*Mark Gahegan*

# *On the Application of Inductive Machine Learning Tools to Geographical Analysis*

*Inductive machine learning tools, such as neural networks and decision trees, offer alternative methods for classification, clustering, and pattern recognition that can, in theory, extend to the complex or "deep" data sets that pervade geography. By contrast, traditional statistical approaches may fail, due to issues of scalability and flexibility. This paper discusses the role of inductive machine learning as it relates to geographical analysis. The discussion presented is not based on comparative results or on mathematical description, but instead focuses on the often subtle ways in which the various inductive learning approaches differ operationally, describing (1) the manner in which the feature space is partitioned or clustered, (2) the search mechanisms employed to identify good solutions, and (3) the different biases that each technique imposes. The consequences arising from these issues, when considering complex geographic feature spaces, are then described in detail. The overall aim is to provide a foundation upon which reliable inductive analysis methods can be constructed, instead of depending on piecemeal or haphazard experimentation with the various operational criteria that inductive learning tools call for. Often, it would appear that these criteria are not well understood by practitioners in the geographic sphere, which can lead to difficulties in configuration and operation, and ultimately to poor performance.*

Over the past thirty years, researchers in computer science, particularly the computer vision and artificial intelligence community, have developed a number of techniques aimed at pattern recognition, classification, cluster analysis, prediction, and simulation that offer potential benefits when applied to the analysis of complex geographic data sets. More recently, some of these techniques have become "packaged" sufficiently for them to be made generally available as analysis tools. Examples of such techniques are neural networks, genetic algorithms, decision trees, and self-organizing maps.[1] Their common property is that they can all act as *inductive learning* tools (Hunt, Marin, and Stone 1966). Induction is described by the philosopher Pierce (1878) as follows. *"By induction, we conclude that facts, similar to observed facts, are true in cases not examined."* In-

[1] Strictly speaking, self-organizing maps are a form of neural network. They are considered separately here since they typically operate as a clustering tool, as opposed to a discriminant classifier. Section 2.2 describes this distinction.

*Mark Gahegan is an associate professor of geography at Pennsylvania State University, and also associate director of the GeoVISTA Center for Geographical Visualization.*

ductive learning offers a means to characterize a category or function without relying on prior knowledge. Inductive learning tools are trained to recognize patterns or to predict outcomes by generalizing from a group of measurements for which the desired outcome is known (training data) to a larger set of circumstances. They perform this task without reliance on any a priori information regarding probability density functions or event likelihoods. Applications in the geographical domain requiring this kind of operation are common and include derivation of landcover information from remotely sensed images, creation of sociodemographic indexes (zones) from census and other variables, detection of clusters in spatiotemporal data sets, prediction of the outflows of stream networks, and so on (Openshaw and Abrahart 1996).

This paper is organized as follows. The first section describes the use and functioning of various inductive tools, applied to a broad range of classification and pattern recognition problems. The approach to this first section is pedagogic, taking what is known from the machine vision community and expressing it in the context of geographical analysis. Subsequent sections examine the operation of inductive learning tools within a geographic setting, concentrating on specific problems that must be addressed before they can be regarded as reliable and effective problem-solving methods, and perhaps move into the mainstream of geographical analysis and modeling techniques. These latter sections are grounded in the theory of the tools described, but born from experience of developing robust methodologies for their use. Interested readers may wish to review Gahegan, German, and West (1998), who describe the development of such a methodology, giving a series of practical results and enhancements to techniques designed at improving both reliability and performance.

Since the beginning of the 1990s there have been many reported examples of the use of inductive learning tools in the geographical analysis, modeling, and remote sensing literature [for example, Benediktsson, Swain, and Ersoy (1990); Byungyong and Landgrebe (1991); Lees and Ritman (1991); Civco (1993); Openshaw (1993); Fischer (1994); Yoshida and Omatu (1994); Paola and Schowengerdt (1995); Foody, McCulloch, and Yates (1995); German and Gahegan (1996); Friedl and Brodley (1997); Evans et al. (1998), and many more examples in the GeoComputation conference proceedings—http://www.ashville.demon.co.uk/geocomp/]. The general trend amongst these reported examples is that substantial performance gains may be possible. However, since the tools are inherently complex, their use often incurs a considerable investment in terms of customization, setup, experimentation, and testing before useful results are obtained. Unfortunately, corresponding progress toward understanding *how* these tools behave with geographic data has been slow to date [with some exceptions, for example, Foody and Arona (1997); Kanellopoulos and Wilkinson (1997); Gahegan, German, and West (1998)] and as a consequence, robust versions of these tools that can be readily applied to geographical analysis problems, or embedded within a GIS, are rare. Furthermore, the lack of understanding of how these tools function, and the inherent *bias* they impart, leads to the danger of misinterpreting results produced.

The purpose of the research reported here is to investigate some of the important issues arising from the application of inductive learning tools to complex geographical data sets, with the aim of building a foundation from which useful functionality can be constructed. The computational theory and algorithmic foundations of the tools have been described many times and are not repeated here except where they are essential for an understanding of behavior. The interested reader is directed to the following books for a full explanation: Pao (1989) and Bishop (1995) for neural networks, Quinlan (1993) for decision trees, Goldberg (1989) and Mitchell (1996) for genetic algorithms, and Kohonen (1995) for self-organizing maps. Mitchell (1997) provides a very useful overview of all these methods.

The development of inductive learning tools is driven by the need to address a range of complex, nondeterministic problems, where a brute-force search for a truly optimal solution becomes computationally intractable (Martin 1991; Moret and Shapiro 1991). Computability is assured via a number of search optimization techniques that examine only a portion of the complete range of solutions. The need for an approximate solution may be as a result of complex input data, complex output characteristics, or a combination of both. For example, the vast increase in attribute dimensionality of a number of digital geospatial data sources (such as hyperspectral remote sensing platforms, digital census, vehicle-borne and wearable computers, airborne electromagnetic induction systems, point-of-sale consumer patterns, and so forth) provide a wealth of data that might be used in geographic analysis and pose some interesting computational challenges. An excellent account of these challenges and the often counterintuitive nature of a high-dimensionality feature space is given by Landgrebe (1999). Established techniques, such as *k*-means clustering measures (MacQueen 1967), principle components analysis (Webster and Oliver 1990), maximum likelihood classification (Richards 1986), and statistically based pattern recognition techniques (Devijver and Kittler 1982), may be less able to address this complexity. It is often claimed that these new inductive tools offer significant advances over traditional techniques in that they are (by comparison at least) robust in the presence of noise, flexible as to the statistical types that can be combined, able to work with feature (attribute) spaces of very high dimensionality, require less training data, and make fewer prior assumptions about data distributions and model parameters. These claims are examined later in the paper with particular attention paid to the characteristics of complex geographical data sets.

In order to maintain a focus narrow enough to allow depth of coverage, most of the examples given relate to the two most popular inductive learning techniques, neural networks (specifically feed-forward, back-propagation networks acting as a multilayer perceptron), and decision trees, set up to function as geographic classifiers. The term "classification" is used here, and in the inductive learning literature, as a surrogate for a broad set of learning tasks. These include cluster detection and analysis (Anderberg 1973) and pattern recognition (Carpenter 1989). This paper begins by explaining the classification process when defined as an inductive learning task, and then introduces the notions of *complexity*, *search*, *generalization*, and *bias* that together determine the outcome. Following from this, each of these notions is discussed, with particular emphasis placed on the kinds of problems and complexities that arise when working with *geographic* data and applications.

## *Why Classify at All?*

Our innate ability to categorize forms the basis of our powers of cognition, leading Lakoff (1987) to remark that without this ability, we would not be able to function at all. Classification has many applications in geography and is a form of categorization where the task is to take the descriptive attributes of an object (or set of objects) and from this to label or identify the object within a different phenomenological domain. The descriptive attributes may be themselves drawn from different data domains, each domain effectively contributing an axis to a combined *feature space* of all possible object descriptions. Hence, the task of the classifier is somehow to partition this feature space into disjoint regions that each represent a particular class, cluster, or pattern.

## 1. TYPES OF CLASSIFIERS

Classifiers are often divided into two groups, *supervised* and *unsupervised.* Supervised techniques have a training phase during which samples of data representing the chosen classes are used to condition a model for the classification process. The commonly used maximum likelihood classifier (MLC) is an example of a supervised technique in which training data is analyzed to derive the parameters of a set of multivariate probability density functions, by which the unseen data is assigned a probabilistic label a posteriori. A full description of the derivation of the MLC is given by Richards (1994). Unsupervised classifiers do not require any preconceived notion of a class, being based purely on grouping or clustering the data, usually formulated on the Euclidean metric. The $k$-means and ISODATA classifiers are examples of the latter technique, as are many of the more recent data mining algorithms. For $k$-means classification, the user supplies only the number of classes to be derived ($k$) and the classifier attempts to locate the $k$ strongest clusters occurring within the data (MacQueen 1967).

A good classification should both *impose* structure and *reveal* the structure already present within the data (Anderberg 1973). With the exception of data reduction tasks, supervised techniques are generally favored in analysis since the user retains control over the classes imposed, that is, over the output domain. The outcome from an unsupervised classifier may have little meaning since the resulting classes are not associated (by design) with any concept arising from the domain of study (although they may be as a consequence of inherent structure in the data).

### *1.1 Advantages of Inductive Classifiers*

Using an inductive learning approach to classification, we attempt to *learn* the classifier, as distinct from *imposing* or conditioning the classifier (as happens with traditional techniques such as maximum likelihood). By learning the classifier we avoid some problems (Gould 1970) whilst making others. To elaborate, the assumption that a predefined distributional model, such as a Gaussian curve or student's $t$ distribution, is a useful template for describing the behavior of each class can itself give rise to a number of difficulties. By attempting, instead, to define the model as the goal of the learning phase, it is possible that the characteristics of a given domain may be represented with greater fidelity.

Specifically, parametrically based classifiers, such as the MLC, suffer from a number of drawbacks. Firstly, they assume that values for each data dimension fit a predetermined distribution. This usually implies that classes are both unimodal and symmetric. Unimodality is a particularly demanding constraint in practice, since each class must be constructed from a single region in feature space. Classes that are, in reality, a series of subclasses must be recognized as such and each distinct region must be defined separately, via its own training examples. This can involve a labor-intensive exploratory study and is problematic to ensure in high-dimensionality feature spaces, due to the difficulty in visualizing or otherwise examining all the feature space at once. Secondly, a large number of training examples are needed to construct the probability density function (PDF) with defensible confidence. The MLC requires a minimum of ten to one hundred samples per dimension in the feature space (Mardia, Kent, and Bibby 1979). Thirdly, nominal (and, to some extent, ordinal) data are handled poorly, if at all. The PDF is defined continuously and is problematic to configure for noncontinuous features. This poses difficulties because such data are common in many geographical problems; for example, noncontinuous ancillary data (such as soil type or geology) can offer significant classification gains if it can be combined with remotely sensed data (Gurney 1983). Finally, computational scaling is poor as the feature space grows; with large feature spaces the computational cost can become prohibitive. As we shall see below, inductive classifiers can go some way to overcoming each of these problems.

## *1.2 The Challenges Posed by Inductive Learning Tools*

Inductive machine learning tools can be characterized by the fact that the problem, be it classification, function approximation, or pattern recognition, is set up as a search for some minimum cost solution across a search space. This search is not exhaustive; the learning stage terminates when the search *converges* on a solution, an approximation of the global minimum of some error function applied to the search. The search mechanism may be one of hyperplane fitting, hierarchical partitioning, or clustering. Irrespective of this mechanism, a number of specific research questions arise:

1. *How can we "map" the geographic problem successfully to the search space used by the tools?* The setup of the problem as one of search and the search mechanisms employed will affect the outcome. In turn, these decisions introduce a *bias* into the search. Many inductive tools do not search directly in feature space (the space defined by all attributes considered) but rather in a more abstract *hypothesis space.* Section 2.1 describes the setup of a hypothesis space and section 2.2 considers the nature of bias and how it affects the classification outcome.
2. *How can we be certain that the tools examine all relevant portions of the hypothesis space—avoiding regions that are unproductive whilst ensuring that a global minimum is still found?* As the attribute dimensionality increases, control over the search becomes more difficult leading to problems in obtaining convergence or possibly convergence on a local, as opposed to a global, minimum. Section 3.2 takes up this issue.
3. *How do we know that the classifier is properly set up, when a good solution has been found, and can we report on statistical reliability?* Since a model is being induced (by contrast with MLC) we must also test the fit of the model, not just the performance of the tool on the data used for training. A methodological shift is needed in evaluation. This becomes a difficult issue where sample or training data is in limited supply. Recognizing a good solution and reporting on its reliability are described in section 3.1.
4. *Are there particular characteristics or nuances of geospatial data that need special consideration in the design of inductive tools?* There is a temptation to treat the geographic data set as providing a feature space where each element is treated in exactly the same manner, that is, as orthogonal, continuous axes, since that is how the tools are typically described and used. Section 3.3 discusses the legitimacy of this assumption.
5. *How do we "mine" the solutions to provide insight or knowledge of the underlying geographic system or problem?* These techniques often operate as "black boxes," with the solutions provided being highly parameterized with little obvious connection to the problem domain. A mapping is constructed from the input domain to the required output in a manner that is not model based. The outcomes provide little or nothing as usable domain knowledge or insight and so do not help to increase the general understanding of a problem. The reporting and mining of outcomes from inductive learning are described in section 3.4.

Without answers to these research questions the gains over established methods are likely to be minimal or erratic.

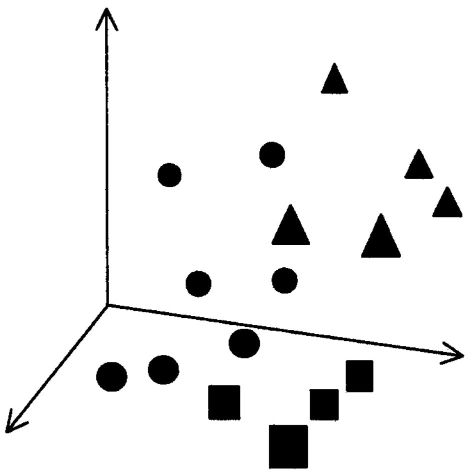

FIG. 1. A Hypothetical Three-Class Separation Problem in a Three-Dimensional Feature Space. Larger symbols are closer.

## *1.3 Classification Complexity*

Classifiers can also be characterized according to the difficulty of the *discrimination* task that they must address. Imagine a three-dimensional feature space, such as that shown by Figure 1, which must be separated out into disjoint regions each containing only one class.

Three possible outcomes can occur (after Sonka, Hlavac, and Boyle 1993). Firstly, if it is possible to position hypersurfaces so that each region formed contains only objects from a single class, then the classes are considered to be *separable*. Secondly, if the hypersurfaces could be replaced with hyperplanes without loss of accuracy, the classes are said to be *linearly separable*. For example, the squares and circles in Figure 1 are linearly separable using a single hyperplane, as shown in Figure 2.

A feature space that is linearly separable requires only a linear classifier (Dunteman 1984, ch. 5). That being the case, inductive learning tools are unlikely to offer any significant advantages over established statistical techniques. Finally, the overlap of samples in feature space may make it impossible (or impractical) to position a set of hypersurfaces so that classes are differentiated.[2] The classes are, to some degree, *inseparable*. Many problems in geographical analysis seem to contain at least some degree of inseparability, with certain classes being inseparable and others requiring a number of hypersurfaces to describe them adequately.

Decision trees and neural networks can both operate as discriminant classifiers, but in rather different ways. The most common form of neural classifier, a feed-forward, back-propagation network, effectively positions a number of hyperplanes within the feature space (where each hyperplane is defined by a single hidden-layer node and its weighted connections).[3] Decision trees, by

[2]Of course, it is always possible to differentiate classes given that (1) feature vectors are not entirely identical and (2) there is no fixed upper bound on the shape complexity and number of discrimination surfaces. However, a classifier trained in this manner is likely to perform very poorly when presented with unseen examples; it will be unable to generalize. Section 3.2.3 takes this point further.

[3]This is in fact a simplification, but for the practical purposes under discussion here it should suffice.

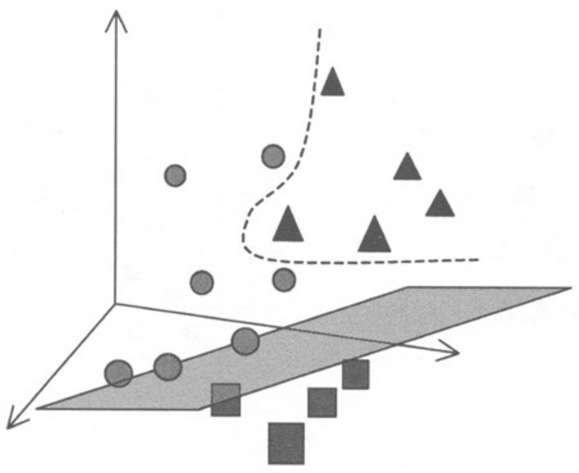

FIG. 2. Linear Separability: A Single Hyperplane Discriminates between the "Square" and "Round" Classes. Separation of circles from diamonds requires at least two hyperplanes or a hypersurface (shown in 2D by the dotted line).

contrast, apply a hierarchy of hard discrimination rules, usually defined orthogonally to the axes in feature space, to effectively construct class boundaries in a piecewise manner. It might appear that both techniques are only capable of linear separation, but further study shows that they are in fact able to approximate arbitrarily complex decision boundaries.

Figure 3 gives examples of how decision trees and neural networks deal with complexity, that is, how they approximate the functioning of hypersurfaces to differentiate complex regions of feature space. The first example, (a), shows a single hyperplane that can be positioned at any orientation, as might be used in a discriminant classifier or neural network. The hyperplane cannot be placed in such a way that classes are perfectly separated, that is, they are not *linearly* separable (see above). Notice that a single diamond remains in error and that any further refinement of the position of the hyperplane is likely to increase this error. Two hyperplanes working together (b) can separate the classes perfectly, provided that the fragmented regions of feature space can be summed together correctly; the use of a separate output layer within the neural network allows the various regions delineated by the hyperplanes to be summed in different ways for each target class, so the hyperplanes do not directly represent decision rules in feature space. By contrast, (c) shows the placement of a single orthogonal decision rule (as used by a decision tree) to partition the feature space and leaving three samples in error. This rule must be parallel to one axis to allow simple definition of the form "*if* $a > \lambda$ *then* $C_1$, *else* $C_2$." As in (a), no improvement can be made by choosing a different rule; to achieve better results, further piecewise approximations to the boundary must be fitted, as shown in (d).

As Figure 3 shows, a single hyperplane can often delineate a boundary with greater accuracy than many decision rules, which should in turn lead to gains in performance and simplicity for neural classifiers. A continuous sloping boundary can require a large number of orthonormal decision rules to make a reasonable approximation, and furthermore, this approximation will tend to "fit" exactly around the training examples (producing a stepwise or staircase boundary), rather than to generalize smoothly between them. Where class boundaries are

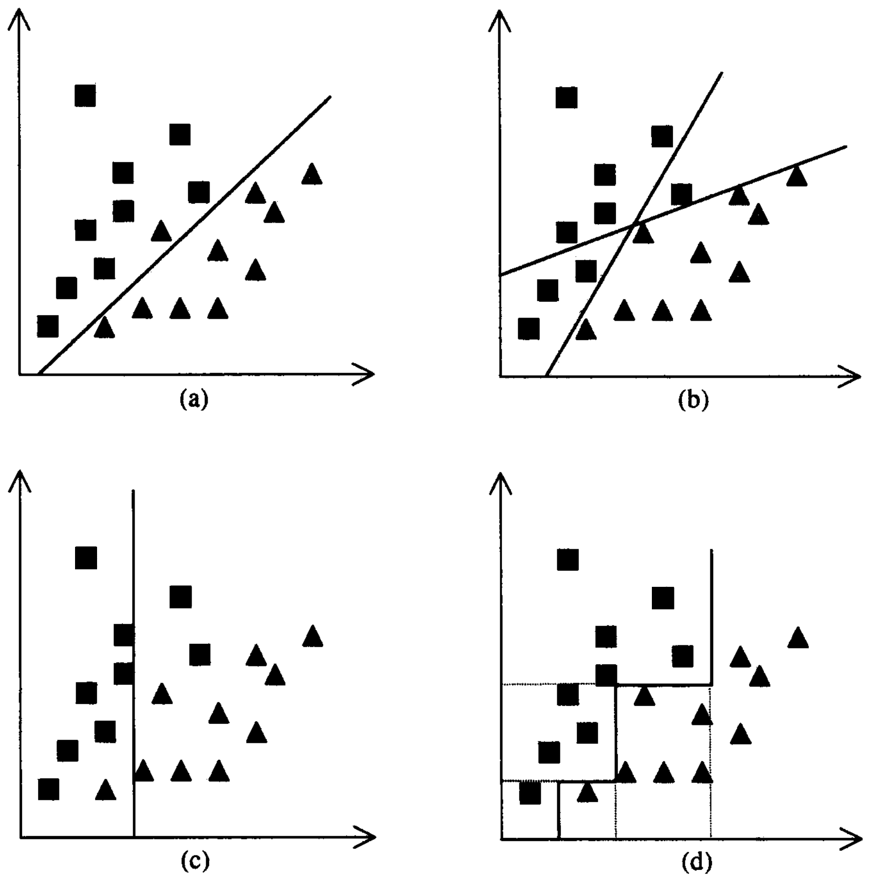

FIG. 3. Contrasting the Approaches of Hyperplanes (a and b) and Decision Rules (c and d) on a Two-Class Separation Problem. See text for a full description.

more complex, hyperplanes may still prove more effective, but only after the additional step of assigning the fragments of feature space to the correct output class. Murphy, Kasif, and Salzberg (1994) describe a decision tree variant that can also support oblique decision rules.

A different approach is taken by techniques that attempt to cluster data, such as the self-organizing map (Grossberg 1991; Kohonen 1995) or the ISODATA method (Kaufman and Rousseeuw 1990). In this case, cluster centroids are introduced into the feature space, and the space is fragmented using ellipsoidal or Voronoi-like regions formed around "close" training samples. Figure 4 gives two examples of these types of clustering using the same set of points as the previous figure. Notice that classes are fragmented as with hyperplane fitting and also that some regions of the space remain undefined in (a), a consideration discussed later in section 3.2. In the self-organizing map, regions of feature space are formed around one or more generator nodes, and the learning task is to position the available nodes optimally within feature space so that they capture the required structure. Some artificial life methods adopt a similar kind of strategy.

One major complicating factor is the freedom that the classifier may or may not have to introduce rules, hyperplanes, or clusters as required. In most neural

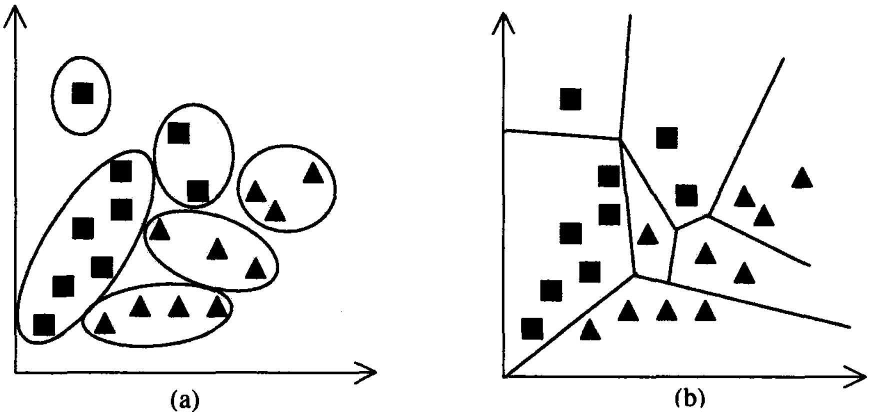

FIG. 4. Classification Based on Clustering in a 2D Feature Space: (a) shows ellipsoidal clusters, (b) shows the tessellated (almost Voronoi) structuring employed by a self-organizing map.

techniques, the provision of a hyperplane or cluster is a specific, dedicated task associated with a single processing unit, a node or neuron (usually in the hidden layer(s)). The number of nodes effectively fixes the number of hyperplanes or clusters. It is vital that this number be well chosen—too few and the network will fail to describe the training data accurately; too many and the feature space will be overfragmented, adding complexity to the final output stage and possibly causing *overtraining* to occur (section 3.2.3).

When all hyperplanes are positioned usefully, the network cannot achieve a performance gain in one region without a sacrifice in another. This explains the practice of repeating the training cycle with various different configurations of hidden layer nodes, to find an optimal arrangement for a given problem. By contrast, decision trees can keep adding rules arbitrarily until the desired level of classifier performance is achieved. So, although neural networks are perhaps capable of better class differentiation there are some practical impediments to realizing their full benefits.

## 2. FEATURE SPACE, HYPOTHESIS SPACE AND THE CLASSIFICATION PROBLEM

This section defines the important components in the design and operation of a classifier. The feature space employed by a classifier is a space of dimensionality $n$, where $n$ is the number of attributes being used, with each attribute supplying a single dimension. Inductive classifiers are sometimes referred to as "distribution free" because they make no assumption as to the shape or form of the distribution of training data within the feature space. In fact, prior statistical assumptions are reduced to a minimum; in many cases, the only assumption retained relating to the feature space is that Euclidean or algebraic distance within this space has meaning, with smaller distances implying a greater degree of similarity and larger distances a greater degree of difference.

As noted above, a classifier is defined as a mapping from one domain to another; for example, when using satellite imagery to predict landcover, the classifier maps a vector of pixel integer values to nominal landcover labels. This mapping can be expressed as a function: $y = f(x)$, where $y$ is the output domain of the class labels, $x$ is the input data drawn from a feature space $X$, and $f(\,)$ is a transformation that maps $x$ to $y$. In most applications, each $x_i$ is in fact a vector of the form $\langle x_{i1}, \ldots, x_{in} \rangle$ and $y$ is a disjoint, crisp set $\{1, \ldots, k\}$. The supervised learning problem is thus to construct the mapping $y = f(x)$ from a set of training data ($S = \{x_i\}$) for which the desired output class is known in advance: $\{(x_1, c(x_1)), \ldots, (x_m, c(x_m))\}$. Since the training data is always only a *sample* of the full range of possibilities then the classifier actually produces a *hypothesis* as to the true value of $f$, that is, a function $y = f'(x)$ where $f' \approx f$. Different inductive learning tools, arrangements of training data and operational parameters will all result in a different mapping, giving rise to a set of possible hypotheses $\{h_1, \ldots, h_j\}$ drawn from a hypothesis space $H$. After the most likely hypothesis ($h^*$) has been determined, this is translated into a function $f'$ by which all unknown data are classified.

## 2.1 *Learning as Search*

With the problem thus specified, the process of learning becomes one of search, specifically, a search through a set of possible hypotheses $\{h\}$ to determine the one that is in best agreement (least conflict) with the training data. Ideally, each successive training example reduces the hypothesis space by invalidating all hypotheses that are at odds with it. For large-scale feature spaces $\{h\}$ is effectively infinite, hence a nonexhaustive search is necessary. Inductive learning tools differ in how they try to locate $h^*$, but they share the fact that the suitability of each $h_i$ is evaluated against some objective function.

Neural networks attempt to examine all attribute dimensions simultaneously in their quest for an "optimal" solution and have routines that search the hypothesis space via a measure of overall classification error (this error is fed back into the network at each iteration as a penalty applied to nodes not performing as desired). A hypothesis is sought that minimizes error by a combination of the position of $d$ hyperplanes and a subsequent summing of the regions of feature space so formed. Sophisticated search routines have been developed specifically for this task, and can be made sensitive to the gradient and direction of change of error over a wide set of parameters (Benediktsson, Swain, and Ersoy 1993; Moller 1993). As a result, the search through $H$ moves smoothly as the current hypothesis is progressively refined.

By contrast, decision trees move toward a solution in a hierarchical manner, making an independent decision in a single data dimension at each iteration, and in the process producing a tree of decision rules. Here, search is the process of determining which dimension, and where in that dimension, to introduce a new rule. A measure of the incremental benefit that a proposed rule adds to the ability of the classifier to explain the training data—known as *information gain*—is used to select the best decision. The order in which the rules are generated determines how $H$ is searched and is dictated by the data alone. Hence, decision trees do not really have a concept of a *global* minimum, but instead attempt to construct $f'$ in a piecewise manner.

Genetic algorithms use an evolutionary approach to the search problem.[4] Search is achieved by the mutation and crossover of attributes, encoded as strings of information. Since the population is usually large, and initially distributed throughout the hypothesis space, the search effectively proceeds in parallel from many starting points, thus overcoming the hierarchical weakness of decision trees. The environment provides an objective function that defines how well an individual solution is adapted (fitness). Here we see a different model of progressive refinement, where useful solutions are able to communicate or pass on their "knowledge," but with an injection of randomness via the way in

[4] Loosely modeled on the genetic adaptation of a population of organisms to an environmental niche.

which such knowledge is recombined. Movement through $H$ is consequently more abrupt than with neural methods, but this helps avoid becoming trapped in local minima. In effect, the technique focuses search toward regions in $H$ that seem favorable, but not exclusively so. Search is controlled via a number of factors working in combination: the initial starting values, the choice of representation, the different types of recombinations possible, their likelihood, and the overall fitness function used for evaluation. A useful account of the resulting hypothesis space in a geographic setting is given by Bennett, Wade, and Armstrong (1999). Genetic algorithms can be used to help other inductive learning tools search $H$. For example, the task of determining the most suitable architecture for a neural network can be expressed using an evolutionary approach (Fischer and Leung 1998). Other forms of artificial life show potential for detecting patterns or clusters by using simple rules that encourage the collaboration of search "agents." For example, Macgill and Openshaw (1998) show how a "flock" of agents can be used to improve performance of a geographical analysis machine.

All techniques need a means to recognize when training is complete, usually via a method to evaluate if there is any likely benefit in continuing the search. For a neural network, termination can be defined using either a minimum overall error value or a number of iterations (epochs) used for training. A network that has reduced its error to a prespecified minimum value is said to have "converged." Decision trees usually employ a *stopping criterion,* where the information gain afforded by the next rule to be added is below some minimum (user-defined) threshold. Artificial life methods can use an overall fitness threshold or a limit on the number of generations or iterations.

With a few exceptions, the tools do not perform an exhaustive search of $H$, so cannot guarantee to find the optimal solution $f$. This is because $H$ is probably very large and contains a limited number of subspaces that are worth exploring. Various stochastic methods are used to control and optimize the search (for example, Gopal and Fischer 1996) thus producing an approximate solution to an intractable problem. An outstanding issue is then to determine how close $f'$ is to $f$, and hence quantify the reliability of the classifier. Because the training data are so critical in shaping $f'$, performance can only be evaluated reliably with respect to statistically independent validation data. Trying to predict performance based on the ability to classify the data used for training is likely to result in an inflated view of accuracy, since no account of overtraining (section 3.2.3) can be taken.

## *2.2 Generalization and Inductive Bias*

All classifiers need to generalize from the limited training data provided to a (usually much larger) set of unknown input vectors. Without generalization, all a classifier can do is "rote learn" the training patterns and their class, basically as a series of $m$ rules. In this case, any input vector that has not been previously encountered cannot be classified. Generalization allows the classifier to move from a model that describes only the training data to one that describes the complete instance space of all possible input examples: that is, $\{x_1 \vee x_2 \vee \ldots \vee x_m\} \xrightarrow{\textit{generalized to}} \{X\}$. A generalization mechanism, such as those demonstrated in Figures 3 and 4, is required, and this mechanism introduces a bias into the classifier. Bias in the classifier provides the mechanism for generalization. Briscoe and Caelli (1996) define bias as *"the set of all factors that influence the selection of a hypothesis, other than the examples themselves."* The manner in which bias is encountered in the tools discussed here is in the setup and the searching of the hypothesis space (Utgoff 1986; Benjamin 1990) and also in the training data used (section 3.1). By contrast, the bias of a maximum likelihood classifier is well understood and its generalization behavior can be expressed mathematically according to posterior probabilities (Johnson and Wichern 1990).

As an example of bias in the setup of the classifier, consider a neural technique where the number of hyperplanes must be chosen prior to training (as explained above). Via a rather roundabout mechanism, this choice imposes an upper bound on the number of decision surfaces that can be placed in feature space, effectively constraining the hypothesis space to those solutions possible using exactly $d$ hyperplanes. Where fewer hyperplanes are available than there are decisions to make (regions to separate), $\{h\}$ fails to contain a good approximation of $f$, irrespective of the search routine used. As an example of bias in searching the hypothesis space, a neural network that contains enough hyperplanes may fail to reach a suitable global minimum due to becoming trapped in some local minimum within $H$ that the search algorithm (for example, gradient descent) cannot overcome. One major problem is the diagnosis of these two separate problems in order that appropriate action can be taken: in the first case, the addition of hyperplanes, in the second case, adjustments to the step size and momentum term or the use of a different search algorithm.

In the geographic realm, as with many others, it is difficult to know beforehand what kinds of bias are likely to produce the most useful generalizations of the training data. As a consequence, some practitioners resort to a brute-force testing of the different architectures and parameter settings available, changing the inductive bias (in some way) to find an arrangement that gives the best performance on testing (for example, Jehng-Jung 1996). Since the space defined by these parameters is itself large in many cases, exhaustive search again is problematic, so there remains the possibility that the classifier's performance is bounded by suboptimal configuration of $H$. In other words, the "best" performance obtained is not guaranteed to be the best that is obtainable.

Another source of bias in the hypothesis space setup arises from the types of class descriptions employed, that is, the representational language of the classifier. Two types of concept description are common: discriminant and characteristic, as exemplified in Figure 3 and Figure 4. Discriminant classifiers concentrate on the separation of concepts one from another whereas characteristic classifiers cluster like concepts by determining commonality. Visualized within the feature space, discriminant classifiers divide up the space by the superposition of decision boundaries between classes whereas characterizing classifiers attempt to form zones that surround similar examples. Thus, a discriminant classifier will offer a high degree of generalization, possibly extending far beyond what is known from the training data, whereas a characteristic classifier may offer little generalization (determined in practice by the type of geometry used to describe each cluster). Figure 5 depicts these two possibilities, known as most generalization and least generalization; in both cases only orthonormal decision boundaries have been used. Notice that the star, which represents an object to be classified, is included in the "square" class using most generalization because it is most like a square, but is excluded from this class using least generalization because it is unlike any square previously encountered. Notice also that the number of decision rules or hyperplanes needed for least generalization is much higher than for most generalization (sixteen versus three). Clearly there is a continuum of generalization that falls between these two extremes.

The degree to which one form of generalization should be favored depends on the confidence we have in the training data. If we are confident that the entire feature space has been representatively sampled, or that in the absence of

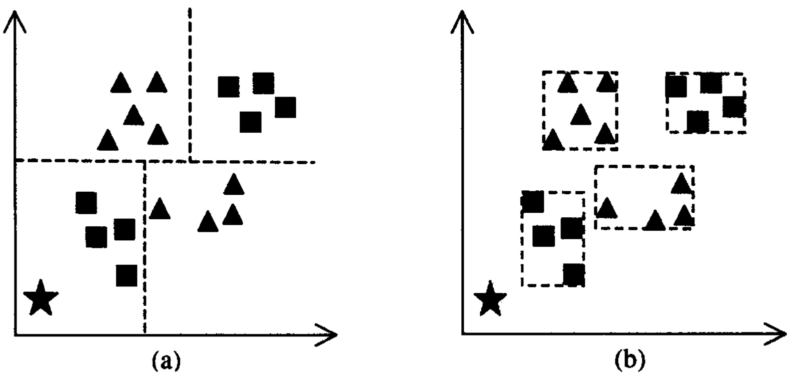

FIG. 5. Examples of Most Generalization (a) and Least Generalization (b) Defined on a 2D Feature Space, taken from Gahegan and West (1998). Squares and diamonds represent training data, and the star an unknown object to be classified.

any known examples, the most likely class is the nearest one, then most generalization would seem a logical choice. However, if the concepts to be classified are highly clustered, or if a large distance from a cluster is indicative of an "unknown" class, or a data error, then least generalization would appear more suitable.

As can be seen, bias results from the choice of class representation and the search for the best hypothesis that the chosen classifier employs. Many other forms of bias affect classifiers, originating from the testing process, the penalization or evaluation of performance, the training data itself, and the reporting of accuracy. These are discussed in the following section.

## 3. PROBLEMS THAT ARISE WITH GEOGRAPHIC APPLICATIONS

Having now described the general functioning of inductive learning tools, this section concentrates on specific problems that are a consequence, at least in part, of the geographic data to be analyzed. It should be borne in mind that many of these tools were originally developed and tested in domains where training and validation data are in abundant supply, and where data quality can be strictly controlled—in short, where the classification problem is considered *well posed.* Of necessity, many geographic problems cannot meet such stringent definition, and therefore might be considered *ill posed.*

The problems described in the following two subsections are aimed at the following question: How do we tell the difference between an ill-posed problem and a poorly configured classifier? Following from this is a subsection describing attribute complexities that are specific to, or characteristic of, geographic data domains and thus require special attention. The final subsection briefly discusses the problem of knowledge acquisition once the learning tools have been configured satisfactorily.

### *3.1 Training Sample*

*3.1.1 Performance Evaluation.* It is usually necessary to establish a figure of merit for the performance of the classifier. The methodology differs here from the typical MLC approach in that it is first necessary to select a classifier (from $\{h\}$), then to evaluate the performance of this classifier. Ideally, selection and evaluation should be performed using data that is statistically independent from that used in training. The reasons evaluation is critical are that (1) there are often many candidate classifiers to choose from and (2) since the classifier has an induced model rather than a predefined PDF, we must also test the goodness of fit of this model to the data (Prechelt 1994). Since the model was itself derived from the data, it follows that training and validation should use separate, ideally disjoint, data sets. We should expect estimates of performance to be inflated if the same data is used for training and validation—this represents a measure of how well the classifier has learned the training patterns themselves, not its ability to generalize from them to unseen examples. Note that a classifier having poor generalization capabilities may often score well when tested on the training data. Indeed, accuracy on the training data must often be sacrificed in order to achieve good generalization capabilities (section 3.2.3 below takes up this point).

*3.1.2 Insufficient Training Data.* Training data can be hard to come by, for a variety of reasons. A lack of training data has two distinct ramifications. Firstly, it restricts the accuracy with which concepts can be defined. The upper bound on the learning abilities of a classifier is determined by the training data and the generalization method. We should not expect a classifier to be able to handle situations for which it has no prior knowledge. Secondly, lack of data restricts the process of validation. If all the samples are required to build the classifier, then there remains no statistically independent source of data with which to validate performance and estimate accuracy. Robust validation techniques such as $k$-fold cross-validation (Schaffer 1993) require $1/k$ of the training examples to be withheld for validation. Large values of $k$ are required to increase statistical reliability of the validation result ($k = 10$ is commonly used). This requires that $1/k$ of the training data still contains a valid, representative sample. Where training data is in short supply, or cannot be obtained with relative ease, then validation can be an expensive enterprise, since it uses up much-needed training data. Hepner et al. (1990) discuss how this lack of training data can be managed using neural classifiers.

Whilst inductive techniques are claimed to be less "data hungry" than their parametric counterparts (section 1.1) they nevertheless require an adequate sampling for the purposes of generalization. Although various rules of thumb have been suggested to quantify "adequate" in this context, this remains an open question. Assuring that sufficient training data is provided is likely to become more difficult as feature spaces expand to match the ever-increasing supply of data attributes. However, the sampling need cover only those regions of feature space over which the classifier is expected to generalize. Data visualization (Tufte 1990; Kraak and MacEachren 1999) can provide a mechanism to explore at least a part of the feature space to examine the validity of training samples for the learning task.

*3.1.3 Bias in the Training Sample.* During learning, an inductive tool attempts to minimize some objective function across all presented training examples. A decision tree attempts to maximize "information gain" through the ability of the rules to describe the training data, a neural network minimizes the sum of errors via backpropagation. Patterns or classes that are underrepresented in the training sample will contribute less to these performance metrics, and hence will have less influence on the search for the best hypothesis. As a consequence, the classifier will concentrate initially on the larger classes, where the biggest performance gains are achievable. Learning will only focus on the smaller classes when overall error has been reduced sufficiently for their contribution (to that error) to become relevant (neural networks) or when the information gains offered by decision rules involving larger classes have already been realized (decision trees). Particularly when considering decision trees, it is uncertain that a global minimum might still be reached starting from the partial solution that will have been constructed *before* smaller classes are considered (section 2.1).

Unequal numbers of samples for each class can also have a dramatic effect on performance evaluation. It is typical to see classifier performance quoted in terms of percentage classified correct (PCC), a global measure of accuracy. Using this metric, all that is necessary to increase performance is to add in more training data for classes that are easily recognized [as shown by Fitzgerald and Lees (1994)], or remove samples from problematic classes. The use of PCC seems to stem from research communities where training data is in abundant supply (German 1999), allowing each class to be represented with an equal (and usually large) set of samples.

Training bias can be used advantageously if training data is abundant and an order of importance is assigned to the output classes; the classifier can be made to concentrate on the important classes simply by providing more training examples for them. However, for many types of geographical analysis (particularly landcover-related problems) the number of training samples often bears no relation to importance. Indeed, the converse may be true—important classes may have few examples, perhaps because the situation they describe is rare, inaccessible, or otherwise problematic to survey. Where sample sizes have no relationship to importance, performance measures must correct the sample bias. This can be achieved by normalizing the performance estimates by the number of classes so that each class contributes the same weight to the overall figure of merit. Examples of unbiased performance measures are the Kappa statistic (Congalton 1991) and the averaged normalized response (Gahegan, German, and West 1998).

*3.1.4 The Definition of Concepts or Classes.* Class descriptions frequently overlap, that is to say, they do not occupy entirely disjoint regions in feature space (a type 3 classifier as defined in section 1.3 above). Overlap can be due to three major factors: (1) the data contain attribute errors or noise; (2) the data are unable to provide a useful signature for the classes (poor characterization) or are inappropriate for the task of separating out the classes (poor differentiation); or (3) the classes overlap by definition. All three of these situations can arise ordinarily when dealing with a real-world problem (as opposed to artificial data sets). They are discussed in turn in the following paragraphs.

Attribute error (noise) is a natural consequence of many forms of data collection, so learning strategies must function under this condition (Angluin and Laird 1988). The cost metrics used to evaluate performance will tend to cause rare, spurious data values to be disregarded, since trying to explain them (via partitioning) is quite literally not worth the cost to the classifier. For a neural network, classifying an outlier involves reassigning one or more hyperplanes, usually leading to a large increase in error. For a decision tree, the information gain for an outlier is likely to be smaller than the gain criterion, so the point will be ignored. This situation breaks down in two cases. Firstly, where errors are systematic, there is likely to be an accumulation of evidence as a result of correlated noise, so the classifier will attempt to learn the pattern. Secondly, if all other data points have been correctly dealt with, these outliers will represent the only source of improving accuracy, so the classifier may attempt to learn them, often with disastrous results (see section 3.2.3 below on overfitting). The deeper issue here is that the choice of gain ratio or number of hyperplanes can offer some control over the classifier's interest in outliers and errors. It is usually wise to experiment with these controls for every new data set encountered.

Problems with characterization and differentiation may be due to lack of appropriate data, and the use of less suitable surrogates, a sometimes unavoidable issue where collection of the perfect data set is expensive or impractical (that is, many cases). Both of these problems will give rise to separability issues, where decision boundaries between classes are either extremely complex (and therefore do not tend to generalize well) or cannot be defined at all. Presented with this kind of problem, a neural network is unlikely to converge, since there exists no path through the hypothesis space that will result in substantial reduction of error. The behavior of a decision tree will depend somewhat on the setting of the gain criterion. If this is high, then the learning phase will terminate early; if it is low, the tree will become very complex (deep) without achieving a corresponding improvement in performance. Problems of class differentiation are often isolated to a pair or small number of classes. In this case, the classifier will concentrate on other regions in feature space where progress can be made, and will tend to ignore the problematic separation tasks. On a positive note, the problem will not cause the classifier to fail, but results will require careful analysis on a per-class basis to detect the problem. "Forced Learning" strategies can be used to focus the learning effort on these problematic areas, by presenting them early, repeatedly and by increasing the penalties associated with incorrect outcomes.

Classes may overlap simply as a consequence of the nature of the underlying phenomena themselves, measured or sampled at a given resolution (for example, transition zones between landcover types or fringe effects around urban regions). Where such overlapping or mixed classes exist, the notion of separate output classes is no longer appropriate. Neural networks are able to represent a more probabilistic outcome by employing continuous reporting (as opposed to discrete) at the output layer (indeed, this is often their standard method of operation). Since decision trees are normally restricted to discrete output, they require explicit definition of mixed classes, or merging of the problematic classes to overcome this problem, that is, $k$ must be made smaller or greater, by changing the fragmentation of the output domain.

*3.1.5 Penalization.* A related problem to overlapping classes arises in neural networks when defining a cost function for error back-propagation. Cost functions usually penalize a network for all but an entirely correct output. Considering a three-class problem tackled by a neural network, with values presented at the output layer for a particular training sample being [0.2, 0.1, 0.7]. Assume that class three (0.7) is indeed the correct class. The network penalizes all the nodes by back-propagating their error, in the hope of achieving an output of [0.0, 0.0, 1.0], (or possibly [0.1, 0.1, 0.9] depending on the type of network). However, in terms of dominant values, the network is already performing satisfactorily. Indeed, it may be performing as well as is possible if the training classes do overlap. Penalization causes the network to adjust hyperplanes where they may already be optimally placed, with no further performance gains possible. This can lead to instability or oscillation in the network, where attempts to improve performance actually have a negative effect that the network must then recover from, only to attempt the same change again. Modification of the cost function, so that only gross errors are penalized (German, Gahegan, and West 1997), can avoid this problem. One approach is known as the winner-takes-all strategy, where penalization results only if the dominant output value arises on an incorrect node. Another is to set a threshold below which a correct (but not maximal) dominant value is not penalized. Unfortunately, such a threshold must be determined experimentally, and there is no reason to suspect that the same threshold would suit all classes. A decision tree avoids the penalization issue altogether. Rules are applied in strict order of information gain, and that is the only criterion used for choice of one rule over another. If the sequential application of rules cannot offer some improvement at each separate iteration, the classes will remain undifferentiated. For some complex classification tasks this constraint can be too limiting, since separation may only begin to occur when several rules are already in place.[5]

## *3.2 Computational Complexity and Search*

*3.2.1 The Complexity of Feature Space—Computational Scaling.* Obviously, as additional features are added, the feature and hypothesis spaces ($X$ and $H$) expand and the scalability of classifiers becomes an issue of some importance. It is problematic to define how increases in $m$, $n$, and $k$ affect a neural network (as before, $m$ is the number of training examples, $n$ is the dimensionality of feature space, and $k$ is the number of output classes). Complexity is a combination of the size of the input vector, the number of hidden layer units, the number of output classes, the search mechanism, the cost function, and the number of training epochs. Furthermore, the complexity varies quite dramatically according to the specific methods used for search (for example, Moller 1993). However, the combination of sophisticated search mechanisms ensures that performance is less than $m^n$. Although computationally demanding, the more sophisticated search methods are able to reduce the number of training epochs required for convergence, sometimes by many orders of magnitude (that is, 100,000s of epochs to 100s of epochs). Similarly, the computational scaling behavior of artificial life methods is difficult to quantify, since it depends on a combination of many search parameters (section 2.1).

The performance of a decision tree such as C4.5 increases as $O(m^3)$, largely because of the need to continually sort attributes at each stage. Optimized tools, such as Ripper (Cohen 1995), can reduce this figure to $O(m(\log_m)^2)$, that is, approximately linearly with $m$ (Dietterich 1997). Decision trees also have the advantage that increases in $n$ also produce only a linear increase in performance, since each dimension is considered independently (section 2.1). Clearly, this gives significant advantages as the scale of the task increases. Self-organizing maps offer similar levels of performance.

It should be noted that, independent of the type of classifier, an additional multiplier is required to account for the number of complete training repetitions required for robust training and validation (section 3.1.1).

*3.2.2 The Size of the Hypothesis Space—Searching and Convergence.* Training is likely to lead to the exclusion of large regions of $H$ from consideration. However, a number of hypotheses that are equally consistent with the training data may still remain. Although this can be a result of problems with training data (see above), it may also be an indication of the difficulty in performing a reliable (but nonexhaustive) search over a complex $H$ space. Poor convergence, and hence poor classification accuracy, can result from data-related or method-related problems, or even a mixture of the two. All methods described can become trapped by local minima in $H$ that *appear* to be global. A large or steep local minimum may prove impossible to overcome; equally, a small but global minimum may be overlooked. Also, if the training data provides only weak characterization and differentiation (section 3.1.4), convergence may not be possible.

[5] It is possible, however, for a decision tree to evaluate not simply the next rule, but the total gain over $n$ consecutive rules, thus providing a mechanism by which decision boundaries can be grouped together, perhaps to tackle a complex region of feature space.

In addition, neural networks are susceptible to the following problems: (1) Oscillating behavior due to penalization, as described above in section 3.1.5. (2) Inappropriate searching methods or inappropriately configured searching methods. The various search strategies available for a neural network use different criteria. Some require the configuration parameters such as the "step size" and "momentum term" that determine the granularity of the search and the sensitivity to local perturbation. Others attempt to compute these terms regularly by examining the local neighborhood for rates of change over distance. Choice of searching method (and its configuration, if required) is yet another task required of the user. (3) Network saturation, where the weights on the interconnections between nodes in a network reach saturation, leading to poor convergence. The **W** and **U** matrices, which describe the weighted connections between nodes (**W** matrix connects the input layer to the hidden layer, **U** matrix connects the output layer to the hidden layer) operate with a restricted dynamic range, typically 0 to 1.0. As the number of connections increases, it becomes more difficult to compute weight values that correctly represent the relative importance of a particular connection (that is, the evidence it propagates). Benediktsson, Swain, and Ersoy (1993) show how performance can be improved by using a more sophisticated search method. Fischer and Staufer (1999) present a comparison of four search methods, with the conclusion that gradient descent techniques perform with greatest reliability.

Self-organizing maps require fewer control parameters, but the user must select the number of reference vectors and this determines the flexibility with which the classifier can partition feature space (analogous to the number of nodes in the hidden layer of a neural network). Decision trees can fail to learn because of their inflexible searching, since each decision enforces a partitioning of the attribute space that cannot be "undone." The initial partitioning may ultimately restrict movement toward a global minimum, because the parts of $H$ that can still be examined are partly determined by previous decisions. This inability to search over more than one dimension concurrently probably accounts for the majority of any accuracy benefit that neural networks can offer when compared to decision trees. Evolutionary approaches perhaps represent the greatest challenge because of their more erratic searching behavior. One problem is that the population tends to congregate around specific sites (crowding). To some extent, this can be overcome by a repulsion factor, or by sharing the measure of fitness between all members competing for a site. These two measures encourage further exploration of the feature space, generating more feasible hypotheses.

Overarching all these considerations is the configuration of $H$ itself. It must contain at least one good approximation of $f$ for there to be any chance of success at all. The usual assumption is that the input variables, together with a suitable configuration for the classifier or pattern detector will ensure $H$ is well configured. As can be seen, there are a number of practical reasons why such an assumption may be unwise.

*3.2.3 Overfitting.* Overfitting is the result of learning past the general pattern encoded in the training data and on to the individual nuances of the training examples used. The graph in Figure 6 shows how accuracy of the classifier changes as learning progresses. Typically, accuracy computed for validation will lag behind that for training since the classifier will show bias toward the actual data values in the training set. When overfitting begins to occur, the validation performance starts to decrease; at this point the classifier is attempting to minimize training error on patterns that cannot be generalized to the validation set. If allowed to continue, accuracies of 100 percent on training are often

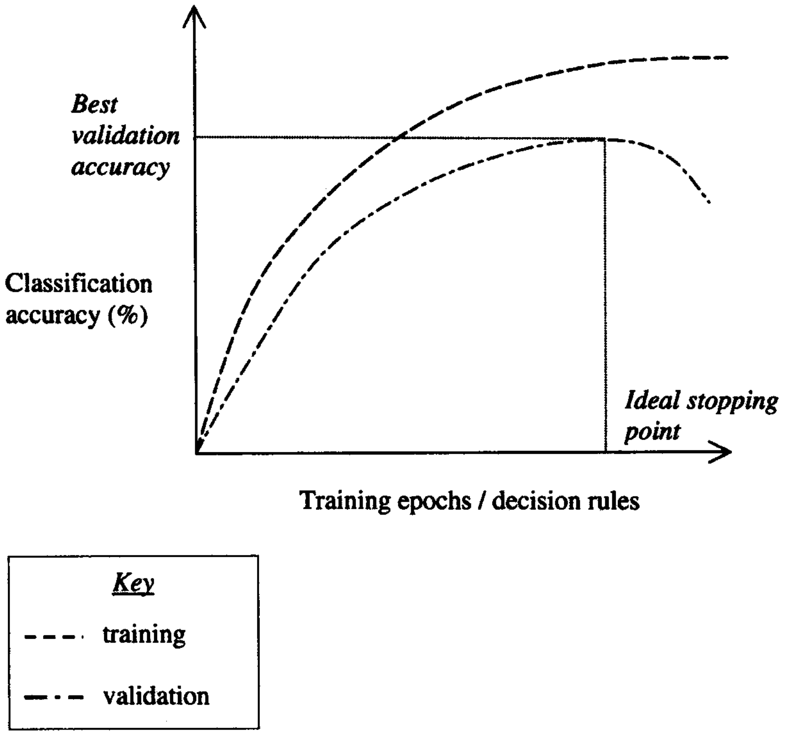

FIG. 6. A Comparison of Training and Validation Accuracy during the Process of Learning. In practice, the curves may exhibit some finer-scale movement for neural classifiers, as the network struggles to overcome local minima at first, then oscillates between progressively smaller improvements.

achievable—given enough hyperplanes or decision rules—but this is highly unlikely to provide the best classifier. The point at which validation performance drops off is not predictable in advance, and in the case of neural networks, will vary with the number of hyperplanes used. One way to avoid overtraining is to periodically test classifier performance on the validation set whilst training is still in progress. When validation begins to decrease, the training is halted and backed up to the configuration where validation performance is maximized. It can be argued, however, that such a method requires yet further validation data; the validation set is no longer independent of the classifier, since it was used to test for overfitting.

Neural networks are particularly problematic where overfitting is concerned because of the importance of ensuring that there are enough hyperplanes available to characterize the general trends in the training data reliably (section 1.3). Indeed, learning the classifier is often reduced to the task of finding the "best" configuration of hyperplanes. A common practice is to provide too many hyperplanes initially, then to prune the network after convergence. Pruning involves removing hyperplanes that are redundant or do not offer a significant improvement to classifier performance, and can be achieved via a systematic testing and removal. A well-pruned network will often achieve better generalization, again by sacrificing training accuracy. Moody (1992) and Gahegan, German, and West (1998) describe methods to make a more informed estimate of hyperplane requirements.

Decision trees are subject to the same problems with a similar solution; post-pruning is carried out on the rules produced during training by removing pre-conditions that do not contribute to the overall accuracy or that are in conflict with the validation data. Self-organizing maps can also overfit the training data but have the advantage that a defined region (cluster) can contain more than one generator, thereby absorbing any surplus learning capacity so that it does not interfere with convergence.

## 3.3 *The Nature of Geographic Domains*

*3.3.1 Geographical Space in Feature Space.* An important consideration for any geographer is the representation of geographic space within the classifier. In the approaches described, all attributes are combined together into the hypercubic feature space described in section 2; no distinction whatsoever is made between the spatial (or temporal) domain and the attributes pertaining to it. In practice, this equates to a geographic space with no imposed structure, except that values can be compared numerically, that is, distance has meaning. Two questions immediately come to mind: does this matter, and if it does, what can be done about it?

Firstly, many traditional classification methodologies ignore spatial attributes anyway, the assumption being that the domain contains little information that can be used easily, or that the problems using it outweigh the benefits. To be of benefit, the data used to train the classifier must first of all have location included, then this data must contain information (structure) that is helpful. Unfortunately, data gathering practices can render the information contained in the spatial domain useless, or even worse, confounding. For example, if training examples are selected from a small known or sampled region then there will be a high degree of locality in the spatial domain. This locality is not at all helpful in identifying other similar regions, and in fact would bias the classifier away from other regions since the classifier learns (incorrectly) that specific ranges of values for $x$ and $y$ evident in the training data are good indicators of a particular class.

In order to be of use, the spatial values must help to characterize the phenomena of interest. This can be achieved if the training sample is randomly distributed within the spatial domain, and hence represents some measure of likelihood of occurrence for the targets. After learning, this will predispose the classifier to assume that an example belongs to class $C$ if its location was helpful in characterizing $C$ during the training phase. If location is a strong indicator of class membership, and particularly if other attribute values are weaker by comparison, then some performance benefit should follow. However, this kind of bias must be used guardedly; it is a reinforcement of any locational prejudice within the training data.

As can be seen from the description so far, the classifier assumes that each training example is independent, any autocorrelation effects within the data are not explicitly handled. Spatial statistics tells us that spatial association can cause overestimation of the strength of relationships. However, the classifier is using the data in a very directed manner, learning a function that combines the data in the most effective way to identify a given target. If autocorrelation influences the predictability of some condition, the classifier will learn to give less weight to the attributes affected. Likewise, if the information contained in specific dimensions of feature space is unhelpful then the classifier will learn to ignore them. However, this complicates the training task; the classifier now has to learn to ignore the "noise" whilst learning to identify the target classes. At best this will slow down training (section 3.2.1); at worst it may interfere with convergence (section 3.2.2).

Where local patterns in space or time contain useful information, they must be represented to the classifier effectively. One method of representing local spatial or temporal structure in data is to construct a neighborhood model for each location, by adding to it the attributes of surrounding locations or by averaging them. Neighborhood classifiers, operating across space and time, are common. They have been shown to offer considerable benefits when analyzing data that has such structure (for example, Jeon and Landgrebe 1992). Inductive learning tools are particularly suited to this kind of problem since they can operate on the large attribute vectors that result from appending spatial or temporal properties to training examples (for example, German and Gahegan 1996).

A deeper and more vexing question relates to the utility of the feature space from which the classifier operates. As currently described, feature space may contain a mixture of disjoint subspaces, perhaps representing traditional geographic themes such as sociodemographics, hydrology, spectral response, and space-time. The internal structure within these subspaces may be very different from their relationships one to another (Aspinall and Lees 1994), so much so that perhaps the classifier should be constructed as independent pieces that are then reconciled at a later stage.

As a final point, the way that space is represented to the classifier influences its usefulness. If each dimension in space (space-time) is represented as a separate variable $(x, y, (z, t))$, the learning task is quite complex. For each cluster the classifier has to learn the ranges of $x$ and $y(z, t)$ that together define the extents of the underlying space. A method that structures the data by location (such as Peano-Hilbert or Morton ordering) may be more effective (for example, Worboys 1995). Tesserral addressing schemes interleave the spatial attributes to form a single, compound key. On average, such schemes reduce the complexity of the spatial encoding, as shown in Figure 7 and so simplify the task of partitioning the spatial domains, making learning easier.

3.3.2 *The Mix of Statistical Types.* The claim that these new forms of classifiers can easily accommodate all statistical data types is not entirely true. Whilst they can be trained with quantitative (that is, interval and ratio), ordinal, and nominal data, the fact remains that search is conducted in a numerical space and all data must be transformed to this space prior to training. The exception here is decision trees, which were originally designed to operate only on nominal data, but were later extended to numeric data. As mentioned previously, al-

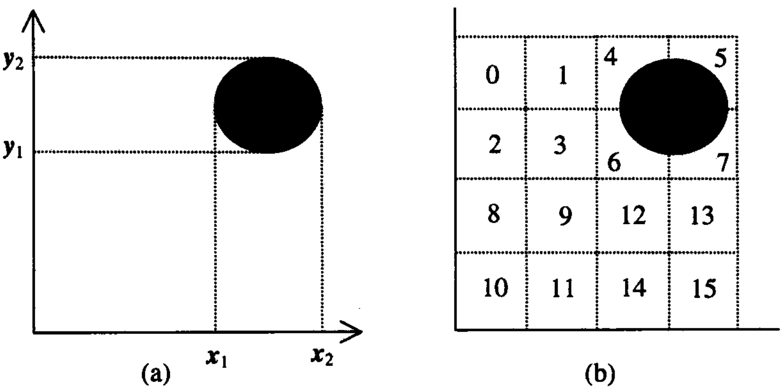

FIG. 7. Addressing Schemes for Spatial Encoding. Using Cartesian coordinates (a) the object is indexed by a range in $x$ and a range in $y$. Using Morton encoding (b), the same object occupies a single range of quadrants (numbers 4–7).

gebraic and Euclidean metrics provide the framework for measuring and comparing distance (from a hyperplane, from a cluster centroid, as a rule). For interval and ratio data types, this causes no real problems. However, ordinal mappings might be somewhat counterintuitive and nominal mappings can cause significant convergence problems.

The general solution adopted for both of these types is to enumerate the category labels in the order they appear, with equally spaced integers (for example, $1, 2, \ldots, n$). Ordinal data are so called precisely because this equal spacing or ordering does not hold, so the mapping is unlikely to be problem free. Consider the example where a data domain has been recorded as a series of categories given by an expert as normal, high, and extremely high. That there is an order is clear, what is unclear is the distance between each category. A straightforward enumeration will treat the intercategory intervals as consistent, leading to a linear penalization on any error function or performance criterion. In reality, these intervals may be nonlinearly spaced. This is not a new problem, but inductive learning tools do not offer a straightforward solution to it.

Nominal data are more problematic. The ordering of the data is now entirely arbitrary. Specifically, the separation complexity (the number of decision rules needed to produce the required classification accuracy) is dependent on *how* this nominal data is ordered when enumerated. Consider the case where several nominal values $\{A, B, C, D, E, F, G\}$ encode some condition such as soil type, three of which $\{A, B, C\}$ are indicators of a particular landcover class, shown as the squares in Figure 8. Depending on the ordering of these values, the number of hyperplanes required might vary between 1 and 6. Attempts to ordinate the data are likely to vary according to the specific task being undertaken; an entirely different ordering may be appropriate when trying to separate out other target landcover classes.

When using neural networks, this ordering problem further complicates the task of determining the correct number of hyperplanes required. With decision trees, the problem is more subtle in its manifestation and depends on how the information is presented to the classifier. If the traditional approach is taken, then a node operating on categorical data will split the domain into a single subtree for each category, whether or not such a total fragmentation is necessary. The learning task now has to proceed from each subtree so formed. Alternatively, the categories can be enumerated, in which case their ordering can become important as Figure 8 shows. If the categories are separated as shown

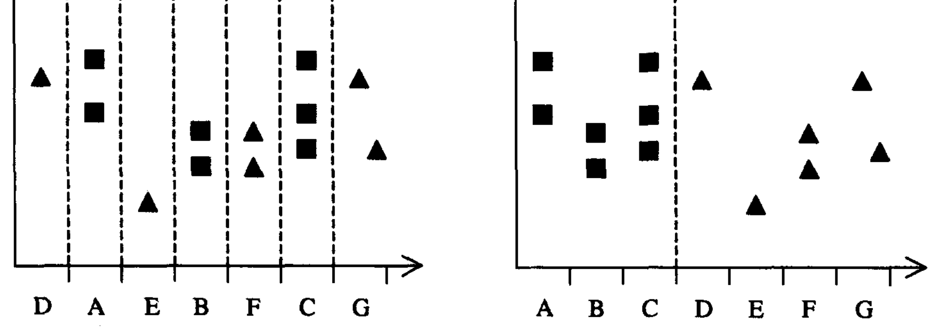

FIG. 8. Two Different Orderings for the Nominal Data Set $\{A, B, C, D, E, F, G\}$ in a Two-Class Problem.

on the left side of the figure, the information gain that each decision rule is able to offer will be reduced, and may even become insignificant. Notice that this is equivalent to the traditional fragmentation described previously. One conclusion is that if nominal data shows a consistent clustering for all target classes then it should be merged or ordered accordingly, to simplify the classification task.

It is worth also noting that although many of the attributes used in geographical analysis are measured on the interval or ratio scale, that is no guarantee of an inherently continuous domain. For example, attribute domains are often fragmented by entirely arbitrary boundaries (for example, height above and below water level) resulting in discontinuities across seemingly continuous axes. Other possible problems can result from variables where phase or state changes are possible. For example, temperature is a continuous domain, but the change at 0°C from water to ice can be highly significant and artificially fragments the domain.[6] A Euclidean distance that spans across such a boundary may not be a reliable measure of similarity.

*3.3.3 Dynamic Range of Input Data.* Since decision trees calculate a standardized measure of information gain in a single dimension at each iteration, the relative dynamic ranges of data domains are of no consequence. For a neural network, differences in dynamic range can be managed through the weights (and node bias term if used) connecting the input nodes to the hidden layer. Weights effectively balance the information content of each data channel by modification through the back-propagation process. However, training times can be lowered and reliability improved if data is scaled or normalized (say, $0 \rightarrow 1$) prior to training. This reduces the complexity of the training task, effectively placing the starting point in $H$ closer to the optimal solution. Under certain assumptions the starting point can be moved even further toward $h^*$ by calculating initial weights according to a measure of discrimination between classes, calculated prior to training by using a conventional linear classifier (German and Gahegan 1996). Self-organizing maps are also sensitive to wide differences in dynamic range and perform better when scaling is applied prior to training.

## *3.4 Knowledge Discovery*

As can be seen from the above descriptions, an inductive learning tool, once trained, contains specialized knowledge about the problem and data used. However, this knowledge is represented in a highly abstract and parameterized form. In the case of neural methods, this form has no direct relation to the feature space, being instead a representation of weight-space. Mapping the learning task to a geographically relevant form offers the possibility of learning about the classifier, and hence, by induction, the domain itself. One possible technique is to report on the performance of the classifier over distinct separation tasks (say, pairwise between all classes), another is to report on the number of rules or hyperplanes that were actually required for a specific task. This is easiest for a decision tree: classes are defined by all rules from each leaf node representing the class to the root of the tree. The complicating factors are likely to be the large number of rules, that they are hierarchically structured, and that they are expressed as tests of inequality, making them difficult to interpret visually. The task is more complex for neural networks since the relationship between controlling parameters and feature space is a good deal more complex, and also more difficult to interpret. For example, it is not trivial to determine which hyperplanes are contributing to a particular class definition, but once it is done, it helps to turn a predicting tool into a modeling tool (Garson 1991).

The final weights associated with the input layer nodes (neural networks) or the depth at which a rule appears (decision trees) also give some indication of the relative importance of particular features to the overall task.

In terms of knowledge discovery, the setting up of the classifier probably represents a richer source of knowledge than the outcome from the tool itself, but this portion of the knowledge is presently not captured by the tool in any useful form. Mining the internal representation of $f'$ is another area of current interest, now referred to as meta-learning or "learning about learning" (Bradsil and Konolige 1990), with the aim of capturing the expertise and parameters in the configuration that may be applicable to other problems.

## 4. CONCLUSIONS

This paper has described a number of approaches to inductive learning in a geographic setting. Inductive approaches may seem inherently inferior because the "best" solution is not guaranteed. It never is. Parametric approaches have to make many assumptions in order to guarantee a best solution. However, inductive learning tools are not a panacea for all the ills of existing statistically based methods, neither does their use guarantee an increase in accuracy. Whilst they certainly can overcome some of the difficulties of more established methods, they introduce problems of their own, namely, computational performance, generalization behavior, and configuration. Neither can they yet overcome all of the difficulties inherent in geographic data without human guidance, but instead place additional demands on the user, particularly in the selection of operational parameters. These considerations underline the need for thoughtful experimentation followed by validation on independent data, to ensure that $H$ is explored fully.

If the complexity of the classification task is completely unknown, then decision trees may present the best alternative, since they are more straightforward to configure, and hence usually more robust. Their search mechanism, although somewhat inflexible, is least sensitive to any operational parameters that must be supplied by the user. If accuracy is the paramount consideration then the global searching employed by neural networks would seem to facilitate better performance, but the increase in accuracy is usually funded via a good deal of experimentation to find a suitable network configuration. If the feature space is complex and irregular then a genetic approach may well meet with the best success because of its many starting points and less smooth searching behavior.

Approaches to classification and clustering based on inductive learning can offer substantial gains on complex problems and equally may not be worth the effort on simple ones. Since parametric techniques do not scale well as the number of attributes increases ($n$ becomes large), the adoption of a different approach is necessitated. And with the increasingly rich data sets available to geographers, the methodological problems described above must now be addressed.

## LITERATURE CITED

Anderberg, M. R. (1973). *Cluster Analysis for Applications*, Boston: Academic Press.

Angluin, D., and P. Laird (1988). "Learning from Noisy Examples." *Machine Learning* 2, 343–70.

Benediktsson, J. A., P. H. Swain, and O. K. Ersoy (1990). "Neural Network Approaches versus Statistical Methods in Classification of Multisource Remote Sensing Data." *IEEE Transactions on Geoscience and Remote Sensing* 28 (4), 540–51.

——— (1993). "Conjugate Gradient Neural Networks in Classification of Multisource and Very High Dimensional Remote Sensing Data." *International Journal of Remote Sensing* 14 (15), 2883–2903.

Benjamin, D. P., ed. (1990). *Change in Representation and Inductive Bias*. Boston, Mass.: Kluwer Academic Press.

Bennett, D. A., G. A. Wade, and M. P. Armstrong (1999). "Exploring the Solution Space of Semistructured Geographical Problems with Genetic Algorithms." *Transactions in GIS* 3 (1), 51–72.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York: Clarendon Press.

Bradsil, P. B., and K. Konolige, eds. (1990). *Meta-Learning, Meta-Reasoning, and Logics*. Boston, Mass.: Kluwer Academic Press.

Briscoe, G., and T. Caelli (1996). *A Compendium of Machine Learning*, vol. 1: *Symbolic Machine Learning*. Norwood, N. J.: Ablex Publishing Corporation.

Byungyong, K., and D. A. Landgrebe (1991). "Hierarchical Decision Tree Classifiers in High Dimensional and Large Class Data." *IEEE Transactions on Geosciences and Remote Sensing* 29 (4), 518–28.

Carpenter, G. A. (1989). "Neural Network Models for Pattern Recognition." *Neural Networks* 2, 243–57.

Civco, D. L. (1993). "Artificial Neural Networks for Landcover Classification and Mapping." *International Journal of Geographical Information Systems* 7 (2), 173–86.

Cohen, W. W. (1995). "Fast, Effective Rule Induction." Proceedings of the 12th International Conference on Machine Learning, pp. 115–23. San Francisco, Calif.: Morgan-Kaufmann.

Congalton, R. G. (1991). "A Review of Assessing the Accuracy of Classifications of Remotely Sensed Data." *Remote Sensing of the Environment* 37, 35–46.

Devijver, P. A., and J. Kittler (1982). *Pattern Recognition: A Statistical Approach*. London: Prentice-Hall International.

Dietterich, T. G. (1997). "Machine Learning Research: Four Current Directions." *AI Magazine*, Winter 1997, pp. 97–136.

Dunteman, G. H. (1984). *Introduction to Multivariate Analysis*. Beverly Hills, Calif.: Sage.

Evans, F. H., H. T. Kiiveri, G. West, and M. Gahegan (1998). "Mapping Salinity Using Decision Trees and Conditional Probabilistic Networks." *Proceedings of the Second IEEE International Conference on Intelligent Processing Systems*. Gold Coast, Queensland, Australia.

Fischer, M. M. (1994). "Expert Systems and Artificial Neural Networks for Spatial Analysis and Modelling." *Geographical Systems* 1, 221–35.

Fischer, M. M., and Y. Leung (1998). "A Genetic-algorithms Based Evolutionary Computational Neural Network for Modeling Spatial Interaction Data." *Annals of Regional Science* 32, 437–58.

Fischer, M. M., and P. Staufer (1999). "Optimization in an Error Backpropagation Neural Network Environment with a Performance Test on a Spatial Pattern Classification Problem." *Geographical Analysis* 31 (April), 89–108.

Fitzgerald, R. W., and B. G. Lees (1994). "Assessing the Classification Accuracy of Multisource Remote Sensing Data." *Remote Sensing of the Environment* 47, 362–68.

Foody, G. M., M. B. McCulloch, and W. B. Yates (1995). "Classification of Remotely Sensed Data by an Artificial Neural Network: Issues Relating to Training Data Characteristics." *Photogrammetric Engineering and Remote Sensing* 61, 391–401.

Foody, G. M., and M. K. Arona (1997). "An Evaluation of Some Factors Affecting the Accuracy of Classification by an Artificial Neural Network." *International Journal of Remote Sensing* 18 (4), 799–810.

Friedl, M. A., and C. E. Brodley (1997). "Decision Tree Classification of Landcover from Remotely Sensed Data." *International Journal of Remote Sensing* 18 (4), 711–25.

Gahegan, M., G. German, and G. West (1998). "Some Solutions to Neural Network Configuration Problems for the Classification of Complex Geographic Data sets." *Geographical Systems* 5 (4).

Gahegan, M., and G. A. W. West (1998). "A Theoretical Approach to the Use of Artificial Intelligence Techniques in the Classification of Complex Geographic Data sets." *Third International Conference on GeoComputation*, edited by P. Longley and B. Abrahart. University of Bristol, England.

Garson, G. D. (1991). "Interpreting Neural Network Connection Weights." *AI Expert*, April 1991, pp. 47–51.

German, G. (1999). "Neural Networks Configured as Classifiers of Landcover." Ph.D. thesis, School of Computing, Curtin University of Technology, Perth, Western Australia.

German, G., and M. Gahegan (1996). "Neural Network Architectures for the Classification of Temporal Image Sequences." *Computers and Geosciences* 22 (9), 969–79.

German, G., M. Gahegan, and G. West (1997). "Predictive Assessment of Neural Network Classifiers for Applications in GIS." In *Second International Conference on GeoComputation*, edited by R. Pascoe, pp. 41–50. University of Otago, Dunedin, New Zealand.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Mass.: Addison-Wesley.

Gopal, Sucharita, and Manfred M. Fischer (1996). "Learning in Single Hidden-Layer Feed-forward Network Models." *Geographical Analysis* 28 (1), 38–55.

Gould, P. R. (1970). "Is Statistix Inferens the Geographical Name for A Wild goose?" *Economic Geography* 46, 539–48.

Grossberg, S. (1991). "Neural Pattern Discrimination." In *Pattern Recognition by Self-Organizing Neural Networks*, edited by G. A. Carpenter and S. Grossberg, pp. 111–56. Cambridge, Mass.: MIT Press.

Gurney, C. M. (1983). "The Use of Contextual Information in the Classification of Remotely Sensed Data." *Photogrammetric Engineering and Remote Sensing* 49 (1), 55–64.

Hepner, G. F., T. Logan, N. Ritter, and N. Bryant (1990). "Artificial Neural Network Classification Using a Minimal Training Set: Comparison to Conventional Supervised Classification." *Photogrammetric Engineering and Remote Sensing* 56, 469–73.

Hunt, E. B., J. Marin, and P. JStone, P. J. (1966). *Experiments in Induction*, New York, Academic Press.

Jehng-Jung, K. (1996). "A Program to Train A Neural Network for Grid Pattern Recognition." *Computers and Geosciences* 22 (9), 1033–49.

Jeon, B., and D. A. Landgrebe (1992). "Classification with Spatio-temporal Interpixel Class Dependency Contexts." *Ieee Transactions on Geoscience and Remote Sensing* 30 (4), 663–72.

Johnson, R. A., and D. W. Wichern (1990). *Applied Multivariate Statistical Analysis*, 2d edition. Englewood, N. J.: Prentice-Hall.

Kanellopoulos, I., and G. Wilkinson (1997). "Strategies and Best Practice for Neural Network Image Classification." *International Journal of Remote Sensing* 18 (4), 711–25.

Kaufman, L., and P. J. Rousseeuw (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: Wiley.

Kohonen, T. (1995). *Self Organisation and Associative Memory*, 3rd edition. Berlin, Germany: Springer-Verlag.

Kraak, M.-J., and A. M. MacEachren, eds. (1999). *International Journal of Geographic Information Science: Special Issue on Exploratory Cartographic Visualization*, 13 (4).

Lakoff, G. (1987). *Women, Fire and Dangerous Things: What Categories Reveal about the Mind*. Chicago, Ill.: University of Chicago Press.

Landgrebe, D. (1999). "Information Extraction Principles and Methods for Multispectral and Hyperspectral Image Data." In *Information Processing for Remote Sensing*, edited by C. H. Chen. River Edge, N.J.: World Scientific Press.

Lees, B. G., and Ritman, K. (1991). "Decision Tree and Rule Induction Approach to Integration of Remotely Sensed and GIS Data in Mapping Vegetation in Disturbed or Hilly Environments." *Environmental Management* 15, 823–31.

Macgill, J., and S. Openshaw (1998). "The Use of Flocks to Drive a Geographic Analysis Machine." Presented at the Third International Conference on GeoComputation, Bristol, England. http://www.ccg.leeds.ac.uk/james/geocomp/ (accessed October 1999).

MacQueen, J. B. (1967). "Some Methods for Classification and Analysis of Multi-variate Observations." In *Proceedings of the 5th Symposium on Mathematics, Statistics, and Probability*, vol. 1, pp. 281–97. Berkeley, Calif.

Mardia, K. V., T. Kent, and J. M. Bibby (1979). *Multivariate Analysis*. London: Academic Press.

Martin, J. C. (1991). *Introduction to Languages and the Theory of Computation*. New York: McGraw Hill.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge, Mass.: MIT Press.

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw Hill.

Moller, M. F. (1993). "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning." *Neural Networks* 6, 525–33.

Moody, J. (1992). "The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems." *Advances in Neural Information Processing Systems* edited by J. E. Moody, S. J. Hanson, and R. P. Lippmann, pp. 847–54. San Mateo, Calif.: Morgan Kaufmann.

Moret, B. M. E., and H. D. Shapiro (1991). *Algorithms from P to NP*. Redwood, Calif.: Benjamin-Cummings.

Murphy, S., S. Kasif, and S. Salzberg (1994). "A System for Induction of Oblique Decision Trees." *Journal of Artificial Intelligence Research* 2, 1–32.

Openshaw, S. (1993). "Modelling Spatial Interaction Using a Neural Net." In *Geographic Information*

*Systems, Spatial Modelling, and Policy Evaluation*, edited by M. M. Fischer and P. Nijkamp, pp. 147–64. London: Springer-Verlag.

Openshaw, S., and R. J. Abrahart (1996). "Geocomputation." *Proceedings of 1st International Conference on GeoComputation*, edited by R. J. Abrahart, pp. 665–66. University of Leeds.

Pao, Y. H. (1989). *Adaptive Pattern Recognition and Neural Networks*. Reading, Mass.: Addison-Wesley.

Paola, J. D., and R. A. Schowengerdt (1995). "A Detailed Comparison of Backpropagation Neural Networks and Maximum-likelihood Classifiers for Urban Land Use Classification." *IEEE Transactions on Geosciences and Remote Sensing* 33 (4), 981–96.

Pierce, C. S. (1878). "Deduction, Induction and Hypothesis." *Popular Science Monthly* 13, 470–82.

Prechelt, L. (1994). "A Study of Experimental Evaluations of Neural Network Learning Algorithms: Current Research Practice." Technical Report, Faculty of Informatics, University of Karlsruhe, Germany.

Quinlan, R. (1993). *C4.5: Programs for Empirical Learning*. San Mateo, Calif.: Morgan Kaufmann.

Richards, J. A. (1994). *Remote Sensing Digital Image Analysis: An Introduction*, 2d edition. Berlin: Springer-Verlag.

Schaffer, C. (1993). "Selecting A Classification Method by Cross-Validation." *Machine Learning* 13, 135–43.

Sonka, M., V. Hlavac, and R. Boyle (1993). *Image Processing, Analysis and Machine Vision*. London: Chapman and Hall.

Tufte, E. R. (1990). *Envisioning Information*. Cheshire, Conn.: Graphics Press.

Utgoff, P. E. (1986). *Machine Learning of Inductive Bias*. Boston, Mass.: Kluwer Academic Press.

Webster, R., and M. A. Oliver (1990). *Statistical Methods in Soil and Land Resource Survey*. New York: Oxford University Press.

Worboys, M. F. (1995). *GIS: A Computing Perspective*. London: Taylor & Francis.

Yoshida, T., and S. Omatu (1994). "Neural Network Approaches to Landcover Mapping." *IEEE Transactions on Geosciences and Remote Sensing* 32 (5), 1103–09.