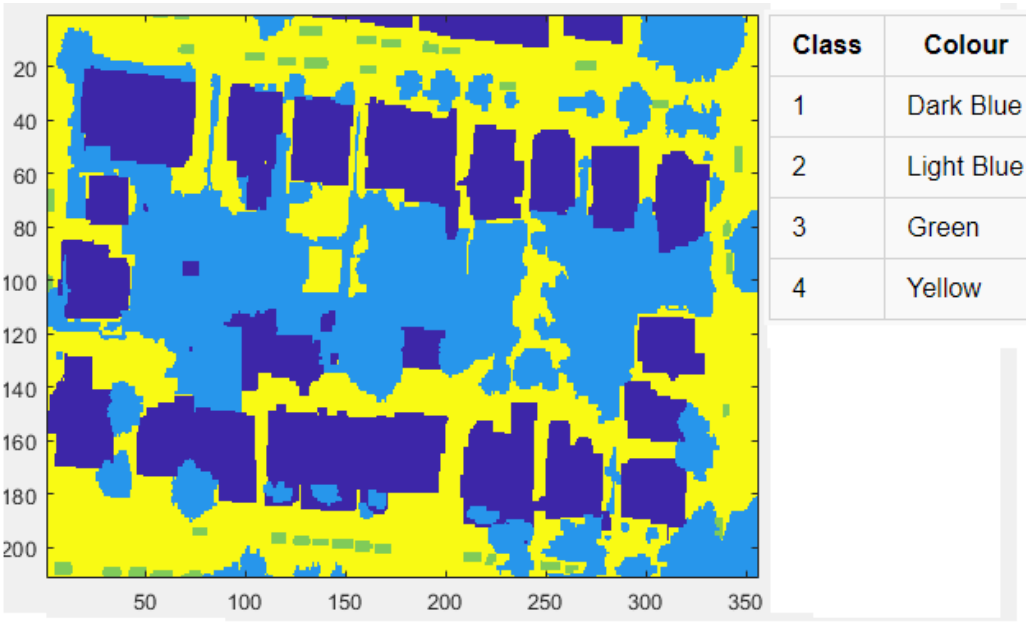
Abstract

Statistical pattern recognition is an important aspect of visual intelligence. This report explores implementing maximum likelihood estimation and applying this to a land classification problem. Using a ground truth image for training along with a set of images (Image bands of red, green, and blue, LIDAR first and last echo, and near infrared) of the same section of land, the objects in the area were able to be classified into four classes with a maximum accuracy of 79.7% when using 100 samples per class for training. It was found that increasing the training sample size generally improves the accuracy of the classification, though exceeding 100 samples begins to once again decrease the accuracy.

Introduction

Statistical pattern recognition (SPR) is a significant part of visual intelligence, in which a general model is produced from statistics and input data, that can be applied to unseen data to predict aspects about it (1). The objective for this report is to implement and explain the Maximum Likelihood method of SPR and, using 6 images of remotely sensed data with the ground truth information (**Fig.1**), classify the given data into 4 classes.



Class	Colour	Description
1	Dark Blue	Building
2	Light Blue	Vegetation
3	Green	Cars
4	Yellow	Ground

Figure 1: *Ground Truth Information and Legend*

In addition to the implementation, some testing and exploration will take place in how the model could be improved to give a closer prediction to the original image.

Methodology

The method for Statistical Pattern Recognition that has been implemented in MATLAB, is the maximum likelihood estimation (MLE). MLE is a method of supervised learning that builds a classifier for the application, from a subset of the overall data. MLE is required in instances where prior probabilities of the data are unknown, and so, by assuming that both the classes and the data are normally distributed, can make it possible to represent the posterior probability as the class conditional probability density function (pdf).

In the implementation of MLE, 6 bitmap images have been provided that show the same landscape, in different forms. This set of images consists of red, green, and blue image bands, LIDAR first echo and last echo, and a near infrared image. Therefore, we have 6 dimensions of the sample's feature space, (**n** in **Fig.2**). We have also been provided with ground truth data that classifies each point of the images into 1 of the 4 classes. (**Fig.1**)

In the ground truth data, 20 points for each class are randomly found, and the values corresponding to those points, in each of the 6 bitmap images are used to build the subset of data required to build this classifier. Therefore, we have 20 6-dimensional vector test samples, for each class (i = {1..20} in **Fig.2**)

The mean vector for each class ($\boldsymbol{\mu}_i$) in this case is a 6-dimension vector that represents the mean of the test samples in each image, and the covariance of a class ($\sum_i$) is a 6x6 matrix. These were found with MATLABs in-built 'mean()' and 'cov()' functions respectively.

$$p(\mathbf{x} \mid \omega_i) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma_i|}} \exp[-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1}(\mathbf{x} - \mu_i)]$$

Figure 2: *Equation for the Class Conditional PDF of a gaussian distribution (2)*

Once finding these values, they can be substituted into the above equation to produce the Probability Density Function of the gaussian distribution, for each class.

```
...for classIndex = 1:4
    prob_vals(classIndex) = PDF(cov_list(:,:,classIndex),reshape(mean_list(:,classIndex),[1,6]),pixel_value);
end

maxValue = max(prob_vals);
new_image(row,col) = find(prob_vals==maxValue);

...

function val = PDF(cov,mean,pixel_value)
    term_1 = 1/((2*pi).^6/2 * sqrt(det(cov)));
    term_2 = exp(-0.5*(pixel_value - mean)*inv(cov)*transpose(pixel_value - mean));
    val = term_1 * term_2;
end
```

Figure 3: *Implementation of the equation in* ***Fig.2*** *(4)*

The above shows the implementation of this function being used to find the best probability of which class the pixel belongs to. Once found, this is assigned to the output image.

When all points in the images have been analysed, the output image can be compared to the original ground truth using a confusion matrix. A confusion matrix is a matrix describing the performance of classifiers on a set of test data for which the true values are known (3). In the MATLAB implementation, the confusion matrix was produced by using the `confusionchart()` and `confusionmat()` in-built functions.

```
subplot(2,2,1) , imagesc(new_image), title('Predicted Results')
subplot(2,2,2) , imagesc(labelled_ground_truth), title('Ground_Truth')
subplot(2,2,3) , confusionchart(confusionmat(reshape(labelled_ground_truth,1,[]),reshape(new_image,1,[]))), title('Confusion
Matrix')
```

Figure 4: *Code to demonstrate the ground_truth data, the predicted data and the confusion matrix (4)*

Results

Confusion Matrix:

A confusion matrix is a good method of visually demonstrating the results of a classification algorithm. It shows the quantity of correctly and incorrectly classified data points plotting the predicted vs true class. The sum of all the cells in the matrix is equivalent to the total number of input data points.

A confusion matrix has been generated of the classification results as shown in **Fig.5**. This matrix shows the results of the classification algorithm when applied with 20 training points per class.

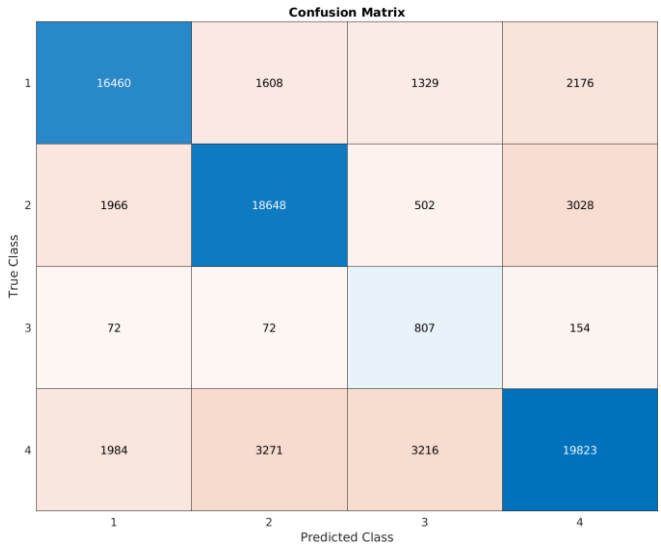


Figure 5: *Confusion matrix of predicted results against true results*

Using the confusion matrix, the success rate of the classification algorithm can be calculated, both overall, and individually for each class. To calculate the overall success rate, the trace of the matrix is calculated and divided by the total number of data points.

With 20 training samples per class, the calculated accuracy is:

$$\frac{16460 + 18648 + 807 + 19823}{75116} = 0.742 = 74.2\%$$

Figure 6: *Accuracy equation of predicted results w/ 20 test samples*

With an accuracy of 74.2% this leaves an overall error rate of 25.8%.

In a similar vein, individual class accuracy can be calculated by taking a single element from the main diagonal and dividing this by either the sum of the column or the row. Dividing by the row shows how accurately a class can be predicted, whereas dividing by the column shows how accurate a given class prediction is. In the table below, the accuracy of a given prediction has been calculated.

Predicted Class (description)	**Accuracy %**	**Error %**
1 (buildings)	80.36%	19.64%
2 (vegetation)	79.02%	20.98%
3 (cars)	13.79%	86.21%
4 (ground)	78.72%	21.28%

Figure 7: *Individual class accuracy rates.*

Fig.7 shows that the reliability of a classification is relatively good for buildings, vegetation, and ground; however, if the prediction shows a car, it is most likely not a car.

Colour Coded Figure Showing the Four Classes:

Fig.8 below shows the difference between the ground truth and the predicted classes when using a training set of 20 samples per class. It seems that the algorithm is generally successful, but a decent amount of noise is still present in the image and the excess of car predictions is noticeable.

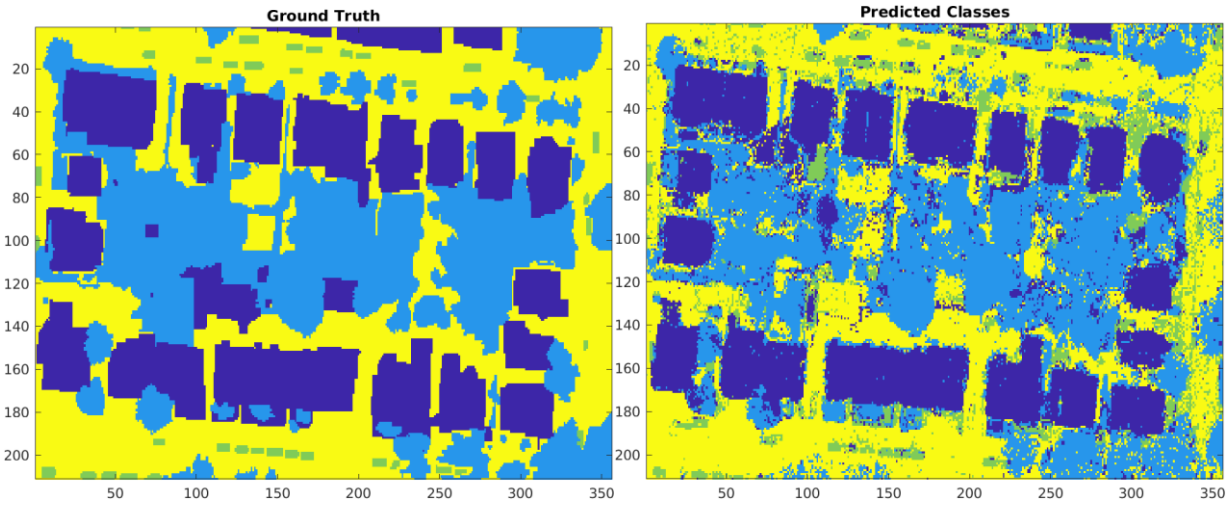


Figure 8: *Ground truth compared to predicted classes*

Effect of Sample Size on Accuracy:

Fig.9 shows how altering the number of samples used in training the algorithm affects the resulting accuracy.

Samples	Accuracy %	Error %
10	50.0%	50.0%
20	74.2%	25.8%
30	77.9%	22.1%
50	78.7%	21.3%
70	79.4%	20.6%
100	79.7%	20.3%
150	79.3%	20.7%

Figure 9: *Table demonstrating the various accuracy rates at different sample sizes*

This information suggests that increasing the number of samples generally increases the accuracy of the classifier. Although, the accuracy very quickly begins to plateau around 80% and the creation of too many training samples leads to an eventual decline in accuracy.

Conclusion

Using MLE It was found that increasing the training samples per class generally had the effect of increasing the accuracy of the algorithm, though exceeding ~100 samples had the opposite effect.

The best classification accuracy of 79.7% was achieved with a training set of 100 samples per class. Though this allows for a generally successful classification, there still remains significant room for improvement that may not be achievable with MLE alone. The main hinderance to the results was the reliability of the algorithm in classifying cars, consistently calculating a significant amount of false car classifications (almost 90% of car classifications were falsely allocated).