LA-UR -92-1262

>141--4

LA-UR--92-1262

DE92 013539

TITLE:

A ONE-CLASS CLASSIFIER FOR IDENTIFYING URBAN AREAS IN REMOTELY-SENSED DATA

AUTHOR(S): Patrick M. Kelly, Don R. Hush, and James M. White

SUBMITTED TO: ISE Conference
Albuquerque, New Mexico
May 13, 1992

A ONE-CLASS CLASSIFIER FOR IDENTIFYING URBAN AREAS IN REMOTELY-SENSED DATA

Patrick M. Kelly, Don R. Hush
University of New Mexico
Albuquerque, New Mexico

James M. White
Los Alamos National Laboratory
Los Alamos, New Mexico

Abstract

For many remote sensing applications, land cover can be determined by using spectral information alone. Identifying urban areas, however, requires the use of texture information since these areas are not generally characterized by a unique spectral signature. We have designed a one-class classifier to discriminate between urban and non-urban data. The advantage to using our c'assification technique is that principles of both statistical and adaptive pattern recognition are used simultaneously. This prevents new data that is completely dissimilar from the training data from being incorrectly classified. At the same time it allows decision boundary adaptation to reduce classification error in overlap areas of the feature space. Results will be illustrated using a LANDSAT scene of the city of Albuquerque.

1 Introduction

Landsat Thematic Mapper (TM) data consists of seven different bands of spectral information. In many remote sensing applications, land cover can be determined by the absorption and reflection of different wavelengths of light. Each pixel for a scene, then, can be classified solely with respect to the spectral values associated with that pixel. A' hough this method works well for identifying many different geographic areas (e.g. deciduous forests, coniferous forests, bodies of water, agricultural fields, etc.), it does not work well when attempting to locate urban areas.

Urban areas do not, in general, have a unique spectral signature. A single pixel in a Landsat scene covers an area of 28 5 square meters. Pixels located in heavily populated downtown areas, therefore, w... contain much different information than pixels located in residential areas. Another problem is that a pixel representing a residential backyard may look the same as a pixel representing a forested area far from any city. For these reasons, spectral information is insufficient to allow discrimination between urban and non-urban data.

	CHANNEL	WAVELENGTH (μm)
1	Visible Blue	0.45-0.52
2	Visible Green	0.52-0.60
3	Visible Red	0.63-0.69
4	Near Infrared	0.76-0.90
5	Mid Infrared	1.55-1.75
6	Thermal Infrared	10.4-12.5
7	Mid Infrared	2.08-2.35

Table 1: Spectral Bands for TM Data

2 Feature Selection

For detecting urban areas, we used textural features instead of spectral features. Textural features are important for this problem, because urban areas look "busy" compared to other areas in a Landsat scene. The features selected for this problem were texture energy measures developed by Laws [1, 2]. They have the advantage of being able to allow discrimination between different types of textures, while being quick and easy to compute.

The texture features were calculated from the TM Band 4 image (near infrared) in 3 steps. Step one convolved the image with a number of convolution kernels. The second step performed an absolute windowing operation on the convolved images; each pixel value was replaced by the sum of the absolute values of the pixel values in a square neighborhood surrounding it:

$$I_{new}(x,y) = \sum_{i=x-N}^{x+N} \sum_{j=y-N}^{y+N} |I_{old}(i,j)| \qquad (1)$$

Finally, related features were added together to provide features invariant to rotation.

L5	=	[	1	4	6	4	1	]
E5	=	[	-1	-2	0	2	1	]
S5	=	[	-1	0	2	0	-1	]
W5	=	[	-1	2	0	-2	1	]
R5	=	[	1	-4	6	-4	1	]

Table 2: Center-Weighted Vectors

Table 2 lists the 5 one-dimensional center-weighted convolution masks which are used to create the 25 two-dimensional 5-by-5 convolution masks. These 5-by-5 masks are created by convolving a horizontal mask with a vertical mask. For instance, an E5L5 mask is formed by convolving a vertical E5 mask with a horizontal L5 mask.

After convolving the base image with one of the 5-by-5 convolution kernels, the associated Texture Energy Measure (TEM) for each pixel is calculated by summing the absolute pixel values of the convolved image within a 15x15 pixel window. A total of 25 TEM images were calculated during this stage of image processing. This set was then reduced by combining related TEM images, such as the L5E5 and E5L5 images, the S5R5 and R5S5 images, etc. All images were divided by the L5L5 image to normalize features for contrast, after which the L5L5 image was discarded. The result was a set of 14 images, each representing some texture feature for the image. Each pixel in the image is now represented by a vector of 14 features.

3 One-Class Classification

When training a pattern classification system to discriminate between different classes of data, it is desirable to use training data representative of all input which is expected to be classified by the system. Problems exist, however, where representative training sets are not easily obtainable. In cases such as these, it may be desirable for systems to classify patterns only when they are similar to the training data, and reject dissimilar data. Otherwise, unfamiliar data may be poorly categorized. Pattern rejection techniques of this kind are important for many applications, especially those where misclassifications are very costly.

The idea of rejecting patterns can also be viewed as part of the one class classifier problem. This problem and some approaches to solving it are discussed in [3]. Consider the problem where patterns from a single class of data need to be detected. Training data from other classes may or may not be available when the classifier is to be designed. A one-class classifier will attempt to define a "closed decision boundary" in the feature space X around the in-class data. New data is then classified as in-class or out-of-class data depending on its location with respect to the decision boundary.

In this paper, we use a one-class classifier to discriminate between urban and non-urban data. This is achieved by first creating a decision boundary that surrounds the areas in X containing urban data. Next, in finding an optimal decision boundary, all available training data for both urban and non-urban data are used by an adaptive algorithm to minimize classification error.

4 Initialization

Our classifier works as follows. Urban data is represented by a set of hyperellipsoidal clusters in the feature space. Data falling inside at least one hyperellipsoid is considered in-class data (urban). A hyperellipsoid is defined by a mean vector, $\underline{\mu}$, a covariance matrix (symmetric and positive definite), Σ, and an effective radius, d. The mean vector determines the location of the hyperellipsoid, while the covariance matrix determines its shape and orientation. The squared Mahalanobis distance [4] is used to determine if a vector $\underline{x}$ lies inside or outside of a given hyperellipsoid. Vectors lying inside of a hyperellipsoid satisfy:

$$(\underline{x}-\underline{\mu})^T\Sigma^{-1}(\underline{x}-\underline{\mu}) < d \qquad (2)$$

An initial clustering of the urban training data is needed as the basis for this classifier. There are numerous ways in which this might be accomplished. We use the k-means clustering algorithm [4]. This algorithm, however, uses the Euclidean distance metric for clustering data, and may not yield representative hyperellipsoidal clusters. In cases such as these, the cluster merging scheme proposed in [5] can be used. After cluster centers have been determined, effective radii are selected for each cluster. We choose the effective radius d for each cluster so that under the assumption that the data is gaussian in nature, P% of the data will fall inside the boundary (typically P=99%.) For an N-dimensional problem, the distribution of the squared Mahalanobis distances to each of the vectors within a given cluster is a χ^2 distribution with N degrees of freedom. If pattern vectors consist of 14 features, for example, then an effective cluster radius of 29.1 would cause the hyperellipsoidal boundaries to contain 99% of the data.

5 Hyperellipsoid Modification

After classifier initialization, urban data is represented by a number of hyperellipsoidal clusters in the fea ture space. An adaptive training algorithm is now employed to reduce classification error in areas of known overlap between different classes. During adaptation, each hyperellipsoid in the classifier will maintain its original orientation, although its position and shape will be modified. The algorithm that we use is similar to the LVQ algorithm [6], and thus is referred to as the LVQ-MM algorithm (LVQ using the Mahalanobis distance Metric).

Assume that we have a hyperellipsoidal decision boundary, and a new vector which we would like to include within the in-class data region (see Figure 1). We are only going to allow the mean vector $\underline{\mu}$ and the eigenvalues of the covariance matrix Σ to change. This means that the orientation of the cluster will remain fixed, although its position and shape may be modified. The cluster will be modified in such a way that the following are true: (1) the new point, $\underline{x}_n$, lies on the new cluster boundary; (2) the original boundary point lying on the opposite side of the hyperellipsoid from $\underline{x}_n$ remains on the cluster boundary; and (3) the eigenvalues of Σ are modified in such a way that the hyperellipsoid is only stretched "towards" $\underline{x}_n$.

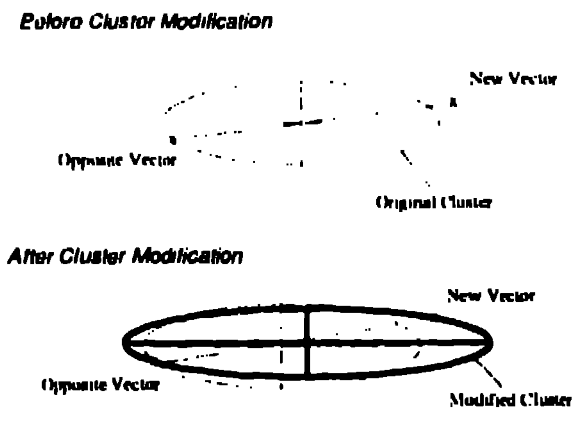

Figure 1: Modifying a Hyperellipsoidal Boundary

The new mean vector for the cluster will lie directly between $\underline{x}_n$ and the point lying opposite it on the hyperellipsoidal boundary, and is given by:

$$\underline{\mu}_n = \frac{1}{2}\left[\left(1+\sqrt{\frac{d}{m_n}}\right)\underline{\mu}+\left(1-\sqrt{\frac{d}{m_n}}\right)\underline{x}_n\right] \qquad (3)$$

where d is the effective cluster radius, and m_n is the squared Mahalanobis distance from the cluster to $\underline{x}_n$.

After computing $\underline{\mu}_n$, we need to modify Σ. In doing so, we are essentially going to stretch the decision boundary towards the new point, $\underline{x}_n$. Note that if the new point lies along one of the hyperellipsoidal axes, that only one eigenvalue for Σ will need to be changed. Otherwise, all (or at least several) eigenvalues will need to be modified. For simplicity, we will restrict our attention to modifying the inverse of Σ. Let us decompose Σ^{-1} as follows:

$$\Sigma^{-1} = M\Lambda M^T \qquad (4)$$

where M is the orthonormal eigenvector matrix for Σ^{-1}, and Λ is the diagonal eigenvalue matrix. To stretch the cluster we will modify the eigenvalues of Σ^{-1}, and keep the eigenvectors fixed. Thus we wish to find a new inverse covariance matrix:

$$\Sigma_n^{-1} = M\Lambda\Lambda_s M^T \qquad (5)$$

where the "stretch matrix", Λ_s, takes on the form:

$$\Lambda_s = \begin{bmatrix} 1+\delta_1 p & 0 & \cdots & 0 \\ 0 & 1+\delta_2 p & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & 1+\delta_N p \end{bmatrix} \qquad (6)$$

The parameter p determines the total stretch, and the parameters δ_i determine the percentage stretch in the direction of the i^{th} principal component. The δ_i satisfy the following constraints:

$$0 \le \delta_i \le 1, \qquad \sum_{i=1}^{N} \delta_i = 1 \qquad (7)$$

Our goal is to determine the parameters p and δ_i so that the hyperellipsoidal boundary is moved to $\underline{x}_n$.

$$(\underline{x}_n - \underline{\mu}_n)^T \Sigma_n^{-1} (\underline{x}_n - \underline{\mu}_n) = d \qquad (8)$$

$$\Sigma_n^{-1} = M\Lambda\Lambda_s M^T \qquad (9)$$

Let us define a new vector $\underline{z}$ to be:

$$\underline{z} = \Lambda^{0.5} M^T (\underline{x}_n - \underline{\mu}_n) \qquad (10)$$

The components of this vector, z_i, $i = 1, 2, \ldots, N$, represent the strength of the projection of $\underline{x}_n$ onto the N principal components. With this, (8) can be rewritten as:

$$\underline{z}^T \Lambda_s \underline{z} = \sum_{i=1}^{N} z_i^2 (1 + \delta_i p) = d \qquad (11)$$

The percentages, δ_i, are chosen to be equal to the relative magnitude of the projection of $\underline{x}_n$ onto each of the principle components:

$$\delta_i = \frac{|z_i|}{\sum_{j=1}^{N} |z_j|} \qquad (12)$$

It is easy to verify that this choice for δ_i satisfies (7). Substituting (12) into (11) and solving for p we get:

$$p = (d - m_n)\frac{\sum_{i=1}^{N} |z_i|}{\sum_{i=1}^{N} |z_i|^3} \qquad (13)$$

In summary, the new inverse covariance matrix is given by (5) where the components of Λ_δ are computed using (10), (12), and (13).

The cluster modification method discussed above provides a foundation for the classifier adaptation algorithm. Using the cluster modification technique to move clusters "toward" and "away from" training vectors, this algorithm will attempt to minimize classification error in regions of known overlap between classes. We will restrict our attention to modifying clusters for a single class of data only. This process is then used independently for each class of data.

6 LVQ-MM Algorithm

Consider the two-dimensional problem shown in Figure 2. The classifier has been initialized using the in-class training data, and consists of three hyperellipsoids in the input space. Notice that based on the training data available to the classifier, there is no overlap in the region "below" the current hyperellipsoids. The only conflicting areas lie on the upper portion of the closed decision region. This suggests that better classifier performance can be achieved by allowing the decision region to include all in-class vectors lying below the current set, and by moving the upper boundary of the classifier to a position minimizing misclassifications in that area.

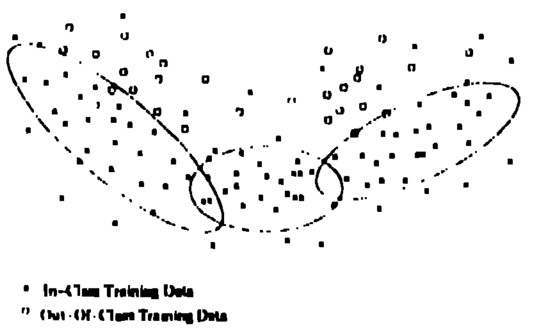

Figure 2: A Two-Dimensional System Before Adaptation

A single step in the LVQ-MM algorithm will basically work as follows. Select a random vector from the training data which is currently misclassified (correctly classified samples do not affect the classifier training). Using the cluster adaptation equations previously derived, move one of the hyperellipsoidal boundaries either toward or away from this vector, depending on its class membership. Note that as the cluster adaptation equations currently stand, this step will always cause the current vector to fall directly on the new hyperellipsoidal boundary.

The question to be addressed now is, "Which of the hyperellipsoids should be modified?" Consider the two cases illustrated in Figure 3. In Case 1 we want to modify cluster 1 to include the new vector even though the mean vector for cluster 1 is farther away from the new vector than the mean vector for cluster 2 in terms of Euclidean distance. In terms of Mahalanobis distance, however, the opposite is true. In Case 2, on the other hand, the roles are reversed. We want to modify cluster 2 to include the new vector. The Euclidean distance to cluster 2 is smaller than the Euclidean distance to cluster 1, and in terms of Mahalanobis distance, cluster 1 is closer. Clearly, the cluster to be modified should be the cluster whose boundary is closest to the new vector (in terms of Euclidean distance). It can be shown that the distance from the boundary to the new vector is given by

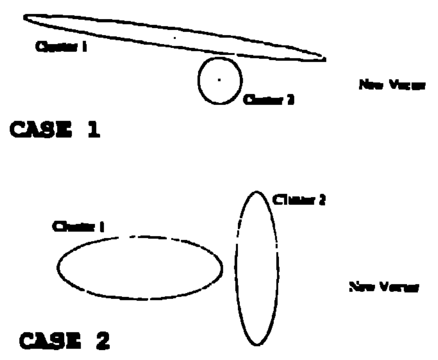

Figure 3: Selecting Desired Boundary

$$dist = |(1 - \sqrt{\frac{d}{m}})| \cdot \|x_n - \mu\| \qquad (14)$$

where d is the effective cluster radius, and m is the squared Mahalanobis distance to the new vector. The adaptation loop, then, works as follows.

LVQ-MM ALGORITHM

```
(1) Select a training vector that is misclassified
(2) Determine which cluster to modify
(3) Modify mean using (3)
(4) Modify inverse covariance matrix using (5)
```

This algorithm is typically run for several passes through the training data.

7 Cooling Algorithm

As with many adaptive pattern recognition techniques, a cooling technique can be used with the LVQ-MM al

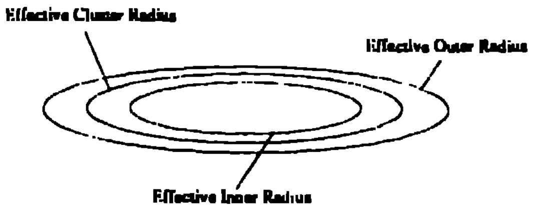

Figure 4: False Effective Cluster Radii

gorithm to ensure convergence of the decision boundary positions. As the LVQ-MM algorithm executes, there will be continuous conflict in the regions of the feature space where urban and nonurban data overlap. Boundaries will continually move outward to "catch" more in-class data, and then move inward to reject out-of-class data. The basic principle for cooling in this algorithm will be that we will only move the decision boundary "towards" the vector in question, instead of moving the boundary on top of it. As the algorithm progresses, the amount which the decision boundary is allowed to move will decrease.

The cooling algorithm for the LVQ-MM algorithm will support a false set of effective cluster radii. For any given cluster j having an effective cluster radius of d_j, two false radii will also be stored. One will be an effective outer radius, denoted d_j^+, and the second will be an effective inner radius, denoted d_j^-. Geometrically speaking, the cluster boundaries generated by the true effective cluster radius, d_j, will lie between the cluster boundaries generated by the false radii (see Figure 4). These false effective radii will be used instead of the actual effective cluster radii during adaptation.

When determining if an in-class vector currently falls outside of a cluster, the false effective radius d_j^+ will be used as opposed to d_j. The cluster modification routine will then adjust the cluster so that the false boundary generated by using d_j^+ as the effective cluster radius falls on top of the misclassified vector. Note that the true cluster boundary is moved toward the misclassified vector, but it does not move all the way up to it. The same idea holds for dealing with misclassified out-of-class training vectors, except that d_j^- is used as the effective cluster radius.

The actual cooling algorithm will be the algorithm which determines how d_j^- and d_j^+ are modified during the execution of LVQ-MM. There are numerous methods which may be used. We present only the one used for obtaining the results presented in this paper. A cooling parameter, *cool*, is selected where $0 \leq cool \ll 1$. (No cooling occurs when $cool = 0$). Each cluster j in the classifier has associated with it values for d_j^- and d_j^+. Both of these values are initially set to the actual effective cluster radius for cluster j, d_j. After cluster j is adapted to include an outside vector, its effective outer radius, d_j^+, is increased by multiplying it by $(1 + cool)$. Similarly, after cluster j is adapted to reject an inside vector, d_j^-, is decreased by multiplying it by $(1 - cool)$. Using this technique, each cluster's movement ability is "cooled" based only on its own recent level of activity, and not that of the entire system. Obvious modifications to this method of "cooling" can be made to change the system behavior.

8 Results

Urban and non-urban data sets were selected from the Albuquerque scene to train and test our classificaton methods. The non-urban training data includes some areas which, although they look different than urban areas, are also highly textured. In selecting a set of test data for this problem, urban areas which look similar to the urban training data were chosen.

We first trained a linear classifier and a 1-hidden layer neural network on the problem. Linear classifiers, by definition, do not define closed decision boundaries, and multi-layer perceptron neural networks are not guaranteed to form closed decision boundaries. Table 3 shows the linear classification results using the LMS algorithm, and Table 4 shows the results using the neural network. When used to classify the entire 1024x1280 Albuquerque test image, these classifiers categorized about 22% of the image as urban.

	URB	NON	Correct
Urban (train)	17763	86	99.52%
Nonurban (train)	322	156569	99.79%
Urban (test)	15459	237	98.49%
Nonurban (test)	192	54048	99.65%

Table 3: Linear Classification Using the LMS Learning Algorithm

	URB	NON	Correct
Urban (train)	17845	4	99.98%
Nonurban (train)	408	156483	99.74%
Urban (test)	15440	250	98.41%
Nonurban (test)	59	54181	99.89%

Table 4: Classification Using the Backpropagation Learning Algorithm

Using the one-class classification approach for this problem, the urban training data was clustered into 5 clusters. An effective cluster radius of 29.1 was chosen for each cluster, and this classifier was used to categorize the data from the Albuquerque test scene. Again, this classifier catogorized about 22% of the image as urban. Numerical results for this classifier are shown in Table 5.

	URB	NON	Correct
Urban (train)	17499	350	98.04%
Urban (test)	13390	2306	85.31%
Nonurban	4394	206737	97.92%

Table 5: Classification Using 5 Clusters (Before Adaptation)

After using the LVQ-MM algorithm to modify the decision boundaries, only 17% of the test scene is categorized as urban. These results are given in Table 6. This classifier separates the training data well, and rejects about 6.63% of the urban test data. Results from the adapted classifier using only 1 cluster are given in Table 7.

	URB	NON	Correct
Urban (train)	17848	1	99.99%
Nonurban (train)	7	156884	100.00%
Urban (test)	14656	1040	93.37%
Nonurban (test)	3	54237	99.99%

Table 6: Classification Using LVQ-MM and 5 Clusters

	URB	NON	Correct
Urban (train)	17848	1	99.99%
Nonurban (train)	1268	!55623	99.19%
Urban (test)	15450	246	98.43%
Nonurban (test)	74	54166	99.86%

Table 7 Classification Using LVQ-MM and 1 Cluster

9 Conclusions

The concept of using closed decision boundaries for pattern recognition allows a system to be considered robust in the sense that anomalous input data will not generate a false "detection" of a given class of interest. The classifier presented here does a good job at rejecting unfamiliar data as being non-urban. Additionally, the classifier can be created by using only in-class training data. Many pattern recognition algorithms are unable to do this.

References

[1] K. Laws. *Textured Image Segmentation*. Ph.D. dissertation, Univ. of Southern Calif., January 1980.

[2] K. Laws. Rapid texture identification. In *SPIE Vol. 238 Image Processing for Missile Guidance*, pages 376–380, 1980.

[3] M. Moya. *A Constrained Second-Order Network with Mean Square Error Minimization and Boundary Size Minimization for One-Class Classification*. PhD Dissertation, University of New Mexico, 1991.

[4] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, NY. 1973.

[5] P.M. Kelly. *A One-Class Classifier Using Hyperellipsoidal Decision Surfaces*. Masters Thesis, University of New Mexico, 1991.

[6] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, Germany, 1988.