by removing x_1 and $x_3, \ldots, x_{10}$). Using Definition 4 we would say simply that there are three relevant features ($r(S, C) = 3$), since this is the number of features relevant to the smallest consistent disjunction. The notion of incremental usefulness in Definition 5 depends on the learning algorithm but, presumably, given the feature set $\{1, 2\}$, the third feature would not be useful but any of features x_{11} to x_{30} would be. We will revisit the question of how Definition 5 is related to the others at the end of Section 2.2 when we discuss a simple specific algorithm.

There are a variety of natural extensions one can make to the above definitions. For instance, one can consider relevant *linear combinations* of features, rather than just relevant individual features. In this case, in analogy to Definition 4 above, one could ask: "What is the lowest-dimensional space such that projecting all the examples in S onto that space preserves the existence of a good function in the class C?" This notion of relevance is often most natural for statistical approaches to learning. Indeed, methods such as principal component analysis (Jolliffe, 1986) are commonly used as heuristics for finding these low-dimensional subspaces.

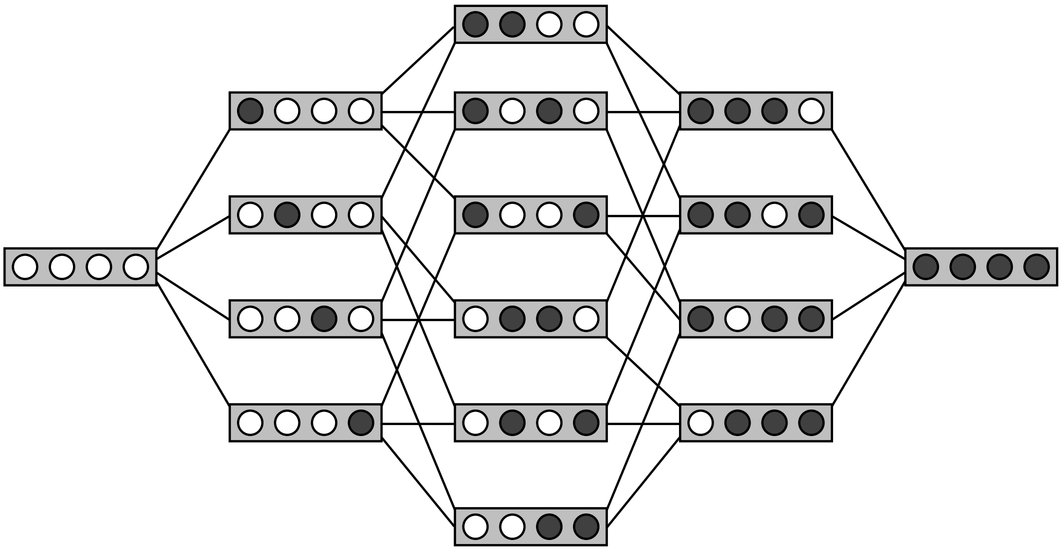

Figure 1. Each state in the space of feature subsets specifies the attributes to use during induction. Note that the states in the space (in this case involving four features) are partially ordered, with each of a state's children (to the right) including one more attribute (dark circles) than its parents.

2.2 Feature Selection as Heuristic Search

We now turn to discussing feature selection *algorithms* and, more generally, algorithms for dealing with data sets that contain large numbers of irrelevant attributes. A convenient paradigm for viewing many of these approaches (especially those that perform explicit feature selection) is that of heuristic search, with each state in the search space specifying a subset of the possible features. According to this view, we can characterize any feature selection method in terms of its stance on four basic issues that determine the nature of the heuristic search process.

First, one must determine the starting point (or points) in the space, which in turn influences the direction of search and the operators used to generate successor states. As Figure 1 depicts, there is a natural partial ordering on this space, with each child having exactly one more feature than its parents. This suggests that one might start with nothing and successively add attributes, or one might start with all attributes and successively remove them. The former approach is sometimes called *forward selection*, whereas the latter is known as *backward elimination*. One can also use variations on this partial ordering: Devijver and Kittler (1982) report an operator that adds k features and takes one away, and genetic operators like crossover produce somewhat different types of connectivity.

A second decision involves the organization of the search. Clearly, an exhaustive search of the space is impractical, as there exist 2^a possible subsets of a attributes. A more realistic approach relies on a greedy method to traverse the space. At each point in the search, one considers local changes to the current set of attributes, selects one, and then iterates. For instance, the hill-climbing approach known as *stepwise* selection or elimination considers both adding and removing features at each decision point, which lets one retract an earlier decision without keeping explicit track of the search path. Within these options, one can consider all states generated by the operators and then select the best, or one can simply choose the first state that improves accuracy over the current set. One can also replace the greedy scheme with more sophisticated methods, such as best-first search, which are more expensive but still tractable in some domains.

A third issue concerns the strategy used to evaluate alternative subsets of attributes. One commonly used metric involves an attribute's ability to discriminate among classes that occur in the training data.

Many induction algorithms incorporate a criterion based on information theory, but others directly measure accuracy on the training set or on a separate evaluation set. A broader issue concerns how the feature selection strategy interacts with the basic induction algorithm, as we discuss shortly in more detail.

Finally, one must decide on some criterion for halting the search. For example, one might stop adding or removing attributes when none of the alternatives improves the estimate of classification accuracy; one might continue to revise the feature set as long as accuracy does not degrade; or, one might continue generating candidate sets until reaching the other end of the search space and then select the best. One simple halting criterion is to stop when each combination of values for the selected attributes maps onto a single class value, but this assumes noise-free training data. A more robust alternative simply orders the features according to some relevancy score, then uses a system parameter to determine the break point.

Note that the above design decisions must be made for any induction algorithm that carries out feature selection. Thus, they provide useful dimensions for describing the techniques developed to address this problem, and we will refer to them repeatedly.

To make this more concrete, let us revisit the scenario given at the end of Section 2.1 (we are considering concepts expressible as a disjunction of Boolean features) with a simple strategy known as the *greedy set-cover* algorithm:

> Begin with a disjunction of zero features (which by convention outputs "negative" on every example). Then, out of those features not present in any *negative* example (and thus are "safe" to add into the hypothesis) choose the one whose inclusion into the current hypothesis most increases the number of correctly classified *positive* examples (breaking ties arbitrarily). Repeat until there are no more "safe" features that would increase the number of correctly classified positives, and then halt.

With respect to our framework, this algorithm begins at the leftmost point in Figure 1, incrementally moves rightward only, evaluates subsets based on performance on the training set with an infinite penalty for misclassifying negative examples, and halts when it can take no further step that strictly improves its evaluated performance.

Given the five data points listed at the end of Section 2.1, this algorithm would first put in x_1, then perhaps x_{11}, then perhaps x_{21}, and then would halt. It is not hard to see that if there exists a disjunction consistent with the training set, then this method will find one. In fact, the number of features selected by this method is at most $O(\log |S|)$ times larger than the number of relevant features using Definition 4 (Johnson, 1974; Haussler, 1986).[1]

We can also use this algorithm to illustrate relationships between some of the definitions in the previous section. For instance, the incrementally useful features for this algorithm (Definition 5) will also be weakly relevant (Definition 3), but the converse is not necessarily true. In fact, if the data is not consistent with *any* disjunction, then even strongly relevant features (Definition 2) may be ignored by the algorithm due to the algorithm's conservative nature (it ignores any feature that may cause it to misclassify a negative example). On the other hand, if the data *is* consistent with some disjunction, then all strongly relevant features *are* incrementally useful (and all will eventually be placed in the algorithm's hypothesis), though the algorithm may prefer a weakly relevant feature to a strongly relevant one due to its evaluation criterion.

We now review some specific feature selection methods, which we have grouped into three classes: those that *embed* the selection within the basic induction algorithm, those that use feature selection to *filter* features passed to induction, and those that treat feature selection as a *wrapper* around the induction process.

1. This is not too hard to see, and follows from the fact that there must always exist some feature to add that captures at least a $1/r(S,C)$ fraction of the still-misclassified positive examples. In the other direction, finding the *smallest* disjunction consistent with a given set of data is NP-hard (Garey & Johnson, 1979); a polynomial-time algorithm to find disjunctions only $c \log n$ times larger than the smallest for $c < 1/4$ would place NP into quasi-polynomial time (Lund & Yannakakis, 1993).

2.3 Embedded Approaches to Feature Selection

Methods for inducing logical descriptions provide the clearest example of feature selection methods embedded within a basic induction algorithm. In fact, many algorithms for inducing logical conjunctions (e.g., Mitchell, 1982; Vere, 1975; Winston, 1975; and the greedy set-cover algorithm given above) do little more than add or remove features from the concept description in response to prediction errors on new instances. For these methods, the partial ordering in Figure 1 also describes the space of hypotheses, and the algorithms typically use this ordering to organize their search for concept descriptions.

Theoretical results for learning pure conjunctive (or pure disjunctive) concepts are encouraging. As mentioned above, the greedy set-cover approach finds a hypothesis at most a logarithmic factor larger than the smallest possible. In fact, Warmuth (personal communication) notes that one can achieve slightly better bounds in the PAC setting by halting earlier so that some training examples are misclassified. Because the resulting hypothesis is guaranteed to be fairly small, the sample complexity grows only logarithmically with the number of irrelevant features. These results apply directly to other settings in which the target concept can be characterized as a conjunction (or disjunction) of a list of functions produced by the induction algorithm. Situations of this form include learning intersections of halfspaces in constant-dimensional spaces (Blumer et al., 1989), and algorithms for learning DNF formulas in $n^{O(\log n)}$ time under the uniform distribution (Verbeurgt, 1990). The above results for the greedy set-cover method are distribution free and worst case, but Pazzani and Sarrett (1992) report an average-case analysis of even simpler methods for conjunctive learning that imply logarithmic growth for certain product distributions.

Similar operations for adding and removing features form the core of methods for inducing more complex logical concepts, but these methods also involve routines for combining features into richer descriptions. For example, recursive partitioning methods for induction, such as Quinlan's ID3 (1983) and C4.5 (1993), and CART (Breiman et al. 1984), carry out a greedy search through the space of decision trees, at each stage using an evaluation function to select the attribute that has the best ability to discriminate among the classes. They partition the training data based on this attribute and repeat the process on each subset, extending the tree downward until no further discrimination is possible.

Dhagat and Hellerstein (1994) have also extended techniques for greedy set cover in a recursive fashion to apply to more complex functions such as k-term DNF formulas and k-alternation decision lists. Blum (1992) describes methods that can be used even when the set of all attributes is unbounded, so long as each individual example satisfies a reasonably small number of them; this is often a good model when dealing with text documents, for instance, that may each contain only a small number of the possible words in the dictionary. For all these cases, the feature-selection process is clearly embedded within another, more complex algorithm.

Separate-and-conquer methods for learning decision lists (Michalski, 1980; Clark & Niblett, 1989; Pagallo & Haussler, 1990) embed feature selection in a similar manner. These techniques use an evaluation function to select a feature that helps distinguish a class C from others, then add the resulting test to a single conjunctive rule for C. They repeat this process until the rule excludes all members of other classes, then remove the members of C that the rule covers and repeat the process on the remaining training cases.

Clearly, both partitioning and separate-and-conquer methods explicitly select features for inclusion in a branch or rule, in preference to other features that appear less relevant or irrelevant. For this reason, one might expect them to scale well to domains that involve many irrelevant features. Although few theoretical results exist for these methods, experimental studies by Langley and Sage (1997) suggest that decision-tree methods scale linearly with the number of irrelevant features for certain target concepts, such as logical conjunctions. However, the same studies also show that, for other targets concepts, they exhibit the same exponential growth as does nearest neighbor. Experiments by Almuallim and Dietterich (1991) and by Kira and Rendell (1992) also show substantial decreases in accuracy, for a given sample size, when irrelevant features are introduced into selected Boolean target concepts.

The standard explanation of this effect involves the reliance of such algorithms on greedy selection of attributes to discriminate among classes. This approach works well in domains where there is little interaction among the relevant attributes, as in conjunctive concepts. However, the presence of attribute interactions, which can lead a relevant feature in isolation to look no more discriminating than an irrelevant one, can cause significant problems for this scheme. Parity concepts constitute the most extreme example of this situation, but it also arises with other target concepts.[2]

Some researchers have attempted to remedy these problems by replacing greedy search with lookahead techniques (e.g., Norton, 1989), with some success. Of course, more extensive search carries with it a significant increase in computational cost. Others have responded by selectively defining new features as combinations of existing ones, so as to make greedy search more powerful by letting it take larger steps (e.g., Matheus & Rendell, 1989; Pagallo & Haussler, 1990). However, neither approach has been directly evaluated in terms of its ability to handle large numbers of irrelevant features, either experimentally or theoretically.

2.4 Filter Approaches to Feature Selection

A second general approach to feature selection introduces a separate process for this purpose that occurs before the basic induction step. For this reason, John, Kohavi, and Pfleger (1994) have termed them *filter* methods, because they filter out irrelevant attributes before induction occurs. The preprocessing step uses general characteristics of the training set to select some features and exclude others. Thus, filtering methods are independent of the induction algorithm that will use their output, and they can be combined with any such method.

Perhaps the simplest filtering scheme is to evaluate each feature individually based on its correlation with the target function (e.g., using a mutual information measure) and then to select the k features with the highest value. The best choice of k can then be determined by testing on a holdout set. This method is commonly used in text categorization tasks (Lewis, 1992a; Lewis, 1992b), often in combination with either a "naive Bayes" or a nearest neighbor classification scheme, and has achieved good empirical success.

Kira and Rendell's (1992) RELIEF algorithm follows this general paradigm but incorporates a more complex feature-evaluation function. Their system then uses ID3 to induce a decision tree from the training data using only the selected features. Kononenko (1994) reports two extensions to this method that handle more general types of features.

Almuallim and Dietterich (1991) describe a filtering approach to feature selection that involves a greater degree of search through the feature space. Their FOCUS algorithm looks for minimal combinations of attributes that perfectly discriminate among the classes. This method begins by looking at each feature in isolation, then turns to pairs of features, triples, and so forth, halting only when it finds a combination that generates pure partitions of the training set (i.e., in which no instances have different classes). FOCUS then passes on the original training examples, described using only the selected features, to an algorithm for decision-tree induction.

Comparative studies with a regular decision-tree method showed that, for a given number of training cases on random Boolean target concepts, FOCUS was almost unaffected by the introduction of irrelevant attributes, whereas decision-tree accuracy degraded significantly. Schlimmer (1993) describes a related method that

2. Note that this problem does not disappear with increasing sample size. Embedded selection methods that rely on greedy search cannot distinguish between relevant and irrelevant features early in the search process even when the entire instance space is available.

Table 1. Characterization of recent work on filter approaches to feature selection in terms of heuristic search through the space of feature sets.

AUTHORS (SYSTEM)	STARTING POINT	SEARCH CONTROL	HALTING CRITERION	INDUCTION ALGORITHM
ALMUALLIM (FOCUS)	NONE	BREADTH FIRST	CONSISTENCY	DEC. TREE
CARDIE	NONE	GREEDY	CONSISTENCY	NEAR. NEIGH.
KOLLER/SAHAMI	ALL	GREEDY	THRESHOLD	TREE/BAYES
KIRA/RENDELL (RELIEF)	—	ORDERING	THRESHOLD	DEC. TREE
KUBAT ET AL.	NONE	GREEDY	CONSISTENCY	NAIVE BAYES
SCHLIMMER	NONE	SYSTEMATIC	CONSISTENCY	NONE
SINGH/PROVAN	NONE	GREEDY	NO INFO. GAIN	BAYES NET

carries out systematic search (to avoid revisiting states) through the space of feature sets, again starting with the empty set and adding features until it finds a combination consistent with the training data.

Although FOCUS and RELIEF follow feature selection with decision-tree construction, one can of course use other induction methods. For instance, Cardie (1993) uses filtering as a preprocessor for nearest neighbor retrieval, and Kubat, Flotzinger, and Pfurtscheller (1993) filter features for use with a naive Bayesian classifier. Interestingly, both used a decision-tree method that relies on an embedded selection scheme as the filter to produce a reduced set of attributes. More recently, Singh and Provan (1996) have used information-theoretic metrics to filter features for inclusion in a Bayesian network, while Koller and Sahami (1996) have employed a cross-entropy measure, designed to find 'Markov blankets' of features, for use in both naive Bayes and decision-tree induction. In a somewhat different vein, Greiner, Grove, and Kogan, in this issue, consider settings where a helpful tutor filters out conditionally irrelevent attributes.

Table 1 characterizes the recent work on filter methods in terms of the dimensions described earlier in the section, along with the induction algorithm that takes advantage of the reduced feature set. The typical results show some improvement over embedded selection methods. Most experiments have focused on natural domains that contain an unknown number of irrelevant features, but a few researchers (Almuallim & Dietterich, 1990; Kira & Rendell, 1992) have studied experimentally the effect of introducing such features.

Another class of filter methods actually constructs higher-order features from the original ones, orders them in terms of the variance they explain, and selects the best such features. The statistical technique of principal components analysis (Jolliffe, 1986), the best-known example of this approach, generates linear combinations of features whose vectors are orthogonal in the original space. Empirically, principal components has successfully reduced dimensionality on a variety of learning tasks. Blum and Kannan (1993) describe theoretical guarantees for methods of this form, when the target function is an intersection of halfspaces and the examples are chosen from a sufficiently benign distribution. The related method of independent component analysis (Comon, 1994) incorporates similar ideas, but insists only that the new features be independent rather than orthogonal.

2.5 Wrapper Approaches to Feature Selection

A third generic approach for feature selection also occurs outside the basic induction method but uses that method as a subroutine, rather than as a postprocessor. For this reason, John et al. (1994) refer to these as *wrapper* approaches (see, also, the paper by Kohavi and John in this issue). The typical wrapper algorithm searches the same space of feature subsets (see Figure 1) as embedded and filter methods, but it evaluates alternative sets by running some induction algorithm on the training data and using the estimated accuracy of the resulting classifier as its metric.[3] Actually, the wrapper scheme has a long history within the literature on statistics and pattern recognition (e.g., Devijver & Kittler, 1982), where the problem of feature selection has long been an active research topic, but its use within machine learning is relatively recent.

The general argument for wrapper approaches is that the induction method that will use the feature subset should provide a better estimate of accuracy than a separate measure that may have an entirely different inductive bias. For example, both Doak (1992) and John et al. (1994) argue in favor of using a wrapper method to improve the behavior of decision-tree induction. Doak reports experimental comparisons of forward selection and backward elimination, as well as the impact of different search-control techniques. John et al. present similar comparative studies, including the effect of using wrappers versus filters. Caruana and Freitag (1994a) report a third set of empirical studies, also focusing on decision trees, that explore variations on wrapper methods.

The major disadvantage of wrapper methods over filter methods is the former's computational cost, which results from calling the induction algorithm for each feature set considered. This cost has led some researchers to invent ingenious techniques for speeding the evaluation process. In particular, Caruana and Freitag describe a scheme for caching decision trees that lets their algorithms search larger spaces in reasonable time. Moore and Lee (1994) describe an alternative scheme that instead speeds feature selection by reducing the percentage of training cases used during evaluation.

Certainly not all work within the wrapper framework has focused on decision-tree induction. Indeed, one might expect methods like nearest-neighbor, which by default take into account all attributes, would benefit more from feature-selection wrappers than algorithms that themselves incorporate embedded schemes. This expectation has led to a substantial body of work on wrapper methods for nearest-neighbor and case-based learning.

Let us consider one such approach and its behavior in some detail. Langley and Sage's (1994a) OBLIVION algorithm combines the wrapper idea with the simple nearest neighbor method, which assigns to new instances the class of the nearest case stored in memory during learning. The feature-selection process effectively alters the distance metric used in these decisions, taking into account the features judged relevant and ignoring the others.

OBLIVION carries out a backward elimination search through the space of feature sets, starting with all features and iteratively removing the one that leads to the greatest improvement in estimated accuracy. The system continues this process until the estimated accuracy actually declines. We characterize OBLIVION as using a wrapper method because its evaluation metric involves running nearest neighbor itself on the training data to measure the accuracy with alternative feature sets. In particular, the system uses leave-one-out cross validation to estimate the accuracy of each feature set on novel test cases.

Although this approach may seem computationally expensive, OBLIVION uses an insight from Moore and Lee (1994) to make it tractable.[4] The leave-one-out technique estimates accuracy on N training cases by holding out each case in turn, constructing a classifier based on the remaining $N-1$ cases, seeing whether the classifier correctly predicts the case, and averaging the results over all N cases. Because nearest neighbor simply stores the training cases in memory, one can implement leave one out by successively removing each case and using the remaining ones to classify it. This scheme is no more expensive than estimating accuracy on the training set itself.

3. One natural metric involves running the induction algorithm over the entire training data using a given set of features, then measuring the accuracy of the learned structure on the training data. However, John et al. argue convincingly that a cross-validation method provides a better measure of expected accuracy on novel test cases.
4. Kohavi (1995) has incorporated the same idea into his technique for inducing decision tables, which has many similarities to OBLIVION.

Table 2. Characterization of recent work on wrapper approaches to feature selection in terms of heuristic search through the space of feature sets.

Authors (System)	Starting Point	Search Control	Halting Criterion	Induction Algorithm
Aha/Bankert (Beam)	Random	Comparison	No Better	Near. Neigh.
Caruana/Freitag (CAP)	Comparison	Greedy	All Used	Dec. Tree
Doak	Comparison	Comparison	Not Enough Better	Tree/Bayes
John/Kohavi/Pfleger	Comparison	Greedy	No Better	Dec. Tree
Langley/Sage (Oblivion)	All	Greedy	Worse	Near. Neigh.
Langley/Sage (Sel. Bayes)	None	Greedy	Worse	Naive Bayes
Moore/Lee (Race)	Comparison	Greedy	No Better	Near. Neigh.
Singh/Provan (K2-AS)	None	Greedy	Worse	Bayes Net
Skalak	Random	Mutation	Enough Times	Near. Neigh.
Townsend-Weber/Kibler	All	Comparison	No Better	Near. Neigh.

Langley and Sage designed a number of experiments to evaluate their system. Results with synthetic domains suggest that, when some features are irrelevant, Oblivion learns high-accuracy classifiers from many fewer instances than simple nearest neighbor. However, they also found that this effect was absent from many of the UCI data sets, suggesting that Holte's (1993) finding about the accuracy of one-level decision trees was due to highly correlated features (which cause no difficulty for nearest neighbor) rather than completely irrelevant ones. Oblivion did fare significantly better on classifying chess end games and predicting a word's semantic class, giving evidence that these domains do contain irrelevant features.

Other researchers have also developed wrapper methods for use with nearest neighbor. For instance, Aha and Bankert (1996) report an a technique much like Oblivion, but their system starts with a randomly selected subset of features and includes an option for beam search rather than greedy decisions. They report impressive improvements on a cloud classification task that involves over 200 numeric features. Skalak's (1994) work on feature selection for nearest neighbor also starts with a random feature set, but replaces greedy search with random hill climbing that continues for a specified number of cycles.

Most research on wrapper methods has focused on classification, but both Moore and Lee (1994) and Townsend-Weber and Kibler (1994) combine this idea with k-nearest neighbor for numeric prediction. Also, most work has emphasized the advantages of feature selection for induction methods that are highly sensitive to irrelevant features. However, Langley and Sage (1994b) have shown that the naive Bayesian classifier, which is sensitive to *redundant* features, can benefit from the same basic approach (as did Doak's earlier work). Singh and Provan (1995) have extended this idea to learning more complex Bayesian networks. This suggests that techniques for feature selection can improve the behavior of induction algorithms in a variety of situations, not only in the presence of irrelevant attributes. As Caruana and Freitag (1994b) argue, most methods for feature selection focus on finding attributes that are *useful* for performance (in the sense of Definition 5), rather than necessarily finding the relevant ones.

Table 2 characterizes the recent efforts on wrapper methods in terms of the dimensions discussed earlier, as well as the induction method used in each case to direct the search process. The table shows the diversity of techniques that researchers have developed, and the heavy reliance on the experimental comparison of variant methods. Unfortunately, few of these experiments directly study the algorithms' ability to deal with increasing numbers of irrelevant features, and few theoretical results are available for them.

2.6 Feature Weighting Methods

So far, we have discussed algorithms that explicitly attempt to select a "most relevant" subset of features. However, another approach, especially for embedded algorithms, is to apply a weighting function to features, in effect assigning them *degrees* of perceived relevance. We have separated this from the explicit feature selection approach because the motivations and uses for these two methods tend to be different. Explicit feature selection is generally most natural when the result is intended to be understood by humans, or fed into another algorithm. Weighting schemes tend to be easier to implement in on-line incremental settings, and are generally more purely motivated by efficiency considerations.

Weighting schemes can be characterized in terms of heuristic search, as we viewed explicit feature-selection methods. However, because the weight space lacks the partial ordering of feature sets, most approaches to feature weighting rely on quite different forms of search. For instance, the most common techniques involve some form of gradient descent, in which successive passes through the training instances lead to iterative changes in all weights.

Perhaps the best-known attribute-weighting method is the perceptron updating rule (Minsky & Papert, 1969), which adds or subtracts weights on a linear threshold unit in response to errors on training instances. The least-mean squares algorithm (Widrow & Hoff, 1960) for linear units and backpropagation (Rumelhart, Hinton, & Williams, 1986), its generalization for multilayer neural networks, also make additive changes to a set of weights in order to reduce error on the training set.[5] Baluja and Pomerleau, in this issue, discuss using a neural-network approach in domains where the degree of feature relevance can vary over time.

Perceptron weighting techniques can have difficulty in settings dominated by truly irrelevant features (for instance, see the paper by Kivinen, Warmuth, and Auer in this issue). In response, Littlestone (1988) developed WINNOW, an algorithm that updates weights in a multiplicative manner, rather than additively as in the perceptron rule. Littlestone showed that, on any on-line stream of data consistent with a disjunction of r features, WINNOW makes at most $O(r \log n)$ mistakes. (This effectively uses the notion of relevance given in Definition 4.) Thus, its behavior degrades only logarithmically with the number of features that are irrelevant to the target concept. More generally, WINNOW achieves this logarithmic degradation for concept classes such as conjunctions, k-DNF formulas, and linear threshold functions with good separation between positive and negative examples.

For concreteness, we present a version of the WINNOW algorithm for the disjunction-learning scenario discussed in Sections 2.1 and 2.2, along with a proof of Littlestone's theorem:

The WINNOW algorithm (a simple version)

1. Initialize the weights $w_1, \ldots, w_n$ of the features to 1.
2. Given an example $(x_1, \ldots, x_n)$, output 1 if $w_1 x_1 + \ldots + w_n x_n \geq n$, and output 0 otherwise.
3. If the algorithm makes a mistake:
   (a) If the algorithm predicts negative on a positive example, then for each x_i equal to 1, double the value of w_i.
   (b) If the algorithm predicts positive on a negative example, then for each x_i equal to 1, cut the value of w_i in half.
4. Go to 2.

5. While most work on embedded weighting schemes has a neural-network flavor, Aha (1990) reports an error-driven method, embedded within a nearest-neighbor learner, that modifies its distance metric by altering weights.

Theorem 1 WINNOW *makes at most* $2 + 3r(1 + \lg n)$ *mistakes on any sequence of examples consistent with a disjunction of* r *features.*

Proof. Let us first bound the number of mistakes that will be made on positive examples. Any mistake made on a positive example must double at least one of the weights in the target function (the *relevant* weights), and a mistake made on a negative example will *not* halve any of these weights, by definition of a disjunction. Furthermore, each of relevant weights can be doubled at most $1 + \lg n$ times, since only weights less than n can ever be doubled. Therefore, WINNOW makes at most $r(1 + \lg n)$ mistakes on positive examples.

Now we bound the number of mistakes made on negative examples. The total weight summed over all features is initially n. Each mistake made on a positive example increases the total weight by at most n (since before doubling, we must have had $w_1x_1 + \ldots w_nx_n < n$). On the other hand, each mistake made on a negative example decreases the total weight by at least $n/2$ (since before halving, we must have had $w_1x_1 + \ldots + w_nx_n \geq n$). The total weight never drops below zero. Therefore, the number of mistakes made on negative examples is at most twice the number made on positive examples, plus 2; that is, $2 + 2r(1 + \lg n)$. Adding this to the bound on the number of mistakes on positive examples yields the theorem. ■

The same general approach of WINNOW has been used in algorithms developed by Littlestone and Warmuth (1994), Vovk (1990), Littlestone, Long, and Warmuth (1991), and Cesa-Bianchi et al. (1993). Kivinen and Warmuth (1995) describe relations between these approaches and additive updating methods such as the least mean squares algorithm. In fact, these multiplicative updating schemes are very similar to the kind of multiplicative probability updates that occur in Bayesian methods, and several of the results provide bounds on the performance of Bayesian updating, even when the probabilistic assumptions of that approach are not met. Experimental tests of WINNOW and related multiplicative methods on natural domains have revealed good behavior (Armstrong et al., 1995; Blum, 1995), and studies with synthetic data show that they scale very well to domains with even thousands of irrelevant features (Littlestone & Mesterharm, 1997).

More generally, weighting methods are often cast as ways of merging advice from different knowledge sources that may themselves be generated through learning. In this light, the weighting process plays an interesting dual role with respect to the filter methods discussed earlier. Filter approaches pass their output (a set of selected features) to a black-box learning algorithm, whereas weighting approaches can take as input the classifiers generated by black-box learning algorithms and determine the best way to combine their predictions.

On the other hand, direct analogs to the filter and wrapper approaches do exist for determining weights. Stanfill (1987) and Ting (1994) describe filter-like methods that use conditional probability distributions to weight attributes for nearest neighbor. Daelemans et al. (1994) present a different weighting scheme that normalizes features based on an information-theoretic metric, and one could use the scores produced by RELIEF (Kira & Rendell, 1992) to the same end. Finally, Kohavi, Langley, and Yun (1997) have adapted the wrapper method to search through a discretized weight space that can be explored in much the same way as feature sets. Each of these approaches shows improvement over use of all features, but only the latter reports comparisons with a simple selection of attributes.

3. The Problem of Irrelevant Examples

Just as some attributes are more useful than others, so may some *examples* better aid the learning process than others. This suggests a second broad type of relevance that concerns the examples themselves, and here we briefly consider techniques for their selection. Some work has assumed the presence of a benevolent tutor who gives informative instances, such as near misses, or provides ideal training sequences (Winston, 1975). However, a more robust approach involves letting the learner select or focus on training cases by itself.

Researchers have proposed at least three reasons for selecting examples used during learning. One is if the learning algorithm is computationally intensive; in this case, if sufficient training data is available, it makes sense to learn only from some examples for purposes of computational efficiency. Another reason is if the cost of labeling is high (e.g., when labels must be obtained from experts) but many unlabeled examples are available or are easy to generate. Yet a third reason for example selection is to increase the rate of learning by focusing attention on informative examples, thus aiding search through the space of hypotheses. Here we should distinguish between examples that are relevant from the viewpoint of *information* and ones that are relevant from the viewpoint of one's algorithm. Most work emphasizes the latter, though information-based measures are sometimes used for this purpose.

As with feature-selection schemes, we can separate example-selection methods into those that embed the selection process within the learning algorithm, those that filter examples before passing them to the induction process, and those that wrap example selection around successive calls to the learning technique. Although we will refer to this dimension below, we will instead organize the section around another distinction: between methods that select relevant examples from *labeled* training instances and ones that select from *unlabeled* instances.

3.1 Selecting Labeled Data

The first generic approach assumes that a set of labeled training data is available for use by the learning system, but that not all of these examples are equally useful. As we noted above, one can embed the process of example selection within the basic learning algorithm, and many simple induction schemes take this approach. For instance, the perceptron algorithm, edited nearest neighbor methods, and some incremental conjunctive methods only learn from an example when their current hypothesis misclassifies it. Such embedded methods, sometimes called *conservative* algorithms, ignore all examples on which their hypothesis is correct.[6]

If one assumes that training data and test data are both taken from a single fixed distribution, then one can guarantee that with high probability, the data used for training will overall be relevant to the success criteria used for testing (Blumer et al., 1989). As learning progresses, however, the learner's knowledge about certain parts of the input space increases, and examples in the "well-understood" portion of the space become less useful. For instance, when a conservative algorithm has a 20% error rate, it will ignore 80% of the training cases, and when it achieves 10% error, it will ignore 90% of the data.

In the PAC model, learning algorithms need to roughly double the number of examples seen in order to halve their error rate (Schapire, 1990; Freund, 1992; Blumer et al., 1989). However, for conservative algorithms, since the number of examples actually used for learning is proportional to the error rate, the number of new examples used by the algorithm each time it wishes to halve its error rate remains (roughly) constant. Thus, the number of examples actually used to achieve some error rate ϵ is really just logarithmic in $1/\epsilon$ rather than linear.

Although this result holds only for conservative algorithms that embed the example selection process within learning, one can use explicit example selection to achieve similar effects for other induction methods. In particular, Schapire (1990) describes a wrapper method called *boosting* that takes a generic learning algorithm and adjusts the distribution given to it (by removing some training data) based on the algorithm's behavior. The basic idea is that, as learning progresses, the booster samples the input distribution to keep the accuracy of the learner's current hypothesis near to that of random guessing. As a result, the learning process focuses on the currently hard data. Schapire has shown that boosting lets one achieve the logarithmic use of examples described above under quite general conditions, and Freund (1990, 1992) has further improved

6. Littlestone and Mesterharm (1997) have shown that a variant of naive Bayes that learns only from errors can deal better with irrelevant features than the standard version, which updates its statistics on each example. This shows there exist interactions between the problems of feature selection and example selection.

on this technique. On the experimental front, Drucker et al. (1992, 1994) have shown that boosting can improve the accuracy of neural network methods on tasks involving optical character recognition. This approach seems especially appropriate for techniques like backpropagation, for which training is much more expensive than prediction.[7]

Another class of wrapper methods for example selection originated in the experimental study of decision-tree induction. Quinlan (1983) reports a *windowing* technique designed to reduce the time needed to construct decision tress from very large training sets. Windowing selects a random sample of the training data to induce an initial decision tree, then uses that tree to classify all the remaining examples. From the misclassified cases, the method selects another random set to augment the original sample, constructs a new decision tree, and so forth, repeating the process until it has a tree that correctly classifies all of the training data. Quinlan reports that windowing led to substantial reduction in processing time on a large collection of chess endgames, and Catlett (1992) describes another wrapper method called *peepholing* designed for even larger training sets. John and Langley (1996) report a much simpler use of wrappers to determine the proper size of a randomly selected training sample.

Lewis and Catlett (1994) describe a filter approach to selection of labeled data, but such techniques are less commmon in the machine learning literature than embedded or wrapper methods. One can imagine simple techniques for cleaning training data, say by removing inconsistent examples that are identical except for their class, but such methods are not widely used. One-pass sampling of the training data would also constitute filtering, but again research has leaned towards iterative versions of sampling like those in boosting and windowing.

3.2 Selecting Unlabeled Data

The learner can also select data even before it has been labeled. This can be useful in scenarios where unlabeled data is plentiful, but where the labeling process is expensive. One generic approach to this problem, which can be embedded within an induction algorithm that maintains a set of hypotheses consistent with the training data, is called *query by committee* (Seung et al., 1992). Given an unlabeled instance, the method selects two hypotheses at random from the consistent set and, if they make different predictions, requests the label for the instance. The basic idea is that informative or relevant examples are more likely to pass the test than those that most hypotheses classify the same way. Unfortunately, to obtain theoretical results for query by committee requires much stronger constraints on the space of hypotheses than does boosting. Specifically, this method requires an ability to sample random consistent hypotheses, which can be quite difficult, although it is also a major topic of algorithmic research (e.g., Sinclair & Jerrum, 1989; Dyer, Frieze, & Kannan, 1989; and Lovasz & Simonovits, 1992).

There has been a larger body of work on algorithms that generate examples of their own choosing, under the heading of *membership query* algorithms within the theoretical community and *experimentation* within the empirical community. A common technique used by algorithms of this sort is to take a known example and slightly alter its feature values to determine the effect on its classification. For instance, one might take two examples with different labels and then "walk" them towards each other to determine at what point the desired classification changes (this, in turn, is often used to determine *relevant features*, tying in with our earlier discussion). Another class of methods effectively designs critical experiments to distinguish among competing hypotheses, letting them eliminate competitors and thus reduce the complexity of the learning task. Mitchell (1982) suggested an information-theoretic approach to example selection, whereas Sammut

7. Although boosting has clear empirical uses, it was originally developed for the theoretical goal of showing that "weak learning implies strong learning" in the PAC model. In other words, if one has an algorithm that will perform somewhat better than guessing over *every* distribution, then there cannot be a hard "core" to the function being learned, and one can boost performance to produce high-quality predictions.

and Banerji (1986) and Gross (1991) used less formal methods but demonstrated their advantage empirically. More recently, work on 'active learning' has continued this tradition; for instance, Cohn, Ghahramani, and Jordan (1996) report successful results with a system that selects examples designed to reduce the learner's variance. In parallel, theoretical researchers (Angluin, 1987; Angluin et al., 1993; Bshouty, 1993; Rivest & Schapire, 1993; Jackson, 1994) have shown that the ability to generate queries greatly enlarges the types of concept classes for which one can guarantee polynomial-time learning.

Although much work on queries and experimentation has emphasized simple classification learning, other efforts have addressed more complex learning tasks. For example, Knobe and Knobe (1977) let their grammar-induction system query an oracle about the legality of candidate strings to distinguish among competing hypotheses, and Kulkarni and Simon's (1990) Kekada and Rajamoney's (1990) Coast design critical experiments to distinguish among competing hypotheses in scientific domains. Finally, both Shen and Simon (1992) and Gil (1993) have explored the uses of experimentation in learning action models for planning tasks.

Other learning systems incorporate strategies for exploring portions of the instance space that have not yet been encountered to obtain more representative information about the domain. For example, Scott and Markovitch (1991) adapt this idea to unsupervised learning situations, and many methods for reinforcement learning include a bias toward exploring unfamiliar parts of the state space (e.g., Lin, 1992). Both approaches can considerably increase learning rates over random presentations.

Most work on selecting and querying unlabeled data has used embedded methods, but Angluin et al. (1993) and Blum et al. (1995) describe theoretical results for a wrapper query method that can be applied to any algorithm. Specifically, they show that when membership queries are available, any algorithm with a polynomial mistake bound for learning a "reasonable" concept class can be converted in an automated way into one in which the number of mistakes plus queries has only a logarithmic dependence on the number of irrelevant features present. The basic idea is to gradually grow a set of features known to be relevant, and whenever the algorithm makes a mistake, to use queries to determine if the mistake results from a missing relevant feature and, if so, to place a new relevant feature into the set.

4. Challenges for Future Relevance Research

Despite the recent activity, and the associated progress, in methods for selecting relevant features and examples, there remain many directions in which machine learning can improve its study of these important problems. Here we outline some research challenges for the theoretical and empirical learning communities.

4.1 Theoretical Challenges

We claim that, in a sense, many of the central open theoretical problems in machine learning revolve around questions of finding relevant features. For instance, consider the well-known question of whether there are polynomial-time algorithms that can guarantee learning of polynomial-size DNF formulas in the PAC or uniform distribution models. Or, consider the similar question of whether polynomial-size decision trees are learnable in either model. These questions both include the following open problem as a special case:

> Does there exist a polynomial time algorithm for learning the class of Boolean functions over $\{0,1\}^n$ that have $\log_2(n)$ relevant features, in the PAC or uniform distribution models?

This is a special case because any function that has only $\log_2 n$ relevant features can, by definition, be written as a truth table having only n entries, and therefore it must have a small decision tree and a small DNF representation (note that the learning problem would be trivial if we knew a priori *which* $\log_2 n$ variables were relevant).[8] On the other hand, this problem appears to be a quite difficult special case. For instance, any algorithm to solve this problem would need to be "unusual" in the sense that the class has been proven impossible to learn in the statistical query model of Kearns (Blum et al., 1994). Thus, issues of finding relevant features seem to be at the core of what makes those classes hard.

As a practical matter, it is unclear how to experimentally test a proposed algorithm for this problem, since no distribution on the *target functions* is given. In fact, functions with random truth tables in this class are generally easy. To allow for easier experimental testing of algorithms for this problem, the following is a *specific* distribution on the target functions that seems quite hard even for uniform random examples (for convenience, the number of relevant features is $2\log_2 n$):

> Select at random two disjoint sets $S, T \subset \{1, \ldots, n\}$ each of size $\log_2 n$. On input x, compute the parity of the bits indexed by S (that is, does S contain an odd number of ones?) and the majority function of the bits indexed by T (that is, does T contain more ones than zeroes?), and output the exclusive-or of the two results.[9]

A second theoretical challenge is to develop algorithms with the focusing ability of Winnow that apply to more complex target classes such as decision lists, parity functions, or general linear threshold functions. This would greatly extend the class of problems for which there exist positive results in on-line settings.

In the framework of example selection, one important direction is to connect the work on membership query models, which have the advantage of generally being algorithmic but assume that arbitrary points in the input space may be probed, with the work on filtering unlabeled instances, which apply when only a fixed data stream is available, but often require solving a computationally hard subproblem. Another challenge is to further theoretically analyze the ways in which example selection can aid the feature selection process.

4.2 Empirical Challenges

Considerable work also remains on the empirical front, with one of the most urgent needs being studies on more challenging data sets. For instance, few of the domains used to date have involved more than 40 features. Two exceptions are Aha and Bankert's study of cloud classification (204 attributes) and Koller and Sahami's work on information retrieval (1675 attributes), but typical experiments have dealt with far fewer features. Moreover, Langley and Sage's (1994) results with the nearest neighbor method suggest that many of the widely-used UCI data sets have few completely irrelevant attributes. In hindsight, this seems natural for diagnostic domains, in which experts tend to ask about relevant features and ignore other ones. However, we believe that many real-world domains do not have this character, and that we must find data sets with a substantial fraction of irrelevant attributes if we want to test adequately our ideas on feature selection.

Experiments with synthetic data also have important roles to play in the study of feature-selection methods. Such data sets can let one systematically vary factors of interest, such as the number of relevant and irrelevant attributes, while holding other factors constant. In this way, one can directly measure the sample complexity of algorithms as a function of these factors, showing their ability to scale to domains with many irrelevant features. However, we distinguish between the use of synthetic data for such systematic experiments and reliance on isolated artificial data sets (such as the Monks problems), which seem much less useful.

8. In fact, this class is easy to learn when the algorithm can make active (membership) queries about examples of its own choosing. Indeed, the algorithm of Bshouty (1993) learns the larger class of decision trees with membership queries in the exact leaning model, and a recent algorithm of Jackson (1994) learns the even larger class of general DNF formulas using membership queries, with respect to the uniform distribution.

9. For instance, if $S = \{1, 2, 3\}$ and $T = \{4, 5, 6\}$ then the classification of the example 011101001010 would be positive, since the first three bits have an even number of ones (making their parity 0), and the next three bits have more ones than zeros (so the majority function is 1), and the XOR of those two quantities is 1.

More challenging domains, with more features and a higher proportion of irrelevant ones, will require more sophisticated methods for feature selection. Although further increases in efficiency would increase the number of states examined, such constant-factor improvements cannot eliminate problems caused by exponential growth in the number of feature sets. However, viewing these problems in terms of heuristic search suggests some places to look for solutions. In general, we must invent better techniques for selecting an initial feature set from which to start the search, formulate search-control methods that take advantage of structure in the space of feature sets, devise improved frameworks for evaluating the usefulness of alternative feature sets, and design better halting criteria that will improve efficiency without sacrificing accuracy. Future research in the area should also compare more carefully the behavior of feature selection and attribute weighting schemes. Presumably, each approach has some advantages, leaving an open question that is best answered by experiment, but preferably by *informed* experiments designed to test specific hypotheses about these two approaches to relevance.

More generally, feature selection and example selection are tasks that seem to be intimately related and we need more studies designed to help understand and quantify this relationship. Much of the empirical work on example selection (e.g., Gross, 1991; Cohn et al., 1996) has dealt with low-dimensional spaces, yet this approach clearly holds even greater potential for domains involving many irrelevant features. Resolving basic issues of this sort promises to keep the field of machine learning occupied for many years to come.

Acknowledgements

This research was supported in part by Grant No. CCR-9357793 from the National Science Foundation, by a Sloan Foundation Research Fellowship, and by Grant No. N00014-94-1-0505 from the Office of Naval Research. Many of the researchers active in the area of feature and example selection contributed, directly or indirectly, to the ideas presented in this paper. We would also like to thank the referees and the editors of this issue for their helpful comments and suggestions.

References

Aha, D. (1990). *A study of instance-based algorithms for supervised learning tasks: Mathematical, empirical, and psychological evaluations*. Doctoral dissertation, Department of Information & Computer Science, University of California, Irvine.

Aha, D. W., & Bankert, R. L. (1996). A comparative evaluation of sequential feature selection algorithms. In D. Fisher & J.-H. Lenz (Eds.), *Artificial intelligence and statistics V*. New York: Springer-Verlag.

Almuallim, H., & Dietterich, T. G. (1991). Learning with many irrelevant features. *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 547–552). San Jose, CA: AAAI Press.

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation*, *75*, 87–106.

Angluin, D., Hellerstein, L., & Karpinski, M. (1993). Learning read-once formulas with queries. *Journal of the ACM*, *40*, 185–210.

Armstrong, R., Freitag, D., Joachims, T., & Mitchell, T. (1995). Webwatcher: A learning apprentice for the world wide web. *AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments*.

Blum, A. (1992). Learning Boolean functions in an infinite attribute space. *Machine Learning*, *9*, 373–386.

Blum, A. (1995). Empirical support for Winnow and weighted-majority based algorithms: Results on a calendar scheduling domain. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 64–72). Lake Tahoe, CA: Morgan Kaufmann.

Blum, A., Furst, M., Jackson, J., Kearns, M., Mansour, Y., & Rudich, S. (1994). Weakly learning DNF and Characterizing Statistical Query learning using Fourier analysis. *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pp. 253–262.

Blum, A., Hellerstein, L., & Littlestone, N. (1995). Learning in the presence of finitely or infinitely many irrelevant attributes. *Journal of Computer and System Sciences*, *50*, 32–40.

Blum, A. & Kannan, R. (1993). Learning an intersection of k halfspaces over a uniform distribution. *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science* (pp. 312–320). IEEE.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1987). Occam's razor. *Information Processing Letters*, *24*, 377–380.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, *36*, 929–965.

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

Bshouty, N. H. (1993). Exact learning via the monotone theory. *Proceedings of the IEEE Symposium on Foundations of Computer Science* (pp. 302–311). IEEE: Palo Alto, CA.

Cardie, C. (1993). Using decision trees to improve case-based learning. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 25–32). Amherst, MA: Morgan Kaufmann.

Caruana, R. A., & Freitag, D. (1994a). Greedy attribute selection. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 28–36). New Brunswick, NJ: Morgan Kaufmann.

Caruana, R. A., & Freitag, D. (1994b). How useful is relevance? *Working Notes of the AAAI Fall Symposium on Relevance* (pp. 25–29). New Orleans, LA: AAAI Press.

Catlett, J. (1992). Peepholing: Choosing attributes efficiently for megainduction. *Proceedings of the Ninth International Conference on Machine Learning* (pp. 49–54). Aberdeen, Scotland: Morgan Kaufmann.

Cesa-Bianchi, N., Freund, Y., Helmbold, D. P., Haussler, D., Schapire, R. E., & Warmuth, M. K. (1993). How to use expert advice. *Proceedings of the Annual ACM Symposium on the Theory of Computing* (pp. 382–391).

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

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Comon, P. (1994). Independent component analysis: A new concept. *Signal Processing*, *36*, 287–314.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*, 21–27.

Daelemans, W., Gillis, S., & Durieux, G. (1994). The acquisition of stress: A data-oriented approach. *Computational Linguistics*, *20*(3) 421-451.

Devijver, P. A., & Kittler, J. (1982). *Pattern recognition: A statistical approach.* New York: Prentice-Hall.

Dhagat, A., & Hellerstein, L. (1994). PAC learning with irrelevant attributes. *Proceedings of the IEEE Symposium on Foundations of Computer Science* (pp. 64–74). IEEE.

Doak, J. (1992). *An evaluation of feature-selection methods and their application to computer security* (Technical Report CSE-92-18). Davis: University of California, Department of Computer Science.

Drucker, H., Schapire, R., & Simard, P. (1992). Improving performance in neural networks using a boosting algorithm. In J. E. Moody, S. J. Hanson, & R. P. Lippmann (Eds.), *Advances in neural information processing systems* (Vol. 4). San Francisco: Morgan Kaufmann.

Drucker, H., Cortes, C., Jackel, L. D., LeCun, Y., & Vapnik, V. (1994). Boosting and other machine learning algorithms. *Proceedings of the Eleventh International Conference on Machine Learning*, (pp. 53–61). New Brunswick, NJ: Morgan Kaufmann.

Dyer, M., Frieze, A., & Kannan, R. (1989). A random polynomial time algorithm for approximating the volume of convex bodies. *Proceedings of the Annual ACM Symposium on the Theory of Computing* (pp. 375–381).

Freund, Y. (1990). Boosting a weak learning algorithm by majority. *Proceedings of the Third Annual Workshop on Computational Learning Theory* (pp. 202–216). San Francisco: Morgan Kaufmann.

Freund, Y. (1992). An improved boosting algorithm and its implications on learning complexity. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory* (pp. 391–398). ACM Press.

Garey, M., & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness.* San Francisco: W. H. Freeman.

Gil, Y. (1993). Efficient domain-independent experimentation. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 128–134). Amherst, MA: Morgan Kaufmann.

Gross, K. P. (1991). *Concept acquisition through attribute evolution and experiment selection.* Doctoral dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Haussler, D. (1986). Quantifying the inductive bias in concept learning. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 485–489). Philadelphia: AAAI Press.

Holte, R. (1993). Very simple classification rules perform well on most commonly used domains. *Machine Learning*, *11*, 63–91.

Jackson, J. (1994). An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Proceedings of the IEEE Symposium on Foundations of Computer Science*. IEEE.

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 121–129). New Brunswick, NJ: Morgan Kaufmann.

John, G. H., & Langley, P. (1996). Static vs. dynamic sampling for data mining. *Proceedings of the Second International Conference of Knowledge Discovery and Data Mining* (pp. 367–370). Portland: AAAI Press.

Jolliffe, I. T. (1986). *Principal component analysis.* New York: Springer-Verlag.

Johnson, D. S. (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, *9*, 256–278.

Kearns, M. J., & Vazirani, U. V. (1994). *An introduction to computational learning theory.* Cambridge, MA: MIT Press.

Kira, K., & Rendell, L. (1992). A practical approach to feature selection. *Proceedings of the Ninth International Conference on Machine Learning* (pp. 249–256). Aberdeen, Scotland: Morgan Kaufmann.

Kivinen, J., & Warmuth, M. K. (1995). Additive versus exponentiated gradient updates for linear prediction. *Proceedings of the 27th Annual ACM Symposium on Theory of Computing* (pp. 209–218). New York: ACM Press.

Kohavi, R. (1995). The power of decision tables. *Proceedings of the Eighth European Conference on Machine Learning*.

Kohavi, R., Langley, P., & Yun, Y. (1997). The utility of feature weighting in nearest-neighbor algorithms. *Proceedings of the Ninth European Conference on Machine Learning*. Prague: Springer-Verlag.

Knobe, B., & Knobe, K. (1977). A method for inferring context-free grammars. *Information and Control*, *31*, 129–146.

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. *Proceedings of the Thirteenth International Conference on Machine Learning*. Bari, Italy: Morgan Kaufmann.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of Relief. *Proceedings of the Seventh European Conference on Machine Learning*.

Kubat, M., Flotzinger, D., & Pfurtscheller, G. (1993). Discovering patterns in EEG signals: Comparative study of a few methods. *Proceedings of the 1993 European Conference on Machine Learning* (pp. 367–371). Vienna: Springer-Verlag.

Kulkarni, D., & Simon, H. A. (1990). Experimentation in machine discovery. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation*. San Francisco: Morgan Kaufmann.

Langley, P., & Iba, W. (1993). Average-case analysis of a nearest neighbor algorithm. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 889–894). Chambery, France.

Langley, P., & Sage, S. (1994a). Oblivious decision trees and abstract cases. *Working Notes of the AAAI94 Workshop on Case-Based Reasoning* (pp. 113–117). Seattle, WA: AAAI Press.

Langley, P., & Sage, S. (1994b). Induction of selective Bayesian classifiers. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence* (pp. 399–406). Seattle, WA: Morgan Kaufmann.

Langley, P., & Sage, S. (1997). Scaling to domains with many irrelevant features. In R. Greiner (Ed.), *Computational learning theory and natural learning systems* (Vol. 4). Cambridge, MA: MIT Press.

Lewis, D. D. (1992a). *Representation and learning in information retrieval*. Doctoral dissertation, Department of Computer Science, University of Massachusetts, Amherst. Also available as Technical Report UM-CS-1991-093.

Lewis, D. D. (1992b). Feature selection and feature extraction for text categorization. *Proceedings of Speech and Natural Language Workshop* (pp. 212–217). San Francsico: Morgan Kaufmann.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 148–156). New Brunswick, NJ: Morgan Kaufmann.

Lin, L. J. (1992). Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning*, *8*, 293–321.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. *Machine Learning*, *2*, 285–318.

Littlestone, N., Long, P. M., & Warmuth, M. K. (1991). On-line learning of linear functions. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing* (pp. 465–475). New Orleans: ACM Press.

Littlestone, N., & Mesterharm, C. (1997). An apobayesian relative of Winnow. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems* (Vol. 9). Denver, CO: MIT Press.

Littlestone, N., & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, *108*, 212–261.

Lovász, L., & Simonovits, M. (1992). On the randomized complexity of volume and diameter. *Proceedings of the IEEE Symposium on Foundations of Computer Science* (pp. 482–491). IEEE.

Lund, C., & Yannakakis, M. (1993). On the hardness of approximating minimization problems. *Proceedings of the Annual ACM Symposium on the Theory of Computing* (pp. 286–293).

Matheus, C. J., & Rendell, L. A. (1989). Constructive induction on decision trees. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 645–650). Detroit, MI: Morgan Kaufmann.

Michalski, R. S. (1980). Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *2*, 349–361.

Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. The MIT Press.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, *18*, 203–226. Reprinted in J. W. Shavlik & T. G. Dietterich (Eds.) (1990), *Readings in machine learning*. San Francisco, CA: Morgan Kaufmann.

Moore, A. W., & Lee, M. S. (1994). Efficient algorithms for minimizing cross validation error. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 190–198). New Brunswick, NJ: Morgan Kaufmann.

Norton, S. W. (1989). Generating better decision trees. *Proceedings of the Eleventh International Conference on Artificial Intelligence* (pp. 800–805). Detroit, MI: Morgan Kaufmann.

Pazzani, M. J., & Sarrett, W. (1992). A framework for the average case analysis of conjunctive learning algorithms. *Machine Learning*, *9*, 349–372.

Pagallo, G., & Haussler, D. Boolean feature discovery in empirical learning. *Machine Learning*, *5*, 71–99.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. San Francisco, CA: Morgan Kaufmann.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco: Morgan Kaufmann.

Rajamoney, S. (1990). A computational approach to theory revision. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation*. San Francisco, CA: Morgan Kaufmann.

Rivest, R. L., & Schapire, R. E. (1993). Inference of finite automata using homing sequences. *Information and Computation*, *103*, 299–347.

Rumelhart, D. E., Hinton, G., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1). Cambridge, MA: MIT Press.

Sammut, C., & Banerji, R. B. (1986). Learning concepts by asking questions. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Francisco, CA: Morgan Kaufmann.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*, 197–227.

Schlimmer, J. C. (1987). Efficiently inducing determinations: A complete and efficient search algorithm that uses optimal pruning. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 284–290). Amherst, MA: Morgan Kaufmann.

Scott, P. D., & Markovitz, S. (1991). Representation generation in an exploratory learning system. In D. H. Fisher, M. J. Pazzani, & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Francisco, CA: Morgan Kaufmann.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (pp. 287–294). New York: ACM Press.

Shen, W. M., & Simon, H. A. (1989). Rule creation and rule learning through environmental exploration. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 675–680). Detroit, MI: Morgan Kaufmann.

Sinclair, A., & Jerrum, M. (1989). Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, *82*, 93–133.

Singh, M., & Provan, G. M. (1995). A comparison of induction algorithms for selective and non-selective Bayesian classifiers. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 497–505). Lake Tahoe, CA: Morgan Kaufmann.

Singh, M., & Provan, G. M. (1996). Efficient learning of selective Bayesian network classifiers. *Proceedings of the Thirteenth International Conference on Machine Learning*. Bari, Italy: Morgan Kaufmann.

Skalak, D. B. (1994). Prototype and feature selection by sampling and random mutation hill-climbing algorithms. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 293–301). New Brunswick, NJ: Morgan Kaufmann.

Stanfill, C. W. (1993). Memory-based reasoning applied to English pronunciation. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 577–581). Seattle, WA: AAAI Press.

Ting, K. M. (1994). *Discretization of continuous-valued attributes and instance-based learning* (Technical Report No. 491). Sydney: University of Sydney, Basser Department of Computer Science.

Townsend-Weber, T., & Kibler, D. (1994). Instance-based prediction of continuous values. *Working Notes of the AAAI94 Workshop on Case-Based Reasoning* (pp. 30–35). Seattle, WA: AAAI Press.

Verbeurgt, K. (1990). Learning DNF under the uniform distribution in polynomial time. *Proceedings of the Third Annual Workshop on Computational Learning Theory* (pp. 314–325). Morgan Kaufmann.

Vere, S. A. (1975). Induction of concepts in the predicate calculus. *Proceedings of the Fourth International Joint Conference on Artificial Intelligence* (pp. 281–287). Tbilisi, USSR: Morgan Kaufmann.

Vovk, V. (1990). Aggregating strategies. *Proceedings of the Third Annual Workshop on Computational Learning Theory* (pp. 371–383). Morgan Kaufmann.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *IRE WESCON Convention Record* (pp. 96–104).

Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.