Improving Rooftop Detection in Aerial Images Through Machine Learning

Marcus A. Maloof[†](maloof@apres.stanford.edu)
Pat Langley[†](langley@newatlantis.isle.org)
Thomas Binford[‡](binford@cs.stanford.edu)
Stephanie Sage[†](sage@icaria.isle.org)

[†]Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306

[‡]Robotics Laboratory, Department of Computer Science
Stanford University, Stanford, CA 94305

Abstract

In this paper, we examine the use of machine learning to improve a rooftop detection process, which is one step in a vision system that recognizes buildings in overhead imagery. We review the problem of analyzing aerial images and describe an existing vision system that automates the recognition of buildings in such images. After this, we briefly review two well-known learning algorithms, representing different inductive biases, that we selected to improve rooftop detection. An important aspect of this problem is that the data sets are highly skewed and the cost of mistakes differs for the two classes, so we evaluate the algorithms under varying misclassification costs using ROC analysis. We report three sets of experiments designed to illuminate facets of applying machine learning to the image analysis task. One set of studies focuses on within-image learning, in which both training and testing data are derived from the same image. Another addresses between-image learning, in which training and testing sets come from different images. A final set investigates learning using all available image data in an effort to determine the best performing method. Experimental results demonstrate that useful generalization occurs when training and testing on data derived from images that differ in location and in aspect. Furthermore, they demonstrate that, under most conditions and across a range of misclassification costs, a trained naive Bayesian classifier exceeded, by as much as a factor of two, the predictive accuracy of nearest neighbor and a handcrafted linear classifier, the solution currently being used in the building detection system. Analysis of learning curves reveals that naive Bayes achieved superiority using as little as 6% of the available training data.

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 focus the analyst's attention on interesting events, but current image understanding systems are not yet 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 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 give 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 two 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

The image analyst interprets 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 faced by the image analyst 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 that confront the analyst.

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 the 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 two learning methods that might give it more robust behavior.

3. An Architecture for Image Analysis

Lin and Nevatia (1996) report 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 edgels, and then invokes a line finder to group edgels into lines. Junctions and parallel lines are identified and combined to form three-sided structures or "Us". The algorithm then groups selected Us 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 from 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 Three 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. These knowledge structures state the conditions under which the concept, in this case "rooftop", is satisfied.

In a previous study (Maloof *et al.* 1997), we evaluated a variety of machine learning methods for the rooftop detection task and selected the two that showed promise of achieving a balance between the true positive and false positive rates: nearest neighbor, and naive Bayes. 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 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; here we will focus on the "simple" nearest neighbor scheme, which uses only the nearest case for prediction.

The naive Bayesian classifier (e.g., 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

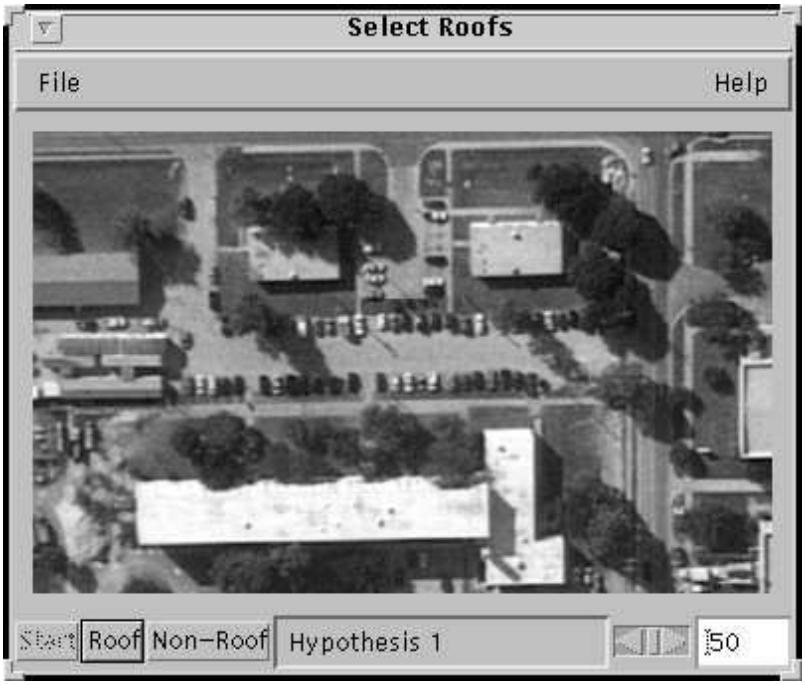

Figure 1. Visualization interface for labeling rooftop candidates. The system presents candidates to a user who labels them by clicking either the 'Roof' or 'Non-Roof' button. It also incorporates a simple learning algorithm to provide feedback to the user about the statistical properties of a candidate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

the class with the highest posterior probability. Like nearest neighbor, naive Bayes has known limitations, such as sensitivity to attribute correlations and an inability to represent multiple decision regions, but in practice it behaves well on many natural domains.

Currently, BUDDS uses a handcrafted linear classifier for rooftop detection (Lin & Nevatia 1996), which is equivalent to a perceptron classifier (e.g., Zurada 1992). Although we did not train this method as we did naive Bayes and nearest neighbor, we included this method in our evaluation for the purpose of comparison. This method represents concepts using a collection of weights $\mathbf{w}$ and a threshold θ. To classify an instance, which we represent as a vector of n numbers $\mathbf{x}$, we compute the output o of the classifier using the formula:

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

For our application, the classifier predicts the positive class if the output is +1 and predicts the negative class otherwise. There are a number of established methods for training perceptrons, but our preliminary studies suggested that they fared worse than the manually set weights, so we did not use the learned perceptrons here. Henceforth, we will refer to the handcrafted linear classifier used in BUDDS as the "BUDDS classifier".

Table 1. Characteristics of the images and data sets. We began with a nadir and an oblique image of an area of Fort Hood, Texas, and derived three subimages from each that contained concentrations of buildings. We then used BUDDS to extract rooftop candidates and labeled each as either a positive or negative example of the concept "rooftop".

Image Number	Original Image	Location	Aspect	Positive Examples	Negative Examples
1	FHOV1027	1	Nadir	197	982
2	FHOV625	1	Oblique	238	1955
3	FHOV1027	2	Nadir	71	2645
4	FHOV625	2	Oblique	74	3349
5	FHOV1027	3	Nadir	87	3722
6	FHOV625	3	Oblique	114	4395

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 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 two images, FHOV1027 and FHOV625, of Fort Hood, Texas, which were collected as part of the RADIUS program (Firschein & Strat 1997). These images cover the same area but were taken from different viewpoints, one from a nadir angle and the other from an oblique angle. We subdivided each image into three subimages, focusing on locations that contained concentrations of buildings, to maximize the number of positive rooftop candidates. This gave us three pairs of images, each pair covering the same area but viewed from different aspects.

Our aim was to improve BUDDS so we used this system to generate candidate rooftops for each image, producing six data sets. Following Lin and Nevatia (1996), 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 shadows near the corners of the candidate. Negative evidence included the existence of lines that cross the candidate, L-junctions adjacent 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.

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 us satisfactory labeling results.

Analysis revealed the difficulties with using such relations to actual rooftops in the labeling process. One is that they ignore information about the candidate's shape; a good rooftop should be a parallelogram, yet nearness of vertices is neither sufficient or necessary 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 in 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 in JAVA, 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.

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[1] 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 confirms 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 Section 7 by the setting sensitivity parameter so that the system presented all candidates as uncertain. Even handicapped in this manner, the interface required 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 still seems quite efficient.

In summary, what began as the 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, which 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

1. The user can set this parameter using the slider bar and number field in the bottom right corner of Figure 1.

candidates before they are passed on to the user (automatically labeling the most confident ones). Techniques such as *selective sampling* (e.g., Freund *et al.* 1997) and *uncertainty sampling* (Lewis & Catlett 1994) should prove useful toward these ends.

6. Cost-Sensitive Learning and Skewed Data

Two aspects of the rooftop selection task influenced our 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 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 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 suggest the need for modified learning algorithms that can achieve high accuracy on the minority class. Second, they require an experimental methodology that lets us compare different methods on domains 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 propose some cost-sensitive learning methods and an approach to experimental evaluation.

6.1 Favoritism Toward the Majority Class

In a previous study (Maloof *et al.* 1997), we evaluated several algorithms without taking into account the cost of classification errors and got confusing experimental results. Some methods, like the standard error-driven algorithm for revising perceptron weights (e.g., Zurada 1992), 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.[2] 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 true positive and false positive rates. Indeed, always predicting the majority class for our problem yields a hit rate of 95 percent, which makes it a misleading measure of performance.

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

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

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. On the other hand, if the class distribution runs counter to the relative cost of mistakes, as in our domain, then we must do something to compensate, both in the learning algorithm itself and in measuring its performance.

Breiman *et al.* (1984) note the close relation between the distribution of classes and the relative cost of errors. In particular, they point 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.

6.2 Cost-Sensitive Learning Methods

Empirical comparisons among 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 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. Turney (1995) presents results from an empirical evaluation of algorithms that take into account both the cost of tests to measure attributes and the cost of classification error.

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. In essence, we want to incorporate a cost heuristic into the algorithms so we can bias them toward making mistakes on the less costly class rather than on the more expensive class.

To accomplish this, we defined a cost for each class on the range $[0.0, 1.0]$ that indicates the relative cost of making a mistake on one class versus another. Zero indicates that errors cost nothing, whereas one means that errors are maximally expensive. To incorporate a cost heuristic into the algorithms, we chose to modify the performance element of the algorithms, rather than the learning element, by using the cost heuristic to adjust the decision boundary at which the algorithm selects one class versus the other.

Recall that naive Bayes predicts the class with the highest posterior probability as computed using Bayes' rule, so we want the cost heuristic to bias prediction in favor of the more expensive class. For a cost parameter $c_j \in [0.0, 1.0]$, we computed the expected cost δ_j for the class ω_j using the formula:

$$\delta_j = P(\omega_j|\mathbf{x}) + c_j(1 - P(\omega_j|\mathbf{x}))$$

where $\mathbf{x}$ is the query, and $P(\omega_j|\mathbf{x})$ is the posterior probability of the jth class given the query. The cost-sensitive version of naive Bayes predicts the class ω_j with the least expected cost δ_j.

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

$$\delta_j = d_E(\mathbf{x}, \mathbf{x}_j) - c_j d_E(\mathbf{x}, \mathbf{x}_j)$$

where $\mathbf{x}_j$ is the closest neighbor from class ω_j to the query point, and $d_E(x, y)$ is the Euclidean distance function. The cost-sensitive version of nearest neighbor predicts the class with the least expected cost. This modification also works for k nearest neighbor, which considers the k closest neighbors when classifying unknown instances.

Finally, because our modifications focused on the performance elements rather than on the learning algorithms, we can make similar changes to the BUDDS classifier. Since this classifier uses 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:

$$\theta' = \theta - \sum_{j=1}^{2} \text{sgn}(\omega_j) c_j \sigma_j$$

where θ is the original threshold for the linear discriminant function, $\text{sgn}(\omega_j)$ returns positive for the positive class and negative 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 BUDDS classifier predicts the positive class if the weighted sum of an instance's attributes surpasses the adjusted threshold θ'; otherwise, it predicts the negative class.

6.3 ROC Analysis for Evaluating Performance

Our second 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 (e.g., Pazzani *et al.* 1994; Fawcett & Provost 1997).

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 ex-

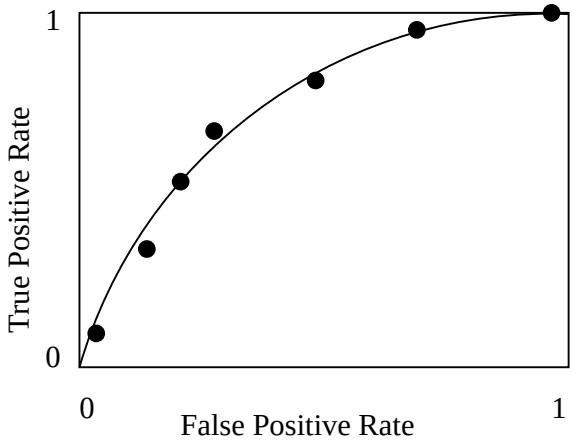

Figure 2. An idealized Receiver Operating Characteristic (ROC) curve.

periments, 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 cost ratio or the class distribution, and to plot the false positive rate against the false negative rate for each situation. Although researchers have used such ROC curves in signal detection and psychophysics for decades (e.g., Green & Swets 1974; Egan 1975), this technique has only recently begun to filter into machine learning research (e.g., Ezawa, Singh, & Norton 1996; Maloof *et al.* 1997; Provost & Fawcett 1997).

Figure 2 shows an idealized ROC curve generated by varying the cost parameter of a cost-sensitive learning algorithm. The lower left corner of the figure 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 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. 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 being 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 raises problems 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 is nested within the other. As we will see, this relation typically holds for our cost-sensitive algorithms in the rooftop detection domain.

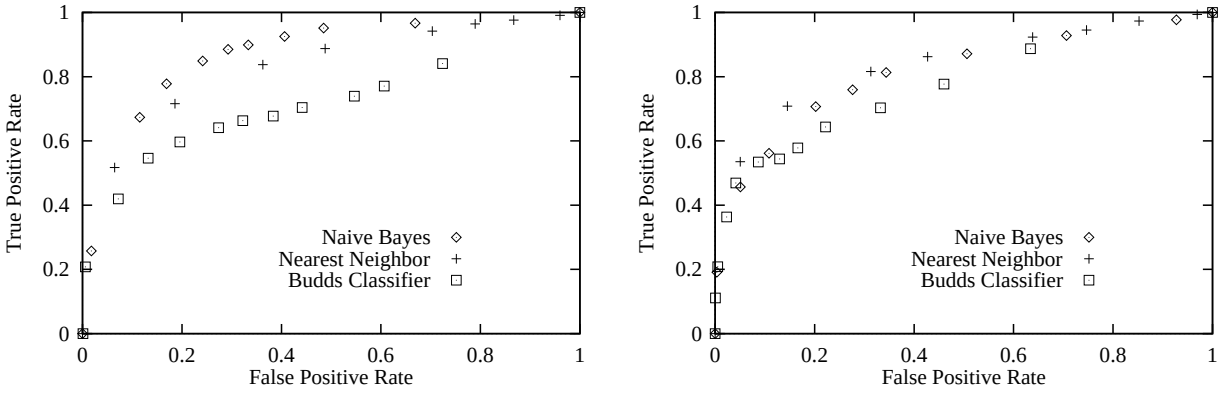

Figure 3. ROC curves for Images 1 and 2. We ran each method by training and testing using data derived from the same image over a range of misclassification costs. We conducted ten such runs and plotted the average true positive and false positive rates. These images are of the same location but different aspects: Image 1 is a nadir view, while Image 2 is an oblique.

7. Experimental Studies

To investigate the use of machine learning for the task of rooftop detection, we conducted experiments using the cost-sensitive versions of naive Bayes, nearest neighbor, and the BUDDS classifier. As typically done in such studies, in each experiment we trained the induction methods on data (rooftop candidates) separate from those used to test the learned classifiers. As we will see, the experiments differed in whether the training and test cases came from the same or distinct images, which let us examine different forms of generalization beyond the training data.

7.1 Within-Image Learning

Our first experimental study 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. Since costs are relative (i.e., $c_+ = 0.0$ and $c_- = 0.5$ is equivalent to $c_+ = 0.25$ and $c_- = 0.75$) and our domain involved only two classes, we varied the cost parameter for only one class at a time and fixed the other at zero. 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. Because the BUDDS classifier was hand-configured, it had no training phase, so we applied it directly to the instances 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 3 presents the ROC curves for Images 1 and 2. Naive Bayes and nearest neighbor give similar results, but both fare better than the BUDDS classifier. Rather than present the curves

Table 2. Results for within-image experiments. For each image, we generated ROC curves by training and testing each method over a range of costs. We used the approximate area under the curve as the measure of performance, which appear with 95% confidence intervals. Naive Bayes performed best overall, with the BUDDS classifier outperforming nearest neighbor on three of the six images.

	Approximate Area under ROC Curve					
Classifier	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6
Naive Bayes	0.870±0.008	0.812±0.017	0.962±0.013	0.908±0.025	0.869±0.016	0.835±0.025
BUDDS Classifier	0.717±0.009	0.773±0.004	0.899±0.015	0.901±0.007	0.833±0.021	0.849±0.010
Nearest Neighbor	0.823±0.019	0.833±0.016	0.911±0.010	0.801±0.028	0.819±0.027	0.739±0.017

for the remaining four images, we follow Swets (1988) and report, in Table 2, the area under each ROC curve, which we approximated by summing the areas of the trapezoids defined by each pair of adjacent points in the ROC curve. For all images except for Image 6, naive Bayes produced curves with areas greater than those for the BUDDS classifier, thus generally supporting our research hypothesis. On Images 4, 5, and 6, nearest neighbor did worse than the handcrafted method, which runs counter to our prediction.

7.2 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 to 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 understand better 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: 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.

One experiment focused on how the methods generalize over aspect. Recall from Table 1 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 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 4. The areas under these curves and their 95% confidence intervals appear in Table 3.

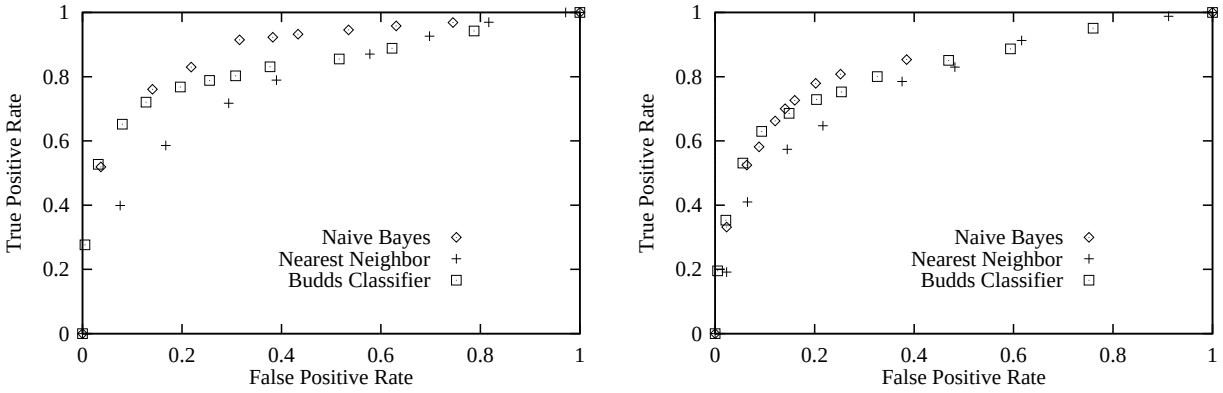

Figure 4. ROC curves for experiments that tested generalization over aspect. Left: For each location, we trained each method on the oblique image and tested the resulting concept descriptions on the nadir image. We plotted the average true positive and false positive rates. Right: We followed a similar methodology, except that we trained the methods on the nadir images and tested on the oblique images.

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 3 shows that naive Bayes generates a curve with an area of 0.878 for the nadir images, but produces a curve with an area of 0.842 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.

Upon comparing the behavior of different methods, we find that, for oblique to nadir generalization, naive Bayes (with an area under the ROC curve of 0.878) performs better than the BUDDS classifier, with an area of 0.837, which in turn did better than nearest neighbor (0.795). For nadir to oblique generalization, naive Bayes performs slightly better than the BUDDS classifier, which produce areas of 0.842 and 0.831, respectively. Nearest neighbor's curve in this situation covers an area of 0.785, which is considerably smaller.

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 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 5.

In this context, we again see evidence that the oblique images presented a more difficult recognition task than the nadir aspect, since the oblique areas are less than those for the nadir images. Comparing the behavior of the various methods, Table 3 shows that, for the nadir aspect, naive Bayes performs slightly better than the BUDDS classifier, which give areas of 0.901 and 0.837. As before, both did better than nearest neighbor, which yielded an area of 0.819 under its ROC curve. When generalizing over location on the oblique images, naive Bayes and the BUDDS classifier produced ROC curves with equal areas of 0.831. These were considerably better than nearest neighbor's, which had an area of 0.697.

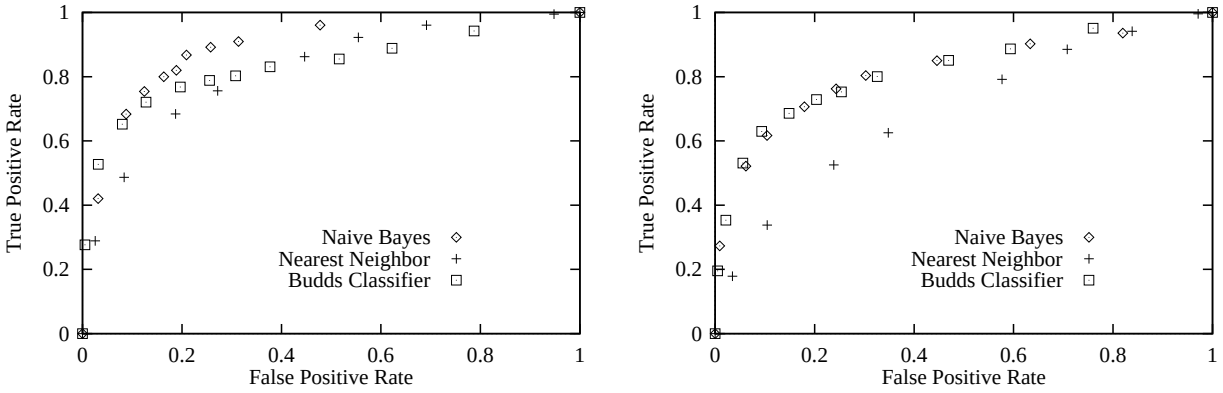

Figure 5. ROC curves for experiment that tested generalization over location. Left: For each pair of images for the nadir aspect, we trained the methods on that pair and tested the resulting concept descriptions on the third image. We then plotted the average true positive and false positive rates. Right: We applied the same methodology using the images for the oblique aspect.

Thus, the results with the naive Bayesian classifier support our main hypothesis. In all experimental conditions this method fared better than or equal to the BUDDS linear classifier. On the other hand, the behavior of nearest neighbor typically gave worse results than the handcrafted rooftop detector, 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 (see Table 3). Simple calculation shows that, for the within-image condition (Table 2), naive Bayes produced an average ROC area of 0.9 for the nadir images and 0.851 for the oblique images. Similarly, nearest neighbor averaged 0.851 for the nadir images and 0.791 for the oblique images. Most of these these areas are substantially higher than the analogous areas that resulted when these methods generalized across location and aspect. The one exception is that naive Bayes actually did equally well when generalizing over location for the nadir image, but the results generally support our prediction.

Also note that naive Bayes' performance degraded less than that of nearest neighbor when generalizing to unseen images. This can be seen by comparing the differences between each method's performance in the within-image condition and in the between-image conditions. For example, naive Bayes' average degradation in performance over all experimental conditions was 0.013, while nearest neighbor's was 0.47. This constitutes further evidence that naive Bayes is better suited for this domain, at least when operating over the nine features used in our experiments.

7.3 Learning from All Available Images

Our next study used all of the rooftop candidates generated from the six Fort Hood images, since we wanted to replicate our previous results in a situation similar to that we envision being used in practice, which would draw on training cases from all images. Based on the earlier experiments, we anticipated that the naive Bayesian classifier would yield an ROC curve of greater area than those of the other methods.

Table 3. Results for between-image experiments. We again used the approximate area under the ROC curve as the measure of performance, along with 95% confidence intervals. Naive Bayes performed the best, while the BUDDS classifier generally outperformed nearest neighbor. 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		Location Experiment		Within Image	
Classifier	Nadir	Oblique	Nadir	Oblique	Nadir	Oblique
Naive Bayes	0.878±0.042	0.842±0.063	0.901±0.079	0.831±0.067	0.900±0.012	0.851±0.022
BUDDS Classifier	0.837±0.085	0.831±0.068	0.837±0.085	0.831±0.068	0.837±0.085	0.831±0.068
Nearest Neighbor	0.795±0.035	0.785±0.053	0.819±0.058	0.697±0.027	0.851±0.019	0.791±0.020

Combining the rooftop candidates from all six images gave us 17,829 instances, 781 labeled positive and 17,048 labeled negative. We ran each algorithm ten times over a range of costs. For each run and set of cost parameters, we randomly split the data into training (60%) and testing (40%) sets, then averaged the results for each cost level over its ten runs.

Figure 6 shows the resulting ROC curves, which plot the true positive and false positive rates, whereas Table 4 gives the approximate area under these curves. As anticipated, naive Bayes performed the best overall, producing a curve with area 0.85. Nearest neighbor fared slightly better than the BUDDS classifier, yielding an area of 0.801, compared to 0.787 for the latter.

In practice, image analysts will not evaluate a classifiers performance using area under the ROC curve but, rather, 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 the naive Bayesian classifier when it achieves a true positive rate of 0.84 and a false positive rate of 0.27, the third diamond from the right in Figure 6. To obtain the same true positive rate, the BUDDS classifier produced a 0.62 false positive rate. This means that, for a given true positive rate, naive Bayes reduced the false positive rate by more than half over the handcrafted classifier. Hence, for the images we considered, the naive Bayesian classifier would have rejected 5,969 more non-rooftops than BUDDS. Similarly, by fixing the false positive rate, naive Bayes improved the true positive rate by 0.12 over the BUDDS classifier. In this case, the Bayesian classifier would have found 86 more rooftops than BUDDS would have detected.

7.4 Rates of Learning

We were also interested in the behavior of the learning methods as they processed increasing amounts of training data. Our long-term goal is to embed the learned classifier in an interactive system that supports an image analyst. For this reason, we would prefer a learning algorithm that achieves high accuracy from relatively few training cases, since this should reduce the load on the human analyst.

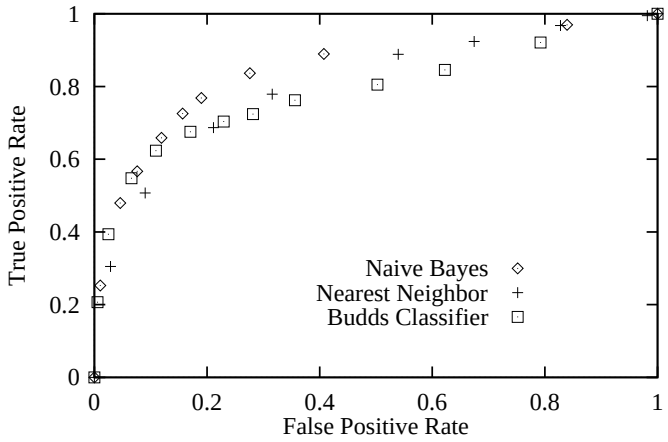

Figure 6. ROC curve for the experiment using all available image data. We ran each method over a range of costs using a training set (60%) and a testing set (40%) and averaged the true positive and false positive rates over ten runs. Naive Bayes produced the curve with the largest area, but nearest neighbor also yielded a curve larger in area than that for the BUDDS classifier.

To this end, we carried out a final experiment in which we systematically varied the number of training cases available to the learning method. We again used all of the available rooftop candidates, splitting the data into training (60%) and test (40%) sets, but further dividing the training set randomly into ten subsets (10%, 20%, ..., 100%). We ran the learning algorithms on each of the training subsets and evaluated the acquired concept descriptions on the reserved testing data, averaging our results over 25 separate training/test splits.

Figure 7 shows the resulting learning curves, each point of which corresponds to the average area under the ROC curves for a given number of training cases. As expected, the learning curve for the the BUDDS classifier is flat, since it involves no training and we simply applied it to the same test set for each number of training cases. However, nearest neighbor produces a curve that starts below that of the BUDDS classifier and then surpasses it after seeing 70% of the training data. Naive Bayes shows similar improvement with increasing amounts of training data, but its performance was better than the BUDDS classifier from the start, after observing only 10% of the training data. This equates to roughly 6% of the available data and is less than the amount of data derived from one image. Not only was naive Bayes the best performing method, but also it was able to achieve this performance using very little of the available training data.

7.5 Summary

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 rooftop detection task 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

Table 4. 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	Approximate Area
Naive Bayes	0.850±0.008
Nearest Neighbor	0.801±0.008
BUDDS Classifier	0.787±0.008

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. Nevertheless, under all circumstances, the performance of naive Bayes was equal to or better than that of the handcrafted linear classifier. Finally, we also discovered that the performance of the methods degraded when generalizing to unseen images, but that the performance of naive Bayes degraded less than that of nearest neighbor.

Our final experiment used all of the available image data for learning and demonstrated that naive Bayes and nearest neighbor outperformed the BUDDS classifier. Further analysis of specific points on the ROC curves revealed that naive Bayes improved upon the false positive rate of the handcrafted solution by more than a factor of two for true positive rates of 0.84 and higher. Learning curves demonstrated that naive Bayes achieved superior performance using very little of the available training data.

8. 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 *et al.* 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 learns to recognize desired objects using these machine-produced classifiers. Shepherd (1983) used decision-tree induction to classify shapes of chocolates for an industrial vision application. Cromwell and Kak (1991) took a similar approach for 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 *et al.* 1996) discussed learning in a multi-step vision system for the same detection problem.

Several researchers have also investigated learning for three-dimensional vision systems. Papers by Conklin (1993), Connell and Brady (1987), Cook *et al.* (1993), Provan, Langley, and Binford

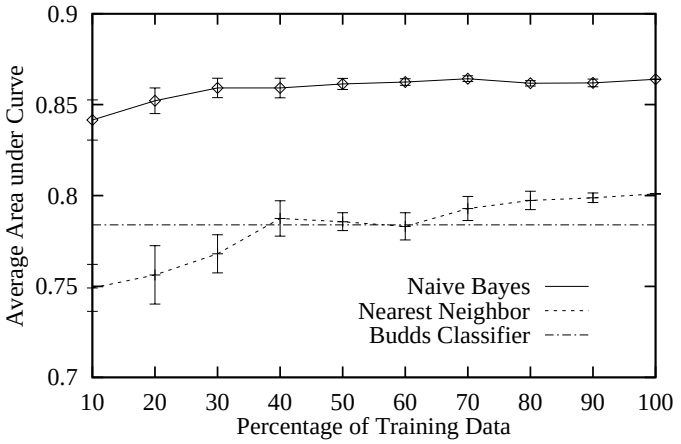

Figure 7. Learning curves for area under the ROC curve using all available image data. We ran each method on increasing amounts of training data and evaluated the resulting concept descriptions on reserved testing data. Each point is an average of ten runs.

(1996), and Sengupta and Boyer (1993) all describe inductive approaches aimed at improving object recognition. The aim here 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).

Most work on visual learning ignores the importance of misclassification costs, but our work along these lines has some precedents. In particular, Draper, Brodley, and Utgoff (1994) incorporate 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, Bowyer, and Kegelmeyer (1996), as well as Rowley, Baluja, and Kanade (1996), report similar work that takes into account the cost of errors.

Much of the research on visual learning uses images of scenes or objects viewed at eye level (e.g., Draper 1997; Teller & Veloso 1997). One exception is Connell and Brady's (1987) work on learning structural descriptions of airplanes from aerial views. Their method converted training images into semantic networks that it then generalized by comparing to descriptions of other instances. However, the authors do not appear to have tested experimentally their algorithm's ability to accurately classify objects in new images. Another example is the SKICAT system (Fayyad *et al.* 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. Asker and Maclin (1997) extend JARTool by using an ensemble of 48 neural networks to improve performance. Using ROC curves, they demonstrate that the ensemble achieved better performance than either the individual learned classifiers or the one used originally in JARTool. They also document 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 problems we encountered.

Finally, Draper (1996) reports a careful study of learning in the context of analyzing aerial images. His approach adapts methods for reinforcement learning to assign credit in multi-stage recognition procedure (for software similar to BUDDS), then uses an induction method (backpropagation in neural networks) to learn conditions on operator selection. He presents initial results on a RADIUS task that also involves the detection of roofs. 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.

9. Concluding Remarks

Although this study has provided some insight into the role of machine learning in image analysis, 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.

In addition, 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.

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.

In conclusion, our studies suggest that machine learning has an important role to play in improving the accuracy, and thus the robustness, of image analysis systems. However, we need additional experiments to give better understanding of the factors affecting between-image generalization and we need to extend learning to additional levels of the image understanding process. Also, 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.

Acknowledgements

The authors thank Ram Nevatia, Andres Huertas, and Andy Lin for their assistance in obtaining the images and data used for experimentation and providing valuable comments and advice. We would also like to thank Dan Shapiro for discussions about decision theory and Wayne Iba for his assistance with naive Bayes. This work 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 research 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.; and Albert, M. 1991. Instance-based learning algorithms. *Machine Learning* 6:37–66.

Asker, L., and Maclin, R. 1997. Feature engineering and classifier selection: a case study in Venusian volcano detection. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 3–11. San Francisco, CA: Morgan Kaufmann.

Beymer, D., and Poggio, T. 1996. Image representations for visual learning. *Science* 272:1905–1909.

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.; Friedman, J.; Olshen, R.; and Stone, C. 1984. *Classification and regression trees.* Belmont, CA: Wadsworth.

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

Chan, L.; Nasrabadi, N.; and 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*, 114–119. Los Alamitos, CA: IEEE Press.

Clark, P., and 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 *Working Notes of the AAAI Fall Symposium on Machine Learning in Computer Vision*, 10–14. Menlo Park, CA: AAAI Press.

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

Cook, D.; Hall, L.; Stark, L.; and Bowyer, K. 1993. Learning combination of evidence functions in object recognition. In *Working Notes of the AAAI Fall Symposium on Machine Learning in Computer Vision*, 139–143. Menlo Park, CA: AAAI Press.

Cromwell, R., and Kak, A. 1991. Automatic generation of object class descriptions using symbolic learning techniques. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 710–717.

Draper, B.; Brodley, C.; and Utgoff, P. 1994. Goal-directed classification using linear machine decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(9):888–893.

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

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

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

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

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

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

Firschein, O., and Strat, T., eds. 1997. *RADIUS: image understanding for imagery intelligence*. San Francisco, CA: Morgan Kaufmann.

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

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

Green, D., and 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 *Working Notes of the AAAI Fall Symposium on Machine Learning in Computer Vision*, 40–44. Menlo Park, CA: AAAI Press.

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

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

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

Langley, P., and Simon, H. 1995. Applications of machine learning and rule induction. *Communications of the ACM* 38:55–64.

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

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

Lin, C., and Nevatia, R. 1996. Building detection and description from monocular aerial images. In *Proceedings of the Image Understanding Workshop*, 461–468. San Francisco, CA: Morgan Kaufmann.

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

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

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

Michalski, R.; Mozetic, I.; Hong, J.; and 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*, 1041–1045. Menlo Park, CA: AAAI Press.

Nayar, S., and Poggio, T., eds. 1996. *Early visual learning.* New York, NY: Oxford University Press.

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

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

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

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

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

Provost, F., and 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*, 43–48. Menlo Park, CA: AAAI Press.

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

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

Segen, J. 1994. GEST: a learning computer vision system that recognizes hand gestures. In Michalski, R., and Tecuci, G., eds., *Machine Learning: A Multistrategy Approach*, volume 4. San Francisco, CA: Morgan Kaufmann. 621–634.

Sengupta, K., and Boyer, K. 1993. Incremental model base updating: learning new model sites. In *Working Notes of the AAAI Fall Symposium on Machine Learning in Computer Vision*, 1–5. Menlo Park, CA: AAAI Press.

Shepherd, B. 1983. An appraisal of a decision tree approach to image classification. In *IJCAI-83*, 473–475.

Soderland, S., and Lehnert, W. 1994. Corpus-driven knowledge acquisition for discourse analysis. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 827–832.

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

Teller, A., and Veloso, M. 1997. PADO: a new learning architecture for object recognition. In Ikeuchi, K., and Veloso, M., eds., *Symbolic visual learning*. New York, NY: Oxford University Press. 77–112.

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 *Working Notes of the AAAI Fall Symposium on Machine Learning in Computer Vision*, 60–64. Menlo Park, CA: AAAI Press.

Woods, K.; Bowyer, K.; and Kegelmeyer, W. 1996. Combination of multiple classifiers using local accuracy estimates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 391–396. Los Alamitos, CA: IEEE Press.

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