Improved Rooftop Detection in Aerial Images with Machine Learning

M.A. Maloof (maloof@cs.georgetown.edu)
Department of Computer Science, Georgetown University, Washington, DC 20057, USA

P. Langley (langley@isle.org)
Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA 94306, USA

T.O. Binford (binford@cs.stanford.edu)
Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305, USA

R. Nevatia (nevatia@iris.usc.edu)
Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California, Los Angeles, CA 90089, USA

S. Sage (sage@isle.org)
Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA 94306, USA

Abstract. In this paper, we examine the use of machine learning to improve a rooftop detection process, one step in a vision system that recognizes buildings in overhead imagery. We review the problem of analyzing aerial images and describe an existing system that detects buildings in such images. We briefly review four algorithms that we selected to improve rooftop detection. The data sets were highly skewed and the cost of mistakes differed between the classes, so we used ROC analysis to evaluate the methods under varying error costs. We report three experiments designed to illuminate facets of applying machine learning to the image analysis task. One investigated learning with all available images to determine the best performing method. Another focused on within-image learning, in which we derived training and testing data from the same image. A final experiment addressed between-image learning, in which training and testing sets came from different images. Results suggest that useful generalization occurred when training and testing on data derived from images differing in location and in aspect. They demonstrate that under most conditions, naive Bayes exceeded the accuracy of other methods and a handcrafted classifier, the solution currently used in the building detection system.

Keywords: supervised learning, learning for computer vision, evaluation of algorithms, applications of learning.

1. Introduction

The number of images available to image analysts is growing rapidly, and will soon outpace their ability to process them. Computational aids will be required to filter this flood of images and to focus the analyst's attention on interesting events, but current image understanding systems are not yet

© 2003 *Kluwer Academic Publishers. Printed in the Netherlands.*

robust enough to support this process. Successful image understanding relies on knowledge, and despite theoretical progress, implemented vision systems still rely on heuristic methods and consequently remain fragile. Handcrafted knowledge about when and how to use particular vision operations can give acceptable results on some images but not on others.

In this paper, we explore the use of machine learning as a means for improving knowledge used in the vision process, and thus for producing more robust software. Recent applications of machine learning in business and industry (Langley & Simon, 1995) hold useful lessons for applications in image analysis. A key idea in applied machine learning involves building an *advisory* system that recommends actions but gives final control to a human user, with each decision generating a training case, gathered in an unobtrusive way, for use in learning. This setting for knowledge acquisition is similar to the scenario in which an image analyst interacts with a vision system, finding some system analyses acceptable and others uninteresting or in error. The aim of our research program is to embed machine learning into this interactive process of image analysis.

This adaptive approach to computer vision promises to greatly reduce the number of decisions that image analysts must make per picture, thus improving their ability to deal with a high flow of images. Moreover, the resulting systems should adapt their knowledge to the preferences of individuals in response to feedback from those users. The overall effect should be a new class of systems for image analysis that reduces the workload on human analysts and gives them more reliable results, thus speeding the image analysis process.

In the sections that follow, we report progress on using machine learning to improve decision making at one stage in an existing image understanding system. We begin by explaining the task domain—identifying buildings in aerial photographs—and then describe the vision system designed for this task. Next, we review four well-known algorithms for supervised learning that hold potential for improving the reliability of image analysis in this domain. After this, we report the design of experiments to evaluate these methods and the results of those studies. In closing, we discuss related and future work.

2. Nature of the Image Analysis Task

Image analysts interpret aerial images of ground sites with an eye to unusual activity or other interesting behavior. The images under scrutiny are usually complex, involving many objects arranged in a variety of patterns. Overhead images of Fort Hood, Texas, collected as part of the RADIUS project (Firschein & Strat, 1997), are typical of a military base and include buildings

in a range of sizes and shapes, major and minor roadways, sidewalks, parking lots, vehicles, and vegetation. A common task analysts face is to detect change at a site as reflected in differences between two images, as in the number of buildings, roads, and vehicles. This in turn requires the ability to recognize examples from each class of interest. In this paper, we focus on the performance task of identifying buildings in satellite photographs.

Aerial images can vary across a number of dimensions. The most obvious factors concern viewing parameters, such as distance from the site (which affects size and resolution) and viewing angle (which affects perspective and visible surfaces). But other variables also influence the nature of the image, including the time of day (which affects contrast and shadows), the time of year (which affects foliage), and the site itself (which determines the shapes of viewed objects). Taken together, these factors introduce considerable variability into the images confronting analysts.

In turn, this variability can significantly complicate the task of recognizing object classes. Although a building or vehicle will appear different from alternative perspectives and distances, the effects of such transformations are reasonably well understood. But variations due to time of day, the season, and the site are more serious. Shadows and foliage can hide edges and obscure surfaces, and buildings at distinct sites may have quite different structures and layouts. Such variations serve as mere distractions to a human image analyst, yet they provide serious challenges to existing computer vision systems.

This suggests a natural task for machine learning: given aerial images as training data, acquire knowledge that improves the reliability of such an image analysis system. However, we cannot study this task in the abstract. We must explore the effect of specific induction algorithms on particular vision software. In the next two sections, we briefly review one such system for image analysis and four learning methods that might give it more robust behavior.

3. An Architecture for Image Analysis

Lin and Nevatia (1998) reported a computer vision package, called the Buildings Detection and Description System (BUDDS), for the analysis of ground sites in aerial images. Like many programs for image understanding, their system operates in a series of processing stages. Each step involves aggregating lower level features into higher level ones, eventually reaching hypotheses about the locations and descriptions of buildings. We will consider these stages in the order that they occur.

Starting at the pixel level, BUDDS uses an edge detector to group pixels into edge elements, and then invokes a linear feature detector to group edge elements into lines. Junctions and parallel lines are identified and combined

to form three-sided structures, or "U-constructs." The algorithm then groups selected U-constructs and junctions to form parallelograms. Each such parallelogram constitutes a hypothesis about the position and orientation of the roof for some building, so we may call this step *rooftop generation*.

After the system has completed the above aggregation process, a *rooftop selection* stage evaluates each rooftop candidate to determine whether it has sufficient evidence to be retained. The aim of this process is to remove candidates that do not correspond to actual buildings. Ideally, the system will reject most spurious candidates at this point, although a final verification step may still collapse duplicate or overlapping rooftops. This stage may also exclude candidates if there is no evidence of three-dimensional structure, such as shadows and walls.

Analysis of the system's operation suggested that rooftop selection held the most promise for improvement through machine learning, because this stage must deal with many spurious rooftop candidates. This process takes into account both local and global criteria. Local support comes from features such as lines and corners that are close to a given parallelogram. Since these suggest walls and shadows, they provide evidence that the candidate corresponds to an actual building. Global criteria consider containment, overlap, and duplication of candidates. Using these evaluation criteria, the set of rooftop candidates is reduced to a more manageable size. The individual constraints applied in this process have a solid foundation in both theory and practice.

The problem is that we have only heuristic knowledge about how to combine these constraints. Moreover, such rules of thumb are currently crafted by hand, and they do not fare well on images that vary in their global characteristics, such as contrast and amount of shadow. However, methods of machine learning, to which we now turn, may be able to induce better conditions for selecting or rejecting candidate rooftops. If these acquired heuristics are more accurate than the existing handcrafted solutions, they will improve the reliability of the rooftop selection process.

4. A Review of Four Learning Techniques

We can formulate the task of acquiring rooftop selection heuristics in terms of supervised learning. In this process, training cases of some concept are labeled as to their class. In rooftop selection, only two classes exist—rooftop and non-rooftop—which we will refer to as positive and negative examples of the concept "rooftop." Each instance consists of a number of attributes and their associated values, along with a class label. These labeled instances constitute training data that are provided as input to an inductive learning routine, which generates concept descriptions designed to distinguish the

positive examples from the negative ones. Such knowledge structures state the conditions under which the concept, in this case "rooftop", is satisfied.

In a previous study (Maloof, Langley, Sage, & Binford, 1997), we evaluated a variety of machine learning methods on the rooftop detection task and selected the three that showed promise of achieving a balance between the true positive and false positive rates: nearest neighbor, naive Bayes, and C5.0, the commercial successor of C4.5. We also included the perceptron because, as we will see, it is similar to the classifier currently used in BUDDS. These methods use different representations, performance schemes, and learning mechanisms for supervised concept learning, and exhibit different inductive biases, meaning that each algorithm acquires certain concepts more easily than others.

The nearest-neighbor method (e.g., Aha, Kibler, & Albert, 1991) uses an *instance-based* representation of knowledge that simply retains training cases in memory. This approach classifies new instances by finding the "nearest" stored case, as measured by some distance metric, then predicting the class associated with that case. For numeric attributes, a common metric (which we use in our studies) is Euclidean distance. In this framework, learning involves nothing more than storing each training instance, along with its associated class. Although this method is quite simple and has a known sensitivity to irrelevant attributes, in practice it performs well in many domains. Some versions select the k closest cases and predict the majority class. For detection tasks, such as ours, one typically sets k to an odd number to prevent ties. We included both variants in our study.

The naive Bayesian classifier (e.g., John & Langley, 1995; Langley, Iba, & Thompson, 1992) stores a probabilistic concept description for each class. This description includes an estimate of the class probability and the estimated conditional probabilities of each attribute value given the class. The method classifies new instances by computing the posterior probability of each class using Bayes' rule, combining the stored probabilities by assuming that the attributes are independent given the class, and predicting the class with the highest posterior probability. Like nearest neighbor, naive Bayes has known limitations, such as sensitivity to attribute correlations, but in practice, it behaves well on many natural domains.

C5.0, the commercial successor of C4.5 (Quinlan, 1993), is a system that uses training data to induce decision trees, which are n-ary trees with leaves representing classes and with internal nodes corresponding to domain attributes. An internal node for a given attribute has links to nodes in the next level of the tree. Each link corresponds to a value or a range of values for the attribute.

The learning element of C5.0 builds a decision tree by selecting the attribute with values that best separate the training examples into the proper classes, by creating a node for that attribute, and by distributing the training

examples into newly created leaf nodes based on the values of that attribute. If all of the examples in a leaf node are of the same class, then construction stops; otherwise, the procedure continues recursively. C5.0 selects attributes by maximizing the *gain ratio criterion*, an information theoretic measure of homogeneity. The learning element also applies a pruning algorithm to induced trees as a post-processing step, which prevents over-fitting of the training data.

To classify an instance, the decision procedure starts at the root of the tree and follows the links as determined by the values that each attribute takes. When it reaches a leaf node, the procedure returns the class label of the node as the decision.

The continuous perceptron (e.g., Zurada, 1992) represents concepts using a vector of weights, $\mathbf{w}$, and a threshold, θ. Training a perceptron involves finding the weight vector and threshold by using a gradient descent technique to minimize the error between the desired output and the actual output of the classifier. Although it is well-known that the training algorithm for a discrete perceptron (i.e., with binary inputs) is guaranteed to converge to the optimal solution for linearly separable patterns, this result does not hold for the continuous version.

To classify an instance, which we represent as a vector of n real numbers, $\mathbf{x}$, the method computes the output, o, using the formula:

$$o = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} w_i x_i > \theta; \\ -1 & \text{otherwise.} \end{cases} \tag{1}$$

For our application, the classifier predicts the positive class if the output is $+1$ and predicts the negative class otherwise. Although this classifier did not fare well in previous studies (Maloof, Langley, Binford, & Sage, 1998; Maloof et al., 1997), we included it here because it is very similar to the classification method used in BUDDS, which we discuss next.

Currently, BUDDS uses a handcrafted linear classifier for rooftop detection (Lin & Nevatia, 1998) that is equivalent to a continuous perceptron classifier (Zurada, 1992). Although we did not train this classifier as we did the other methods, we included it in our evaluation for the purpose of comparison. Recall that one motivation of this study was to use learning algorithms to improve rooftop selection, so it was important to include this method as a baseline. Henceforth, we will refer to the handcrafted linear classifier used in BUDDS as the "BUDDS classifier."

5. Generating, Representing, and Labeling Rooftop Candidates

We were interested in how well the various induction algorithms could learn to classify rooftop candidates in aerial images. This required three things: a

Table I. Characteristics of the images and data sets.

Image Number	Location	Image Size	Aspect	Positive Examples	Negative Examples
1	A	2055×375	Nadir	71	2645
2	A	1803×429	Oblique	74	3349
3	B	670×645	Nadir	197	982
4	B	704×568	Oblique	238	1955
5	C	1322×642	Nadir	87	3722
6	C	1534×705	Oblique	114	4395

set of images that contain buildings, some means to generate and represent plausible rooftops, and labels for each such candidate.

As our first step, we selected six overhead images of Fort Hood, Texas, collected as part of the RADIUS program (Firschein & Strat, 1997). These images were acquired in the visible range of the light spectrum at resolutions between 1.2–1.7 pixels per meter, and quantized to 256 levels of intensity. They covered three different areas but were taken from two different viewpoints, one from a nadir aspect (i.e., directly overhead) and the other from an oblique aspect, as summarized in Table I. We selected images that contained concentrations of buildings to maximize the number of positive rooftop candidates.

In addition to differences in aspect and in location, these images also differed in their dimensions and in the number, size, shape, and height of the buildings therein. Some buildings were rectangular, both small and large, but others were L-shaped or irregularly shaped with rooftops formed of multiple rectangular sections of differing heights. For some buildings, BUDDS extracted rooftop candidates that aligned perfectly to actual rooftops, but it also produced candidates that only partially covered actual rooftops and that corresponded to clusters of cars, to sections of parking lots, to sidewalks and lawns, and to sides and shadows of buildings.

Our aim was to improve rooftop selection in BUDDS, so we used this system to process the images and generate candidate rooftops, thereby producing six data sets, one for each image. Following Lin and Nevatia (1998), the data sets described each rooftop candidate in terms of nine continuous features that summarize the evidence gathered from the various levels of analysis. For example, positive indications for the existence of a rooftop included evidence for edges and corners, the degree to which a candidate's opposing lines are parallel, support for the existence of orthogonal trihedral vertices, and evidence of shadows near the corners of the candidate. Negative evidence included the existence of lines that cross the candidate, L-junctions adja-

cent to the candidate, similarly adjacent T-junctions, gaps in the candidate's edges, and the degree to which enclosing lines failed to form a parallelogram. L-junctions are configurations of two linear features extracted from an image that form an L-shape. Similarly, T-junctions are configurations that form a T-shape.

We should note that induction algorithms are often sensitive to the features one uses to describe the data, and we make no claims that these nine attributes are the best ones for recognizing rooftops in aerial images. However, because our aim was to improve the robustness of BUDDS, we needed to use the same features as Lin and Nevatia's handcrafted classifier. Moreover, it seemed unlikely that we could devise better features than the system's authors had developed during years of research.

The third problem, labeling the generated rooftop candidates, proved the most challenging and the most interesting. BUDDS itself classifies each candidate, but since we were trying to improve on its ability, we could not use those labels. Thus, we tried an approach in which an expert specified the vertices of actual rooftops in the image, then we automatically labeled candidates as positive or negative depending on the distance of their vertices from the nearest actual rooftop's corners. We also tried a second scheme that used the number of candidate vertices that fell within a region surrounding the actual rooftop. Unfortunately, upon inspection neither approach gave satisfactory labeling results.

Analysis revealed the difficulties with using such relations to actual rooftops in the labeling process. One is that these schemes ignore information about the candidate's shape: A good rooftop should be a parallelogram; yet nearness of vertices to a true rooftop is neither necessary nor sufficient for this form. A second drawback is that they ignore other information contained in the nine BUDDS attributes, such as shadows and crossing lines. The basic problem is that such methods deal only with the two-dimensional space that describes location within the image, rather than the nine-dimensional space that we want the vision system to use when classifying a candidate.

Reluctantly, we concluded that manual labeling by a human was necessary, but this task was daunting, as each image produced thousands of candidate rooftops. To support the process, we implemented an interactive labeling system, shown in Figure 1, that successively displays each extracted rooftop to the user. The system draws each candidate over the portion of the image from which it was extracted, then lets the user click buttons for 'Roof' or 'Non-Roof' to label the example. Table I also lists the number of positive and negative examples extracted from each of the six images.[1]

[1] Oblique images yielded more negative examples, because in these, a building's walls were visible. Walls produced parallel, linear features, which BUDDS grouped into parallelograms. During labeling, we designated these constructs as negative.

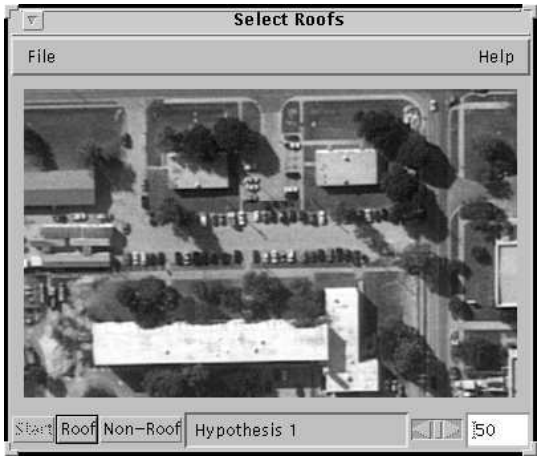

Figure 1. Visualization interface for labeling rooftop candidates. The system presents candidates to a user who labels them by clicking either the 'Roof' or 'Non-Roof' button. It also incorporates a simple learning algorithm to provide feedback to the user about the statistical properties of a candidate based on previously labeled examples.

The visual interface itself incorporates a simple learning mechanism—nearest neighbor—designed to improve the labeling process. As the system obtains feedback from the user about positive and negative examples, it divides unlabeled candidates into three classes: likely rooftops, unlikely rooftops, and unknown. The interface displays likely rooftops using green rectangles, unlikely rooftops as red rectangles, and unknown candidates as blue rectangles. The system includes a sensitivity parameter that affects how certain the system must be before it proposes a label. After displaying a rooftop, the user either confirms or contradicts the system's prediction by clicking either the 'Roof' or 'Non-Roof' button. The simple learning mechanism then uses this information to improve subsequent predictions of candidate labels.

Our intent was that as the interface gained experience with the user's labels, it would display fewer and fewer candidates about which it was uncertain, and thus speed up the later stages of interaction. Informal studies suggested that the system achieves this aim: By the end of the labeling session, the user typically confirmed nearly all of the interface's recommendations. However, because we were concerned that our use of nearest neighbor might bias the labeling process in favor of this algorithm, during later studies, we generated the data used in the experimental sections—Sections 6 and 8—by setting the sensitivity parameter so the system presented all candidates as uncertain. Even handicapped in this manner, it took the user only about five hours to label the 17,829 roof candidates extracted from the six images. This comes to under one second per candidate, which is quite efficient.

The consistency of labeling is an important issue, not just for a given expert, but also for different experts. Incorporating a learning method into the labeling system was an attempt to improve consistency for a particular individual. However, more interesting and challenging were the disagreements between experts. For instance, one argued that a parallelogram delineating a parking lot should be labeled as a rooftop because it was well-shaped, and BUDDS would remove it in later stages of processing when the system failed to find evidence of walls and shadows. Another argued to designate a misshapen parallelogram as a rooftop because it was the only rooftop candidate present in the data set for a particular building.

We have investigated disagreements among experts when labeling rooftops and the effect these have on learning and performance (Ali, Langley, Maloof, Sage, & Binford, 1998). In spite of disagreements on non-rooftops of 31% between experts and of 14% between labeling sessions for one expert, naive Bayesian classifiers built from these data sets performed almost identically. Moreover, these performances were notably better than that of the BUDDS classifier. This implies that at this stage in our work, such disagreements have little effect. However, if it becomes a greater problem in the future, then one solution is to train classifiers with each expert's labeled data and use ensemble methods to combine the outputs into a single decision. Such an approach may be impractical given the amount of data we have, so we could also use a voting scheme to either remove or weight the rooftop candidates on which they disagree.

In summary, what began as a simple task of labeling visual data led us to some of the more fascinating issues in our work. To incorporate supervised concept learning into vision systems that can generate thousands of candidates per image, we must develop methods to reduce the burden of labeling these data. In future work, we intend to measure more carefully the ability of our adaptive labeling system to speed this process. We also plan to explore extensions that use the learned classifier to order candidate rooftops (showing the least certain ones first) and even to filter candidates before they are passed on to the user (automatically labeling the most confident ones). Techniques such as *selective sampling* (Freund, Seung, Shamir, & Tishby, 1997), *uncertainty sampling* (Lewis & Catlett, 1994), or methods for learning from both labeled and unlabeled training data (Miller & Uyar, 1997) should prove useful toward these ends.

6. Experiment I: Evaluating the Methods Traditionally

After constructing a labeled data set and identifying four learning algorithms, we evaluated the methods empirically to determine which might outperform the BUDDS classifier. To accomplish this, we randomly split the labeled roof-

Table II. Results for the experiment using all of the image data. Measures are accuracy, true positive (TP) rate, false positive (FP) rate with 95% confidence intervals. Italics type shows the best measure in each column.

Method	Accuracy	TP Rate	FP Rate
C5.0	*0.963±0.003*	0.23±0.022	0.0034±0.0011
k-NN (k = 17)	0.961±0.001	0.19±0.015	0.0037±0.0003
k-NN (k = 11)	0.960±0.001	0.21±0.017	0.0056±0.0006
k-NN (k = 5)	0.957±0.001	0.23±0.010	0.0097±0.0009
Perceptron	0.957±0.001	0.02±0.011	*0.0001±0.0001*
BUDDS Classifier	0.917±0.001	0.54±0.018	0.0657±0.0008
Naive Bayes	0.908±0.003	*0.56±0.008*	0.0761±0.0036

top data into training (60%) and testing (40%) sets. We then ran each algorithm by training using the examples in the training set, by testing the resulting classifier using the examples in the testing set, and by computing the accuracy, the true positive rate, and the false positive rate. We conducted ten of these learning runs, averaging the performance metrics, which are shown in Table II. We ran k-NN for $k = 3,5,\ldots,19$, but for the sake of brevity, we present only the three best results for this method. We report these omitted measures elsewhere (Maloof et al., 1998).

An analysis of variance (Keppel, Saufley, & Tokunaga, 1992) indicated that these results were statistically significant at $p < .01$. We also used Duncan's test (Walpole, Myers, & Myers, 1998) to identify statistically significant subgroups of performance, also at $p < .01$. The means of the perceptron and of k-NN, for $k = 5$, were not significantly different. Further, the means of k-NN, for $k = 7,9,\ldots,19$, were not significantly different. However, the differences between naive Bayes and the BUDDS classifier and between C5.0 and k-NN, for $k = 17$, were statistically significant.

Using accuracy as our measure of performance, we concluded that C5.0 outperformed the BUDDS classifier by roughly five percent. However, looking at the false positive rate, we see that much of C5.0's superiority in performance was due to its success in identifying non-rooftops and that it was only fair at detecting rooftops, as the true positive rate indicates.

Since we were attempting to develop a better rooftop detector, we considered choosing the method that maximized the true positive rate, but that method, naive Bayes, while performing significantly better in the statistical sense, did not significantly eclipse the original BUDDS classifier in any other sense. As we will see in the following sections, the dissatisfying results that we obtained in this experiment were not due to a failing of the learning

methods. Rather, they were due to a shortcoming in our initial choice of an evaluation methodology.

7. Cost-Sensitive Learning and Skewed Data

Two aspects of the rooftop selection task influenced our subsequent approach to implementation and evaluation. First, BUDDS works in a bottom-up manner, so if the system discards a rooftop, it cannot retrieve it later. Consequently, errors on the rooftop class (false negatives) are more expensive than errors on the non-rooftop class (false positives), so it is better to retain a false positive than to discard a false negative. The system has the potential for discarding false positives in later stages of processing when it can draw upon accumulated evidence, such as the existence of walls and shadows. However, since false negatives cannot be recovered, we need to minimize errors on the rooftop class.

Second, we have a severely skewed data set, with training examples distributed non-uniformly across classes (781 rooftops vs. 17,048 non-rooftops). Given such skewed data, most induction algorithms have difficulty learning to predict the minority class. Moreover, we have established that errors on our minority class (rooftops) are most expensive, and the extreme skew only increases such errors. This interaction between a skewed class distribution and unequal error costs occurs in many computer vision applications, in which a vision system generates thousands of candidates but only a handful correspond to objects of interest. It also holds in many other applications of machine learning, such as fraud detection (Fawcett & Provost, 1997), discourse analysis (Soderland & Lehnert, 1994), and telecommunications risk management (Ezawa, Singh, & Norton, 1996).

These issues raise two challenges. First, they highlight the need to achieve higher accuracy on the minority class, whether through modified learning algorithms or altered distributions. Second, they require an experimental methodology that lets us compare different methods on tasks like rooftop detection, in which the classes are skewed and errors have different costs. In the remainder of this section, we further clarify the nature of the problem, after which we describe our cost-sensitive learning methods and an approach to experimental evaluation.

7.1. Favoritism Toward the Majority Class

In Section 6 and in a previous study (Maloof et al., 1997), we evaluated several algorithms without taking into account the cost of classification errors and obtained confusing experimental results. Some methods, like the standard error-driven algorithm for revising perceptron weights, learned to

always predict the majority class. The naive Bayesian classifier found a more comfortable trade-off between the true positive and false positive rates, but still favored the majority class. For data sets that are skewed, an inductive method that learns to predict the majority class will often have a higher overall accuracy than a method that finds a balance between the true positive and false positive rates.[2] Indeed, always predicting the majority class for our problem yields an accuracy of 0.95, which makes it a misleading measure of performance (see also Provost, Fawcett, & Kohavi, 1998).

This bias toward the majority class only causes difficulty when we care more about errors on the minority class. For the rooftop domain, if the error costs for the two classes were the same, then we would not care on which class we made errors, provided we minimized the total number of mistakes. Nor would there be any problem if mistakes on the majority class were more expensive, since most learning methods are biased toward minimizing such errors anyway. However, if the class distribution runs counter to the relative cost of mistakes, as in our domain, then we must take actions both to improve accuracy on the minority class and to refine our performance measure.

Breiman, Friedman, Olshen, and Stone (1984) noted the close relation between the distribution of classes and the relative cost of errors. In particular, they pointed out that one can mitigate the bias against the minority class by duplicating examples of that class in the training data. This also helps explain why most induction methods give more weight to accuracy on the majority class, since skewed training data implicitly places more weight on errors for that class. In response, several researchers have explored approaches that alter the distribution of training data in various ways, including use of weights to bias the performance element (Cardie & Howe, 1997), removing unimportant examples from the majority class (Kubat & Matwin, 1997), and "boosting" the examples in the under-represented class (Freund & Schapire, 1996). However, as we will see shortly, one can also modify the algorithms themselves to more directly respond to error costs.

7.2. Cost-Sensitive Learning Methods

Empirical comparisons of machine learning algorithms seldom focus on the cost of classification errors, possibly because most learning methods do not provide ways to take such costs into account. Happily, some researchers have explored variations on standard algorithms that effectively bias the method in favor of one class over others. For example, Lewis and Catlett (1994) introduced a *loss ratio* into C4.5 (Quinlan, 1993) to bias it toward under-represented classes. Pazzani et al. (1994) have also done some preliminary

[2] Covering algorithms, like AQ15 (Michalski, Mozetic, Hong, & Lavrac, 1986) or CN2 (Clark & Niblett, 1989), may be less susceptible to skewed data sets, but this is highly dependent on their rule selection criteria.

work along these lines, which they describe as addressing the costs of different error types. Their method finds the minimum-cost classifier for a variety of problems using a set of hypothetical error costs. Bradley (1997) presented results from an empirical evaluation of algorithms that take into account the cost of classification error, whereas Turney (1995) addressed the cost of tests to measure attributes. Domingos (1999) recently proposed a meta-learning approach for making classifiers cost-sensitive that involves relabeling training examples so their distribution is consistent with the cost of errors.

If we have n classes, then we can specify a cost matrix C, where c_{ij}, for $i, j = 1, \ldots, n$, is the cost incurred by mistakenly assigning the class label i to an instance from the class j. If $\mathbf{x}$ is an example, then the *overall risk* (Duda & Hart, 1973) of a classifier is simply

$$\sum_{i=1}^{n} \sum_{j=1}^{n} P(j|\mathbf{x}) c_{ij}, \tag{2}$$

where $P(j|\mathbf{x})$ is the conditional probability that the example $\mathbf{x}$ will be labeled j. Naturally, we want to build a classifier that minimizes our risk.

In practice, it is difficult to find the optimal set of boundaries that partition the set of examples in a way that minimizes the overall risk. For example, there is no guarantee that the labels of the training examples will coincide with the true cost of errors (Domingos, 1999; Maloof et al., 1998). Duda and Hart (1973) suggested that we can take error costs into account by adjusting a class's prior probability, but for some methods, it is not clear how to use altered priors to influence the process of concept formation. As we mentioned previously, we can indirectly change a class's prior probability by duplicating or removing examples from the training set (Breiman et al., 1984). But as Domingos (1999) observed, this stratification method is not without its problems. If we remove examples, then we have less data for training, and if we duplicate examples, then we increase the time required for training.

These difficulties have lead several researchers to devise heuristic approaches for constructing cost-sensitive classifiers. For example, Pazzani and his colleagues (Pazzani et al., 1994) used a post-processing step to select and order rules in a decision list to minimize costs. Bradley (1997), for the perceptron, simulates different misclassification costs by using a heuristic to vary the decision threshold.

When implementing cost-sensitive learning methods, the basic idea is to change the way the algorithm treats instances from the more expensive class relative to the other instances, either during the learning process or at the time of testing. We want to incorporate a heuristic into the algorithms so that we can bias them toward making mistakes on the less costly class rather than on the more expensive class.

Recall that naive Bayes predicts the class with the highest posterior probability as computed using Bayes' rule, so in our implementation, we simply

computed the risk for each class, selecting the one with the least risk. That is, for an example, $\mathbf{x}$, we computed the expected risk, $R(i|\mathbf{x})$, for the class i using the formula:

$$R(i|\mathbf{x}) = \sum_j P(j|\mathbf{x})c_{ij}, \tag{3}$$

where $P(j|\mathbf{x})$ is the posterior probability of the jth class given the example. The cost-sensitive version of naive Bayes predicts the class i with the least expected risk.

C5.0 uses a method similar to that in CART (Breiman et al., 1984) to grow decision trees in a way that minimizes error cost. C5.0 first estimates the prior probability of each class from the training set. We have already established the link between the cost of errors and the prior probabilities of the classes. C5.0 then uses the estimated priors and the cost matrix to compute altered priors, which are used to bias the selection of attributes during the tree growing process.

For the remaining algorithms—the perceptron and nearest neighbor classifiers—we chose to incorporate a *cost parameter* into the performance element of the algorithms, rather than the learning element, so we could vary the decision threshold, thus simulating different costs of misclassification. For each class, we defined a parameter, τ, in the range $[0.0, 1.0]$ to indicate the cost of making a mistake on the class. Zero indicates that errors cost nothing, and one means that errors are maximally expensive.

Nearest neighbor, as normally used, predicts the class of the example that is closest to the query. Any cost heuristic should have the effect of moving the query point closer to the closest example of the more expensive class, and the magnitude of this change should be proportional to the magnitude of the cost parameter. Therefore, we computed the altered distance, δ_j, for the class j using the formula:

$$\delta_j = d_E(\mathbf{x}, \mathbf{x}_j) - \tau_j \, d_E(\mathbf{x}, \mathbf{x}_j), \tag{4}$$

where $\mathbf{x}_j$ is the closest neighbor from class j to the query point, and $d_E(x, y)$ is the Euclidean distance function. The cost-sensitive version of nearest neighbor returns as its prediction the class label of the closest instance as measured by the altered distance. This modification also works for k-nearest neighbors, which considers the k closest neighbors as measured by the altered distance when classifying unknown instances.

Since the perceptron is a linear discriminant function, we want the cost heuristic to adjust the threshold so the hyperplane of discrimination is farther from the hypothetical region of examples of the more expensive class, thus enlarging the decision region of that class. The degree to which the algorithm adjusts the threshold is again dependent on the magnitude of the cost

parameter. The adjusted threshold θ' is computed by:

$$\theta' = \theta - \sum_{j=1}^{2} \mathrm{sgn}(j)\tau_j\sigma_j, \tag{5}$$

where θ is the original threshold for the linear discriminant function, $\mathrm{sgn}(j)$ returns $+1$ for the positive class and -1 for the negative class, and σ_j is the maximum value the weighted sum can take for the jth class. The cost-sensitive version of the perceptron predicts the positive class if the weighted sum of an instance's attributes surpasses the adjusted threshold θ'; otherwise, it predicts the negative class.

Finally, because our modifications focused on the performance elements rather than on the learning algorithms, we made similar changes to the BUDDS classifier. Like the perceptron, it is a linear discriminant function, so we made the same modifications to the BUDDS classifier that we made to the perceptron algorithm.

7.3. ROC Analysis for Evaluating Performance

Our next challenge was to identify an experimental methodology that would let us compare the behavior of our cost-sensitive learning methods on the rooftop data. We have already seen that comparisons based on overall accuracy are not sufficient for domains that involve non-uniform costs or skewed distributions. Rather, we must separately measure accuracy on both classes, in terms of false positives and false negatives. Given information about the relative costs of errors, say, from conversations with domain experts or from a domain analysis, we could then compute a weighted accuracy for each algorithm that takes cost into account (Fawcett & Provost, 1997; Pazzani et al., 1994).

However, in this case, we had no access to image analysts or enough information about the results of their interpretations to determine the actual costs for the domain. In such situations, rather than aiming for a single performance measure, as typically done in machine learning experiments, a natural solution is to evaluate each learning method over a *range* of cost settings. ROC (Receiver Operating Characteristic) analysis (Swets, 1988) provides a framework for carrying out such comparisons. The basic idea is to systematically vary some aspect of the situation, such as the misclassification costs, the class distribution, or the decision threshold, apply the classifier to test cases, and plot the false positive rate against the true positive rate for each situation. (See Appendix A for more detail.) Although researchers have used such ROC curves in signal detection and psychophysics for decades (Egan, 1975; Green & Swets, 1974), this technique has only recently begun to filter into machine learning research (e.g., Bradley, 1997; Ezawa et al., 1996; Maloof et al., 1997; Provost & Fawcett, 1997; Provost et al., 1998).

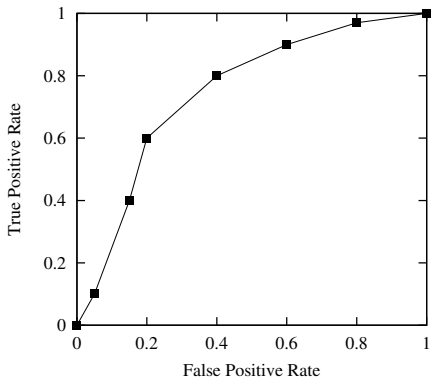

Figure 2. A hypothetical Receiver Operating Characteristic (ROC) curve.

Figure 2 shows a hypothetical ROC curve generated by varying the decision threshold of a cost-sensitive learning algorithm. The lower left corner of the figure, point (0, 0), represents the situation in which mistakes on the negative class are maximally expensive (i.e., $c_+ = 0.0$ and $c_- = 1.0$). Conversely, the upper right corner of the ROC graph, point (1, 1), represents the situation in which mistakes on the positive class are maximally expensive (i.e., $c_+ = 1.0$ and $c_- = 0.0$). By varying over the range of cost parameters and plotting the classifier's true positive and false positive rates, we produce a series of points that represents the algorithm's accuracy trade-off, which is unconfounded by inductive bias, unequal error costs, and skewed data sets. The point $(0, 1)$ is where classification is perfect, with a false positive rate of zero and a true positive rate of one, so we want ROC curves that "push" toward this corner.

Traditional ROC analysis uses area under the curve as the preferred measure of performance, with curves that cover larger areas generally viewed as better (Hanley & McNeil, 1982; Swets, 1988). Given the skewed nature of the rooftop data, and the different but imprecise costs of errors on the two classes, we decided to use area under the ROC curve as the dependent variable in our experimental studies. This measure is problematic when two curves have similar areas but are dissimilar and asymmetric, and thus occupy different regions of the ROC space. In such cases, other types of analysis are more useful (e.g., Provost & Fawcett, 1997), but area under the curve appears to be most appropriate when curves have similar shapes and when one curve dominates the other. As we will see, this relation typically holds for our cost-sensitive algorithms in the rooftop detection domain.

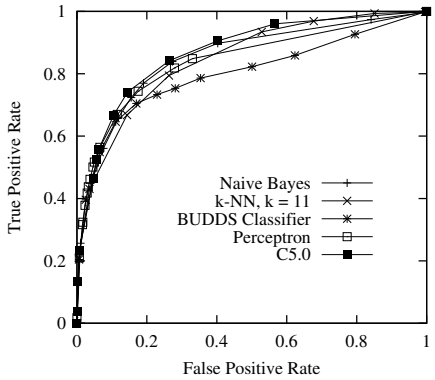

Figure 3. ROC curve for the experiment using all available image data. We ran each method over a range of costs using a training set (60%) and a testing set (40%) and averaged the true positive and false positive rates over ten runs. C5.0 produced the curve with the largest area, but all of the other learning methods yielded curves larger in area than that of the BUDDS classifier.

8. Experiment II: Cost-Sensitive Algorithms and ROC Analysis

With this new perspective, we revisited the rooftop detection task by conducting an experiment using the cost-sensitive versions of C5.0, naive Bayes, *k*-NN, the perceptron, and the BUDDS classifier. As before, we used all of the rooftop candidates generated from the six Fort Hood images, since we wanted to replicate our previous experiment reported in Section 6, and trained the induction methods on data (rooftop candidates) separate from those used to test the learned classifiers.

Combining the rooftop candidates from all six images yielded 17,829 instances, 781 labeled positive and 17,048 labeled negative. We ran each algorithm ten times over a range of costs, randomly splitting the data each run into training (60%) and testing (40%) sets. Because the BUDDS classifier was hand-configured, it had no training phase, so we applied it directly to the instances in the test set.

Since our domain involved only two classes and costs are relative (i.e., $\tau_+ = 0.0$ and $\tau_- = 0.5$ is equivalent to $\tau_+ = 0.25$ and $\tau_- = 0.75$), we varied the cost parameter for only one class at a time and fixed the other at zero. Furthermore, because cost settings differed between algorithms and between data sets due to differences in inductive bias and in the distribution of examples, respectively, we empirically determined for each algorithm the settings required to yield the desired ROC curve. (See Appendix A for more details.) For each of the ten runs, we used the trapezoid rule to approximate the area under each ROC curve. Upon completing the runs, we averaged the ten areas for each method and computed 95% confidence intervals. We also averaged

Table III. Results for the experiment using all of the image data. We split the data into training (60%) and test (40%) sets and ran each method over a range of costs. We then computed the average area under the ROC curve and 95% confidence intervals over ten runs.

Classifier	Area under ROC Curve
C5.0	0.867±0.006
Naive Bayes	0.854±0.009
Perceptron	0.853±0.010
k-NN (k = 11)	0.847±0.006
BUDDS Classifier	0.802±0.014

the true positive and false positive rates for each method over the ten runs to produce an ROC curve.

Figure 3 shows the resulting ROC curves, which plot the averaged true positive and false positive rates, whereas Table III gives the average approximate area under these curves. To determine if these results were significant, we conducted an analysis using LabMRMC (Dorfman, Berbaum, & Metz, 1992). This uses the Jackknife method (Hinkley, 1983) on case ratings to account for case-sample variance and then applies traditional analysis of variance (ANOVA) to determine significance. This analysis showed that the means of C5.0, naive Bayes, the BUDDS classifier, the perceptron, and k-NN, for $k = 11$, were significantly different ($p < .001$).

C5.0 performed the best overall, producing a curve with area 0.867. Naive Bayes and the perceptron, while not performing quite as well as C5.0, produced curves with areas roughly equal to 0.85. Of the k-NN classifiers, $k = 11$ performed the best with an area of 0.847. Finally, the BUDDS classifier produced a curve of area 0.802. The important result from this experiment is not whether C5.0 performed better than naive Bayes. Indeed, the important result is that all of the learning methods outperformed the handcrafted BUDDS classifier, which supports our research hypothesis: that learning methods can outperform carefully handcrafted heuristics.

In practice, image analysts will not evaluate a classifier's performance using area under the ROC curve, but will have specific error costs in mind, even if they cannot state them formally. We have used ROC curves because we do not know these costs in advance, but we can inspect behavior of the various classifiers at different points on these curves to give further insight into how much the learned classifiers are likely to aid analysts during actual use.

For example, consider the behavior of C5.0 when it achieves a true positive rate of 0.84 and a false positive rate of 0.26. For the same true positive rate, the BUDDS classifier obtained a false positive rate of 0.5. This means that for this true positive rate, C5.0 reduced the false positive rate by about half. Hence, for the images we considered, the C5.0 classifier would have rejected 4,432 more non-rooftops than the BUDDS classifier. Similarly, by fixing the false positive rate, C5.0 improved the true positive rate by 0.1 over the BUDDS classifier. In this case, the C5.0 classifier would have found 78 more rooftops than the BUDDS classifier. Furthermore, if we were willing to tolerate more false positives in an effort to increase the true positive rate, then we could select points higher on C5.0's ROC curve where the differences in performance between C5.0 and the BUDDS classifier are even greater (e.g., the points at which the false positive rate is 0.4).

8.1. WITHIN-IMAGE LEARNING

We also examined how the various methods behaved given *within-image* learning, that is, when generalizing to test cases taken from the same image on which we trained them. Our research hypothesis was that the learned classifiers would be more accurate over a range of misclassification costs than the handcrafted linear classifier. Because our measure of performance was area under the ROC curve, this translates into a prediction that the ROC curves of the learned rooftop classifiers would have larger areas than those of the BUDDS classifier.

For each image and method, we varied the error costs and measured the resulting true positive and false positive rates for ten runs. Each run involved partitioning the data set randomly into training (60%) and test (40%) sets, running the learning algorithms on the instances in the training set, and evaluating the resulting concept descriptions using the data in the test set. For each cost setting and each classifier, we plotted the average false positive rate against the average true positive rate over the ten runs.

Figure 4 presents the ROC curves for Images 1 and 2, on which, naive Bayes produced the best results. The areas under these curves, which we approximated using the trapezoid rule, appear in Table IV. Rather than present curves for the remaining four images, we report the areas under each ROC curve in Table V. For Images 3 and 4, C5.0 produced curves with areas greater than those of the other methods. For Images 5 and 6, naive Bayes again produced curves with the greatest area, although the differences between naive Bayes and the BUDDS classifier for Images 2 and 6 were not notable. Moreover, with the exception of these two images, all of the learning methods outperformed the handcrafted classifier, and this outcome generally supports our research hypothesis.

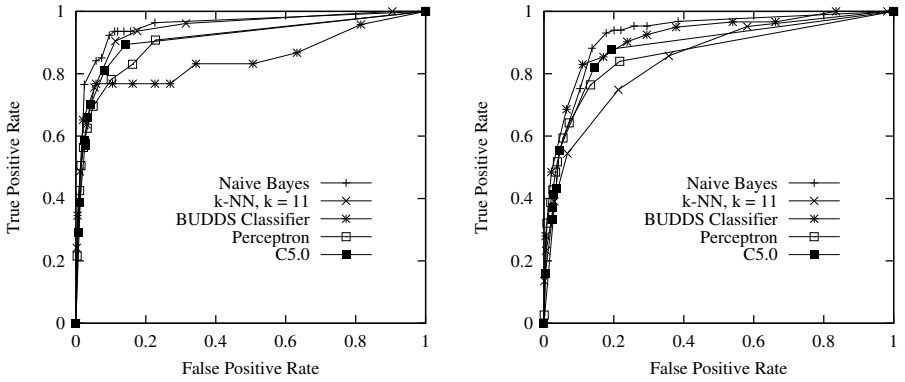

Figure 4. ROC curves for two images from within-image experiments. We ran each method by training and testing using data derived from the same image over a range of misclassification costs. We conducted ten such runs and plotted the average true positive and false positive rates. Left: Image 1, a nadir-view image. Right: Image 2, an oblique-view image.

Table IV. Results for within-image experiments for Images 1 and 2. Approximate areas under the ROC curve appear with 95% confidence intervals.

	Approximate Area under ROC Curve	
Classifier	Image 1	Image 2
C5.0	0.913±0.005	0.882±0.008
Naive Bayes	*0.952±0.010*	*0.918±0.007*
Perceptron	0.923±0.014	0.874±0.012
k-NN, *k* = 11	0.941±0.009	0.858±0.014
BUDDS Classifier	0.846±0.036	*0.915±0.009*

8.2. BETWEEN-IMAGE LEARNING

We geared our next set of experiments more toward the goals of image analysis. Recall that our motivating problem is the large number of images that the analyst must process. In order to alleviate this burden, we want to apply knowledge learned from some images to many other images. But we have already noted that several dimensions of variation pose problems for transferring such learned knowledge to new images. For example, one viewpoint of a given site can differ from other viewpoints of the same site in orientation or in angle from the perpendicular. Images taken at different times and images of different areas present similar issues.

We designed experiments to let us better understand how the knowledge learned from one image generalizes to other images that differ along such dimensions. Our hypothesis here was a refined version of the previous one:

Table V. Results for within-image experiments for Images 3–6. Approximate areas under the ROC curve appear with 95% confidence intervals.

	Approximate Area under ROC Curve			
Classifier	Image 3	Image 4	Image 5	Image 6
C5.0	*0.894±0.004*	*0.834±0.*008	0.861±0.018	0.837±0.009
Naive Bayes	0.838±0.004	0.823±0.012	*0.876±0.007*	*0.852±0.007*
Perceptron	0.858±0.011	0.807±0.023	0.860±0.020	0.743±0.028
k-NN, k = 11	0.846±0.005	0.828±0.010	0.830±0.009	0.783±0.009
BUDDS Classifier	0.750±0.007	0.771±0.012	0.829±0.022	0.850±0.007

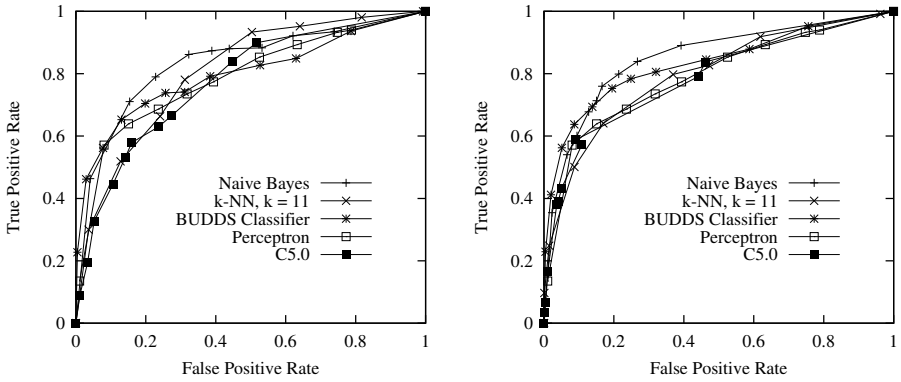

Figure 5. ROC curves for experiments that tested generalization over aspect. Left: For each location, we trained each method on the oblique image and tested the resulting concept descriptions on the nadir image. We plotted the average true positive and false positive rates. Right: We followed a similar methodology, except that we trained the methods on the nadir images and tested on the oblique images.

Classifiers learned from one set of images would be more accurate on unseen images than handcrafted classifiers. However, we also expected that between-image learning would give lower accuracy than the within-image situation, since differences across images would make generalization more difficult.

8.2.1. *Generalizing over Aspect*

One experiment focused on how the methods generalize over aspect. Recall from Table I that we had images from two aspects (i.e., nadir and oblique) and from three locations. This let us train the learning algorithms on an image from one aspect and test on an image from another aspect but from the same location. As an example, for the nadir aspect, we chose Image 1 and then tested on Image 2, which is an oblique image of the same location. We ran the algorithms in this manner using the images from each location, while varying

Table VI. Results for between-image experiment in which we tested generalization over aspect. The labels 'Nadir' and 'Oblique' indicate the testing condition. We derived analogous results for the within-image experiments by averaging the results for each condition. Approximate areas appear with 95% confidence intervals.

	Aspect Experiment		Average Within Image	
Classifier	Nadir	Oblique	Nadir	Oblique
C5.0	0.837±0.016	0.806±0.015	*0.889±0.008*	0.851±0.007
Naive Bayes	0.839±0.020	*0.858±0.028*	*0.889±0.012*	*0.865±0.010*
Perceptron	0.828±0.026	0.835±0.019	0.880±0.011	0.808±0.017
k-NN, $k = 11$	*0.856±0.028*	0.814±0.017	0.872±0.012	0.823±0.009
BUDDS Classifier	0.801±0.029	0.841±0.032	0.809±0.016	0.846±0.014

their cost parameters and measuring their true positive and false positive rates. We then averaged these measures across the three locations and plotted the results as ROC curves, as shown in Figure 5. The areas under these curves and their 95% confidence intervals appear in Table VI.

One obvious conclusion is that the nadir images appear to pose an easier problem than the oblique images, since the curves for testing on nadir candidates are generally higher than those for testing on data from oblique images. For example, Table VI shows that C5.0 generated a curve with an area of 0.837 for the nadir images, but produced a curve with an area of 0.806 for the oblique images. The other two methods show a similar degradation in performance when generalizing from nadir to oblique images rather than from oblique to nadir images. The exception is naive Bayes, which achieved better performance when generalizing to oblique images.

Upon comparing the behavior of different methods, we find that for oblique to nadir generalization, k-NN, for $k = 11$, with an area under the ROC curve of 0.856, performed better than the BUDDS classifier, with an area of 0.801. In this experimental condition, all of the learning methods outperformed the BUDDS classifier. For nadir to oblique generalization, naive Bayes performed slightly better than the BUDDS classifier, which produced areas of 0.858 and 0.841, respectively. In this experimental condition, BUDDS outperformed three of the learning methods: the perceptron, k-NN, and C5.0.

8.2.2. *Generalizing over Location*

A second experiment examined generalization over location. To this end, we trained the learning methods on pairs of images from one aspect and tested on the third image from the same aspect. As an example, for the nadir images, one of the three learning runs involved training on rooftop candidates from Images 1 and 3, then testing on candidates from Image 5. We then ran each

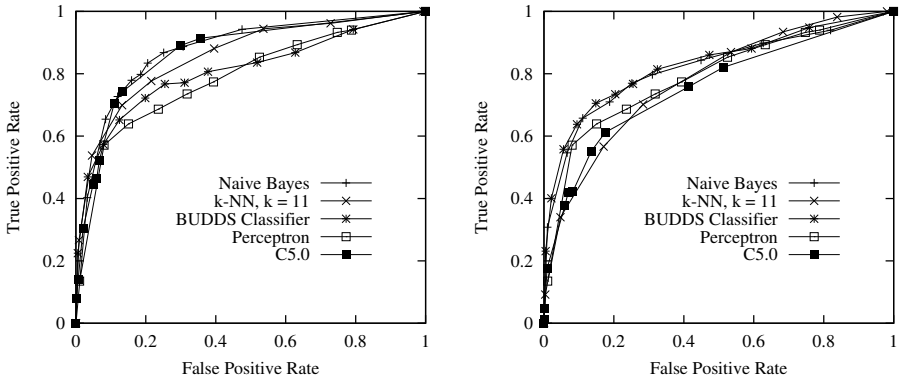

Figure 6. ROC curves for experiment that tested generalization over location. Left: For each pair of images for the nadir aspect, we trained the methods on that pair and tested the resulting concept descriptions on the third image. We then plotted the average true positive and false positive rates. Right: We applied the same methodology using the images for the oblique aspect.

Table VII. Results for between-image experiment in which we tested generalization over location. The labels 'Nadir' and 'Oblique' indicate the testing condition. We derived analogous results for the within-image experiments by averaging the results for each condition. Approximate areas appear with 95% confidence intervals.

	Location Experiment		Average Within Image	
Classifier	Nadir	Oblique	Nadir	Oblique
C5.0	0.872±0.025	0.779±0.037	*0.889±0.008*	0.851±0.007
Naive Bayes	*0.887±0.032*	0.833±0.028	*0.889±0.012*	*0.865±0.010*
Perceptron	0.877±0.031	0.802±0.052	0.880±0.011	0.808±0.017
k-NN, *k* = 11	0.860±0.024	0.796±0.021	0.872±0.012	0.823±0.009
BUDDS Classifier	0.813±0.025	*0.840±0.033*	0.809±0.016	0.846±0.014

of the algorithms across a range of costs, measuring the false positive and true positive rates. We plotted the averages of these measures across all three learning runs for one aspect in an ROC curve, as shown in Figure 6.

In this context, we again see evidence that the oblique images presented a more difficult recognition task than the nadir aspect, since areas for the oblique images are less than those for the nadir images. Comparing the behavior of the various methods, Table VII shows that for the nadir aspect, naive Bayes performs better than the BUDDS classifier, yielding areas of 0.887 and 0.813, respectively. As before, all of the learning methods performed better than the BUDDS classifier. When generalizing over location with the oblique images, the BUDDS classifier performed the best (0.84), but naive Bayes was

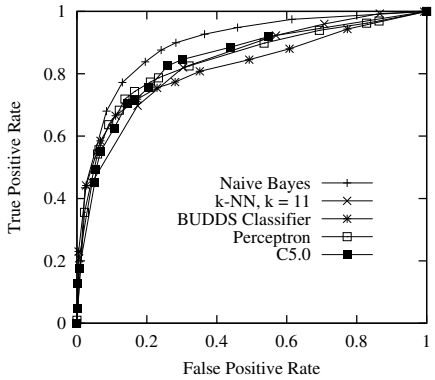

Figure 7. ROC curve for the experiment in which we left each image out as a test image and trained using the remaining five images. Naive Bayes produced the curve with the largest area, but all of the other learning methods yielded curves larger in area than that of the BUDDS classifier.

Table VIII. Results for the experiment in which we left images out for testing and trained using the remaining five images. We held out each image, each nadir image, and each oblique image. Average areas under the ROC curve appear with 95% confidence intervals.

	Area under ROC Curve		
Classifier	Each	Nadir	Oblique
Naive Bayes	*0.887±0.040*	*0.911±0.065*	*0.863±0.051*
Perceptron	0.874±0.044	0.909±0.071	0.855±0.082
C5.0	0.854±0.042	0.880±0.062	0.828±0.059
k-NN ($k = 11$)	0.845±0.043	0.872±0.075	0.819±0.042
BUDDS Classifier	0.828±0.039	0.829±0.068	0.828±0.069

a close second with an area under its curve of 0.833. Generally speaking, the learning methods did not fare as well in this experimental condition.

8.2.3. *Leaving One Image Out*

Our third and final experiment was an attempt to evaluate how the methods might perform in a real-world setting. In this scenario, developers train a system using a set of images, and then image analysts use the system to identify objects in a new image that may differ in location, aspect, or both.

For this experiment, we evaluated the learning methods under three experimental conditions. In the first, we selected each of the images for testing and used the remaining images for training. In the second, we proceeded similarly, but left out each of the nadir images. In the third, we left out each of the oblique images. For each data set in each condition, we applied each

learning method, testing the resulting classifiers over a range of error costs on rooftop candidates from the test image, with each method producing an ROC curve. We repeated this procedure for each image remaining in the set and averaged over the runs. Plots of the ROC curves for the first condition appear in Figure 7. We also approximated the area under the curves for all three conditions and computed 95% confidence intervals, and these measures appear in Table VIII.

As in the previous experiments, the learning methods generally outperformed the handcrafted BUDDS classifier, which produced a curve with an area of roughly 0.828 for each of the three experimental conditions. Naive Bayes, as in the previous settings, performed the best with an area under the ROC curve of 0.887 for the first condition, 0.911 for the condition in which we held our nadir images, and 0.863 for the condition in which we held out oblique images. In this experiment, we again see evidence that the oblique images posed a more difficult learning problem than the nadir images. Finally, for the condition in which we left out each of the oblique images, naive Bayes and the perceptron outperformed the BUDDS classifiers, but this was not true of C5.0 and k-NN, for $k = 11$.

8.2.4. *Summary*

In experiments testing performance on unseen images, the results of the naive Bayesian classifier support our main hypothesis. In most experimental conditions, this method fared better than the BUDDS linear classifier. On the other hand, we were disappointed that the learning methods performed worse than the BUDDS classifier in the hardest experimental condition: generalizing to new locations with an oblique viewpoint, which went against our original expectations.

Recall that we also anticipated that generalizing across images would give lower accuracies than generalizing within images. To test this hypothesis, we must compare the results from these experiments with those from the within-image experiments. Simple calculation shows that for the within-image condition, naive Bayes produced an average ROC area of 0.889 for the nadir images and 0.865 for the oblique images. Similarly, C5.0 averaged 0.889 for the nadir images and 0.851 for the oblique images. Most of these average areas, which appear in the two rightmost columns of Tables VI and VII, are larger than the analogous areas that resulted when these methods generalized across location and aspect. One exception is that naive Bayes, C5.0, and the perceptron performed almost as well when generalizing over location for the nadir image (see Table VII), but the results generally support our prediction.

Finally, in perhaps the most realistic experimental condition in which we trained methods on a collection of images and tested on new images, the learning methods, especially naive Bayes, performed quite well. Although

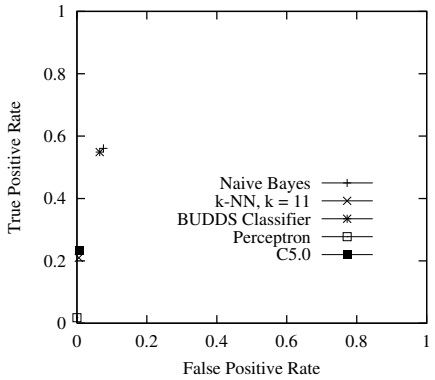

Figure 8. Results from Experiment I plotted in an ROC graph.

this outcome supports our hypothesis that learning methods can outperform handcrafted heuristics, it highlights the need for incorporating learning mechanisms into systems for image analysis.

9. Discussion of the Experimental Results

In the first experiment, we evaluated the four learning methods traditionally, without taking into account the cost of errors and using accuracy as our measure of performance. Based on these results, there was no clear choice for the best performing classifier. C5.0 did achieve the highest overall predictive accuracy, but naive Bayes, while doing poorly overall, performed the best on the most important class, rooftops. Nevertheless, naive Bayes' true positive rate was not significantly better than the handcrafted BUDDS classifier's, the method we were attempting to improve upon.

If we plot the true positive and false positive rates from the first experiment in an ROC graph, as shown in Figure 8, we gain additional insights into the relative performances of the methods. We see that each of the points lie on some unknown ROC curve and that the first experiment charted little of the ROC space.

In the second experiment, because of an improved evaluation methodology that involved cost-sensitive learning algorithms, ROC analysis, and a single measure of performance—area under the ROC curve—we had a much better understanding of how each method performed. Although we may have been able to conclude based on the results of the first experiment that C5.0 was the best method for detecting rooftops, the results from the second experiment more clearly demonstrated this fact, and we were able to draw this conclusion with greater certainty.

Comparing the performances of naive Bayes in the first and second experiments, we see how the ROC methodology elucidated important but hidden aspects of performance. Based on the results from the first experiment, we concluded that the performance of naive Bayes was no better than that of the BUDDS classifier. Yet, the results from the second experiment revealed a very different picture and showed that over a range of costs and in an important region of the ROC space (i.e, where the true positive rate is greater than 0.6 and the false positive rate is less than 0.4), naive Bayes significantly outperformed the BUDDS classifier.

As we discussed previously, the skewed distribution of the data set considerably affected the performance of the perceptron, which simply learned to predict the negative class, but did not affect as greatly the performance of naive Bayes, which found a more acceptable trade-off between the false positive and true positive rates. We speculate that each learning method operates under the influence of an inherent but unknown set of cost parameters associated with its inductive bias. These differences may have caused naive Bayes to be less effected than the perceptron by the skewed distribution of the data set and may partially account for the differences in performance of the methods in Experiment I.

Although the perceptron did quite poorly in the first experiment, the results from the second experiment showed ultimately that its ROC curve did indeed dominate that of the BUDDS classifier. Therefore, a problem with using accuracy as the sole measure of performance is that under certain conditions, we may conclude that a given method is superior to other methods, when in fact, the ROC curves for the same methods, for the same task, reveal completely different phenomena.

From the within-learning experiments, in which we trained and tested the learning methods using data derived from the same image, it was apparent that at least one machine learning method, naive Bayes, showed promise of improving the detection of rooftops over the handcrafted linear classifier. The results from this experiment also established baseline performance conditions for the methods because they controlled for differences in aspect and location.

In an effort to test the learning methods for their ability to generalize to unseen images, we found that rooftop detection for oblique images posed a more difficult problem than for nadir images. This could be because BUDDS was initially developed using nadir images and then extended to handle oblique images. Thus, the features may be biased toward nadir-view rooftops. A more likely explanation is that oblique images are simply harder than nadir images. As we indicated previously, oblique images yield more negative examples, and we believe these additional negative examples pose a more difficult learning problem. Nevertheless, under all but one circumstance, the performance of naive Bayes was better than that of the handcrafted linear classifier.

10. Related Work

Research on learning in computer vision has become increasingly common in recent years. Some work in visual learning takes an image-based approach (e.g., Beymer & Poggio, 1996), in which the images themselves, usually normalized or transformed in some way, are used as input to a learning process, which is responsible for forming the intermediate representations necessary to transform the pixels into a decision or classification. Researchers have used this approach extensively for face and gesture recognition (e.g., Chan, Nasrabadi, & Mirelli, 1996; Gutta, Huang, Imam, & Weschler, 1996; Osuna, Freund, & Girosi, 1997; Segen, 1994), although it has seen other applications as well (e.g., Nayar & Poggio, 1996; Pomerleau, 1996; Viola, 1993).

A slightly different approach relies on handcrafted vision routines to extract relevant image features, based on intensity or shape properties, then recognizes objects using learned classifiers that take these features as inputs. For example, Shepherd (1983) used decision-tree induction to construct classifiers for chocolate shapes in an industrial vision application. Cromwell and Kak (1991) took a similar approach to recognizing electrical components, such as transistors, resistors, and capacitors. Maloof and Michalski (1997) examined various methods of learning shape characteristics for detecting blasting caps in X-ray images, whereas additional work (Maloof, Duric, Michalski, & Rosenfeld, 1996) discussed learning in a multi-step vision system for the same detection problem.

Researchers have also investigated structural approaches, which we divide into two categories: model-based and hierarchical. The former approach relies on features computed from images, but combines elementary visual constructs extracted from images to form higher-level constructs, which the system compares to a database of object models. Connell and Brady (1987) incorporated learning into a model-based vision system. Using the vision task of classifying airplanes from aerial views, their method converts objects into semantic networks, which serve as training examples for a learning process that produces generalized object descriptions. However, the authors do not appear to have tested experimentally their algorithm's ability to classify objects in new images. Binford, Levitt, and colleagues, in a series of papers, examined Bayesian methods for inference in a model-based system using generalized cylinders as the primitive (Binford, Levitt, & Mann, 1987; Levitt, Agosta, & Binford, 1989).

Hierarchical vision systems work similarly to model-based systems but eliminate the requirement of a model-base and model matching by using perceptual grouping operations to combine visual constructs and by using heuristics to select the most promising constructs for further processing (Mohan & Nevatia, 1989). Grouping and selection continues until the system forms a view-dependent representation of the object detected in the image.

Researchers have successfully applied this approach to building detection in single images (Lin & Nevatia, 1998) and in multiple images (Noronha & Nevatia, 1997). Recent work on learning within this framework has concentrated on single levels of the hierarchy (Maloof, Langley, Binford, & Nevatia, 1998; Maloof et al., 1997; Kim & Nevatia, 1999, 2000), with two exceptions being that of Maloof (2000) and of Pope and Lowe (2000). Finally, Sarkar and Soundararajan (2000) used learning automata to acquire the knowledge necessary for perceptual grouping.

Several researchers have also investigated learning for three-dimensional vision systems. Papers by Conklin (1993), Cook, Hall, Stark, and Bowyer (1993), Woods, Cook, Hall, Bowyer, and Stark (1995), Provan, Langley, and Binford (1996), and Sengupta and Boyer (1993) all describe inductive approaches aimed at improving object recognition. The objective of this approach is to learn the three-dimensional structure that characterizes an object or object class, rather than its appearance. Another line of research, which falls midway between this approach and image-based schemes, instead attempts to learn a small set of *characteristic views*, each of which can be used to recognize an object from a different perspective (e.g., Gros, 1993; Pope & Lowe, 1996).

Much of the research on visual learning uses images of scenes or objects viewed at eye level (e.g., Draper, 1997; Sarkar & Soundararajan, 2000; Teller & Veloso, 1997). One exception is the SKICAT system (Fayyad, Smyth, Burl, & Perona, 1996), which catalogs celestial objects, such as galaxies and stars, using images from the Second Palomar Observatory Sky Survey. A related system, JARTool (Fayyad et al., 1996), also analyzes aerial images, in this case to detect Venusian volcanos, using synthetic aperture radar on the Magellan spacecraft.

Burl and his colleagues extended JARTool by using an ensemble of 48 neural networks to improve performance (Burl et al., 1998). Using ROC curves, they demonstrated that the ensemble achieved better performance than either the individually learned classifiers or the one used originally in JARTool. They also documented some of the difficulties associated with applying machine learning techniques to real-world problems, such as feature selection and instance labeling, which were similar to those we encountered.

Kubat, Holte, and Matwin (1998) used the SHRINK system (Kubat, Holte, & Matwin, 1996) to detect oil spills in overhead images. They too dealt with imbalanced data sets, but also coped with a scarcity of data that was not present in our domain. Luckily, we were able to select images that contained sufficient concentrations of buildings, but oil spills, especially those captured in satellite images, proved to be a rarity. Their images from a synthetic aperture radar covered 3500 square kilometers and contained as few as two oil spills. Although they had more images than we—nine versus our six—our system extracted many more objects: 17,829 positive and negative examples

of rooftops as compared to 937 positive and negative examples of oil spills. Interestingly, the relative proportion of positive examples in these data sets was similar: approximately 4.38%.

Finally, Draper (1996) conducted a careful study of learning in the context of analyzing aerial images. His approach adapted methods for reinforcement learning to assign credit in multi-stage recognition procedure (for software similar to BUDDS), then used an induction method (back-propagation in neural networks) to learn conditions on operator selection. He presented initial results for detecting rooftops as part of the RADIUS project (Firschein & Strat, 1997), which we discussed previously in Section 2. Our framework shares some features with Draper's approach, but assumes that learning is directed by feedback from a human expert. We predict that our supervised method will be more computationally tractable than his use of reinforcement learning, which is well known for its high complexity. Our approach does require more interaction with users, but we believe this interaction will be unobtrusive if cast within the context of an advisory system for image analysis. Indeed, we have recently formulated a design for incorporating machine learning algorithms into hierarchical vision systems, and preliminary results for simple, compositional objects are encouraging (Maloof, 2000).

Interest in cost-sensitive learning and in ROC analysis has become increasingly common in recent years, and our work along these lines has some precedents. We have already mentioned the work of Pazzani et al. (1994), who used a hypothetical cost analysis and constructed minimal-cost classifiers. Similarly, Draper, Brodley, and Utgoff (1994) incorporated the cost of errors into their algorithm for constructing and pruning multivariate decision trees. They tested this approach on the task of labeling pixels from outdoor images for use by a road-following vehicle. They determined that in this context, labeling a road pixel as non-road was more costly than the reverse, and showed experimentally that their method could reduce such errors on novel test pixels. Woods, Kegelmeyer, and Bowyer (1997), as well as Rowley, Baluja, and Kanade (1996), reported similar work taking into account the cost of errors.

Researchers have begun using ROC analysis to measure performance of classifiers when costs are unknown or when data sets are skewed. Bradley (1997) incorporated mechanisms to adjust the decision threshold of several learning algorithms and used ROC analysis to measure performance on several UCI data sets (Blake & Merz, 1998). Provost et al. (1998) conducted a similar study but emphasized the problems with using accuracy as a measure of performance. We have used ROC analysis to measure the ability of various classifiers to generalize on a rooftop detection task when unseen images varied in aspect and in location (Maloof et al., 1998). As mentioned previously, we have also conducted a comparative study that examined the errors experts

made when labeling training data using our visual interface and the effect of such errors on learning (Ali et al., 1998).

In terms of the statistical analysis of ROC curves, Bradley (1997) used ANOVA and Duncan's test to analyze the means and subgroups of means, respectively, of classifier performance, as measured by area under the curve. However, this analysis treats a classifier's performance on an individual case in the testing set as a fixed effect. As a result, we cannot generalize these results to the population from which the test cases were drawn (Metz, 1989); we can generalize them only to the population of test sets. Furthermore, empirical results from a Monte Carlo simulation suggest that, since ANOVA does not adequately account for case-sample variance, especially when sample sizes are small, it fails to reject the null hypothesis more frequently than methods designed to account for this variance. Therefore, ANOVA may be overly optimistic for machine-learning experiments using area under the ROC curve as the performance metric (Maloof, 2002).

Estimating or accounting for case-sample variance can be difficult, but as we have described, Dorfman et al. (1992) detailed a methodology in which one uses the Jackknife method (Hinkley, 1983) to account for case-sample variance and then applies traditional ANOVA to test significance. Recently, Beiden, Wagner, and Campbell (2000) proposed a nonparametric method for comparing means of areas that involves conducting a series of bootstrap experiments to estimate components of variance, like the case-sample variance. Although developed for human-reader experiments in medical imaging, researchers have begun to apply this method to the analysis of learning algorithms (Beiden, Maloof, & Wagner, 2002; Maloof, Beiden, & Wagner, 2002). There has also been work on the analysis of portions of ROC curves (Thompson & Zucchini, 1986; Woods et al., 1997) and in situations in which no single ROC curve dominates in all parts of the ROC space (e.g., Provost & Fawcett, 1997). Researchers have also proposed extensions to ROC analysis for the case of multiple decision categories (Hand & Till, 2001; Mossman, 1999; Swets & Pickett, 1982).

11. Concluding Remarks

Although this study has provided some insight into the role of cost-sensitive algorithms and ROC analysis in systems for image understanding, much still remains to be done. For example, we may want to consider other measures of performance that take into account the presence of multiple valid candidates for a given rooftop. Classifying one of these candidates correctly is sufficient for the purpose of image analysis.

As we mentioned earlier, in order to automate the collection of training data for learning, we also hope to integrate learning routines into BUDDS. This

system was not designed initially to be interactive, but we intend to modify it so that the image analyst can accept or reject recommendations made by the image understanding system, generating training data in the process. At intervals the system would invoke its learning algorithms, producing revised knowledge that would alter the system's behavior in the future and, hopefully, reduce the user's need to make corrections. The interactive labeling system described in Section 5 could serve as an initial model for this interface.

Although the rooftop selection stage was a natural place to start in applying our methods, we intend to work at both earlier and later levels of the building detection process. The goal here is not only to increase classification accuracy, which could be handled entirely by candidate selection, but also to reduce the complexity of processing by removing poor candidates before they are aggregated into larger structures. With this aim in mind, we plan to extend our work to all levels of the image understanding process. We must address a number of issues before we can make progress on these other stages. One involves identifying the cost of different errors at each level, and taking this into account in our modified induction algorithms. Another concerns whether we should use the same induction algorithm at each level or use different methods at each stage. We could also use multiple methods at each level and apply ensemble methods, such as stacking (Wolpert, 1992) or bagging (Breiman, 1996), to produce a single decision and to take into account contextual cues, such as aspect.

In conclusion, our studies suggest that machine learning has an important role to play in improving the accuracy and the robustness of image analysis systems. We need to extend learning to additional levels of the image understanding process, and before we can build a system that truly aids the human image analyst, we must further develop unobtrusive ways to collect training data to support learning. Nevertheless, as more and more vision systems rely on learning and adaptive mechanisms, it will become increasingly important to properly measure the performance of such systems.

Appendix

A. Generating ROC Curves

We are aware of three methods of generating ROC curves (Metz, 1978). First, if we modify the performance element of a learning method to produce a numeric *rating* for each test case—for example, with naive Bayes, we might select the posterior probability of the positive class—then we can use a parametric approach by assuming a binormal distribution (i.e., observations of each of the two classes are normally distributed) and fitting the ROC curve of maximum likelihood (Dorfman & Alf, 1969). It is then a simple matter of

computing the area under this curve (Bamber, 1975; Thompson & Zucchini, 1986).

Second, with such ratings, we can also use a nonparametric approach in which we map the sorted case ratings into, say, 10–12 discrete categories (Wagner, Beiden, & Metz, 2001) and use the number of actually positive cases and actually negative cases of each category to calculate the true positive and false positive rates of an ROC curve.

With these points, we can use the trapezoid rule to compute the approximate area under the curve, $\hat{A}$, but researchers have shown this measure to be equal to the Mann-Whitney two-sample statistic (DeLong, DeLong, & Clarke-Peterson, 1988), which is convenient for numeric case ratings. Given m ratings of negative cases, $\mathbf{r}^-$, and n ratings of positive cases, $\mathbf{r}^+$,

$$\hat{A} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} I(r_i^-, r_j^+),$$

where

$$I(r^-, r^+) = \begin{cases} 1 & \text{if } r^- > r^+; \\ \frac{1}{2} & \text{if } r^- = r^+; \\ 0 & \text{if } r^- < r^+. \end{cases}$$

For some algorithms, it is easy to modify the performance element to produce these ratings. For this purpose, we have used for naive Bayes the posterior probability of the positive class, for k-NN, the number of majority votes for the positive class, and for linear and quadratic discriminants, the output of the discriminant function (Maloof et al., 2002).

However, when we attempted to modify C4.5 (Quinlan, 1993) in this manner by using as ratings the weights that the performance element computes for test instances, the resulting ROC curves were either degenerate or not of the correct area (which we established through further experimentation). It appears the problem is that C4.5, like many learning methods, partitions the representation space to form decision regions with instances in each region being equiprobable. If the number of decision regions is few, then there will be little information present in the case ratings for constructing the ROC curve.

The third method of generating ROC curves, which we used in this study, involves varying some aspect of the learning method or the experiment to bias the learner toward one class versus the other. Unfortunately, such schemes tend to be specific to a learning method (cf. Domingos, 1999) and can increase the running time, but include varying the prior probabilities of each class, the error cost of each class, the decision threshold, and the class distribution of examples in the training set.

In this study, we chose to introduce a mechanism into the performance element that adjusts the decision threshold. Once we have modified the algorithm in this fashion, then it should be a simple matter to select various

thresholds, to evaluate the examples in the test set at each, and to produce a set of true positive and false positive rates that form an ROC curve. Unfortunately, many things conspire to make selecting a set of decision thresholds difficult, such as the distribution of the training examples and the inductive bias of the learner. For instance, it can be difficult to predict which threshold will produce a given point on an ROC curve. A set of parameters for varying the decision threshold for one algorithm will not necessarily produce the same ROC curve for another algorithm. Finally, for a given algorithm and set of decision thresholds, different splits of the data set will yield different ROC curves. This was especially true when we trained on nadir images and tested on oblique images.

Reluctantly, we concluded that setting these thresholds by trial and error for each algorithm would be necessary. We split all of the available data into single training (60%) and testing (40%) sets, and applied each learning algorithm. For each method, we began with the decision threshold implying that mistakes on the positive class were maximally expensive, which produced the point $(0,0)$ on the ROC graph. We repeatedly tried new cost parameters that increasingly favored the positive class until the performance element swept out a curve that was well-shaped with points distributed evenly along its length.

This third method of generating ROC curves can be disadvantageous for algorithms with an expensive performance element, such as k-NN. We must cycle through the training examples, finding the k closest neighbors, for each cost parameter. For our data set, this was expensive, but other methods performed quickly. For instance, the perceptron computes a simple weighted sum, and evaluating this sum for each example and over the cost parameters required little time. Nevertheless, we contend that the nonparametric method of producing ROC curves is more efficient and more principled, but as we have described, it may not work well for all learning methods without significant modification.

Acknowledgements

The authors thank Wayne Iba for assistance with naive Bayes, Dan Shapiro for discussions about decision theory, Yan Bulgak for proofreading a draft of the paper, Ross Quinlan for describing the cost-sensitive mechanisms in C5.0, and Andres Huertas and Andy Lin for advice and assistance in obtaining the images and data used for experimentation. Robert Wagner provided valuable comments regarding ROC analysis, and Sergey Beiden conducted the analysis of the learning algorithms using LabMRMC. We are grateful to Doug Fisher and the anonymous reviewers, who provided constructive and collegial advice that notably improved an earlier version of the manuscript. We thank

the Department of Computer Science at Georgetown University for its support of this work. This research was conducted at the Institute for the Study of Learning and Expertise and in the Computational Learning Laboratory, Center for the Study of Language and Information, at Stanford University. The work was supported by the Defense Advanced Research Projects Agency, under grant N00014-94-1-0746, administered by the Office of Naval Research, and by Sun Microsystems through a generous equipment grant.

References

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

Ali, K., Langley, P., Maloof, M., Sage, S., & Binford, T. (1998). Improving rooftop detection with interactive visual learning. In *Proceedings of the Image Understanding Workshop* (pp. 479–492). San Francisco, CA: Morgan Kaufmann.

Bamber, D. (1975). The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology*, *12*, 387–415.

Beiden, S., Maloof, M., & Wagner, R. (2002). Analysis of competing classifiers in terms of components of variance of ROC summary accuracy measures: Generalization to a population of trainers and a population of testers. In *Proceedings of the SPIE International Symposium on Medical Imaging: Image Processing* (Vol. 4684).

Beiden, S., Wagner, R., & Campbell, G. (2000). Components-of-variance models and multiple-bootstrap experiments: An alternative method for random-effects Receiver Operating Characteristic analysis. *Academic Radiology*, *7*, 341–349.

Beymer, D., & Poggio, T. (1996). Image representations for visual learning. *Science*, *272*, 1905–1909.

Binford, T., Levitt, T., & Mann, W. (1987). Bayesian inference in model-based machine vision. In *Proceedings of the Third Annual Conference on Uncertainty in Artificial Intelligence* (pp. 73–97). New York, NY: Elsevier Science.

Blake, C., & Merz, C. (1998). *UCI Repository of machine learning databases* ([http://www.ics.uci.edu/∼mlearn/MLRepository.html]). Department of Information and Computer Sciences, University of California, Irvine.

Bradley, A. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*, 1145–1159.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees.* Boca Raton, FL: Chapman & Hall/CRC Press.

Burl, M., Asker, L., Smyth, P., Fayyad, U., Perona, P., Crumpler, L., & Aubele, J. (1998). Learning to recognize volcanoes on Venus. *Machine Learning*, *30*, 165–194.

Cardie, C., & Howe, N. (1997). Improving minority class prediction using case-specific feature weights. In *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 57–65). San Francisco, CA: Morgan Kaufmann.

Chan, L., Nasrabadi, N., & Mirelli, V. (1996). Multi-stage target recognition using modular vector quantizers and multilayer perceptrons. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 114–119). Los Alamitos, CA: IEEE Press.

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

Conklin, D. (1993). Transformation-invariant indexing and machine discovery for computer vision. In *Papers from the the AAAI Fall Symposium on Machine Learning in Computer Vision: What, Why, and How?* Technical Report No. FS-93-04 (pp. 10–14). Menlo Park, CA: AAAI Press.

Connell, J., & Brady, M. (1987). Generating and generalizing models of visual objects. *Artificial Intelligence*, *31*, 159–183.

Cook, D., Hall, L., Stark, L., & Bowyer, K. (1993). Learning combination of evidence functions in object recognition. In *Papers from the the AAAI Fall Symposium on Machine Learning in Computer Vision: What, Why, and How?* Technical Report No. FS-93-04 (pp. 139–143). Menlo Park, CA: AAAI Press.

Cromwell, R., & Kak, A. (1991). Automatic generation of object class descriptions using symbolic learning techniques. In *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 710–717). Menlo Park, CA: AAAI Press.

DeLong, E., DeLong, D., & Clarke-Peterson, D. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, *44*, 837–845.

Domingos, P. (1999). MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining* (pp. 155–164). New York, NY: ACM Press.

Dorfman, D., & Alf, E., Jr. (1969). Maximum likelihood estimation of parameters of signal-detection theory and determination of confidence intervals—rating method data. *Journal of Mathematical Psychology*, *6*, 487–496.

Dorfman, D., Berbaum, K., & Metz, C. (1992). Receiver Operating Characteristic rating analysis: Generalization to the population of readers and patients with the Jackknife method. *Investigative Radiology*, *27*, 723–731.

Draper, B. (1996). Learning grouping strategies for 2D and 3D object recognition. In *Proceedings of the Image Understanding Workshop* (pp. 1447–1454). San Francisco, CA: Morgan Kaufmann.

Draper, B. (1997). Learning control strategies for object recognition. In K. Ikeuchi & M. Veloso (Eds.), *Symbolic visual learning* (pp. 49–76). New York, NY: Oxford University Press.

Draper, B., Brodley, C., & Utgoff, P. (1994). Goal-directed classification using linear machine decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*, 888–893.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis.* New York, NY: John Wiley & Sons.

Egan, J. (1975). *Signal detection theory and ROC analysis.* New York, NY: Academic Press.

Ezawa, K., Singh, M., & Norton, S. (1996). Learning goal-oriented Bayesian networks for telecommunications risk management. In *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 139–147). San Francisco, CA: Morgan Kaufmann.

Fawcett, T., & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, *1*, 291–316.

Fayyad, U., Smyth, P., Burl, M., & Perona, P. (1996). Learning to catalog science images. In S. Nayar & T. Poggio (Eds.), *Early visual learning* (pp. 237–268). New York, NY: Oxford University Press.

Firschein, O., & Strat, T. (Eds.). (1997). *RADIUS: Image understanding for imagery intelligence.* San Francisco, CA: Morgan Kaufmann.

Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 148–156). San Francisco, CA: Morgan Kaufmann.

Freund, Y., Seung, H., Shamir, E., & Tishby, N. (1997). Selective sampling using the Query by Committee algorithm. *Machine Learning*, *28*, 133–168.

Green, D., & Swets, J. (1974). *Signal detection theory and psychophysics.* New York, NY: Robert E. Krieger Publishing.

Gros, P. (1993). Matching and clustering: Two steps towards automatic object model generation in computer vision. In *Papers from the the AAAI Fall Symposium on Machine Learning in Computer Vision: What, Why, and How?* Technical Report No. FS-93-04 (pp. 40–44). Menlo Park, CA: AAAI Press.

Gutta, S., Huang, J., Imam, I., & Weschler, H. (1996). Face and hand gesture recognition using hybrid classifiers. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition* (pp. 164–169). Los Alamitos, CA: IEEE Press.

Hand, D., & Till, R. (2001). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, *45*, 171–186.

Hanley, J., & McNeil, B. (1982). The meaning and use of the area under a Receiver Operating Characteristic (ROC) curve. *Radiology*, *143*, 29–36.

Hinkley, D. (1983). Jackknife methods. In S. Kotz, N. Johnson, & C. Read (Eds.), *Encyclopedia of statistical sciences* (Vol. 4, pp. 280–287). New York, NY: John Wiley & Sons.

John, G., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (pp. 338–345). San Francisco, CA: Morgan Kaufmann.

Keppel, G., Saufley, W., & Tokunaga, H. (1992). *Introduction to design and analysis* (2nd ed.). New York, NY: W.H. Freeman.

Kim, Z., & Nevatia, R. (1999). Uncertain reasoning and learning for feature grouping. *Computer Vision and Image Understanding*, *76*, 278–288.

Kim, Z., & Nevatia, R. (2000). Learning Bayesian networks for diverse and varying numbers of evidence sets. In *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 479–486). San Francisco, CA: Morgan Kaufmann.

Kubat, M., Holte, R., & Matwin, S. (1996). Learning when negative examples abound. In *Proceedings of the 1997 European Conference on Machine Learning* (pp. 146–153). Berlin: Springer-Verlag

Kubat, M., Holte, R., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite images. *Machine Learning*, *30*, 195–215.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 179–186). San Francisco, CA: Morgan Kaufmann.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 223–228). Menlo Park, CA: AAAI Press.

Langley, P., & Simon, H. (1995). Applications of machine learning and rule induction. *Communications of the ACM*, *38*, 54–64.

Levitt, T., Agosta, J., & Binford, T. (1989). Model-based influence diagrams for machine vision. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence* (pp. 371–388). New York, N.Y.: Elsevier Science.

Lewis, D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 148–156). San Francisco, CA: Morgan Kaufmann.

Lin, C., & Nevatia, R. (1998). Building detection and description from a single intensity image. *Computer Vision and Image Understanding*, *72*, 101–121.

Maloof, M. (2000). An initial study of an adaptive hierarchical vision system. In *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 567–573). San Francisco, CA: Morgan Kaufmann.

Maloof, M. (2002). On machine learning, ROC analysis, and statistical tests of significance. In *Proceedings of the Sixteenth International Conference on Pattern Recognition.* Los Alamitos, CA: IEEE Press.

Maloof, M., Beiden, S., & Wagner, R. (2002). *Analysis of competing classifiers in terms of components of variance of ROC accuracy measures* (Technical Report No. CS-02-01). Washington, DC: Department of Computer Science, Georgetown University. (http://www.cs.georgetown.edu/∼maloof/pubs/cstr-02-01.html)

Maloof, M., Duric, Z., Michalski, R., & Rosenfeld, A. (1996). Recognizing blasting caps in X-ray images. In *Proceedings of the Image Understanding Workshop* (pp. 1257–1261). San Francisco, CA: Morgan Kaufmann.

Maloof, M., Langley, P., Binford, T., & Nevatia, R. (1998). Generalizing over aspect and location for rooftop detection. In *Proceedings of the Fourth IEEE Workshop on Applications of Computer Vision* (pp. 194–199). Los Alamitos, CA: IEEE Press.

Maloof, M., Langley, P., Binford, T., & Sage, S. (1998). *Learning to detect rooftops in overhead imagery* (Technical Report No. 98-1). Palo Alto, CA: Institute for the Study of Learning and Expertise.

Maloof, M., Langley, P., Sage, S., & Binford, T. (1997). Learning to detect rooftops in aerial images. In *Proceedings of the Image Understanding Workshop* (pp. 835–845). San Francisco, CA: Morgan Kaufmann.

Maloof, M., & Michalski, R. (1997). Learning symbolic descriptions of shape for object recognition in X-ray images. *Expert Systems with Applications*, *12*, 11–20.

Metz, C. (1978). Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, *VIII*, 283–298.

Metz, C. (1989). Some practical issues of experimental design and data analysis in radiological ROC studies. *Investigative Radiology*, *24*, 234–245.

Michalski, R., Mozetic, I., Hong, J., & Lavrac, H. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three

medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 1041–1045). Menlo Park, CA: AAAI Press.

Miller, D., & Uyar, H. (1997). A mixture of experts classifier with learning based on both labeled and unlabeled data. In M. Mozer, M. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems* (Vol. 9). Cambridge, MA: MIT Press.

Mohan, R., & Nevatia, R. (1989). Using perceptual organization to extract 3-D structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *11*, 1121–1139.

Mossman, D. (1999). Three-way ROCs. *Medical Decision Making*, *19*, 78–89.

Nayar, S., & Poggio, T. (Eds.). (1996). *Early visual learning.* New York, NY: Oxford University Press.

Noronha, S., & Nevatia, R. (1997). Detection and description of buildings from multiple aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 588–594). Los Alamitos, CA: IEEE Press.

Osuna, E., Freund, R., & Girosi, F. (1997). Training Support Vector Machines: An application to face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 130–136). Los Alamitos, CA: IEEE Press.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. (1994). Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 217–225). San Francisco, CA: Morgan Kaufmann.

Pomerleau, D. (1996). Neural network vision for robot driving. In S. Nayar & T. Poggio (Eds.), *Early visual learning* (pp. 161–181). New York, NY: Oxford University Press.

Pope, A., & Lowe, D. (1996). Learning probabilistic appearance models for object recognition. In S. Nayar & T. Poggio (Eds.), *Early visual learning* (pp. 67–97). New York, NY: Oxford University Press.

Pope, A., & Lowe, D. (2000). Probabilistic models of appearance for 3-D object recognition. *International Journal of Computer Vision*, *40*, 149–167.

Provan, G., Langley, P., & Binford, T. (1996). Probabilistic learning of three-dimensional object models. In *Proceedings of the Image Understanding Workshop* (pp. 1403–1413). San Francisco, CA: Morgan Kaufmann.

Provost, F., & Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 43–48). Menlo Park, CA: AAAI Press.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 445–453). San Francisco, CA: Morgan Kaufmann.

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

Rowley, H., Baluja, S., & Kanade, T. (1996). Neural network-based face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 203–208). Los Alamitos, CA: IEEE Press.

Sarkar, S., & Soundararajan, P. (2000). Supervised learning of large perceptual organization: Graph spectral partitioning and learning automata. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 504–525.

Segen, J. (1994). GEST: A learning computer vision system that recognizes hand gestures. In R. Michalski & G. Tecuci (Eds.), *Machine learning: A multistrategy approach* (Vol. 4, pp. 621–634). San Francisco, CA: Morgan Kaufmann.

Sengupta, K., & Boyer, K. (1993). Incremental model base updating: Learning new model sites. In *Papers from the the AAAI Fall Symposium on Machine Learning in Computer Vision: What, Why, and How?* Technical Report No. FS-93-04 (pp. 1–5). Menlo Park, CA: AAAI Press.

Shepherd, B. (1983). An appraisal of a decision tree approach to image classification. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (pp. 473–475). San Francisco, CA: Morgan Kaufmann.

Soderland, S., & Lehnert, W. (1994). Corpus-driven knowledge acquisition for discourse analysis. In *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 827–832). Menlo Park, CA: AAAI Press.

Swets, J. (1988). Measuring the accuracy of diagnostic systems. *Science*, *240*, 1285–1293.

Swets, J., & Pickett, R. (1982). *Evaluation of diagnostic systems: Methods from signal detection theory.* New York, NY: Academic Press.

Teller, A., & Veloso, M. (1997). PADO: A new learning architecture for object recognition. In K. Ikeuchi & M. Veloso (Eds.), *Symbolic visual learning* (pp. 77–112). New York, NY: Oxford University Press.

Thompson, M., & Zucchini, W. (1986). On the statistical analysis of ROC curves. *Statistics in Medicine*, *18*, 452–462.

Turney, P. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, *2*, 369–409.

Viola, P. (1993). Feature-based recognition of objects. In *Papers from the the AAAI Fall Symposium on Machine Learning in Computer Vision: What, Why, and How?* Technical Report No. FS-93-04 (p. 60–64). Menlo Park, CA: AAAI Press.

Wagner, R., Beiden, S., & Metz, C. (2001). Continuous versus categorical data for ROC analysis: Some quantitative considerations. *Academic Radiology*, *8*, 328–334.

Walpole, R., Myers, R., & Myers, S. (1998). *Probability and statistics for engineers and scientists* (6th ed.). Upper Saddle River, NJ: Prentice-Hall.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, *5*, 241–259.

Woods, K., Cook, D., Hall, L., Bowyer, K., & Stark, L. (1995). Learning membership functions in a function-based object recognition system. *Journal of Artificial Intelligence Research*, *3*, 187–222.

Woods, K., Kegelmeyer, W., & Bowyer, K. (1997). Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 405–410.

Zurada, J. (1992). *Introduction to artificial neural systems.* St. Paul, MN: West Publishing.