Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. http://www.jstatsoft.org/

MATLAB Software for Supervised Classification in Remote Sensing and Image Processing

J. M. Bardsley
Univ. of Montana

Marylesa Wilde
Univ. of Montana

Chris Gotschalk
UC Santa Barbara

M. S. Lorang
Univ. of Montana

Abstract

We present a software package for the supervised classification of images. By supervised, we mean that the user has in hand a representative subset of the pixels in the image of interest. A statistical model is then built from this subset to assign every pixel in the image to a best fit group based on reflectance or spectral similarity. In remote sensing, this approach is typical, and the subset of known pixels is called the ground-truth data.

Ideally, a classifier incorporates both spectral and spatial information. In our software, we implement quadratic discriminant analysis (QDA) for spectral classification and a choice of three spatial methods – mode filtering, probability label relaxation, and Markov random fields – for the incorporation of spatial context after the spectral classification has been computed. Each of these techniques is discussed with some detail in the text.

Finally, we introduce a graphical-user-interface (GUI) that facilitates the creation of ground-truth data subsets – based on individual pixels, lines of pixels, or polygons of pixels – that appear to the user to have spectral similarity. Once a ground-truth subset is created, histogram plots for each band are outputted in order to aid the user in determining whether to accept or reject it. Therefore the GUI makes the software quantitatively robust, broadly applicable and easily usable. We test our classification software on several examples.

Keywords: Markov random fields, probability label relaxation, quadratic discriminant analysis, remote sensing, supervised classification.

1. Introduction

Passive remote sensing is the act of making observations from afar of light reflected from an object. This type of remote sensing data is typically collected by means of a sensor mounted on an aircraft or spacecraft (Richards and Jia 2006), and collected data must be statistically grouped or classified to extract quantitative information about the object.

Supervised classification refers to a class of methods used in the quantitative analysis of remote sensing image data. These methods require that the user provide the set of cover types in the image—e.g., water, cobble, deciduous forest, etc.—as well as a *training field* for each cover type. The training field typically corresponds to an area in the image that contains the cover type, and the collection of all training fields is known as the training set or *ground-truth data*. The ground-truth data is then used to assign each pixel to its most probable cover type.

In this paper, we present MATLAB software for the classification of remotely sensed images. Several well-known software packages exist for classifying remote sensing imagery, e.g. ArcGIS, however, these packages are expensive and tend to be complicated enough that a GIS specialist is required for their use. One of our primary goals in this work is to provide the applied scientist with a few of the most effective, and also cutting edge, classification schemes currently available. The software includes spectral-based classification as well as classifiers that incorporate the spatial correlation within typical images. The software is also unique in that it allows the user to create ground-truth data from a particular image via a graphical user interface (GUI). Finally, the software makes supervised classification (segmentation) techniques available to the broader imaging science community, which uses MATLAB extensively.

To test our software, we consider images of river flood plains, which are among the most dynamic and threatened ecosystems on the planet (Tockner and Stanford 2002). Flood plains are endangered worldwide because most have been permanently inundated by reservoirs which also reduce geomorphic forming flows or are disconnected by dikes and armored by bank stabilization structures (Tockner *et al.* 2010). River ecosystems are in a state of constant change, both on the seasonal and yearly scales, as well as over the course of decades (Stanford *et al.* 2005). Remote sensing of flood plain imagery has become a valuable tool for conserving and protecting ecosystem goods and services that are important to human well being through modeling river dynamics (Lorang *et al.* 2005; Stanford *et al.* 2005). Modeling this change is crucial to river ecologists. In particular, analyzing cover type changes over time allows the river ecologist to address physical and ecological questions at scales relevant to large river ecology and management of those ecosystem components. To perform such an analysis, the collection of remotely sensed images is required, as is the classification of the collected imagery.

A remotely sensed image contains both spectral and spatial information. In our approach, we advocate first computing a spectral-based classification using quadratic discriminant analysis (QDA) (also called maximum likelihood classification (Richards and Jia 2006)). QDA is popular due to its simplicity and compact training process (Pal and Mather 2003). It assumes that the spectral information contained in the pixels within each cover type follows a multivariate normal distribution. This distributional assumption suggests a natural thresholding method, which the user has the choice of implementing.

A vast number of other supervised classification schemes exist Hastie *et al.* (2001), however in (Hand 2006) it is argued that while advances in the performance of newer methods continue to be achieved, it is often the case that the simpler methods, such as QDA, achieve over 90% of the predictive power obtained by the most advanced methods. Moreover, in the comparison studies of algorithms found in (Pal and Mather 2003), it is shown that the QDA classifier excels over artificial neural networks and decision trees in design effort, training time, and classification accuracy; and in (Foody and Arora 1997; Wilkinson 1997) it was shown that although neural/connectionist algorithms make no assumption on statistical distribution, as does QDA, training time can be lengthy and the choice of design architecture is rather

complex.

Nonetheless, a major drawback of QDA is that it ignores the spatial arrangement of the pixels within an image (Tadjudin and Landgrebe 1998). The correlation that exists between neighboring pixels within an image (see, e.g., (Congalton 1988; Spiker and Warner 2007; Yin *et al.* 2008)) should, in many instances, be incorporated into a classification scheme (Richards and Jia 2006; Switzer 1980). The effect of spatial autocorrelation and its relation to spatial resolution in an image has been explored in remote sensing (Chen and Wei 2009; Spiker and Warner 2007; Yin *et al.* 2008), but less has been done to directly incorporate spatial autocorrelation into classification. Methods presented in (Chen and Wei 2009; Richards and Jia 2006) suggest pre-classification filters and neighborhood exploitation methods for dealing with spatial autocorrelation. In this paper, and in our software, we present filtering techniques as well as the methods of probabilistic label relaxation and Markov random fields for spatial-based classification. Each of these techniques may or may not reclassify a given pixel based on information about its neighbors. Finally, in order to test the effectiveness of our classification schemes, we have implemented k-fold cross validation.

This paper is arranged as follows. First, in Section 2, we derive both non-spatial and spatial algorithms for supervised classification and present k-fold cross validation, as well as a technique for thresholding. In Section 3, classification is carried out on a variety of images using a sampling of the methods from Section 2. A discussion and summary of the methods and the results is given in Section 4.

2. Methods

We begin with a presentation our spectral-based classifier, QDA. The spatial-based techniques follow, and then we present a thresholding method, as well as k-fold cross validation for error assessment.

2.1. Spectral-Based Classification: Quadratic Discriminant Analysis

We begin by letting ω_i be the i^{th} of the M spectral classes, or cover types, in the image to be classified. When considering to which spectral class, or cover type, a particular pixel vector $\boldsymbol{x} \in \mathbb{R}^k$ belongs, we denote the conditional probability of $\boldsymbol{x}$ belonging to ω_i by

$$p(\omega_i|\boldsymbol{x}), \qquad i = 1, \ldots, M.$$

We are interested in the spectral class that attains the greatest probability given $\boldsymbol{x}$. Thus classification will yield $\boldsymbol{x} \in \omega_i$ when

$$p(\omega_i|\boldsymbol{x}) > p(\omega_j|\boldsymbol{x}) \qquad \text{for all} \quad j \neq i. \tag{1}$$

It remains to define $p(\omega_i|\boldsymbol{x})$ for $i = 1, \ldots, M$, which can be related to the conditional probability $p(\boldsymbol{x}|\omega_i)$ via Bayes' theorem

$$p(\omega_i|\boldsymbol{x}) \quad = \quad p(\boldsymbol{x}|\omega_i)p(\omega_i)/p(\boldsymbol{x}), \tag{2}$$

where $p(\omega_i)$ is the probability that the i^{th} spectral class occurs in the image; and $p(\boldsymbol{x})$ is the probability density associated with the spectral vector $\boldsymbol{x}$ over all spectral classes. In

Bayes' terminology, $p(\omega_i)$ is referred to as the prior probability and is used to model *a priori* knowledge.

Substituting (1) into (2), the method will classify $\boldsymbol{x}$ as an element of spectral class ω_i if

$$p(\boldsymbol{x}|\omega_i)p(\omega_i) > p(\boldsymbol{x}|\omega_j)p(\omega_j) \qquad \text{for all} \quad j{\neq}i. \tag{3}$$

The probability $p(\boldsymbol{x}|\omega_i)$ can be determined using ground-truth data supplied by the analyst, while $p(\omega_i)$ is determined using prior knowledge of the image. Here we assume a particular vector $\boldsymbol{x}$ has the same probability of being classified into any of the spectral classes. However, if we have reason to believe that a pixel is more likely to be in one class than another, then different priors can be assigned.

To simplify the above equation, we use the discriminant functions $g_i(\boldsymbol{x})$ defined to be the natural logarithm of (3), which preserves the inequality since the logarithm is a monotone increasing function. Thus our method classifies $\boldsymbol{x}$ as an element of ω_i when

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \qquad \text{for all} \quad j{\neq}i. \tag{4}$$

In what follows, we will assume that $\ln p(\omega_i)$ is the same for all i. Thus, it remains to define $p(\boldsymbol{x}|\omega_i)$.

We assume the probability distributions for the classes are multivariate normal, though this is not verifiable. We make this assumption since the properties of the multivariate normal are well-known. Thus we have

$$p(\boldsymbol{x}|\omega_i) = \frac{1}{\sqrt{(2\pi)^N|\Sigma_i|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_i)'\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{m}_i)\right), \tag{5}$$

where $\boldsymbol{m}_i \in \mathbb{R}^k$ and $\Sigma_i \in \mathbb{R}^{k\times k}$ are the mean vector and covariance matrix, respectively, of the data in class ω_i. Then the discriminant function has the form

$$g_i(\boldsymbol{x}) = -\frac{N}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_i)'\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{m}_i) + \ln p(\omega_i).$$

The $-\frac{N}{2}\ln 2\pi$ term appears in all $g_i(\boldsymbol{x})$ and the term $\ln p(\omega_i)$ is assumed to be equal for all i, thus they both can be ignored. Moreover, since we assumed equal prior probabilities for each class, and it is (4) that is of interest, we can equivalently use the discriminant function

$$g_i(\boldsymbol{x}) = -\ln|\Sigma_i| - (\boldsymbol{x}-\boldsymbol{m}_i)'\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{m}_i). \tag{6}$$

From here, a variety of different classifiers can be derived based on assumptions on the discriminant function and the probability model. However, we have found quadratic discriminant analysis (QDA) to be both very effective, computationally efficient, as well as straightforward to implement. QDA is given by (4), (6), with the mean and covariance in (6) approximated using the empirical mean and covariance of the data in ground-truth class i. The effective k-nearest neighbor method (Hastie *et al.* (2001)) was not implemented due to its greater computational burden.

2.2. Spatial-Based Techniques

In QDA, the classification of a particular pixel does not depend on that of it's neighbors. In certain instances, it is clear that neighbor information should be taken into account. For

instance, in an image containing a river, the pixels in the "water class" should be distributed in a spatially uniform manner in the classified image. In order to take into account such prior information, additional techniques are needed.

Four separate approaches were taken for incorporating spatial autocorrelation of the remotely sensed data into classification: pre-classification filtering, probabilistic label relaxation, post-classification filtering, and Markov random Fields. Each of the four are described in detail in the following subsections.

Pre-classification filtering

Pre-classification filtering can be viewed as a denoising technique (low pass filter) for the image to be classified. The filtering is implemented prior to classification and hence does not directly incorporate spatial context into the classification scheme. However, it does impose a degree of homogeneity among neighboring pixels, increasing the probability that adjacent pixels will be classified into the same spectral class.

It requires a neighborhood matrix. In our implementation, we include a first-, second-, and third-order neighborhood:

$$N_1 = \frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad N_2 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad N_3 = \frac{1}{36}\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 1 & 4 & 8 & 4 & 1 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{7}$$

Convolution of an image by one of these neighborhood matrices yields a new image in which each pixel value has been replaced by a weighted average of the neighboring pixels (including itself) of the original image, with the weighting given explicitly by the values in the chosen neighborhood matrix. If the neighborhood is too large comparative to the degree of spatial autocorrelation, the quantitative distinction between cover types could be lost, so the hope is that the neighborhoods used are slight enough to incorporate neighboring information yet not grand enough to lose individuality of pixels.

To incorporate pre-classification filtering into the algorithm, the `conv2` function in MATLAB is used. Each band of the image was convolved separately with the neighborhood matrix and the same size matrix was asked to be returned so the buffer placed on the image was not returned as well.

Post-Classification Filtering

To filter post-classification, a classification labeling must first be performed. The classified map is then filtered with a defined neighborhood to develop a degree of spatial context. This is a similar idea to the pre-classification filtering method, however, the `conv2` function from MATLAB cannot be used in this case due to the categorical response of the classified map; e.g., the spectral classes are not ordered such that a pixel in class 1 is closer in form to a pixel in class 2 than it is to a pixel in class 5. Thus a weighted average of neighboring pixels will not give us information about the central pixel.

Instead we must filter using a different approach (Townsend 1986). First, a window size is chosen to define the neighborhood size – in our implementation, we use a 3×3 window surrounding the central pixel – and each pixel is relabeled by selecting the spectral class that occurs most often in the neighborhood window.

To implement this in MATLAB, a 1-pixel buffer was placed around the entire image by replicating its closest pixel, since it is probable that the boundary pixel's adjacent pixel is of the same spectral class. The `nlfilter` function was then used on the classified map with the `mode` function, and the buffer pixels were removed.

Probabilistic Label Relaxation

While post-classification filtering can be effective, it only uses the classified image to incorporate spatial context. Given that we have the values of the discriminant function (and hence the corresponding probabilities), it is tempting to want to use this information as well. Probability label relaxation (PLR) is a technique that uses this information to incorporate spatial homogeneity in the classification (Richards and Jia 2006).

The PLR algorithm begins with a classification – in our case using QDA – as well as with the corresponding set of probabilities $p_{\boldsymbol{x}}(\omega_i)$ that pixel $\boldsymbol{x}$ belongs to spectral class ω_i. A neighborhood is then defined for $\boldsymbol{x}$ and neighboring pixels are denoted by $\boldsymbol{y}$. The neighborhood should be large enough that all pixels considered to have spatial correlation with pixel $\boldsymbol{x}$ are included (Richards and Jia 2006); in our implementation, we simply use the pixels to the left, right, above, and below.

A neighborhood function $Q_{\boldsymbol{x}}(\omega_i)$ is then defined so that all pixels in the neighborhood have some influence over the probability of pixel $\boldsymbol{x}$ belonging to ω_i. The probabilities defined by $p_{\boldsymbol{x}}(\omega_i)$ are then updated and normalized using $Q_{\boldsymbol{x}}(\omega_i)$. This is done iteratively so that the PLR algorithm has the form

$$p_{\boldsymbol{x}}^{k+1}(\omega_i) = \frac{p_{\boldsymbol{x}}^{k}(\omega_i) Q_{\boldsymbol{x}}^{k}(\omega_i)}{\sum_i p_{\boldsymbol{x}}^{k}(\omega_i) Q_{\boldsymbol{x}}^{k}(\omega_i)}. \tag{8}$$

It remains to define the neighborhood function $Q_{\boldsymbol{x}}$. We begin with the compatibility coefficient defined by the conditional probability $p_{\boldsymbol{xy}}(\omega_i|\omega_j)$, i.e. the probability that ω_i is the correct classification of pixel $\boldsymbol{x}$ given that ω_j is the correct classification for neighboring pixel $\boldsymbol{y}$. We assume that $p_{\boldsymbol{xy}}(\omega_i|\omega_j)$ is independent of location, so that it is only the relative location of $\boldsymbol{y}$ with respect to $\boldsymbol{x}$, and whether or not it is one of its neighbors, that matters. Then, using the previously computed classification, we can compute $p_{\boldsymbol{xy}}(\omega_i|\omega_j)$ by looking at the ratio of occurrences of a pixel $\boldsymbol{x}$ and its neighbors $\boldsymbol{y}$ being classified into ω_i and ω_j, respectively. A location dependent definition for $p_{\boldsymbol{xy}}(\omega_i|\omega_j)$ can also be given, but we do not do that in our implementation.

The probabilities $p_{\boldsymbol{y}}(\omega_j)$, which we have in hand, are then modified via the equation

$$\sum_j p_{\boldsymbol{xy}}(\omega_i|\omega_j) p_{\boldsymbol{y}}(\omega_j). \tag{9}$$

In the approach for computing $p_{\boldsymbol{xy}}(\omega_i|\omega_j)$ mentioned in the previous paragraph, this amounts to simple matrix multiplication. Our neighborhood function is then defined as a spatial average of the values of (9), i.e.

$$Q_{\boldsymbol{x}}(\omega_i) = \sum_{\boldsymbol{y}} d_{\boldsymbol{y}} \sum_j p_{\boldsymbol{xy}}(\omega_i|\omega_j) p_{\boldsymbol{y}}(\omega_j) \tag{10}$$

where $d_{\boldsymbol{y}}$ are the weights for each neighboring pixel $\boldsymbol{y}$. Generally the weights will all be the same – as is the case for us – unless there is prior knowledge about stronger spatial autocorrelation in one direction than another.

It is recommended that algorithm (8) be iterated until the probabilities have stabilized. There is no predetermined number of iterations for this process to terminate, though in practice it has been observed that beyond the first several iterations the relaxation process can begin to deteriorate (Richards *et al.* 1981). Richards mentions that five to ten iterations will typically produce satisfactory results (Richards and Jia 2006).

Markov Random Fields

The neighborhood function used in PLR, though intuitive, can only be justified heuristically. Another approach that incorporates spatial context, and that can be justified mathematically, from a Bayesian viewpoint, is Markov random fields (MRF).

Suppose the image of interest has pixel data $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, where N is the number of pixels. Let ω_{in} denote the i^{th} spectral class of $\boldsymbol{x}_n$. Then the set of class labels for the entire image can be represented as $\Omega = \{\omega_{i1}, \ldots, \omega_{iN}\}$, where i can be one of $i = 1, \ldots, M$. Our task is to compute a random field Ω that well-approximates the "true" class labeling Ω^* of the image under investigation.

We now take a Bayesian point of view and assume that Ω is a random array with (prior) distribution $p(\Omega)$. In this context, Ω is sometimes referred to as a random field. Through classification, we seek the random field Ω that correctly identifies the ground-truth, and thus hopefully correctly identifies the entire image. For this we maximize the posterior probability $p(\Omega|\boldsymbol{X})$, i.e. from Bayes' theorem we compute

$$\widehat{\Omega} = \arg\max_{\Omega}\{p(\boldsymbol{X}|\Omega)p(\Omega)\}.$$

Since spatial autocorrelation typically exists in images, we incorporate it into the posterior probability. Considering a neighborhood $\mathcal{N}$ around a pixel vector $\boldsymbol{x}_n$, we denote the labels on this neighborhood $\omega_{\delta n}$, $\delta \in \mathcal{N}$ and our objective is now to maximize $p(\omega_{in}|\boldsymbol{x}_n, \omega_{\delta n})$. Using probability theory we can manipulate the conditional probability as follows

$$\begin{aligned} p(\omega_{in}|\boldsymbol{x}_n, \omega_{\delta n}) &= p(\boldsymbol{x}_n, \omega_{\delta n}, \omega_{in})/p(\boldsymbol{x}_n, \omega_{\delta n}) \\ &= p(\boldsymbol{x}_n|\omega_{\delta n}, \omega_{in})p(\omega_{\delta n}, \omega_{in})/p(\boldsymbol{x}_n, \omega_{\delta n}) \\ &= p(\boldsymbol{x}_n|\omega_{\delta n}, \omega_{in})p(\omega_{it}|\omega_{\delta n})p(\omega_{\delta n})/p(\boldsymbol{x}_n, \omega_{\delta n}). \end{aligned} \tag{11}$$

To simplify (11), note that $p(\boldsymbol{x}_n|\omega_{\delta n}, \omega_{in})$ is the probability distribution of $\boldsymbol{x}_n$ conditional on both its labeling as the i^{th} class and its given neighborhood labelings $\omega_{\delta n}$ for $\delta \in \mathcal{N}$. We assume that this conditional distribution is independent of the neighborhood labelings, i.e., $p(\boldsymbol{x}_n|\omega_{\delta n}, \omega_{in}) = p(\boldsymbol{x}_n|\omega_{in})$. Then we also have that the measurement vector $\boldsymbol{x}_n$ is independent of its neighborhood labeling, and hence, $p(\boldsymbol{x}_n, \omega_{\delta n}) = p(\boldsymbol{x}_n)p(\omega_{\delta n})$. Substituting these expressions into equation (11) then yields

$$\begin{aligned} p(\omega_{in}|\boldsymbol{x}_n, \omega_{\delta n}) &= p(\boldsymbol{x}_n|\omega_{in})p(\omega_{it}|\omega_{\delta n})p(\omega_{\delta n})/p(\boldsymbol{x}_n)p(\omega_{\delta n}) \\ &= p(\boldsymbol{x}_n|\omega_{in})p(\omega_{it}|\omega_{\delta n})/p(\boldsymbol{x}_n) \\ &\propto p(\boldsymbol{x}_n|\omega_{in})p(\omega_{it}|\omega_{\delta n}). \end{aligned}$$

The conditional prior probability $p(\omega_{in}|\omega_{\delta n})$ is the probability for class i of pixel n given the neighborhood labels $\omega_{\delta n} \in \mathcal{N}$. This conditionality forces our random fields of labels to be

known as Markov Random Fields (MRF). Knowing that we have a MRF gives us the ability to express $p(\omega_{it}|\omega_{\delta n})$ in the form of a Gibbs distribution

$$p(\omega_{it}|\omega_{\delta n}) = \frac{1}{Z}e^{-U(\omega_{in})},$$

where $U(\omega_{in})$ is based on the Ising model

$$U(\omega_{in}) = \sum_{\delta n} \beta[1 - \delta(\omega_{in}, \omega_{\delta n})].$$

The term $\delta(\omega_{in}, \omega_{\delta n})$ is the Kroneker delta, which takes value 1 if the arguments are equal and 0 otherwise. We will take $Z = 1$ as suggested in (Richards and Jia 2006). The parameter $\beta > 0$, fixed by the user, controls the neighborhood effect. We can now take $p(\boldsymbol{x}_n|\omega_{in})$ to be given by (5).

Taking the natural log of $p(\omega_{in}|\boldsymbol{x}_n, \omega_{\delta n})$, as in our derivation of QDA, yields the discriminant function for classification

$$g_{in}(\boldsymbol{x}_n) = -\frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{m}_i)'\Sigma_i^{-1}(\boldsymbol{x}_n - \boldsymbol{m}_i) - \sum_{\delta n} \beta[1 - \delta(\omega_{in}, \omega_{\delta n})]. \tag{12}$$

Here β is sometimes referred to as a regularization parameter, and it must be specified by the user.

Notice that before the Kroneker delta can be computed in the above MRF discriminant function, an initial classification must be computed. We do this using the QDA classifier. Thus the MRF discriminant function modifies a given set of labels based on a neighborhood effect. This process should be iterated until there are no further changes in labeling (Richards and Jia 2006); 5-10 iterations is typically sufficient.

2.3. Error Assessment

Thresholding

The QDA method can only classify a pixel into one of the originally given spectral classes. The question arises, what happens if the ground-truth data is incomplete, i.e., there are missing classes? In such cases, it is useful to have an additional spectral class designed for pixels that, with some degree of certainty, do not fit into one of the original spectral classes noted by the user. The technique of thresholding addresses this problem.

Thresholding compares the probability value of $p(\boldsymbol{x}|\omega_i)$ to a user specified threshold. If the probability value is lower than the threshold, the pixel will be assigned to a *discard* class. Thus, thresholding extracts the pixels with a low probability of belonging to any of the training classes and places them into the *discard class*.

With thresholding, our classification schema now returns $\boldsymbol{x}$ as an element of ω_i when

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \forall\, j{\neq}i \quad \text{and} \quad g_i(\boldsymbol{x}) > T_i, \tag{13}$$

where T_i is the threshold associated with spectral class ω_i. It is only when the above does not hold for any i that $\boldsymbol{x}$ is then classified into the discard class.

To determine T_i, we examine the discriminant function; for QDA, it is given by (6) and the second inequality in (13) becomes

$$(\boldsymbol{x} - \boldsymbol{m}_i)'\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{m}_i) < -T_i - \ln|\Sigma_i|. \tag{14}$$

Assuming $\boldsymbol{x}$ is normally distributed with mean $\boldsymbol{m}_i$ and covariance $\boldsymbol{\Sigma}_i$, the left hand side of (14) is known to have a χ^2 distribution with k degrees of freedom (recall, $\boldsymbol{x} \in \mathbb{R}^k$)) (Anderson 1984). The corresponding χ^2_k value can be found using MATLAB's `chi2inv` function for the desired confidence, yielding a corresponding value for T_i.

K-fold Cross Validation

Once a classification has been made, we are interested in how accurately it performed. Since the image in its entirety is unknown, we look to the ground-truth data. First, the ground-truth data set is partitioned into K approximately equal-sized parts. Remove the first part and set it aside to be used as the testing set while the remaining $K - 1$ sets become the adjusted training set. The classifier is created with the adjusted training set and is then used to classify the testing set. For example, if $K = 10$, this is equivalent to using 90% of the ground-truth to classify the remaining 10% of the ground-truth. The classified set is then compared to the ground-truth pixels, respectively, and the percent of misclassified pixels is noted. This process is repeated K times (folds) so that each of the K sets is used exactly once as a testing set. The average percent of misclassified pixels is reported.

The recommended K for K-fold cross validation (CV) is $K = 5$ or 10 (Hastie *et al.* 2001). If instead, K is taken to be the total number of ground-truth pixels, N, this process is known as the leave-one-out cross validation method. This is approximately unbiased, however it may result in high variance due to the similarities between the N training sets and can be computationally costly. With $K = 5$ or 10, the variance is lower, but bias may play a role in the error assessment. If the classifier is highly affected by the size of the training set, it will overestimate the true prediction error (Hastie *et al.* 2001).

3. Numerical Experiments

We test our algorithms on a variety of images. Since ground classification in remote sensing is the application that motivated this work, we begin with these types of images.

3.1. Multi-Banded Imagery

The classification methods were tested on two different flood plain images from Quickbird. The first image is of the Middle Fork of the Flathead River in the Nyack flood plain bordering Glacier National Park, Montana (left image in Figure 1). It is an 8-bit RBG-infrared (4 banded) image of $2,208,129$ pixels. The 8-bit training data set consisted of 1540 pixels. The location of the training data can be seen in the right-hand image in Figure 1. Training data for the Nyack flood plain consists of the following common cover types that could be easily identified by non-expert users: pasture land, willows, cottonwoods, conifers, water, cobble, regenerative riparian trees, deciduous forest, and road.

To classify this image, we prefilter using neighborhood matrix N_1 and implemented QDA both with and without post-classification smoothing. The straight spectral classification is given

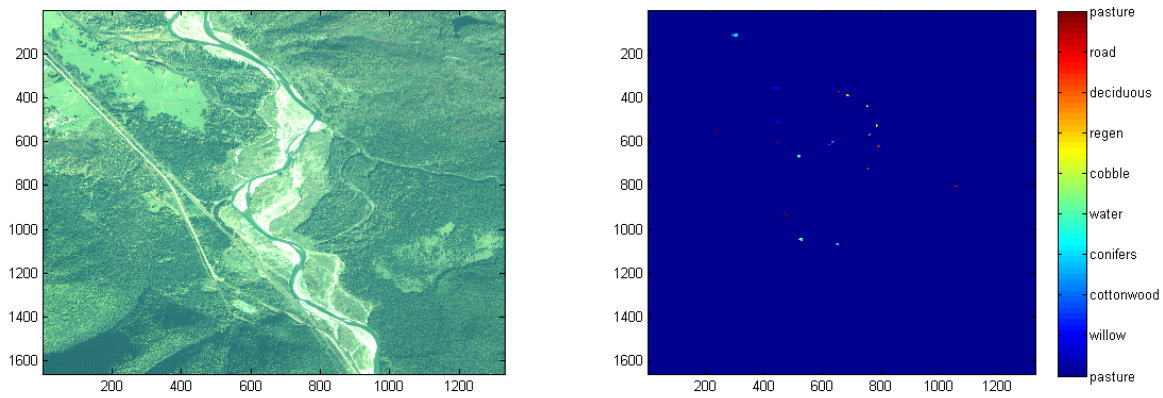

Figure 1: On the left is an image, used in this numerical experiment, of the Nyack flood plain located on the Middle Fork of the Flathead River, Montana. On the right, the polygons denote the location of the training data set, taken from the image on the left.

on the left in Figure 2; it took approximately 9.5 seconds in our MATLAB implementation. The 10-fold cross validation error was approximately 3%.

Implementing 5 iterations of the Markov random field (MRF) took approximately 49 seconds, including the QDA classification step. The β parameter in (12) was taken to be 10. Eight iterations of probability label relaxation (PLR) took approximately 55 seconds. Note the increased spatial homogeneity obtained when PLR, and more noticeably when MRF, are used. We emphasize that this may not always be desirable in practice, but if the user knows, prior to classification, that spatial homogeneity exists, this can be enforced using PLR or MRFs in an efficient manner.

We also test the software on an image of the Kwethluk flood plain – an RBG image containing 132126 pixels – given on the left in Figure 3, and on the right is the ground-truth data, which was hand chosen using a graphical user interface (GUI), which will be introduced in detail below. We note that the chosen ground-truth is not necessarily optimal in the sense of yielding the most accurate classification, but was chosen to test the software.

To classify the image, we implement QDA both with and without post-classification smoothing. The straight spectral classification is given on the left in Figure 4; it took approximately 0.32 seconds in our MATLAB implementation. The 10-fold cross validation error was approximately 4.6%. Implementing QDA plus 8 iteration of PLR took approximately 1 second. QDA plus 8 MRF iterations with $\beta = 10$ took approximiately 1.8 seconds.

3.2. Thresholding

To use thresholding on the QDA classifier, we must first calculate T_i. From equation (14), since the Nyack data set is four-banded, we know the left hand side has a χ^2 distribution with 4 degrees of freedom. For a 99% acceptance rate, $T_i - \ln|\Sigma_i| = 13.2767$ is the value left of which lies 99% of the area under the χ^2 curve with 4 degrees of freedom. Therefore, $T_i = -13.2767 - \ln|\Sigma_i|$, and this is used in equation (14) for classification.

Because the multi-variate Gaussian assumption can be inaccurate for the spectral information

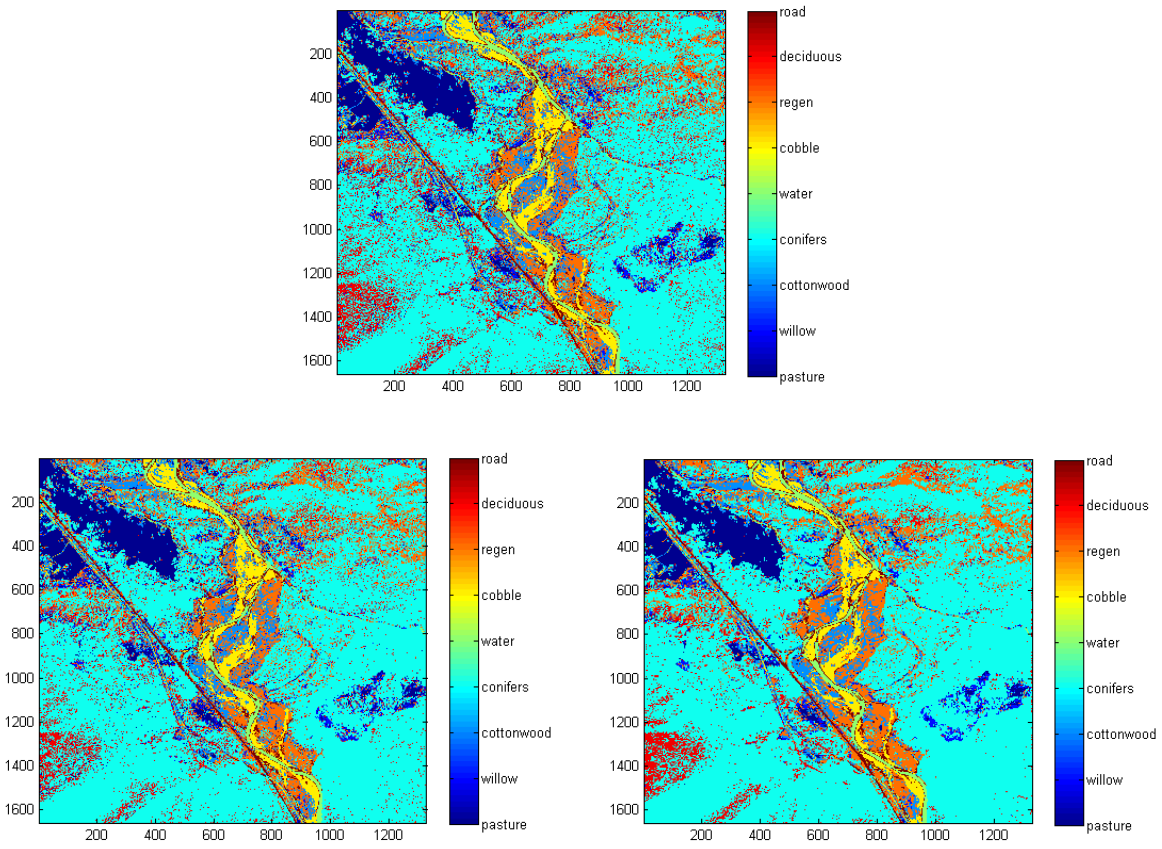

Figure 2: Above is the QDA classification. The lower figures are the classifications obtained using probability label relaxation (left) and Markov random fields (right), respectively, applied to the QDA classification.

corresponding to certain cover types, thresholding is not effective in all instances. However, for certain types of problems it is useful. For example, suppose that we want to extract the water from the Nyack flood plain image in Figure 1. In this case, it seems unnecessary to ground-truth every cover type appearing in the given image, since we are not interested in any cover type other than water. Thus using only water as ground-truth input, the technique of thresholding at the 99% level was combined with QDA to produce Figure 5. The dark blue on the left image represents the water that has been extracted. Ground-truth for the water extraction (light blue polygons) is displayed in the right image. We created the ground-truth using the GUI discussed below. The 10-fold CV error was 0% indicating all of the supplied ground-truth was classified as water and not discarded.

3.3. Classifying General Images

More generally, the software can be used to segment any RGB image – an important problem in image processing – very efficiently. As an example, consider the color image in Figure 6, the ground-truth image obtained using the ground-truth GUI, and the resulting QDA classification that cannot be improved upon, visually, using a spatial technique such as MRF

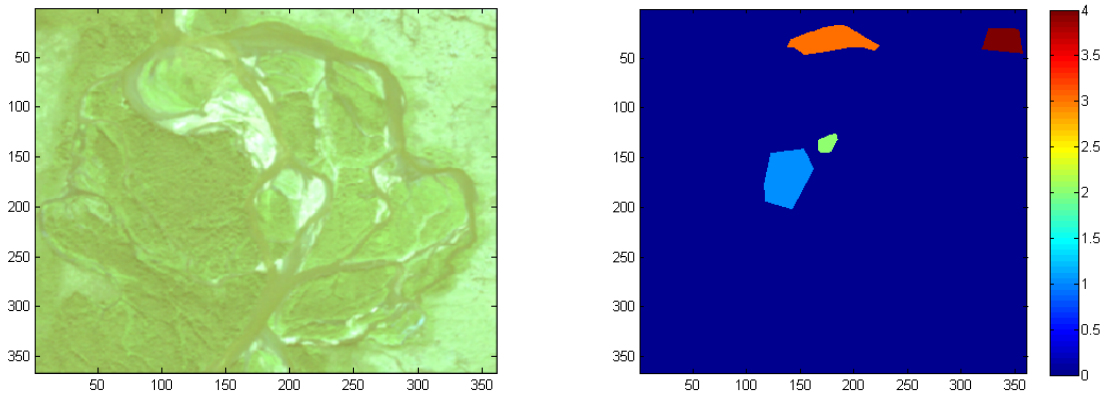

Figure 3: On the left, is an image of a flood plain on the Kwethluk River, Alaska. On the right, the polygons denote the location of the training data set, taken from the image on the left; 1 denotes conifer trees, 2 denotes cobble, 3 denotes water, and 4 denotes deciduous trees.

or PLR. This classification took approximately 2.5 seconds.

Finally, the software can be used on gray-scale and binary images as well for denoising, which is an important problem in image processing. For our gray-scale example, we use the noisy gray-scale image of Mickey Mouse on the left in Figure 7. For this example, no ground-truth data was used, which is allowed in the codes provided the mean vectors and the covariance matrices (both scalars in this case) are provided. For our classification, we chose the five means 1, 90, 188, 200, and 255, and the variance values were taken to be 1 for every class. The denoising is then accomplished using 10 iteration of the MRF algorithm with $\beta = 7000$, which took approximately 1 second. Implementing the mode filtering, on the other hand, yielded similar results but at approximately 15 seconds, a significant increase in computational time.

Our binary example is given in Figure 8. In this case, the means were taken to be 1 and 0, and the variances were both taken to be 1. Ten MRF iterations with $\beta = 1$ yielded the denoised image in the middle in Figure 8 and took approximately 2 seconds. On the right in Figure 8 is the denoised image obtained using mode filtering, which took approximately 55 seconds to compute.

3.4. Using the Software and Ground-truth GUI

First, download the zip file, unzip it and open MATLAB in the resulting directory. Type "startup" at the MATLAB prompt, which adds the directories CLASSIFY&TESTS, GUI, and IMAGES to the MATLAB path. The examples above can be reproduced by typing `TestNyack`, `TestKweth`, `TestHotAirBalloon`, `TestMickey`, or `TestZebra` at the MATLAB prompt and hitting enter. These m-files can be found in the CLASSIFY&TESTS directory and can be used as a template for the implementation of the code on other problems.

In the files mentioned above, ground-truth images were used that were created by the authors using the GUI. We now discuss, in detail, how to use the GUI to create your own ground-truth TIFF files. The GUI, coupled with the classification codes, makes the segmentation of images

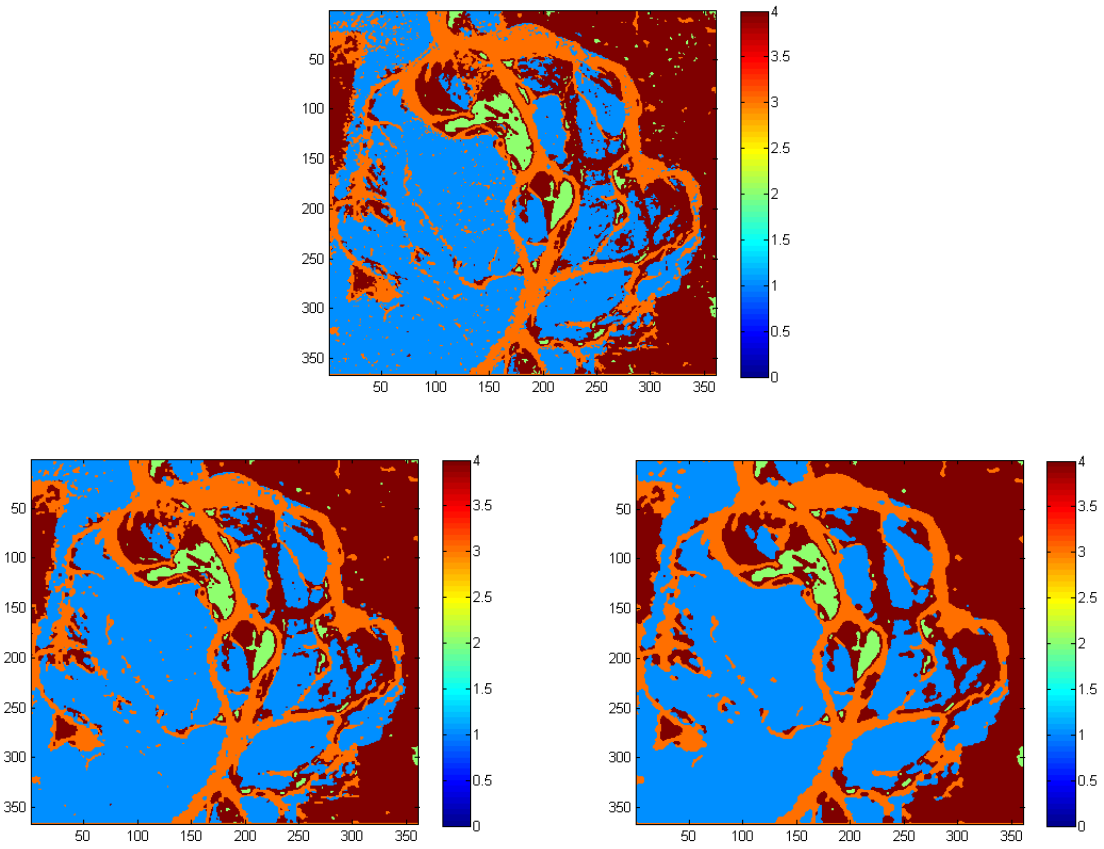

Figure 4: Above is the QDA classification. The lower figures are the classifications obtained using probability label relaxation (left) and Markov random fields (right), respectively, applied to the QDA classification.

extremely simple and efficient.

To create ground-truth using the GUI, after following the directions in the first two sentences of this section, type RUN_GUI at the MATLAB prompt and hit enter. The GUI will open in a separate box. First, click the "Load a raw tiff" button and choose the image to be classified; it will open as Figure 1.

Next, either choose a cover type from the drop down menu labeled "Pick from cover type list" or create a new cover type by typing it into the box labeled "Add new cover type to list" and then pressing the ENTER key. If the latter was performed, the newly created cover type is now available in the drop down menu "Pick from cover type list". Once a cover type is selected, a corresponding color needs to be chosen. Color choices that contrast with the background image allow the user to more easily locate small cover type regions.

Ground truth pixels that represent a cover type (e.g. groups of similar trees) are isolated by enclosing within a polygon or by intersecting with a line or point. If the image requires being zoomed in to view ground-truth pixels for selection, this adjustment needs to be completed before selecting the preferred button for delineating ground-truth. Once the image is at the necessary resolution, select the button for method of ground-truthing: "polygon", "line" or

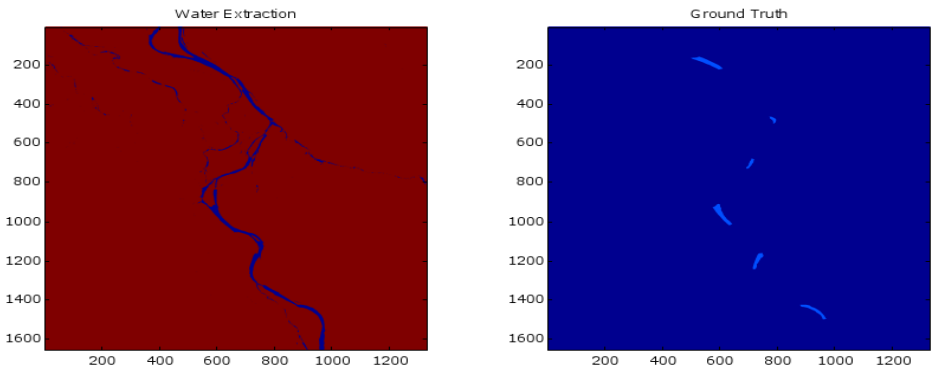

Figure 5: Left: River (blue) has been extracted using 99% threshold under QDA. Right: Light blue polygons denote location of water ground-truth from original image

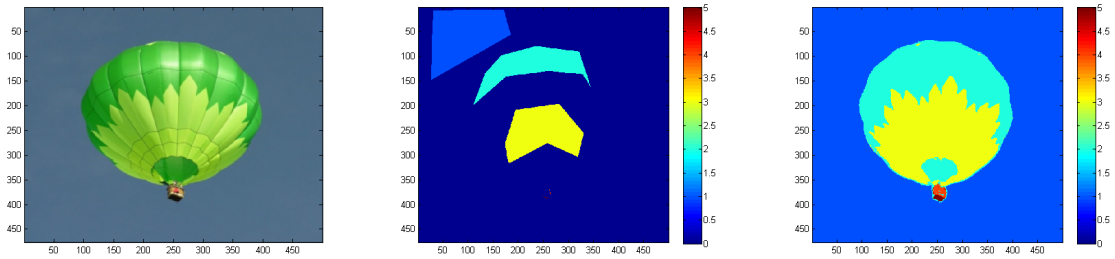

Figure 6: On the left is the original RGB image. In the middle is the ground-truth obtained using the GUI; class 4 and 5, corresponding to the basket, are hardly visible. On the right is the classification obtained using QDA.

"points". This brings up cross-hairs on the image. A left click selects vertices to initiate the ground-truth selection process and a right click ends it. Once the selection is ended, use a left click to edit vertices or pixels. A double click ends the session for that cover type.

Two new figures will open. Figure 2 is a repeated image of Figure 1, though Figure 1 now contains delineation of the recently acquired ground-truth. Figure 3 contains a frequency distribution of the recently acquired ground-truth for each band of data. The ground-truth must be accepted or redone using the corresponding buttons on the GUI. If "Redo" is chosen, the most recently acquired ground-truth is rejected. If "Accept" is chosen, the user has the choice of generating more ground-truth – either for the same or a different cover type – or of finishing the session by selecting "Save/Exit". A snapshot of the GUI can be seen in Figure 9.

Once the "Save/Exit" button is clicked, the ground-truth is saved in the same directory as the original in a *.mat file beginning with the name of the image used, then "_GTOutput_", followed by the date and time. To save the ground-truth as an 8-bit *.tif file, first load it into MATLAB with the command "load", followed by the name of the *.mat file sans *.mat. The command "imwrite(output.groundTruth.matrix,'groundtruth.tif','tiff')" writes the ground-truth to the file groundtruth.tif. To call the corresponding classes, use the command "output.groundTruth.name".

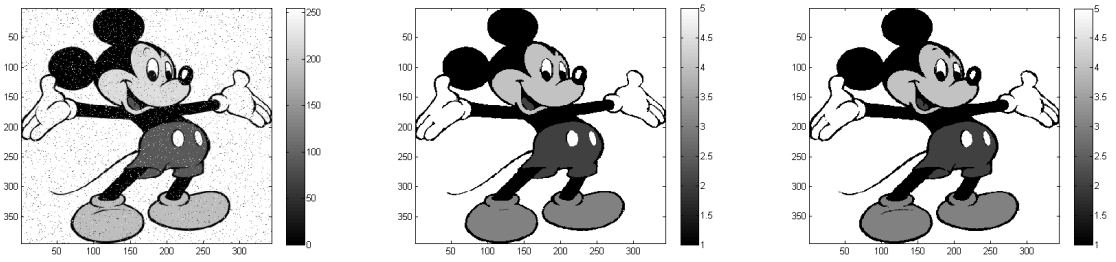

Figure 7: On the left is the noisy image of Mickey. The denoised images were obtained using MRFs (middle) and mode filtering (right).

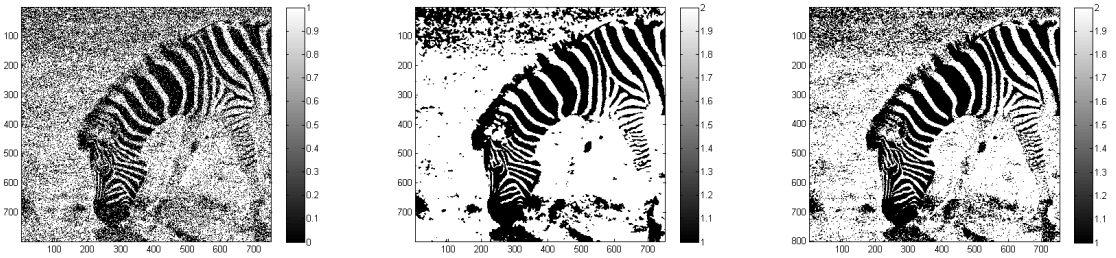

Figure 8: On the left is the noisy binary image of a Zebra. The denoised images were obtained using MRFs (middle) and mode filtering (right).

4. Conclusions

In this paper, we outline a MATLAB software package for the supervised classification of images. The software uses quadratic discriminant analysis (QDA) for spectral classification and contains three methods for spatial smoothing after a spectral classification has been computed. These spatial methods include mode filtering, probability label relaxation, and Markov random fields. Also included is a thresholding technique based on the multi-variate Gaussian assumption used in the motivation of QDA, as well as k-fold cross validation for error analysis.

The software contains a graphical user interface (GUI) that enables the user to generate subsets, known as ground-truth data, for building statistical classifiers. Current versions of MATLAB do not contain such software, hence the use of the ground-truth GUI is essential in allowing our classification software to be widely and easily useable.

The software is tested on a number of images – two examples from remote sensing and three generic images – in order to illustrate its usefulness on a broad range of image processing applications. The examples also show that the software is useful and efficient on large-scale problems.

5. Acknowledgements

Support for PhD student M. Wilde came from an NSF grant to JMB and MSL, *Mathematical methods for habitat classification of remote sensing imagery from river flood plains*, NSF-

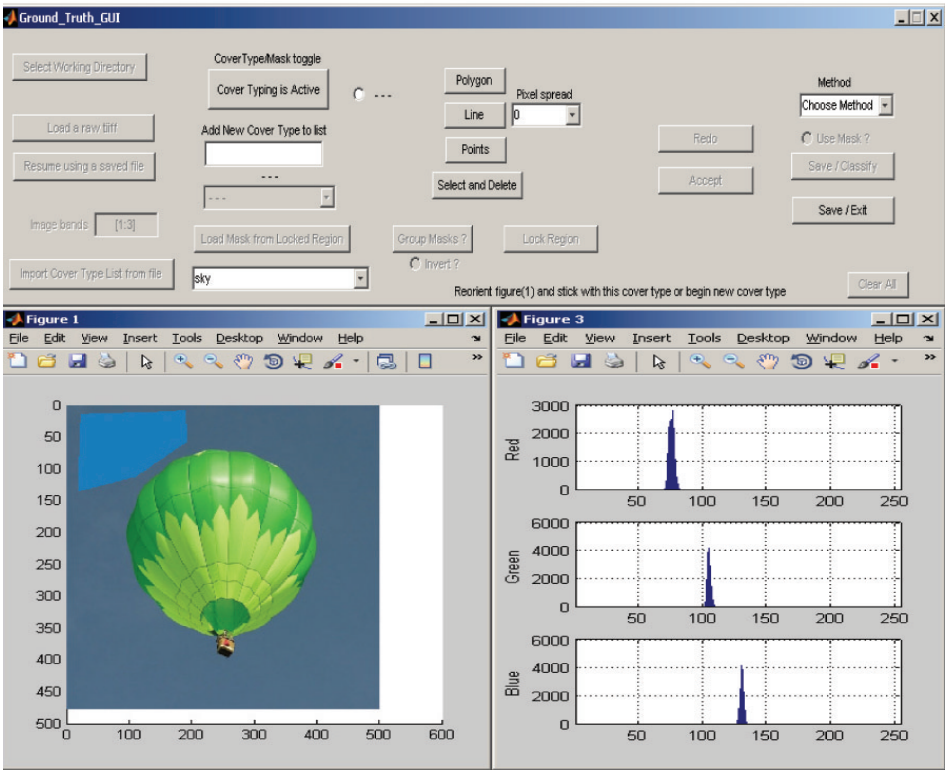

Figure 9: A snapshot of the GUI. On the lower-right are the histograms for each band of the spectral values for the chosen pixels.

EPSCoR Large River Ecosystems Grant EPS-0701906. Partial support for JMB and MSL came from the Gordon and Betty Moore Foundation (UM grant #344.01) The Salmonid Rivers Observatory Network: Relating Habitat and Quality to Salmon Productivity for Pacific Rim Rivers, Stanford PI, Hauer, Kimball, Lorang, Poole Co-PI's, and support for C. Gotschalk to write the GUI came from a grant to MSL (UM grant PPL#443216) from Pennslyviana Power and Light.

References

Anderson TW (1984). *An Introduction to Multivariate Statistical Analysis*. Wiley, New York.

Chen DM, Wei H (2009). "The Effect of Spatial Autocorrelation and Class Proportion on the Accuracy Measures from Different Sampling Designs." *ISPRS Journal of Photogrammetry and Remote Sensing*, **64**, 140–150.

Congalton TG (1988). "Using Spatial Autocorrelation Analysis to Explore the Errors in Maps Generated from Remotely Sensed Data." *Photogrammetric Engineering and Remote Sensing*, **54**(5), 587–592.

Foody GM, Arora MK (1997). "An Evaluation of Some Factors Affecting the Accuracy of

Classification by an Artificial Neural Network." *International Journal of Remote Sensing*, **18**, 799–810.

Hand DJ (2006). "Classifier Technology and the Illusion of Progress." *Statistical Science*, **21**(1), 1–14.

Hastie T, Tibshirani R, Friedman J (2001). "The Elements of Statistical Learning." pp. 214–216.

Lorang MS, Whited DC, Hauer FR, Kimball JS, Stanford JA (2005). "Using airborne multispectral imagery to evaluate geomorphic work across floodplains of gravel-bed rivers." *Ecological Applications*, **15**, 1209–1222.

Pal M, Mather P (2003). "An Assessment of the Effectiveness of Decision Tree Methods for Land Cover Classification." *Remote Sensing of the Environment*, **86**, 554–565.

Richards J, Jia X (2006). *Remote Sensing Digital Analysis, an Introduction*. Springer-Verlag, Berlin.

Richards J, Landgrebe D, Swain P (1981). "On the Accuracy of Pixel Relaxation Labelling." *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-6**, 420–433.

Spiker JS, Warner TA (2007). "Scale and Spatial Autocorrelation From a Remote Sensing Perspective." In *Geo-Spatial Technologies in Urban Environments*, chapter 10, pp. 197–213. Springer, Berlin/Heidelberg, Germany.

Stanford JA, Lorang MS, Hauer FR (2005). "The Shifting Habitat Mosaic of River Ecosystems." *Verh. Internat. Verein. Limnol.*, **29**, 123–136.

Switzer P (1980). "Extensions of Linear Discriminant Analysis for Statistical Classification of Remotely Sensed Satellite Imagery." *Mathematical Geology*, **12**(4), 367–376.

Tadjudin S, Landgrebe D (1998). "Classification of high dimensional data with limited training samples." *Technical Report 98-08*, Electrical and Computer Engineering Department, Purdue University.

Tockner K, Lorang M, Stanford J (2010). "River flood plains are model ecosystems to test general hydrogeomorphic and ecological concepts." *River Research and Applications*, **26**, 76–86.

Tockner K, Stanford JA (2002). "Riverine floodplains: present state and future trends." *Environmental Conservation*, **29**, 308–330.

Townsend F (1986). "The Enhancement of Computer Classifications by Logical Smoothing." *Photogrammetric Engineering and Remote Sensing*, **52**, 213–221.

Wilkinson GG (1997). "Open Questions in Neuro-Computing for Earth Observation." In *Neuro-Computational in Remote Sensing Data Analysis*, pp. 3–13. Springer-Verlag, Berlin.

Yin S, Chen X, Yu Z, Sun Y, Cheng Y (2008). "Scale Dependence of Autocorrelation from a Remote Sensing Perspective." *Proceedings of SPIE, Geoinformatics 2008 and Joint Conference on GIS and Built Environment: Advanced Spatial Data Models and Analyses*, **7146**, 71461T–1 to 71461T–9.

Affiliation:
Johnathan M. Bardsley
Associate Professor
Department of Mathematical Sciences
University of Montana
Missoula, MT, 59812
E-mail: johnathan.bardsley@umontana.edu
Supported by grant NSF-DMS 0915107.

Marylesa Wilde
NSF-EPSCoR PhD Fellow
Department of Mathematical Sciences
University of Montana
Missoula, MT, 59812, USA
E-mail: marylesa.wilde@umontana.edu

Chris Gotschalk
Research Analyst
Marine Science Institute
University of Calafornia
Santa Barbara, CA, 93106
E-mail: gots@lifesci.ucsb.edu

Mark Lorang
Associate Research Professor
Flathead Lake Biological Station
University of Montana
32125 Bio Station Lane
Polson, MT 59860-9659
E-mail: mark.lorang@umontana.edu