Object Classification in Astronomical Images

Richard L. White

ABSTRACT
Automated classification methods are needed for processing the huge quantities of data generated by modern astronomical instruments. The star-galaxy classification problem and some techniques that have been applied to it are briefly reviewed. Methods for constructing training sets and selecting parameters are described.
A new method of scaling parameter values using ranks has been developed. This approach is found to be of great utility for distinguishing stars and galaxies on digitized photographic plates. It should be widely applicable to other classification problems, especially when the data being classified are not completely homogeneous.

1 Introduction

The combination of ever larger detector formats, powerful computers capable of processing vast amounts of data, and on-line access to large databases has made an ocean of data available to astronomers. Because of the large size of individual databases, and the large number of databases, automated techniques must be used to search a single or combined database for particular classes of interesting or rare objects. Moreover, automated methods are required to construct the databases themselves from the raw data collected at observatories.

In this paper I focus mainly on the problem of star-galaxy separation. For our purposes, the main difference between stars and galaxies is that stars look completely unresolved and so have sharp images, while galaxies look fuzzy. Even though stars are not fuzzy, images taken through telescopes have finite resolution and the stellar images do have non-zero sizes. Figure 1 shows a small section from a digitized photographic plate with galaxies marked. Distinguishing stars from galaxies becomes difficult when both are faint and the galaxies are small; it is also sometimes difficult for very bright objects, when the stars can saturate the detector and no longer look compact. All galaxies are centrally concentrated, and a significant fraction of galaxies have activity near their centers and so appear to have a stellar core surrounded by faint fuzz. Such objects are especially difficult to classify correctly.

Two examples will suffice to demonstrate the scope of the problem. The Sloan Digital Sky Survey, scheduled to begin in 1997, will generate a sur-

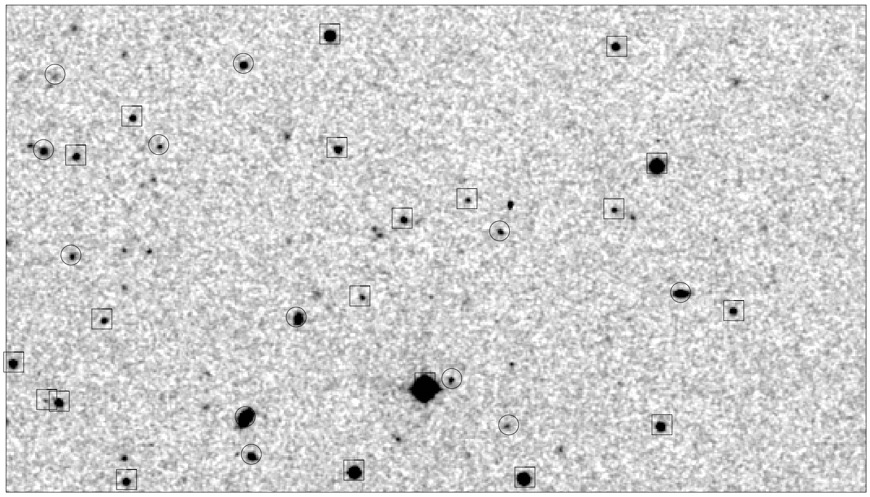

FIGURE 1. 530×300 pixel section of digitized Palomar Sky Survey II plate. Pixels are one arcsecond. The contrast has been enhanced to make visible both faint objects and the noise in the sky. Objects brighter than $V = 20.5$ from Postman's deep CCD images are marked (squares are stars, circles are galaxies.) Our goal is to develop a classification algorithm that can distinguish the stars from the galaxies. There are more than 3 million image patches this size across the entire sky.

vey of 10,000 square degrees (1/4 of the sky) in 5 colors using an array of 30 2048×2048 pixel CCD detectors [GK92]. The ~ 12 terabytes of raw pixel values generated by this survey will be processed to produce a catalog of $> 10^8$ objects with positions, brightness in 5 colors, and classifications as stars or galaxies. The classification will be based on both the objects' morphological parameters (shape, central concentration, etc.) and on their colors. Accurate classification of this vast number of objects down to the detection limit of the survey is a daunting task that will require new classification methods.

The Space Telescope Science Institute constructed a catalog of guide stars for pointing the Hubble Space Telescope [LSM+90]. The original catalog contained about 2×10^7 objects to a brightness of 15th magnitude and was constructed from a digitized version of photographic plates in a single color taken at the Palomar and the UK Schmidt telescopes. A second generation catalog is now under construction [LMJ+95]; it will have nearly one billion objects and will be based on digitized photographic data taken in a variety of colors and at several different epochs, so it will include both color and information on proper motions of objects in the catalog. This project also needs an accurate, automatic star-galaxy classifier.

Although these projects may sound similar, they present rather different classification problems because of the non-linear response of photographic emulsions to light.

The next section (§2) briefly describes some of the common methods used for classification, with some attention to ways to handle noise in the data. The steps required during the development of a classifier for a particular problem are discussed (§3). A new method of constructing more robust features by using ranks is described (§4), with examples from the Guide Star Catalog II star/galaxy discrimination problem. The concluding section (§5) discusses promising areas for future work.

2 Methods for Classification

Any classification method uses a set of *features* or *parameters* to characterize each object, where these features should be relevant to the task at hand. We consider here methods for *supervised* classification, meaning that a human expert both has determined into what classes an object may be categorized and also has provided a set of sample objects with known classes. This set of known objects is called the *training set* because it is used by the classification programs to learn how to classify objects. There are two phases to constructing a classifier. In the training phase, the training set is used to decide how the parameters ought to be weighted and combined in order to separate the various classes of objects. In the application phase, the weights determined in the training set are applied to a set of objects that do *not* have known classes in order to determine what their classes are likely to be.

If a problem has only a few (two or three) important parameters, then classification is usually an easy problem. For example, with two parameters one can often simply make a scatter-plot of the feature values and can determine graphically how to divide the plane into homogeneous regions where the objects are of the same classes. The classification problem becomes very hard, though, when there are many parameters to consider. Not only is the resulting high-dimensional space difficult to visualize, but there are so many different combinations of parameters that techniques based on exhaustive searches of the parameter space rapidly become computationally infeasible. Practical methods for classification always involve a heuristic approach intended to find a "good-enough" solution to the optimization problem.

2.1 Neural Networks

There are a number of standard classification methods in use. Probably neural network methods are most widely known. Odewahn et al. [OSP+92] applied neural network methods to the star-galaxy classification problem on digitized photographic plates. They obtained good results for objects in a limited brightness range. The biggest advantage of neural network methods is that they are general: they can handle problems with very many parameters, and they are able to classify objects well even when the distribution of objects in the N-dimensional parameter space is very complex. The disadvantage of neural networks is that they are notoriously slow, especially in the training phase but also in the application phase. Another significant disadvantage of neural networks is that it is very difficult to determine *how* the net is making its decision. Consequently, it is hard to determine which of the image features being used are important and useful for classification and which are worthless. As I discuss below (§3), the choice of the best features is an important part of developing a good classifier, and neural nets do not give much help in this process.

2.2 Nearest-Neighbor Classifiers

A very simple classifier can be based on a nearest-neighbor approach. In this method, one simply finds in the N-dimensional feature space the closest object from the training set to an object being classified. Since the neighbor is nearby, it is likely to be similar to the object being classified and so is likely to be the same class as that object. Nearest neighbor methods have the advantage that they are easy to implement. They can also give quite good results if the features are chosen carefully (and if they are weighted carefully in the computation of the distance.) There are several serious disadvantages of the nearest-neighbor methods. First, they (like the neural networks) do not simplify the distribution of objects in parameter space to a comprehensible set of parameters. Instead, the training set is retained in its entirety as a description of the object distribution. (There are some thinning methods that can be used on the training set, but the result still does not usually constitute a compact description of the object distribution.) The method is also rather slow if the training set has many examples. The most serious shortcoming of nearest neighbor methods is that they are very sensitive to the presence of irrelevant parameters. Adding a single parameter that has a random value for all objects (so that it does not separate the classes) can cause these methods to fail miserably.

2.3 Decision Trees

Decision tree methods have also been used for star-galaxy classification problems [FWD93, DWF94, Wei94]. In axis-parallel decision tree methods, a binary tree is constructed in which at each node a single parameter is compared to some constant. If the feature value is greater than the threshold, the right branch of the tree is taken; if the value is smaller, the left branch is followed. After a series of these tests, one reaches a leaf node of the tree where all the objects are labeled as belonging to a particular class. These are called axis-parallel trees because they correspond to partitioning

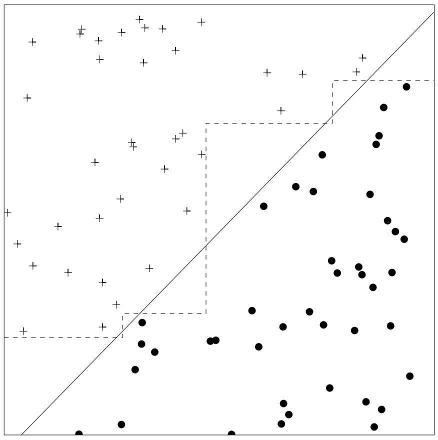

FIGURE 2. Sample classification problem with only two features. *Solid line*: oblique decision tree. *Dashed line*: axis-parallel decision tree. The true dividing line for the simulated data is the square's diagonal. The oblique decision tree is both simpler and more accurate than the axis-parallel tree.

the parameter space with a set of hyperplanes that are parallel to all of the feature axes except for the one being tested.

Axis-parallel decision trees are usually much faster in the construction (training) phase than neural network methods, and they also tend to be faster during the application phase. Their disadvantage is that they are not as flexible at modeling parameter space distributions having complex distributions as either neural networks or nearest neighbor methods. In fact, even simple shapes can cause these methods difficulties. For example, consider a simple 2-parameter, 2-class distribution of points with parameters x, y that are all of type 1 when $x > y$ and are of type 2 when $x < y$ (Fig. 2). To classify these objects with an axis-parallel tree, it is necessary to approximate the straight diagonal line that separates the classes with a series of stair-steps. If the density of points is high, very many steps may be required. Consequently, axis-parallel trees tend to be rather elaborate, with many nodes, for realistic problems.

2.4 Oblique Decision Trees

Oblique decision trees attempt to overcome the disadvantage of axis-parallel trees by allowing the hyperplanes at each node of the tree to have any orientation in parameter space [HKS93, MKS94]. Mathematically, this means that at each node a linear combination of some or all of the parameters is computed (using a set of feature weights specific to that node) and the sum is compared with a constant. The subsequent branching until a leaf node is reached is just like that used for axis-parallel trees.

Oblique decision trees are considerably more difficult to construct than axis-parallel trees because there are so many more possible planes to consider at each tree node. As a result the training process is slower. However, they are still usually much faster to construct than neural networks. They have one major advantage over all the other methods: they often produce very simple structures that use only a few parameters to classify the objects. It is straightforward through examination of an oblique decision tree to determine which parameters were most important in helping to classify the objects and which were not used.

Most of our work has concentrated on oblique decision trees using the freely available software OC1[1] [MKS94], though we have used all the methods mentioned above. We find that oblique decision trees represent a good compromise between the demands of computational efficiency, classification accuracy, and analytical value of the results.

2.5 Adapting Classification Methods to Noise in Data

Noise is common in astronomical data. In the presence of noise, classification becomes a statistical estimation problem. Objects should not be classified as either stars or galaxies, but should be assigned probabilities of being in one or the other class.

All of the above methods can be modified to give a probabilistic estimate of class rather than a simple yes or no answer. Neural nets can have an output parameter that represents probability. It is also possible to provide to a neural network (or the other methods, for that matter) the noise in parameters as additional object features. It is hard to know what the neural network actually does with this information, though — this approach asks the classifier to discover that a particular parameter measures the noise on another parameter. Why should the classifier be burdened with deducing this when we already know what the parameters mean?

The nearest-neighbor method can be generalized to use the K nearest neighbors to an object, which can vote for the object's class. If the vote is not unanimous, it gives an estimate of the uncertainty of the object's majority classification. The same approach can be applied to decision trees. If the search algorithm used to construct the decision tree has a randomized component (which is typical for optimization searches in high dimensional spaces and is in fact the case for OC1), one can construct a number of different decision trees and let them vote on an object's class.

For decision trees, it is also possible to use known noise estimates for the parameters to determine at each node the *probability* that an object belongs on the left branch or the right branch. These probabilities, multiplied down to the leaf nodes and summed over all the leaves belonging

[1]OC1 is available at `http://www.cs.jhu.edu/s̃alzberg/announce-oc1.html` and `ftp://ftp.cs.jhu.edu/pub/oc1`.

to a particular class, give a probability that an object belongs that class. We are just beginning to explore this approach, which has great intuitive appeal and appears to be a promising avenue for further work.

3 Steps in Developing a Classifier

The choice of an algorithm for classification is in many ways the easiest part of developing a scheme for object classification. The discussion above demonstrates that there are several "off-the-shelf" approaches available (though there is obviously still room for improvement.) There are two major hurdles to be faced before these methods can be used, though: a training set must be constructed for which the true classifications of the objects are known, and a set of object parameters must be chosen that are powerful discriminators for classification. Once a possible classifier has been identified, it is necessary to measure its accuracy.

3.1 Training Sets

A training set must contain a list of objects with known classifications. Ideally the training set should contain many examples (typically thousands of objects) so that it includes both common and rare types of objects. Creating a training set requires a source of true object classifications, which is usually difficult even for human experts to generate if it must rely on the same data being used by the classifier.

To construct a training set for the star-galaxy classification problem, the best approach we have found is to use a more sensitive, higher resolution image to get true classifications of the objects seen in the survey data. In some cases (e.g. for the Sloan Digital Sky Survey), it may be quite difficult and time-consuming to acquire images that are significantly better than the survey images; in that case it may be possible to use instead simulated data (where the true classifications are known.) A risk in using simulated images is that they may not include all the effects that make classification difficult from the real data.

3.2 Feature Selection

Adding many irrelevant parameters makes classification harder for all methods, not just the nearest neighbor methods. Training classifiers is an optimization problem in a many-dimensional space. Increasing the dimensionality of the space by adding more parameters makes the optimization harder (and the difficulty grows exponentially with the number of parameters.) It is always better to give the algorithm only the necessary parameters rather than expecting it to learn to ignore the irrelevant parameters.

One should not ask the classifier to rediscover everything you already know about the data. Not only should irrelevant parameters be omitted, but highly correlated parameters should be combined when possible to produce a few powerful features. For example, if you expect the shapes of images of a particular class of object to be similar, include a brightness-independent shape parameter rather than simply giving the classifier raw pixel values and expecting it to figure out how to extract shape information from the pixel values.

If the training process does not require too much computation, a useful approach to identifying the best parameters is to train many times on subsets of the features. We have used this method in two ways, both starting from the complete list of features and reducing it by removing parameters, and starting from a minimal list, augmenting it by adding parameters. Both methods have proven effective at pruning unnecessary parameters [SCF+95]. This procedure can be very fast if axis-parallel trees are used for the exploration. (Note that OC1 can construct either axis-parallel or oblique trees.)

Another useful approach with the decision tree methods is to examine directly the weights assigned to the various features. Important features are given a high weight, while unimportant features may not be used at all. This information can be used as a guide for pruning experiments.

3.3 *Assessing Classifier Accuracy*

Once a potentially useful classifier has been constructed, the accuracy of the classifier must be measured. Knowledge of the accuracy is necessary both in the application of the classifier and also in comparison of different classifiers.

The accuracy can be determined by applying the classifier to an independent training set of objects with known classifications. This is sometimes trickier than it sounds. Since training sets are usually difficult to assemble, one rarely has the resources to construct yet another set of objects with known classifications purely for testing. One must avoid the temptation to train and test on the same set of objects, though. Once an object has been used for training, any test using it is necessarily biased.

We normally use five-fold cross-validation to measure the accuracy of our classifiers. The training set is divided into five randomly selected subsets having roughly equal numbers of objects. The classifier is then trained five times, excluding a single subset each time. The resulting classifier is tested on the excluded subset. Note that each training session must be completely independent of the excluded subset of objects; one cannot, for example, use the results of an earlier training session as a starting point.

The advantage of cross-validation is that all objects in the training set get used both as test objects and as training objects. This ensures that the classifier is tested on both rare and common types of objects. The cost of cross-validation is that the training process must be repeated many times, adding to the computational cost of the training. In most applications, though, the computer time necessary to repeat the training is more readily available than is the human expert time required to generate completely independent test and training sets.

4 Use of Ranks to Make Parameters More Robust

Inhomogeneous data sets can make developing a classifier devilishly difficult. In the Guide Star Catalog II (GSC-II) star/galaxy classification problem, for example, there is a considerable variation in the characteristics of images from different photographic plates. The sky brightness and atmospheric "seeing" (which determines the stellar image size) change with time. The telescope optical characteristics also cause a slow variation in these parameters from the center to the edges of the plates. The atmosphere causes the images of objects photographed directly overhead (at the zenith) to differ from images of the same objects photographed near the horizon. Finally, the photographic plates themselves do not all respond identically when exposed to light. All of these factors combine to make the image features of identical objects appear to change from one plate to another.

Almost all large-scale surveys have similar problems. The Sloan Digital Sky Survey does not suffer plate emulsion variations, but it will have seeing, sky, and zenith-angle variations.

The straightforward solution to this problem is to generate many training sets to cover all the possible variations in object parameters. That is almost never a viable approach, however. For the GSC-II project, we have many thousands of photographic plates. Both the human expense of generating training sets and the computational expense of retraining the classifier on each plate are prohibitive.

A better approach is to find image parameters that are independent of the plate-to-plate variations. This section describes a new approach to generating robust parameters for classification. This approach appears to solve our problems for the GSC-II; it is a very general technique that will be applicable to many other problems as well.

It is not terribly difficult to find image parameters that separate stars and galaxies for a single GSC-II plate. Figure 3(a) shows a scatter-plot of the total density[2] (summed over all pixels in the object) versus the peak density (brightest pixel) for objects from one plate. Stars and galaxies for this training set were identified using the deep CCD catalog of Postman [PLG+96].

[2] Density measures the darkness of the photographic plate and is a non-linear function of the intensity of light to which the plate was exposed.

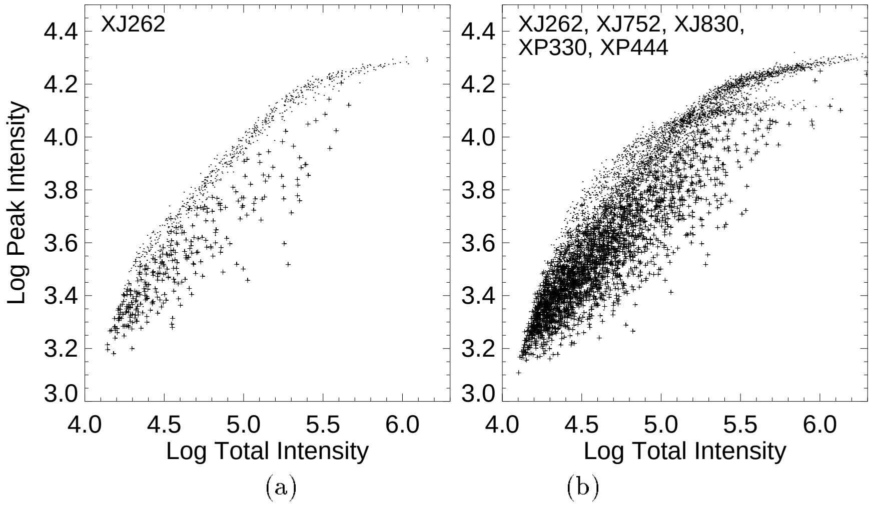

FIGURE 3. Total density versus peak density for stars (dots) and galaxies (plusses) measured on GSC-II photographic plates. (a) The distribution for a single plate (XJ262) is well-defined and could be used for classification. (b) The distribution for five plates differs for different plates and so a classifier for all plates could not be constructed from these parameters.

Stars are (mainly) well-separated from galaxies: stars have sharper images and so have brighter peak density values for a given total density. The distribution is non-linear as a result of the non-linear response of the photographic plate to light, but otherwise a reasonably simple and accurate classifier can be constructed using only these two parameters.

The problem gets very messy when one compares objects from different plates. Figure 3(b) shows the same plot with objects from five different plates. The well-defined distributions for individual plates can be discerned here, but it would be practically impossible to develop an accurate classifier based on these parameters.

We are currently using what appears to be a new approach for scaling these parameters so that they are plate-independent. The idea for this scaling grew from some methods used in robust statistics. A standard "trick" in robust statistics is to use not the value of a parameter but its rank. For example, the Spearman rank-order correlation coefficient can be used to test for a correlation between two variables using not the actual values of the variables but their ranks.

The advantages of ranks in statistical applications are well-known: by using ranks, we are able to construct statistical tests that do not rely on the probability distribution of the variables. The ranks are by definition uniformly distributed. Similarly, using ranks for classification allow us to use parameters such as the total and peak intensities even though their detailed distributions vary from plate to plate.

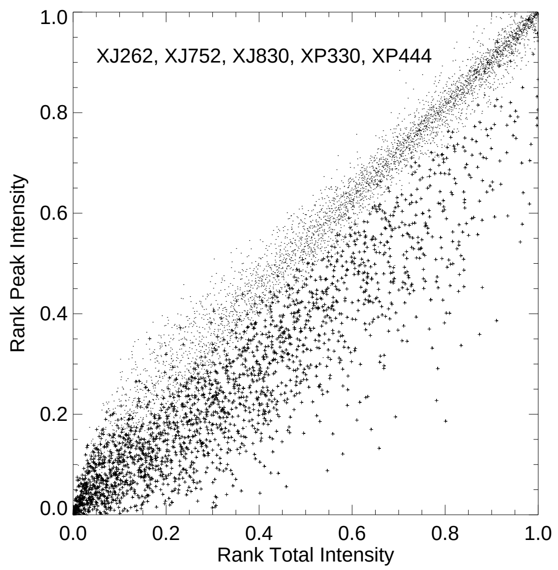

FIGURE 4. Distribution of total density versus peak density for objects from Fig. 2(b) after transformation using ranks. Stars and galaxies from different photographic plates separate very well. An accurate classifier can be constructed based on these rank parameters.

In the current application, we proceed as follows. We compute the "raw" feature values for all objects on a plate (or on a portion of the plate if there are variations within the plate.) We then sort the raw values of each parameter and determine the rank of each raw value within the sorted list. These ranks are scaled to be in the range zero to one. Thus each of the raw features is transformed into a corresponding rank feature.

Note that the ranks are computed separately for objects from each plate. Once converted to ranks, the features for all objects can be combined and a single classifier can be used for objects from different plates.

Figure 4 shows the distribution of the total density rank versus the peak density rank for the same five plates shown in Figure 3(b). Using the rank transformation results in excellent separation between stars and galaxies. Transforming these parameters using their ranks has three significant benefits for our classifier. (1) The distributions become much more similar on different plates. (2) The distribution of objects with feature values becomes uniform, so for example there are equal number of objects in the intervals (0,0.1) and (0.9,1). That makes separating objects in feature space easier. (3) The line separating the stars from galaxies becomes nearly perfectly linear, which makes the decision tree classifiers even more effective than they would be for the untransformed features.

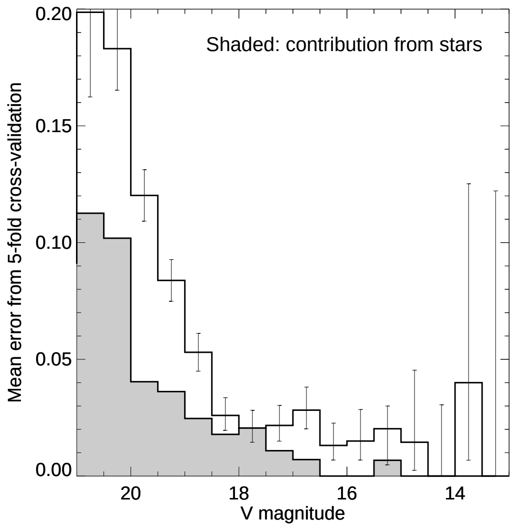

FIGURE 5. Error rate as a function of object brightness for oblique decision tree based on parameters transformed using ranks. All objects in the training set shown in Figures 3 and 4 are included. Excellent accuracy is achieved for brighter objects, and even for faint objects the accuracy is good.

A single decision tree classifier with an accuracy of 91.5% on all five plates can be constructed using only these two parameters. Most of the classification errors are faint, noisy objects. When additional features are included, we are able to construct a classifier with an accuracy of 95–96% for all objects and 98–99% for moderately bright objects (see Fig. 5.) We apply the rank transformation to most of the additional parameters. The complete details of our classifier and the features we are using will be reported elsewhere.

This method was developed only recently, and some details are still being worked out. What is the best way to store the rank transformation so that small object lists (e.g., from small images extracted from the plate) can be classified? How can we best adapt this method to cases where the population of objects changes substantially from one data set to another? For example, in the plane of the Milky Way, the sky images are densely crowded with many stars; galaxies are rare or non-existent. Looking straight up out of the plane of our galaxy, stars are scattered much more sparsely across the sky and a larger fraction of the objects are galaxies. It may be necessary to have independent training sets for the two extremes and to somehow interpolate between them for intermediate regions of the sky.

Despite the uncertainty about how some details should be properly handled, it is clear that this new method is a powerful tool for converting parameters to a more useful form that in most cases leads to simpler and more accurate classifiers.

5 Summary

In this paper I have briefly reviewed methods for object classification, focusing on the problem of distinguishing stars from galaxies. There are several methods available for classification; the oblique decision tree method is well-suited for our problem, but the other methods are also useful and may be better for other problems. Choice of parameters and the construction of high quality training/test data sets are important steps in the problem.

A new method has been described for scaling feature values based on ranks to make the features more robust. This approach should be generally applicable to many classification problems and is especially useful when the data being classified are not completely homogeneous.

There are several interesting avenues for future work. Two topics related to this conference come to mind. First, since the use of ranks (borrowed from statistical methods) looks like a powerful tool for classification, perhaps there are other transformation methods from robust or nonparametric statistics that should be explored for similar applications. Second, most of the classifiers discussed here do not view the classification as a statistical estimation problem. When the object features are noisy, though, classification really is a statistical problem. All the classification methods we have used may be adapted to utilize noise estimates on parameters (§2.5), but there has been relatively little work on this problem to date.

Acknowledgements: This project is an outgrowth of an on-going collaboration with Steven Salzberg, Holland Ford, Rupali Chandar, and Sreerama Murthy on applications of classification methods to astronomical problems. Thanks to all of them for many useful discussions. I am especially grateful to Murthy and Steven for freely distributing the OC1 decision tree program, which I have used extensively in this work. The development of the high quality training sets was done with my GSC-II colleague, Andrea Zacchei, whose participation in the project is supported by the Italian Council for Research in Astronomy. Marc Postman's CCD galaxy catalog has proved invaluable in this effort.

6 References

[DWF94] S. Djorgovski, N. Weir, and U. Fayyad. Processing and analysis of the Palomar-STScI Digital Sky Survey using a novel software technology. In D. R. Crabtree, R. J. Hanisch, and J. Barnes, editors, *Astronomical Data Analysis Software and Systems III*, pages 195–204, San Francisco, 1994. ASP.

[FWD93] U. M. Fayyad, N. Weir, and S. Djorgovski. SKICAT: A machine learning system for automated cataloging of large scale sky surveys. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 112–119, Amherst, MA, 1993. Morgan Kaufmann.

[GK92] J. E. Gunn and G. R. Knapp. The Sloan Digital Sky Survey. *Proceedings of the Astronomical Society of the Pacific*, 43:267–279, 1992.

[HKS93] D. Heath, S. Kasif, and S. Salzberg. Learning oblique decision trees. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1002–1007, Chambery, France, 1993. Morgan Kaufmann.

[LMJ+95] B. M. Lasker, B. J. McLean, H. Jenkner, M. G. Lattanzi, and A. Spagna. Potential application of GSC-II for GAIA operations. In F. van Leeuven and M. Perryman, editors, *Future Possibilities for Astrometry in Space*, ESA SP-379, 1995.

[LSM+90] B. M. Lasker, C. R. Sturch, B. J. McLean, J. L. Russell, H. Jenkner, and M. M. Shara. The Guide Star Catalog. I. Astronomical foundations and image processing. *Astronomical Journal*, 99:2019, 1990.

[MKS94] S. K. Murthy, S. Kasif, and S. Salzberg. Induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–33, 1994.

[OSP+92] S. C. Odewahn, E. B. Stockwell, R. L. Pennington, R. M. Humphreys, and W. A. Zumach. Automated star/galaxy descrimination with neural networks. *Astronomical Journal*, 103:318–331, 1992.

[PLG+96] M. Postman, L. M. Lubin, J. E. Gunn, J. B. Oke, J. G. Hoessel, D. P. Schneider, and J. A. Christensen. The Palomar distant cluster survey: I. The cluster catalog. *Astronomical Journal*, 111:615, 1996.

[SCF+95] S. Salzberg, R. Chandar, H. Ford, S. K. Murthy, and R. White. Decision trees for automated identification of cosmic rays in Hubble Space Telescope images. *Proceedings of the Astronomical Society of the Pacific*, 107:279–288, 1995.

[Wei94] N. Weir. *Automated analysis of the digitized Second Palomar sky survey: system design, implementation, and initial results.* PhD thesis, California Institute of Technology, Pasadena, California, 1994.