CLASSIFICATION OF FOREST AREAS BY K NEAREST NEIGHBOR METHOD: CASE STUDY, ANTALYA.

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FERAY ÖZSAKABAŞI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
GEODETIC AND GEOGRAPHIC INFORMATION TECHNOLOGIES

JUNE 2008

Approval of the thesis:

CLASSIFICATION OF FOREST AREAS BY K NEAREST NEIGHBOR METHOD: CASE STUDY, ANTALYA.

submitted by **FERAY ÖZSAKABAŞI** in partial fulfillment of the requirements for the degree of **Master of Science in Geodetic and Geographic Information Technologies Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Assoc. Prof. Dr. Şebnem Düzgün
Head of Department, **Geodetic and Geographic Information Tech.** _____

Assoc. Prof. Dr. Zuhal Akyürek
Supervisor, **Geodetic and Geographic Information Tech. Dept., METU** _____

Examining Committee Members:

Prof. Dr.Vedat Toprak _____
Geological Engineering Dept., METU

Assoc. Prof. Dr. Zuhal Akyürek _____
Civil Engineering Dept., METU

Assoc. Prof. Dr. Şebnem Düzgün _____
Mining Engineering Dept., METU

Assoc. Prof. Dr. Nurünnisa Usul _____
Civil Engineering Dept., METU

Assist. Prof. Dr. Pınar Şenkul _____
Computer Engineering Dept., METU

Date: 05.06.2008

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Feray Özsakabaşı

Signature :

ABSTRACT

CLASSIFICATION OF FOREST AREAS BY K NEAREST NEIGHBOR METHOD: CASE STUDY, ANTALYA.

Özsakabaşı, Feray

M.Sc., Department of Geodetic and Geographic Information Technologies

Supervisor: Assoc. Prof. Dr. Zuhal Akyürek

June 2008,101 pages

Among the various remote sensing methods that can be used to map forest areas, the K Nearest Neighbor (KNN) supervised classification method is becoming increasingly popular for creating forest inventories in some countries. In this study, the utility of the KNN algorithm is evaluated for forest/non-forest/water stratification.

Antalya is selected as the study area. The data used are composed of Landsat TM and Landsat ETM satellite images, acquired in 1987 and 2002, respectively, SRTM 90 meters digital elevation model (DEM) and land use data from the year 2003. The accuracies of different modifications of the KNN algorithm are evaluated using Leave One Out, which is a special case of K-fold cross-validation, and traditional accuracy assessment using error matrices. The best parameters are found to be Euclidean distance metric, inverse distance weighting, and k equal to 14, while using bands 4, 3 and 2. With these parameters, the cross-validation error is 0.009174, and the overall accuracy is around 86%. The results are compared with those from the Maximum Likelihood algorithm. KNN results are found to be accurate enough for practical applicability of this method for mapping forest areas.

Keywords: K-nearest neighbor method (KNN), Classification, Forest, Accuracy Assessment, Antalya.

ÖZ

ORMAN ALANLARININ K EN YAKIN KOMŞU METODUYLA SINIFLANDIRILMASI: ÖRNEK ÇALIŞMA, ANTALYA

ÖZSAKABAŞI, FERAY
Yüksek Lisans, Jeodezi ve Coğrafi Bilgi Teknolojileri Bölümü
Tez Yöneticisi: Doç. Dr. Zuhal Akyürek

Haziran 2008, 101 sayfa

Orman alanlarını haritalandırmak amacıyla kullanılmakta olan farklı uzaktan algılama metodları arasında, K En Yakın Komşu (KNN) kontrollü sınıflandırma metodu, bazı ülkelerde orman envanterlerini oluşturmak için gittikçe daha popüler hale gelmektedir. Bu çalışmada, KNN algoritmasının orman/orman olmayan/su katmanlaşması için kullanılabilirliği değerlendirilmiştir.

Çalışma alanı olarak Antalya seçilmiştir. Kullanılan veriler, sırasıyla 1987 ve 2002 yıllarına ait Landsat TM and Landsat ETM uydu görüntüleri, SRTM 90 metre sayısal yükseklik modeli (SYM) ve 2003 yılına ait arazi kullanımı verisinden oluşmaktadır. KNN algoritmasının farklı modifikasyonları, K-katlı çapraz doğrulamanın bir türü olan Leave One Out ve hata matrislerinin kullanıldığı bilinen doğruluk analizleri ile değerlendirilmiştir. En iyi parametreler, 4., 3. ve 2. bandları kullanırken, Euclidean uzaklık ölçümü, ters uzaklık ağırlıkları, ve k değeri 14'e eşit olarak bulunmuştur. Bu parametrelerle, çapraz doğrulama hatası 0.009174 ve toplamdaki doğruluk yaklaşık %86'dir. Sonuçlar, Maksimum Benzerlik sonuçlarıyla karşılaştırılmıştır. KNN sonuçlarının, orman alanlarının haritalandırılması için pratik uygulanabilirliğine yetecek kadar doğru olduğu görülmüştür.

Anahtar Kelimeler: K En Yakın Komşu Metodu, Sınıflandırma, Orman, Doğruluk Analizi, Antalya.

To my family …

Thank you for your love and support

ACKNOWLEDGEMENTS

I would like to express my deepest thanks to my parents and my family for their unbelievable patience and support. Without your steadfastness and encouragement, this study would not have been completed in time. Your tolerance during difficult times is unforgettable.

I am eternally grateful to my supervisor, Assoc. Prof. Zuhal Akyürek, for her helpful guidance, her wise supervision, her genuine interest, her sincere help and encouragement. You helped me out in all of my difficulties, technical or otherwise. Without your critical and constructive reviews, this study would have been impossible. I am glad to have started my future academic life under your supervision. I learned a lot from you, and I consider myself very fortunate.

Great thanks go to my other examining committee members, Prof. Vedat Toprak, Assoc. Prof. Şebnem Düzgün, Assoc. Prof. Nurünnisa Usul and Assist. Prof. Pınar Şenkul. Your supportive help to my problems, your critical review of this study and your suggestions helped make it a lot better.

I also wish to thank all GGIT Department assistants for their support throughout my graduate studies and during the thesis. Your outstanding help was important and influential for me, and what I learned from you was very valuable.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

CHAPTER 1

INTRODUCTION

Creating an accurate forest inventory by mapping forest vs. non-forest areas is an essential requirement for forest management and planning. Using field analysis for creating forest inventories is a difficult task, especially due to economical constraints. In field sampling, the density of sample plots has to be low or even very low in hard-to-reach areas. Therefore, attempts to estimate forest variables for small areas turn out to be inaccurate, and mapping large areas is infeasible. In the last decades, the availability of high-resolution satellite images has opened up new possibilities for mapping forest areas in an objective, accurate and comprehensive way. In particular, intelligent image classification algorithms and increasing computing power make forest mapping possible to be performed automatically with the help of computers.

In remote sensing, a “class” can be defined as a land cover type. For example, to map a forest inventory, the classes “forest,” “non-forest” and “water” are of interest because the training samples are collected only from forest areas with plots. For a pixel in a satellite image, image classification algorithms help us to detect which class that pixel belongs to. Each pixel in the satellite image belongs to only one of these classes. Thus, the classified image is a computer-generated land cover map. Here, one can easily detect the expanse covered by each land cover class.

There have been a lot of researches in image classification algorithms and their application to forest detection. These researches are especially active in Scandinavian countries and in the USA. Suggestions for algorithmic improvement and optimization, mainly to increase classification accuracy, are an important focus in these studies. Although the various classification methods are well-established, there is still room for improvement. In Turkey, there are some studies which determine forest stand parameters using remotely sensed data. The projects focus

on classifying and mapping the stand parameters such as development stages, crown closure, stand types, and land cover.

The two most common algorithms employed in forest area detection are K-nearest-neighbor (KNN) and Maximum Likelihood, which are both classification algorithms. In recent years, KNN is becoming more and more significant and is attaining widespread use, especially in mapping forest areas.

In this study, the primary objective is to evaluate the utility of the KNN method for forest/non-forest/water stratification. In the process, various modifications of the KNN algorithm are tested for detecting forest areas. By comparing the resulting accuracies, the optimal parameters of the classification for KNN forest detection are found. Also, KNN results are compared with Maximum Likelihood results, to illustrate its relative effectiveness. The comparison is done mainly from the perspectives of accuracy and computing performance.

Research activity in the area of satellite image classification is expected to continue for a long time, since it offers vast potential. The automation of complex classification tasks will allow us to obtain meaningful knowledge from vast amounts of data, with the possibility of speed never before imagined. Decision makers will be supported with reliable, up-to-date land cover data. In the particular case of forest mapping, it will be possible to track the change of forest areas over the years, by comparing the classified satellite images acquired at different times.

Deforestation is the conversion of forested areas to non-forest areas, by cutting down trees and making room for alternative land uses, mainly for urban, industrial or agricultural growth. Large-scale deforestation is very harmful since it reduces biodiversity, affects climate (global warming), increases the probability of flooding and erosion, and ultimately, reduces the quality of human life (URL1). During the last decades, this has become an important problem, especially in developing countries like Turkey. Better image classification algorithms will help these countries make better planning decisions, which will ultimately improve the quality of life.

In this study the area is selected as the Center District of Antalya, which is a rapidly growing city in Turkey. Namely, it has a population growth of 41.79% for the last

decade. Change of forest areas during this growth phase is expected to be significantly large, due to urbanization. Therefore, two satellite images of Antalya, belonging to different years 1987 and 2002, are classified using KNN and Maximum Likelihood algorithms. Thus, the change of forest areas over these years of high growth can be analyzed. The performance of KNN algorithm against Maximum Likelihood algorithm in terms of accuracy is evaluated for detecting deforestation.

The study is presented in six chapters:

In the first chapter, the Introduction, the importance of improving land use detection algorithms is outlined.

In the second chapter, Theoretical and Practical View of Classification on Forest Areas, previous studies on supervised classification algorithms that are relevant to this particular study are summarized.

In the third chapter, Materials and Methodology, the study area is illustrated, preparation of the data is explained, and any difficulties are described. Also, flowcharts for the classification algorithms are given.

In the fourth chapter, Analyses, the procedures in implementing and carrying out the algorithms are described in detail. The advantages and disadvantages of each algorithm are discussed. The explanations convey an approximate sense of the relative implementation complexity of these algorithms. Solutions to the difficulties encountered during image classification are explained, which should help further related studies greatly. The classification results can also be found here.

In the fifth chapter, Discussion of the Results, the results obtained from the large number of accuracy assessment experiments are presented, and their significance is discussed. Change detection is performed to compare different results. With careful interpretation, the reasons are explained.

In the sixth and final chapter, Conclusions and Recommendations, a summary and evaluation of the study are presented.

CHAPTER 2

THEORETICAL AND PRACTICAL VIEW OF CLASSIFICATION ON FOREST AREAS

2.1. Classification Algorithms and Accuracy Assessment

It is a painstaking process to detect land cover features in satellite images by hand. If we are able to automate this process with the help of computers, we can facilitate this task immensely. Therefore, in the long run, we will be able to generate land cover / land use maps over unlimited areas and over arbitrarily long timeframes. This will help us to observe past trends and predict future tendencies very rapidly.

In order for the computer to simulate human recognition of land cover features, artificial intelligence (AI) must come into play. Computer AI is not perfect, and is a dynamic, unending field of research. A common technique to develop AI is “machine learning,” where computers can “learn” with the use of some advanced algorithms inspired from human recognition. Using this intelligence, computers can extract information from data automatically.

In our particular case, the computer must be able to classify a satellite image into its features. This is a typical case of the statistical classification problem. Here, the computer reads in a training set of previously classified objects and sees a number of input/output (object/class) examples. Afterwards, it predicts the class labels of individual, unclassified input objects, based on one or more quantitative variables in these objects. This is a supervised (machine) learning task. Therefore, the most common supervised learning algorithms are applied in this study suited to this task:

There are different classification techniques used in forest inventories. For example, in Scandinavian forest inventory studies KNN is used mostly. On the other hand, in some countries different classification techniques are used, for example, in Canadian Forest inventory studies, K-Means clustering as an unsupervised classification technique, and maximum likelihood as a supervised classification technique (Wulder et al., 2001).

2.1.1. KNN Algorithm

The nearest neighbor algorithm is one of the simplest machine learning algorithms. It is a non-parametric technique. Before detailing the algorithm, some definitions are necessary:

- The "distance" between two objects is taken as the Euclidean distance between them. (In some cases, the Manhattan or Mahalanobis distance can be used as well.) To calculate this distance, every object must be represented by a position vector in a multidimensional feature space. Let the vectors $\vec{x}$ and $\vec{y}$ be two input samples (objects) with p features $(x_1, x_2, \ldots, x_p)$. The Euclidean distance between sample $\vec{x}$ and sample $\vec{y}$ is defined as in Equation 2.1.

$$d(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_p - y_p)^2} \quad (2.1)$$

- An object is the "neighbor" of another object if the distance between them is below a predefined threshold.
- The "nearest neighbor" of an object x is the sample object whose distance to x is the lowest among all input samples.
- The "2nd nearest neighbor" of an object x is the sample object whose distance to x is the second lowest among all input samples. The "nth nearest neighbor" is defined analogously.
- "k nearest neighbors" of an object x are the collection of sample objects x_i where $i = \{1, 2, \ldots, k\}$ and x_i is the ith nearest neighbor of x.

The nearest neighbor algorithm steps can be described as follows:

1. Training phase
    a) A human being classifies a number of objects manually. This is the training set. The feature vectors and class labels of these samples are stored.
    b) The computer reads in this set of objects. The correct classification for these objects is known.
2. Classification phase
    a) A new, unclassified input object (test sample) is classified by a majority vote of its neighbors:
        - The neighbors are taken from the training set.
        - Distances from the test sample object to all stored sample objects are calculated, and the k nearest neighbors of the object are selected. k is a small integer.
        - There are different ways to assign a particular class to the object. Usually, the most common class among these k neighbors is assigned to the object. In other words, an object is assigned to the class c if it is the most frequent class label among the k nearest training samples. If k = 1, then the class of the nearest neighbor is assigned to the object. This special case (k = 1) is called the “nearest neighbor” algorithm.

Normally, the training phase is executed once, and the classification phase is executed any number of times afterwards.

Drawbacks:

- If there is a class with a very large number of training samples compared to the other classes, then its samples come up more frequently among the k nearest neighbors of a new object when these are calculated. This class dominates the classification of new objects, by overwhelming samples belonging to other classes. This can be avoided by slightly enhancing the majority vote. For example, one can modify it so that the distance of each neighbor to the test sample determines the “strength” or “closeness” of that

neighbor. So, the shorter the distance, the more effect the sample has on the majority vote.

- The accuracy drops severely when there are noisy or irrelevant features, or if the feature scales are inconsistent with their importance.
- The algorithm does not report confidence or class probabilities.
- A classification is always made. There are no objects that cannot be assigned to a class.

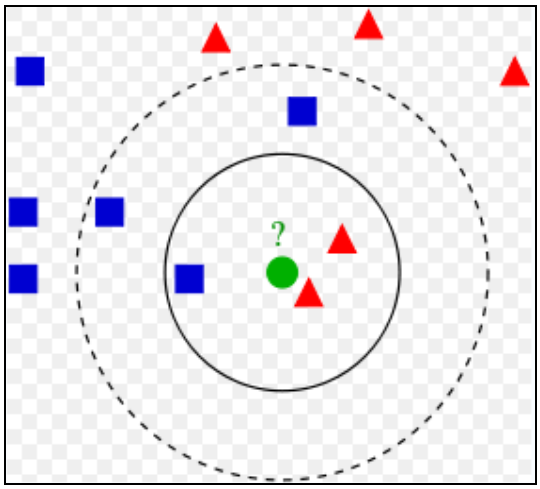

Figure 2.1. Example of KNN classification

As an example given in Figure 2.1, the test sample (green circle) should be classified either to class “blue square” or to class “red triangle.” If *k = 3* it is classified to the “red triangle” because there are 2 triangles and only 1 square inside the inner circle. If *k = 5* it is classified to “blue square” (3 squares vs. 2 triangles inside the outer circle).

2.1.1.1. Choosing k value

If there are only two different classes, an even number of k can cause a tie. Choosing an odd value for k prevents this problem (URL 8). The size of k value is important. Small and large values of k value have different characteristics. The small and large k values are compared in Table 2.1. Heuristic techniques such as cross-validation can help in selecting a good value for k. Ultimately, of course, the best value of k depends on the data at hand.

Table 2.1. Features of small and large values of k

Small values of k	**Large values of k**
Cause over-fit	Cause over-generalization
Increase negative effect of noise	Reduce negative effect of noise
Create distinct class boundaries	Create indistinct class boundaries

2.1.1.2. Distance functions

Different distance metrics can be used when calculating distances for the KNN algorithm. First of all, it is helpful to explain a general class of metric called as Minkowski metric which is given in Equation 2.2.

$$d(\vec{x},\vec{y}) = \left(\sum_{i=1}^{p} |x_i - y_i|^k \right)^{1/k} \tag{2.2}$$

where different values of $k \geq 1$ result in different commonly-used metrics. Here are the most common metrics used for calculating distances in KNN:

Euclidean: Explained as in Equation 2.1 in section 2.1.1. This is a special case of the Minkowski metric (Equation 2.2) where k = 2. According to Berrueta et al. (2007), this metric should be used when the different features are not strongly correlated.

Mahalanobis: Let the vectors $\vec{x}$ and $\vec{y}$ be two input samples of the same distribution with the covariance matrix P. The Mahalanobis distance between sample $\vec{x}$ and sample $\vec{y}$ is defined as

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T \Sigma^{-1} (\vec{x}-\vec{y})} \tag{2.3}$$

According to Berrueta et al. (2007), this metric should be used when the different features are strongly correlated. The covariance matrix Σ represents this correlation.

If the standard deviation of any of the features is 0, then the determinant of Σ is 0, and the inverse of Σ is undefined. Thus, in cases where one or more features of a

training set consist of a single value, its standard deviation is 0, and this distance metric cannot be used. When the covariance matrix Σ is equal to I (the identity matrix), the Mahalanobis distance becomes the similar Euclidean distance (URL 6). Figure 2.2 illustrates Euclidean and Mahalanobis distances on the same graph, in two-dimensional space.

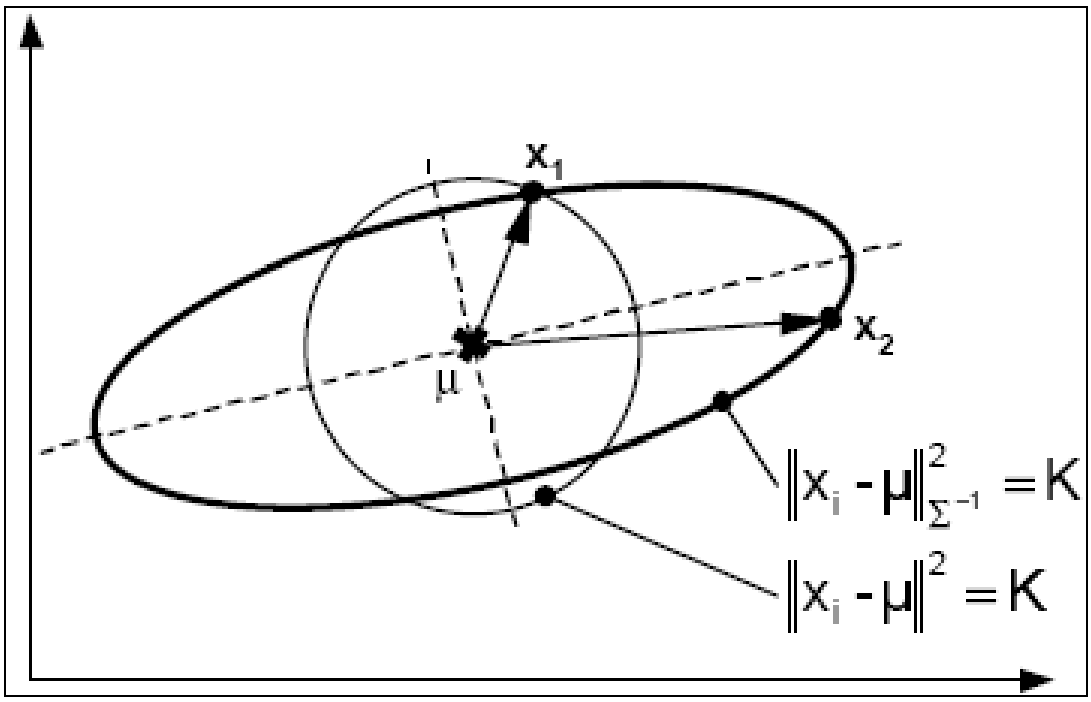

Figure 2.2. The graph of Euclidean(circle) and Mahalanobis distances (URL 6)

Diagonal (Class-Dependent) Mahalanobis: (Also called normalized Euclidean distance) Let the vectors $\vec{x}$ and $\vec{y}$ be two input samples of the same distribution, with p features. Let σ_i be the standard deviation of feature i. The Diagonal Mahalanobis distance between sample $\vec{x}$ and sample $\vec{y}$ is defined as in Equation 2.4.

$$d(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{p} \frac{(x_i - y_i)^2}{\sigma_i^2}} \tag{2.4}$$

As with the Euclidean distance, the correlation of different features is not taken into account here.

If the covariance matrix Σ in the Mahalanobis distance is diagonal (the features are not correlated), it reduces to the Diagonal Mahalanobis distance. As in the

Mahalanobis distance, this metric cannot be used if the standard deviation of any of the features of a training set is 0.

Manhattan: This is a special case of the Minkowski metric (Equation 2.2) where k = 1. Let the vectors $\vec{x}$ and $\vec{y}$ be two input samples (objects) with p features $(x_1, x_2, \ldots, x_p)$. The Manhattan distance between sample $\vec{x}$ and sample $\vec{y}$ is defined as

$$d(\vec{x}, \vec{y}) = |x_1 - y_1| + |x_2 - y_2| + \ldots + |x_p - y_p| \quad (2.5)$$

As with the Euclidean distance, the correlation of different features is not taken into account here.

2.1.1.3. Weight functions

After the k nearest neighbors of a test sample are found, these can be evaluated using different weighting methods. For each neighboring pixel, the pixel's weight is added to the total weight of that pixel's class. At the end, the class with the largest total weight wins.
The goal of weight functions is to cause distant neighbors to have less effect on the majority vote than the closer neighbors.
Here are the most common weight functions:

i. None: All neighbors have equal weight

ii. Fraction: Let i be the order of the neighbor in the list of k neighbors, i = 1..k. The weight function is 1/i. Therefore, the weight of the pixel is inversely proportional to its rank in the neighbor list. The fraction weights decrease steeply as the order(i) of nearest neighbor increases (Figure 2.3).

iii. Stairs: Let i be the order of the neighbor in the list of k neighbors, i = 1..k. The weight function is (k – i + 1) / k. Again, the weight of the pixel is inversely proportional to its rank in the neighbor list. The stairs weights slightly decrease as the order (i) of nearest neighbor increases (Figure 2.4).

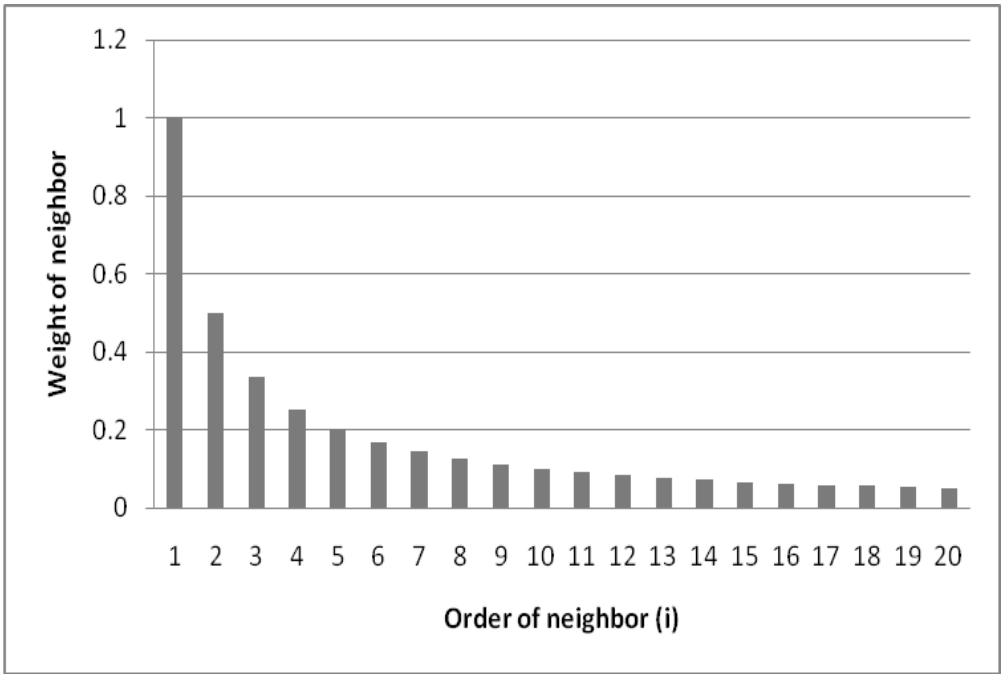

Figure 2.3. The Fraction weights for Weighted KNN

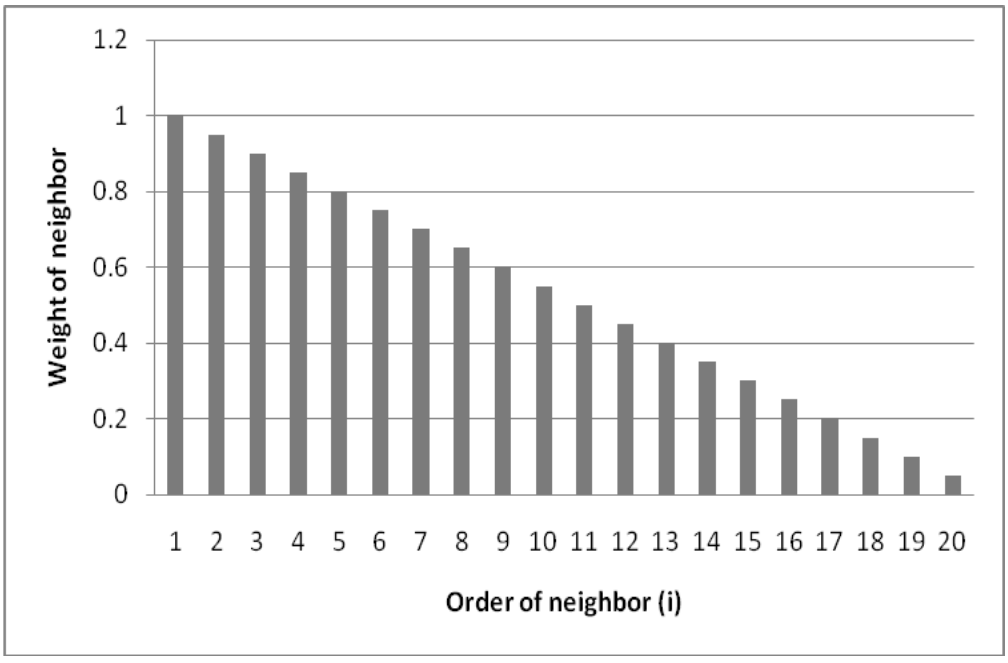

Figure 2.4. The Stairs weights for Weighted KNN

iv. InverseDistance: Let d be the distance of the neighbor from the test sample. The weight function is 1/d. Therefore, the weight of the pixel is inversely proportional to its distance from the test sample.

v. InverseSquareDistance: Let d be the distance of the neighbor from the test sample. The weight function is $1/d^2$. Again, the weight of the pixel is inversely proportional to its distance from the test sample.

2.1.2. Maximum Likelihood Algorithm

Although slightly complex, the maximum likelihood algorithm is related to many familiar estimation methods in statistics. Before detailing the algorithm, the definition of “distance” is necessary, which can be taken from the corresponding description under the KNN algorithm.

The maximum likelihood algorithm steps can be described as follows:

1. Training phase
    a) A human being classifies a number of objects manually. This is the training set. The feature vectors and class labels of these samples are stored.
    b) The computer reads in this set of objects. The correct classification for these objects is known.
    c) A probability model is picked for each class (uniform distribution, normal distribution, etc.).
    d) For each class, those probability model parameters are selected which make the class’s training data “more likely” than any other model parameters would make them. For example, if uniform prior distribution was picked, then its model parameters would be its most probable values.
2. Classification phase
    a) A new, unclassified input object (test sample) is classified by its distance to the parameterized probability model of each class.

- The probability that a test sample with value x belongs to class i is given by Bayes' theorem in Equation 2.6.

$$P(i \mid x) = \frac{P(x \mid i)P(i)}{P(x)} \tag{2.6}$$

where $P(x \mid i)$ is the probability that a sample in class i has the value x, $P(i)$ is the prior probability that a test sample belongs to class i, and $P(x)$ is the probability that a sample has the value x. $P(x)$ is the sum of $P(x \mid i)$ values for all i.

- Equation 2.6 is the decision rule. For each test sample x, $P(i \mid x)$ is calculated for all classes i. It is decided that x belongs to the class having the highest $P(i \mid x)$ (likelihood). Since $P(x)$ is the same for all classes, it can be ignored here.

Choosing a probability model:

If normal distribution is selected, then maximum likelihood estimation gives a unique solution. Let's take the normal distribution $N(\mu, \sigma^2)$ which has the probability density function defined as in Equation 2.7.

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \tag{2.7}$$

We can calculate the corresponding probability density function for *n* normal random variables that are correlated, using Equation 2.8. Let $\vec{x}$ be a vector of random variables $(x_1, \ldots, x_n)$, where each variable has means given by $\vec{\mu} = (\mu_1, \ldots, \mu_n)$. Also, Σ is the covariance matrix of the random variables. Then,

$$f(\vec{x}) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma)}} \exp\left(-\frac{(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})}{2} \right) \tag{2.8}$$

Here, $\vec{x}$ can be viewed as a test sample having *n* features. The parameters $\vec{\mu}$ and Σ are calculated once for the training samples of each class. Thus, the probability that a sample in class i has the value $\vec{x}$ can be calculated with Equation 2.8, where $\vec{\mu}$ and Σ are different for each class i. Inserting this into Equation 2.6, the probability that a test sample with value $\vec{x}$ belongs to class i is given by Equation 2.9.

$$P(i \mid \vec{x}) = f(\vec{x} \mid i)P(i) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma_i)}} \exp\left(-\frac{(\vec{x}-\vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x}-\vec{\mu}_i)}{2}\right)P(i) \qquad (2.9)$$

If the standard deviation of any of the features in class i is 0, then the determinant of Σ_i is 0, and the inverse of Σ_i is undefined. Thus, in cases where one or more features of a training set consist of a single value, its standard deviation is 0, and the maximum likelihood algorithm cannot be used.

2.1.3. Accuracy Assessment

The following sections detail the various assessment methods that were used to calculate the accuracy of the classification methods detailed in the previous chapter.

2.1.3.1. Leave One Out Cross Validation

In the field of supervised learning, it could occur that the algorithm at hand tunes itself too closely to the training data. This means that the algorithm does not gain strong abilities to predict the class of a new and unfamiliar test object correctly. Of course, we cannot know for sure how the algorithm is going to perform in a real-world situation, where unclassified objects arrive all the time. However, we can at least estimate the algorithm's future success by attempting to simulate this real-world situation artificially.

If we can expect that future data will be taken from the same distribution as the training data, we can use a method called cross-validation to estimate the accuracy of the algorithm's future predictions. We can apply this validation method to a range of different classification algorithms, and obtain cross-validation error values

for each one. Then, we can compare these values to find the lowest error rate and so the best classification algorithm, having the highest “generalization ability.”

The cross-validation algorithm steps can be described as follows:

1. The available data is divided into *k* disjoint sets.
2. The classification algorithm is executed *k* times. For each execution:
    a) A partition is selected out of these partitions.
    b) The algorithm is trained with k – 1 partitions. The training data excludes the selected partition.
    c) The algorithm is tested on the selected partition. That is, the algorithm, not knowing the actual class of the selected partition, attempts to classify it.
    d) The performance statistic is evaluated for this trained model.
3. The mean of the performance statistic of k trained models is calculated. This is the *k*-fold cross-validation estimate.

Advantages:

- Normally, for split-sample training, one would use training, validation and test partitions. These would represent the real distribution of objects for each class. Obviously, these partitions require a high amount of data. Sometimes, however, this data is not available. Cross-validation is especially useful here. Since each sample object is used as both training data and test data, cross-validation uses the available data very conservatively.

If k is equal to the number of training objects, the algorithm is called “Leave One Out cross validation.” This particular case has been the focus of many studies. Here is a simplified algorithm for this validation method:

For *i* = 1 to *k* (where *k* is the number of training set objects)

- Temporarily remove the i^{th} object from the training set.
- Train the learning algorithm on the remaining *k* - 1 points.
- Test the removed object against the trained algorithm and note your error.

- Calculate the mean error over all *k* objects.

The advantages of Leave One Out accuracy assessment can be listed as follows:

- Well-suited for selecting a classification model, by giving an almost unbiased estimate of its generalization ability.
- Uses almost all available objects when training. The resulting classification model is virtually the same as it would have been if all objects had been used.
- Does not waste data.

It costs a lot of computing power, since the algorithm must be repeated for every single training set object. Therefore, this can be considered as disadvantage of the method.

The error mentioned above is usually measured in terms of Root Mean Square Error (RMSE) for continuous variables. On the other hand, for class variables (such as forest/non-forest/water classification), the error rate is calculated using the following formula defined as in Equation 2.10.

$$Err = \sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{n} \tag{2.10}$$

where $\hat{y}$ is the predicted value, y is the actual value, and n is the number of classifications made. For each classification, if y and $\hat{y}$ belong to the same class (i), then the difference is 0. Otherwise, the difference is 1. Then, overall accuracy is defined using the following formula defined as in Franco-Lopez et al. (2001) in Equation 2.11.

$$OA = 1 - Err \tag{2.11}$$

2.1.3.2 Error Matrix

An error matrix is a very effective way to represent map accuracy in that the individual accuracies of each category are plainly described along with both the errors of inclusion (commission errors) and errors of exclusion (omission errors) present in classification. A commision error is simply defined as including an area into category when it does not belong to that category. An omission error is excluding that area from the category in which it truly does belong. Every error is an omission from the correct category and a commission to a wrong category (Congalton and Green 1999).

In addition to clearly showing errors of omission and comission, the error matrix can be used to compute other accuracy measures, such as overall accuracy, producer's accuracy and user's accuracy (Story and Congalton 1986, in Congalton and Green 1999).
Additionally, the error matrix is helpful for calculating other accuracy metrics. Three of these are described below:

- Overall Accuracy: Indicates how well the map identifies all land cover types on the ground and is defined as in Equation 2.14.

- Producer's Accuracy: Indicates what percentage of the time a particular land cover type on the ground was identified as that land cover type on the map. It expresses how well the map producer identified a land cover type on the map from the satellite imagery data and is defined as in Equation 2.15.

- User's Accuracy: Indicates what percentage of the time a particular land cover type on the map is really that land cover type on the ground. It expresses how well a person using the map will find that land cover type on the ground and is defined as in Equation 2.16.

The calculation of these three different types of accuracy is rather simple. But before detailing them, some definitions are necessary:

- There are k categories and n sample objects.
- n sample objects are distributed into k^2 cells.
- Each sample object is assigned to one of k categories in the remotely sensed classification (usually the rows) and, independently, to one of the same k categories in the reference data set (usually the columns).
- n_{ij} denotes the number of samples classified into category $i\left(i=1,2,\ldots,k\right)$ in the remotely sensed classification and category $j\left(j=1,2,\ldots,k\right)$ in the reference data set.
- p_{ij} denotes the proportion of samples in the i,j th cell, corresponding to n_{ij}. In other words, $p_{ij}=n_{ij}/n$.

$$n_{+j}=\sum_{i=1}^{k}n_{ij} \tag{2.12}$$

$$n_{i+}=\sum_{j=1}^{k}n_{ij} \tag{2.13}$$

$$\text{Overall accuracy}=\frac{\sum_{i=1}^{k}n_{ii}}{n} \tag{2.14}$$

$$\text{Producer's accuracy}=\frac{n_{ii}}{n_{+j}} \tag{2.15}$$

$$\text{User's accuracy}=\frac{n_{ii}}{n_{i+}} \tag{2.16}$$

$$p_{+j}=\sum_{i=1}^{k}p_{ij} \tag{2.17}$$

$$p_{i+}=\sum_{j=1}^{k}p_{ij} \tag{2.18}$$

The number of samples which are used in error matrix is important for accuracy assessment. Consider a population of units divided into k mutually exclusive and exhaustive categories. (Congalton and Green 1999). The calculation of the sample size is as follows;

$$N = B * \Pi i * \frac{(1 - \Pi i)}{b_i{}^2} \quad (2.19)$$

N= Sample size

i= categories 1 to k

Π = the proportion of the population in the i^{th} category

B= upper (α/k)*100^{th} percentile of the X^2 distribution with degree of freedom.

b= desired precision

2.1.3.3 Kappa Analysis

The Kappa analysis is a discrete multivariate technique used in accuracy assessment for statistically determining if one error matrix is significantly different than another (Bishop et al. 1975, in Congalton and Green 1999). The result of performing a Kappa analysis is a KHAT statistic (an estimate of Kappa), which is another measure of agreement or accuracy (Cohen 1960, in Congalton and Green 1999). This measure of agreement is based on the difference between the actual agreement in the error matrix (i.e., the agreement between the remotely sensed classification and the reference data as indicated by the major diagonal) and the chance agreement which is indicated by the row and column totals (Congalton and Green 1999).

The following equations are used for computing the KHAT statistic and its variance.

$$p_0 = \sum_{i=1}^{k} p_{ii} \quad (2.20)$$

$$p_c = \sum_{i=1}^{k} p_{i+} p_{+j} \quad (2.21)$$

$$\hat{K} = \frac{p_0 - p_c}{1 - p_c} \quad (2.22)$$

p_{ii} denotes the proportion of samples in the i,i th cell, corresponding to n_{ii}. In other words, $p_{ii} = n_{ii}/n$. p_{i+} and p_{+j} are defined in Equation 2.17 and Equation 2.18, respectively.

2.2. Practical View on KNN on Forest Areas

In this section, the previous studies about KNN classification in remote sensing analysis are given. There are numerous references about KNN supervised classification method and accuracy assessment. The literature about KNN classification on forest areas is given here and details on the steps of the flowchart is presented in Figure 2.5.

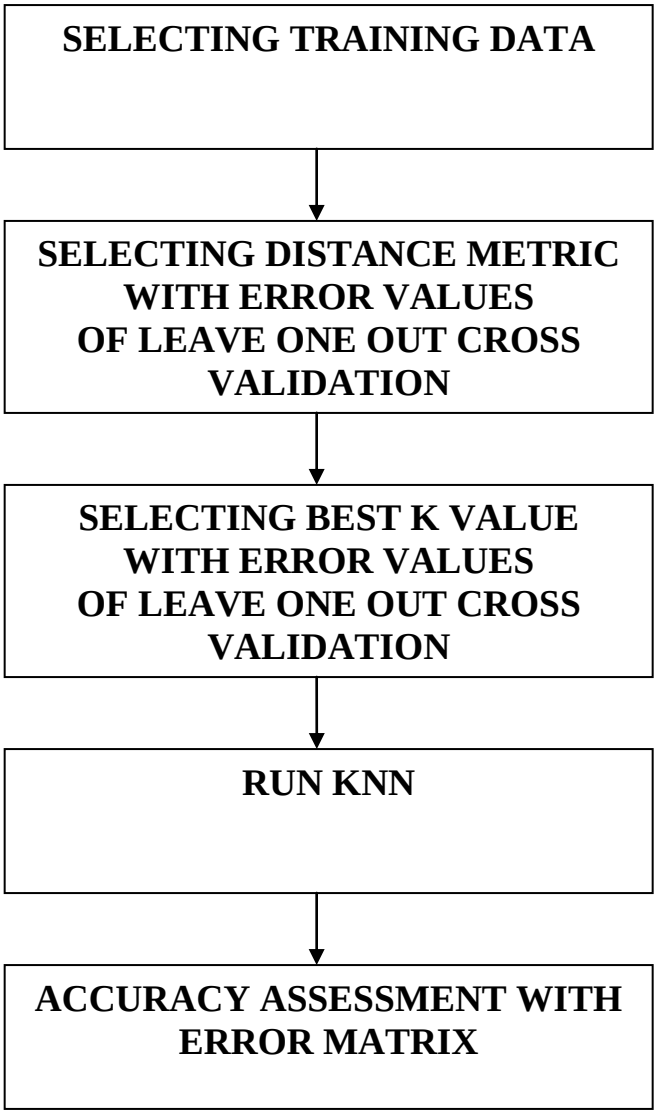

Figure 2.5. Flowchart of KNN Supervised Classification steps.

2.2.1. KNN usage in the world forestry

Practical success of the KNN method is described by Gjertsen (2007). It is being used as a part of the National Forest Inventory (NFI) in Finland for 10 years. The NFI reference data is based on sample plots. These are explained in Gjertsen (2007). To sample a plot, a predefined spot in a forest area is visited. Then, forest variables (such as tree species, age, basal area, volume, etc.) are measured inside a 250 m^2 circle centered in this spot. Using this process, a plot is sampled once every 3 km inside the forest, forming a 3 km by 3 km network. The resulting grid of plots makes up the reference data for forest variables. Before remote sensing was introduced, the plots were used to estimate forest variables and the location of forest resources, where each plot had equal weight. However, this system did not give reliable estimates. Multi-source forest inventory (MSFI) method was developed as a solution (Gjertsen 2007). In MSFI, remotely sensed data were integrated using the KNN algorithm as follows: Both ancillary data such as digital elevation models (DEMs) and NFI plot data were combined and used as reference data for KNN classification. For each pixel in the area to be estimated, the k nearest neighbors were found among the available NFI plots. The distance between a test pixel and a plot was calculated using the spectral values of the test pixel and of the pixel covering the plot. The plot weights depended on these distances.

When continuous land use data is available, then any number of sample pixels corresponding to the various land use classes can be selected as training data. These pixels are analogous to the plots collected in forest fieldwork, but they are much larger in number.

According to Gjertsen (2007), this KNN method was able to produce wall-to-wall maps showing the location of different forest resources. These maps are being used by the forest industry for timber procurement. Ecologists use these maps for habitat analyses. In Sweden, the method has been used to produce a complete map database for the whole country, called KNN Sweden. It has been used to improve forest statistics from the National Forest Inventory by using post stratification based on stem volume strata derived from the database (Gjertsen,

2007). It is reported that the standard errors for estimates of total stem volume, stem volume for spruce, stem volume for pine, and woody biomass have been reduced by 10% to 30% at the county level (Gjertsen, 2007).

In Gjertsen (2007), the KNN method has been tested for preharvest inventory of a forest plantation with pine trees in New Zealand. Estimates were made at pixel level and stand level. Cross validation tests showed that the estimates were unbiased but with high root mean square errors (RMSEs). The pixel-level accuracy was tested in Finland as well. It was found that the errors at pixel level for volume estimates were relatively high, with RMSE for total volume around 75 m^3/ha or between 62% and 68% of the mean estimated value. For volumes of species groups, the relative errors were even higher and above 100%. However, the bias was found to be low, and it was concluded that the accuracy would improve for large area estimates. In (Tomppo and Katila, 1991), KNN-based volume estimates for three municipalities were compared with independent surveys of the municipalities made for forest planning purposes. The estimates from the latter survey varied more from municipality to municipality than the KNN estimates.

According to Gjertsen (2007), a KNN -based method was tested based on Landsat TM data on a site in Germany. Area proportions of single tree species groups were estimated for forest stands. It was found that the KNN method improved the estimated values from 1.7% to 25.2% relative to estimates based on the mean values of the sample of reference plots. Reduction of RMSE was used as indicator of improvement. It was concluded that the method does not provide sufficient information for a forest management plan but that it provides a good overview of the spatial distribution of the main tree types.

Since 1990, optical area satellite images and digital maps, in addition to field plot data, have been used by the Finnish multisource National Forest Inventory (MSNFI) (Gjertsen, 2007). In Katila (2001), it is explained that a set of parameters are chosen for the KNN method in the operative MSNFI, such as:

(1) the image features;
(2) the distance measure;
(3) the value of k, i.e., the number of the nearest neighbors;

(4) parameters related to the possible use of digital elevation model;
(5) stratification of the image and field plots to mineral land and peatland on the basis of a digital site class map, produced by the National Land Survey (NLS);
(6) the geographical reference area from which the nearest field plots are selected. The geographical reference area is crucial for the estimation procedure and is selected separately for each pixel in the Finnish MS-NFI.

In Franco-Lopez et al. (2001), distances between neighbors were computed using two different distance metrics, Mahalanobis and Euclidean. RMSE values of the classified images were found for continuous variables, and overall accuracy values were found for class variables, calculating with Euclidean and Mahalanobis distances. There was a typical KNN classification result which was good for the first few k values. Even though there is a well-known correlation among TM band values, the use of Mahalanobis distance did not benefit the quality of the estimation in these trials (Franco-Lopez et al., 2001). RMSE values at least 5% smaller than those of Mahanolobis distance metric were obtained, for any number of neighbors, and it was then noted that this is contrary to the results reported by Nilsson, (1997), in Franco-Lopez et al. (2001). Usually, the Euclidean distance is used, but for strongly correlated variables, correlation-based measures, like the Mahalanobis distance, are preferred (Berrueta et al., 2007).

2.2.2. KNN and its features

In Berrueta et al. (2007), it is explained that nearest neighbor methods are based on the determination of the distances between an unknown object and each of the objects of the training set. Usually, the Euclidean distance is used, but for strongly correlated variables, correlation-based measures are preferred. Then, the lowest distance is selected for the assignment of the class membership. It is described that in KNN, the k-nearest objects to the unknown sample are selected and a majority rule is applied: the unknown is classified in the group to which the majority of the k objects belong. The choice of k is optimized by calculating the prediction ability with different k values. It is claimed by Berrueta et al. (2007) that frequently,

small k values (3 or 5) should be preferred. It has been suggested to choose k near sqrt(n_i) for a typical n_i, the number of plots in class i (Gjersten, 2007). However, this is more applicable in estimating continuous forest variables in the presence of a low number of sample plots. For class variables, it can happen that a much larger number of samples are available, since numerous sample pixels can be selected based on large-scale land use data. In such cases, sqrt(n_i) becomes too large to be useful.

The KNN method is a nonparametric classifier in which there are no assumptions about the distributions of the variables involved in the classification (Franco-Lopez et al., 2001). In Franco-Lopez et al. (2001), it is explained that all the digital number information of pixels for all training classes are obtained and the unlabeled pixel is classified among the closer neighboring training pixels. There is a summary of a substantial body of literature regarding the statistical characteristics of nearest-neighbor rules and the statement ‘‘when the proportion of pixels in each training class is identical to the actual proportion of each class in the population, the KNN rule is a maximum-likelihood classifier”. According to Mc Roberts and Tomppo (2006), both parametric and non-parametric estimation methods have been tested and applied in forest inventory applications. The keen interest in the non-parametric KNN method is partly motivated by the desire to estimate simultaneously the large number of variables of interest.

The K-Nearest Neighbor (KNN) method can be used in a wide range of estimation and classification applications. In the past decade, the KNN method has been advanced for estimation of forest variables and is now operational in Finland’s national forest inventory (Haapanen et al., 2004).

In Gjertsen (2007), it is summarized that the main reason to use the KNN method is that it is very flexible. A highlighted fact is that KNN produces statistical estimates and wall-to-wall maps of the inventory area at the same time. In particular, to produce wall-to-wall maps, many other methods could be used, including parametric methods such as the maximum likelihood classifier. However, there are references to other studies which have demonstrated that the KNN method performs well in comparison to other methods. Some examples are given

by Gjertsen (2007): KNN, maximum likelihood, and several other classifiers were tested using Landsat TM data and very small differences in the classifiers' performance were found. Also, KNN has been compared with artificial neural network (ANN) and traditional statistical classifiers. There, it was concluded that KNN performed as well as ANN classifiers and better than the traditional statistical classifiers.

In Gjertsen (2007), a very common drawback of the KNN method in practice was detailed as well. The KNN method makes no assumptions on the distribution of pixels in feature space as a function of forest variables. It is noted that for every forest variable value (e.g. for every forest cover type), a sufficient number of plot pixels must be available in order to find similar ones. These plot pixels must lie on the same image as the unknown pixel, otherwise the spectral similarity is disturbed by external factors such as sun elevation and atmospheric conditions. It is mentioned that this problem particularly occurs in countries that are elongated in east–west direction. The reason for this is those scenes from neighboring image acquisition paths are usually acquired on different dates. The same concerns apply to forest/non-forest/water stratification as well. Here, a sufficient number of training data pixels must be available, which must lie on the same image as the unknown pixel. This is usually not a problem when selecting training pixels based on large-scale land use data.

Another explanation of the KNN algorithm is given by Katila and Tomppo (2001): KNN searches the feature space for the k nearest pixels, whose field data vectors are known, applying a distance measure, d, defined in the feature space. Then, field data from the k nearest pixels is transferred to the unknown pixel. The method has been widely studied in pattern recognition and statistics (Katila and Tomppo, 2001). The importance in choosing an appropriate value of k is explained in Katila and Tomppo (2001): The KNN estimator may give biased estimates as the value of k increases, but the bias can be reduced with weighted averages of the k neighbors. The error rate asymptotically approaches the optimal rate of the Bayes decision rule for discrete variables when both the k and n (number of observations) tend to infinity in such a way that $k/n \rightarrow 0$ (Katila and Tomppo, 2001).

According to Berrueta et al. (2007), the KNN method has the following advantages:

(i) It has mathematical simplicity. Still, KNN achieves classification results as good as (or even better than) other more complex pattern recognition techniques.

(ii) It does not make any prior statistical assumptions, such as the normal distribution of the variables.

(iii) Its effectiveness does not depend on the space distribution of the classes.

In Franco-Lopez et al. (2001), the KNN method was tested for propagating forest stand density, volume, and cover type through the landscape, and it was found to be very promising. The method could be easily integrated within the procedures of existing forest monitoring systems. An important difference between the KNN and traditional classification and estimation techniques was pointed out: The KNN method is a form of poststratification constrained to the range of plot values of the inventory. In effect, after field plots are taken, they comprise strata with associated variable values. These values are then assigned to the remaining nonselected plot locations according to the similarity of certain features among the sampled and nonsampled plots. As an example that is given, a mature pine plot and its variable values are distributed (assigned) across the landscape to nonsampled locations that are determined to be similar in some sense. On the other hand, traditional classification attempts to establish strata according to the inventory plots they may contain. Thus, the KNN retains the full set of inventory specifications and values, while traditional classification typically does not (Franco-Lopez et al., 2001).

2.2.3. Distance-weighted k-nearest-neighbor (DW- KNN) rule

According to Berrueta et al. (2007), KNN cannot work well if there are large differences in the number of samples in each class. In such cases, when using equal weights, highly populated classes tend to dominate other classes in the list of neighbors. To solve this problem, an alternative criterion is suggested instead of a simple majority criterion. For instance, another choice of criterion in KNN consists of weighing the importance as a neighbor of a known object to an unknown sample (inverse distance or inverse square distance). This will cause the nearest

neighbors to influence the classification more than the farthest ones. Three drawbacks to the KNN method are mentioned:

- It provides poor information about the structure of the classes and of the relative importance of each variable in the classification.
- It does not allow a graphical representation of the results,
- In the case of large number of samples, the computation can become excessively slow.

In Dudani (1976), in Yang and Chou (2005), the distance weighted-KNN (DW-KNN) rule is proposed as a modification of the KNN rule. It is suggested that training samples closest to the test sample should be given more weight than training samples that are more distant. In general, the neighborhood of the DW-KNN model is comparable to the KNN model, but its decision criterion is different. In applying the DW-KNN rule, an unlabeled sample is assigned to the class producing the highest total weight among its reference neighbors (Yang and Chou, 2005).

A similar explanation of the distance-weighted k-nearest-neighbor algorithm, which is a refinement of the original k-nearest neighbor algorithm, is given by Dudani (1976), in Tsiriga and Virvouin (2003): In general, nearest neighbor learning algorithms typically store all of the n available training examples during learning. These algorithms use a distance function to determine how close a new query instance is to each stored instance, and use the nearest instance or instances to classify the query instance (Tsiriga and Virvouin, 2003). In other words, the basic idea of the distance weighted k-nearest neighbor algorithm is to weigh the contribution of each of the k neighbors according to their distance to the query point, thereby giving greater weight to neighbors that are closer than neighbors that are farther (Tsiriga and Virvouin, 2003)

A summary of the main decisions that must be made when applying a distance weighted nearest neighbor algorithm is given by Tsiriga and Virvouin (2003):

1. Select the features that would be used to formulate the input space of the distance function.
2. Identify a distance function to estimate the similarity between two instances.

3. Define the number of neighbors (k) that would participate in the classification task.
4. Design a function to classify new instances.

In Dudani's paper, several weighting functions are listed (Yang and Chou, 2005). One of them was found meaningful and original by Yang and Chou (2005): A function tied to the weight to the inverted distance from the unlabeled sample to the reference neighbors. If the distance between the unlabeled sample and a certain reference neighbor was shorter than the distances from the other reference neighbors, it gave the largest weight and asserted the strongest relation among all of the reference neighbors. Also, an extended, more comprehensive version was used, where a weighting factor inversely proportional to a w-powered Mahalanobis distance from the unlabeled sample was utilized.

In Gjertsen et al. (2007), it is mentioned that the National Forest Inventory plots are typically located outside the inventory area; however, using ancillary data, the method calculates the representativeness of the external National Forest Inventory plots in the form of new area weights. The development in Finland has inspired a similar development of the Norwegian National Forest Inventory. According to Gjertsen et al (2007), the KNN method is very attractive because it works in a manner familiar to the National Forest Inventory by making estimates based on sample plots with associated area weights. It is explained that the basic difference is that in NFI, all plots have the same area weights, while in KNN, the plots receive different area weights according to how similar they are to the pixels of the inventory area. Similarity is not based on forest variables, but rather on the vector of spectral values from the image pixel covering the plot.

The KNN method for forest/nonforest/water stratification and its ultimate application in developing forest area estimates for the USDA Forest Service's Forest Inventory and Analysis (FIA) program was studied by Haapanen et al. (2004). The method couples field-based inventory and satellite imagery data to produce continuous digital layers of measured forest or land use attributes. The KNN algorithm assigns each unknown (target) pixel the field attributes of the most similar reference pixel(s) for which field data exists. The similarity is defined in terms of the feature

space (e.g., Euclidean distance in spectral space). Attributes of interest are imputed to target pixels by calculating a weighted average of measurements of each of the (k) reference pixels. Class variables such as cover type or land use are estimated as a weighted mode. These weights can be applied as some function of spectral distance between each target and reference pixel. Because forest attributes are imputed based solely on spectral similarity, the method can be used to simultaneously impute all field-measured attributes to target pixels.

In Franco-Lopez et al. (2001), it is mentioned that Hardin (1994) compared the performance of parametric and nonparametric classifiers, particularly nearest neighbor rules. This study concluded that the neighborhood-based classifiers, in particular, the distance weighted neighbor classifier, are superior to the best parametric classifiers (such as the maximum likelihood classifier) when the training sets are large and contain the same class proportions as the population to be classified. When this condition is severely violated, there was not a clear advantage in using the KNN algorithm.

2.2.4. Accuracy Assessment in KNN algorithm applications

The accuracy of a classification algorithm in predicting the correct class for a pixel is the most important measure of its success, or its “performance.” Accuracy can only be measured for images where the land use data is available, which allows comparing the estimated class with the real class.

In general, accuracy assessment is done with the help of traditional error matrix and various cross validation methods such as Leave One Out, k-fold. Most widely used cross validation method is Leave One Out. This method is based on extracting a single observation from the original sample as the validation data, and the remaining observations as the training data.

The evaluation of KNN’s utility for forest/nonforest/water stratification was studied by Haapanen et al., (2003). The errors were estimated by Leave One Out cross validation. For each omission (Haapanen et al, 2004), the KNN prediction rule was applied to the remaining sample. Subsequently, the errors from these predictions

were summarized. In total, the prediction rule was applied n times and predicted the outcome for n units. Such estimates of prediction error are nearly unbiased.

Ways to develop statistical methods to integrate real coarse scale variation of forest variables into KNN estimation were studied by Tomppo and Halme (2004). Also, work was done on selection of the neighbors with the minimum bias and RMSE and to develop methods to estimate a weight vector for the feature vector applied in the KNN estimation. The parameters were selected which are based on pixel level validation of the KNN predictions using Leave One Out and jackknifing.

The estimation of forest stand volumes by means of satellite image data and stand-level field data was investigated by Makela and Pekkarinen (2004). In order to determine the appropriate value of k for the KNN estimation, different values of k were tested with employing the cross-validation (Leave One Out) technique. After the selection of parameter k and the best features, the estimations of the volumes for actual forest were observed with the help of existing ones. Also, the results of the actual forest stand volumes were compared with the estimated ones.

In Thessler et. al (2007), KNN and discriminant analyses to classify rain forest types in a Landsat TM image over northern Costa Rica were studied. It was emphasized that the level of forest classification accuracy from a given satellite sensor's data depends on the classification algorithm and the resolution (pixel window or segment size) applied in the process. Leave One Out cross validation was used in the accuracy assessment of both classification methods.

CHAPTER 3

MATERIALS AND METHODOLOGY

In this chapter, the case study area, the preparation of the input data required for the K nearest neighbor (KNN) classification and the methodology used in this study are described.

3.1. Definition of the Study Area

The study area is in the city of Antalya, Turkey. This area lies in the rectangle between the $30^0$31'E - $37^0$01' N, $30^0$53' E -$36^0$49' N latitudes & longitudes (Figure 3.1). Antalya has fifteen districts: Center district, Akseki, Alanya, Elmalı, Finike, Gazipaşa, Gündoğmuş, İbradi, Kale, Kaş, Kemer, Korkuteli, Kumluca, Manavgat, and Serik.

For the last decade, Antalya has been the fastest growing metropolitan city in Turkey (Sevik, 2006). This information shows that due to the very high speed of urbanization, the forest areas in Antalya may be in danger of decline. Therefore, the region was a natural candidate for this study. In addition, it was advantageous that satellite data of this region from different years were readily available.

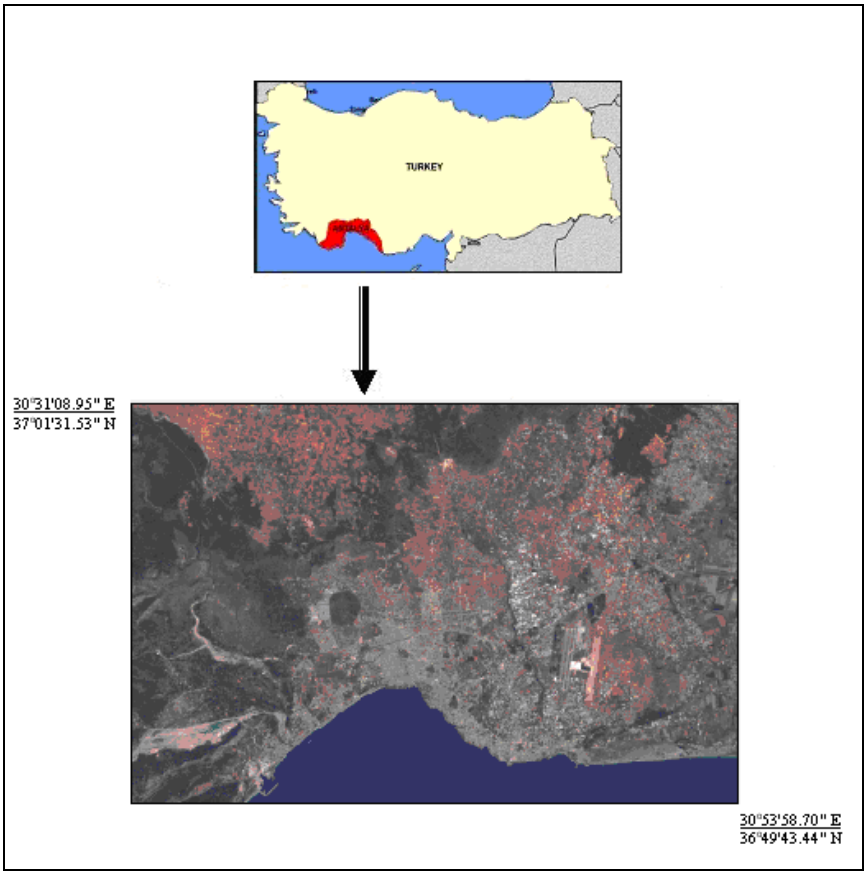

Figure 3.1. Location of the Study Area

3.2. Data Used in the Study

1987 and 2002 satellite images were acquired in the summer time namely June and August (Sevik, 2006). The year of 1987 satellite image Thematic Mapper (TM) Landsat 30 meter image with 7 bands) and the year of 2002 satellite image Enhanced Thematic Mapper (ETM) Landsat 30 meter image with 8 bands) were used in this study. Then they were projected by using projection system of UTM WGS 84, Zone 36. In this study ETM (RGB; 432) (Figure 3.2), ETM (RGB; 321) (Figure 3.3), TM (RGB; 432) (Figure 3.4) band composites are used for comparison of classification results. The year of 2003 landuse map (Sevik, 2006) (Figure 3.5) and SRTM 90 meters DEM (Digital Elevation Model) data (Figure 3.6) were used as ancillary data. They were also projected into the same projection system. No atmospheric conditions were available for these satellite images.

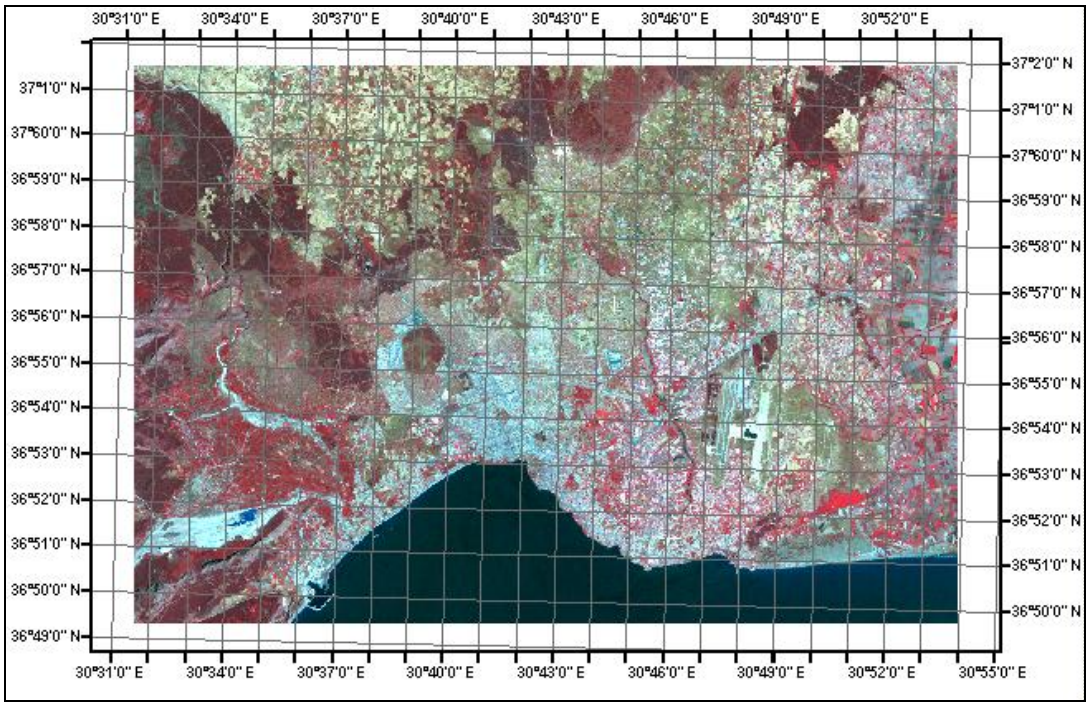

Figure 3.2. 2002 Landsat ETM satellite imagery of the study area with 30-meter spatial resolution (RGB; 432).

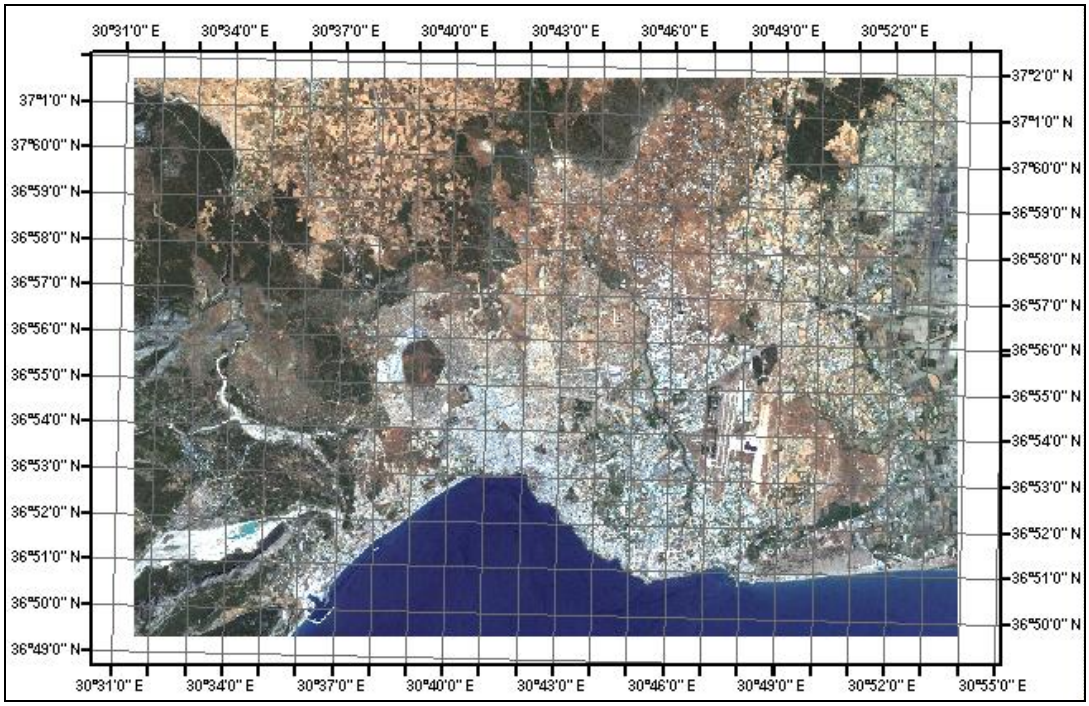

Figure 3.3. 2002 Landsat ETM satellite imagery of the study area with 30-meter spatial resolution (RGB; 321).

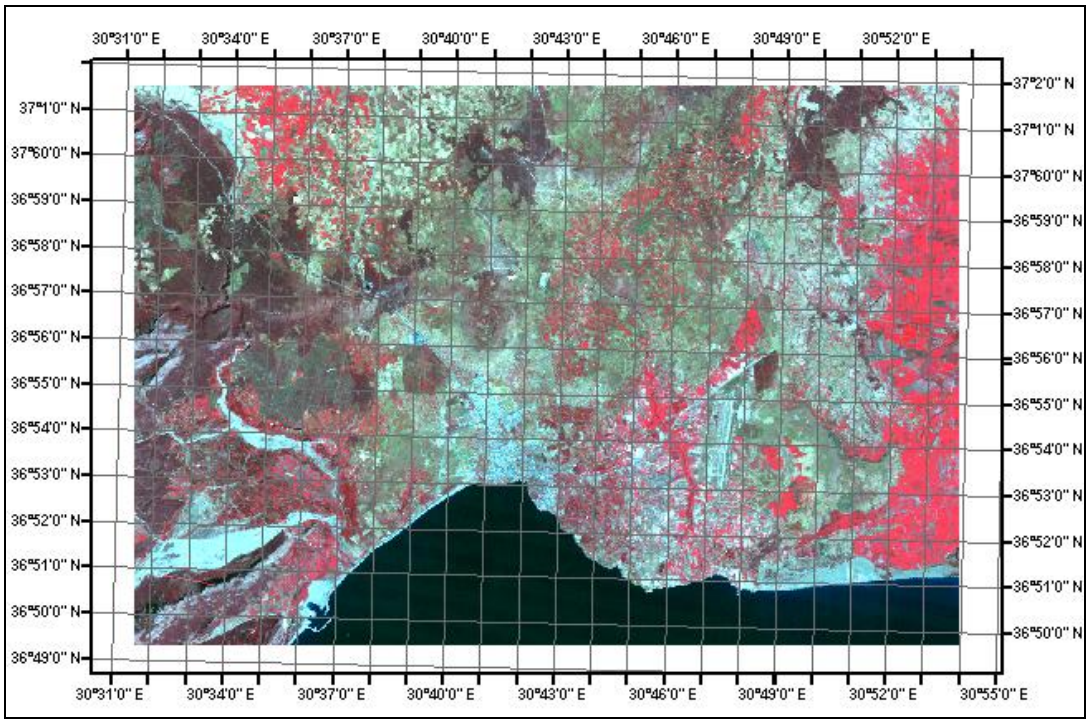

Figure 3.4. 1987 Landsat TM satellite imagery of the study area with 30 meter spatial resolution (RGB; 432).

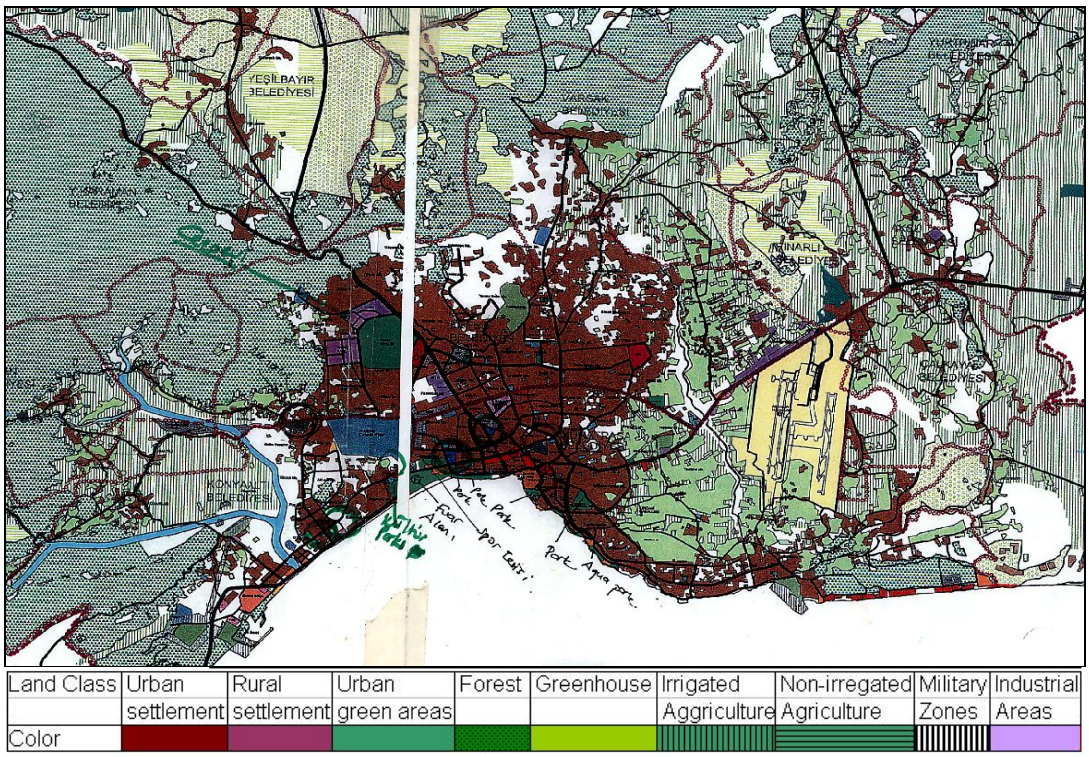

Figure 3.5. 2003 Landuse Data of the study area

Figure 3.6. SRTM 90meters DEM Data of the study area

3.3. Data normalization

Satellite images from two different dates of the same area can have completely different spectral values, due to atmospheric conditions and illumination geometry. In order to be able to classify two different satellite images with the same training data and perform change detection, they must undergo radiometric correction.

The Landsat ETM and Landsat TM satellites have different sensor settings, although they produce images with the same resolution. This causes the 2002 ETM image to have a different color distribution than the 1987 TM image. In addition, atmospheric conditions that are present when the satellite image is being taken cause strange effects on the resulting image. Because of these reasons, two different satellite images of the same location taken at different times always have different color distributions.

Supervised classification methods require training data which is a small representative set out of the data that needs to be classified. In satellite images, one can create the training data by extracting pixels whose classes are already known. This training data can then be used to classify the satellite image that it

was extracted from. Ideally, one set of training data should be sufficient to classify any satellite image. This should minimize manual intervention in the classification process. However, because of the reasons mentioned above, each satellite image has its own color distribution. Therefore, each satellite image can be classified with its own training data only.This was exactly the case with 2002 ETM and 1987 TM images. This behavior is not desired since it is impractical to do such manual work with every image. It would be much better if one set of training data could be used for all images.

This can be achieved by equalizing satellite images. There are different methods for this:

1) Atmospheric correction
2) Radiometric correction.

1) In order to perform atmospheric correction, the exact atmospheric conditions at the time of the satellite image snapshot must be known. Then, using tools such as ERDAS IMAGINE and PCI Geomatics, these variables can be input to perform atmospheric correction for all satellite images at hand. As a result, all satellite images have the same color distribution, and can be classified with the same set of training data. However, most of the time, the atmospheric conditions are not available for a given satellite image. This was the case with our ETM and TM images. In such cases, radiometric correction must be applied.

2) One of the radiometric correction techniques is histogram matching explained in (URL3). If two different satellite images have different histograms, they are corrected to have the same histogram. Normally, there is a reference image that is not changed. All other images are modified so that their histograms match that of the reference image. In Nelson et al. (2005), it is mentioned that distribution-based relative radiometric correction techniques, such as histogram matching, eliminate the problem of subjectivity and reduce the dependence on a geometrically accurate spatial match between multi-date images through their use of the entire dataset. In this study, histogram matching is applied as radiometric correction technique.The histgorams of Landsat TM before and after histogram

matching can be seen in Figure 3.7. Landsat ETM histogram can be seen in Figure 3.8. The scaled vertical axis shows the amount of histogram in the view.

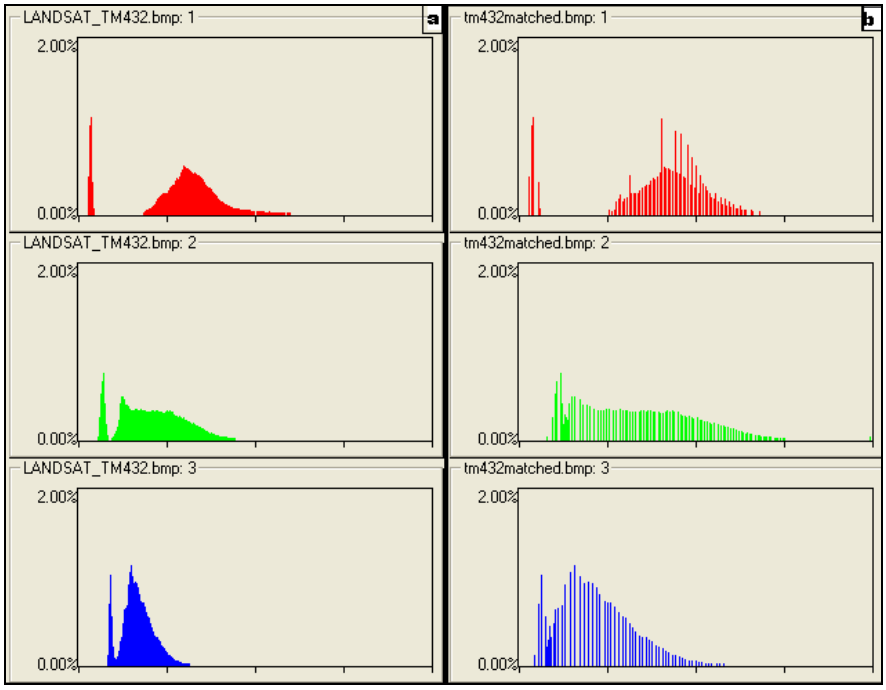

Figure 3.7. Histograms of Landsat TM before (a) and after (b) the histogram matching

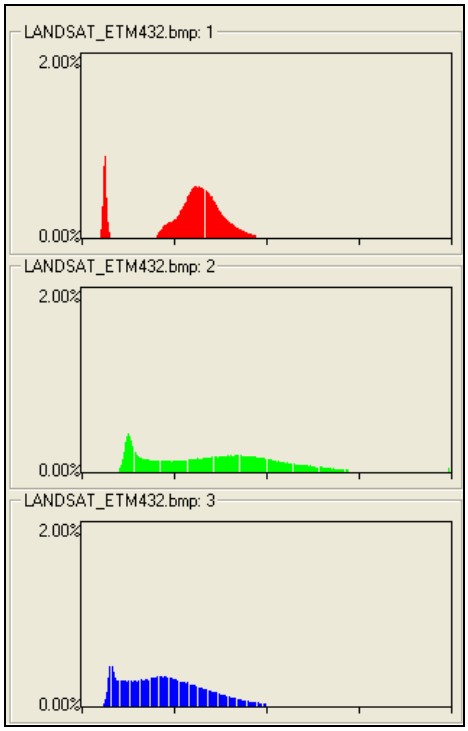

Figure 3.8. Histogram of Landsat ETM

3.4. Land Use Classes

According to the official land use data which is obtained from (Sevik, 2006), the area of interest (Center District) has mainly the following classes (Figure 3.5) :

- Urban settlement
- Rural settlement
- Urban green areas
- Forest
- Greenhouses
- Water agriculture
- Dry agriculture
- Military zones
- Industrial area
- Commercial areas
- Airport

There is also a portion of the Antalya Bay (Mediterranean Sea) inside this area.
Since this study focuses on measuring forest areas with classification algorithms, none of the specific non-forest classes were of interest. All of these were viewed as a single class, “Non-forest.” In addition to “Forest” and “Non-forest,” a third class, “Water” was also included in the study. The reasons for this are the very distinctive pixel values of the sea and the lakes, and also the ability to identify the city coastline more easily in the classified image.

3.5. Training Data Selection

Three sets of training data were prepared, for each of the classes Forest, Non-forest and Water (Figure 3.9). In Haapanen (2004), it is emphasized that the training data should capture the range of spectral variability within a class, in order to obtain good class estimates with the KNN method. So, in the process, rectangular areas were selected out of specific areas of the ETM satellite image which were believed to be representative of the corresponding class. Since the non-forest class is actually a collection of many different land use classes. In this study, nonforest training data were collected from the corresponding land use

classes except rural settlements and industrial areas, whose pixels are very similar to urban settlement pixels. The size of the training data was about 1% (approximately ten thousand pixels) of the ETM image (1170x793 pixels) which has to be classified. The Quantile-Quantile plots are used to check the normality of training data. It shows sample quantiles of training samples versus theoretical quantiles from a normal distribution. If the distribution of samples is normal, the plot will be close to linear. The QQ plots of forest, nonforest and water training samples can be seen in this section (Figure 3.10-3.18). It can be seen that some training pixels have deviations from the line. Therefore, if training pixels had been normally distributed exactly, then Maximum Likelihood classification would have better accuracy.

The training data was selected out of the 2002 ETM image since official land use data was only available for the year 2002. By using image normalization, it was possible to use this same training data to classify the 1987 TM image. A striping (banding) problem was discovered in the 1987 TM image, which was most obvious in the water (sea) areas. The striping effect is caused by a miscalibration in the satellite sensor. So the sea areas in the TM image had abnormal pixel values. In order to detect the sea areas correctly, a part of this abnormal sea was added to the water training data during the TM classification. The training data was subjected to Leave One Out accuracy assessment, before and after the consideration of this abnormal sea area. It was found that the addition of this area to the training data affected the accuracy by a negligible amount.

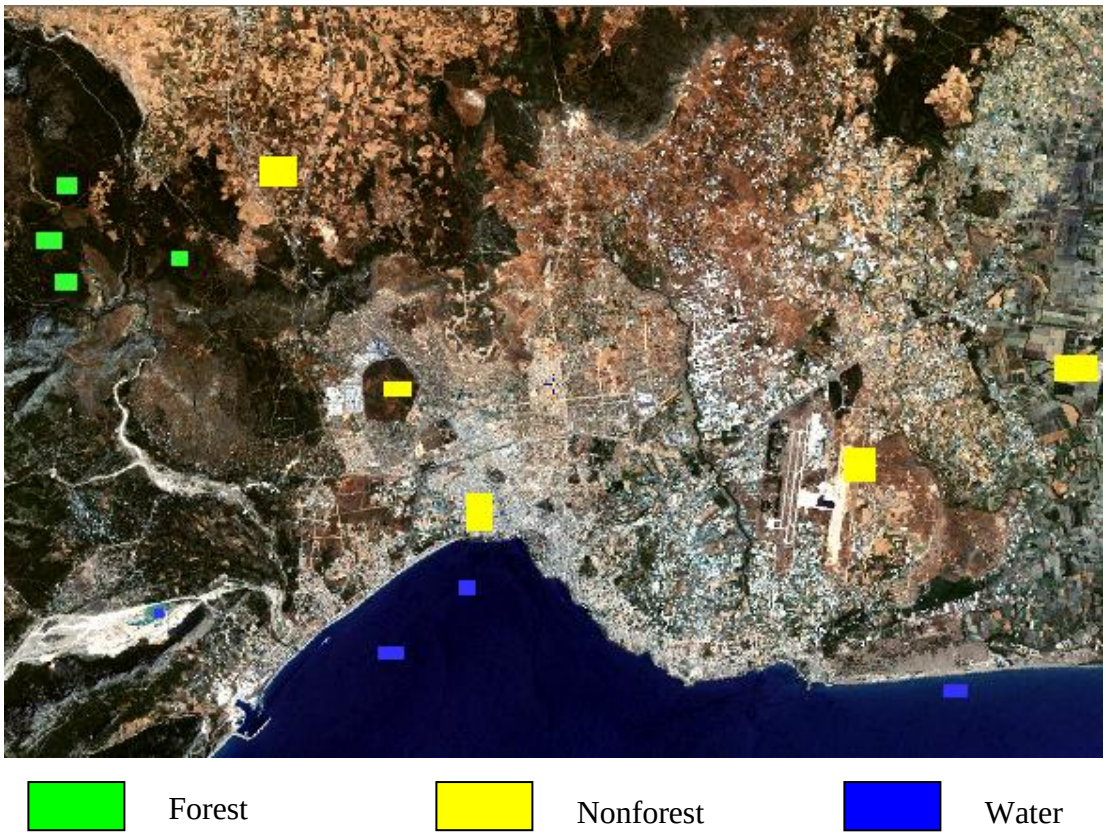

Figure 3.9. Training data selected for this study

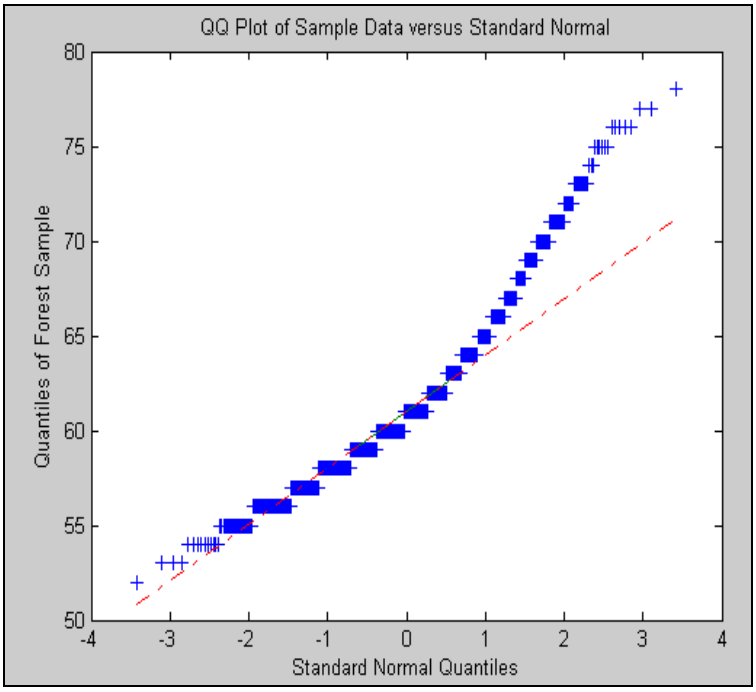

Figure 3.10. Forest training data 4th band QQ plot

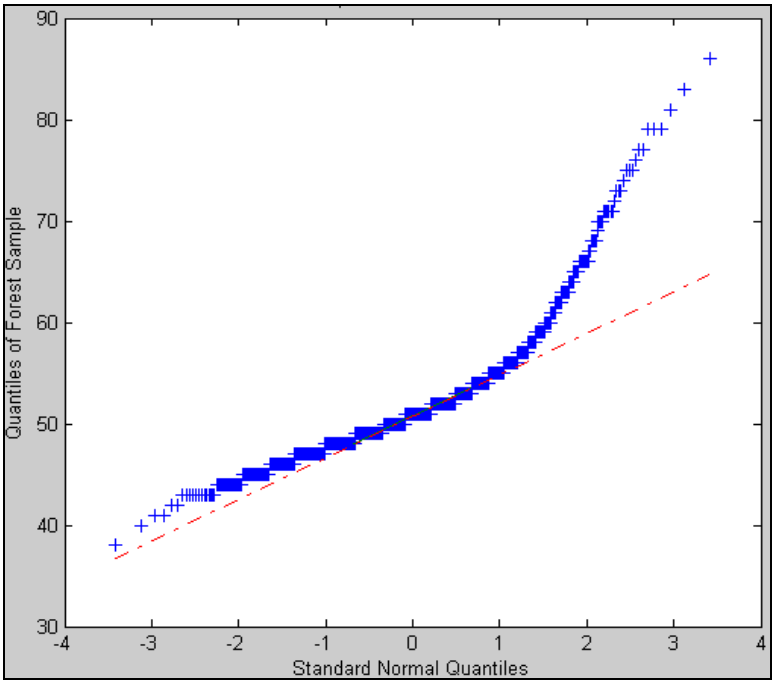

Figure 3.11. Forest training data 3rd band QQ plot

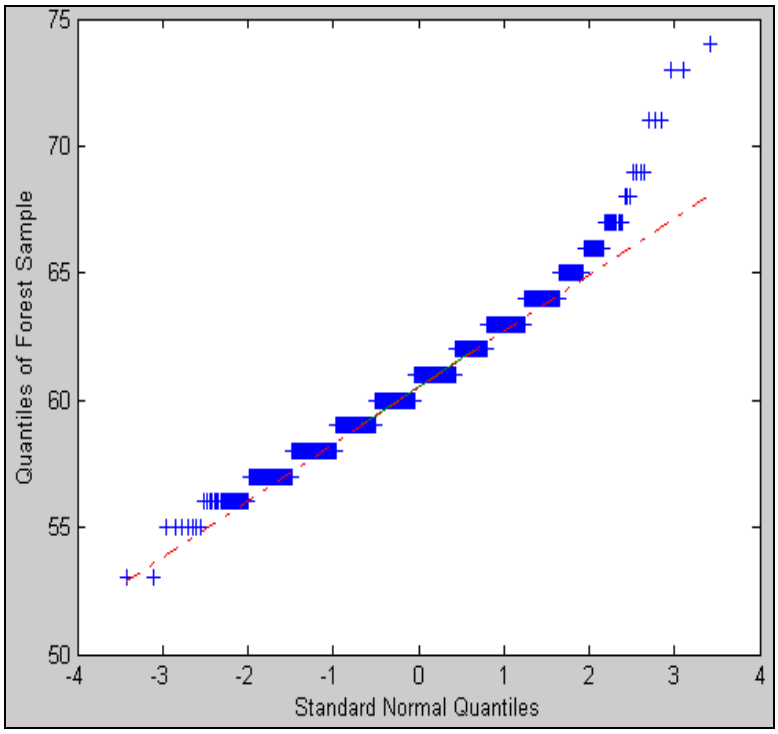

Figure 3.12. Forest training data 2nd band QQ plot

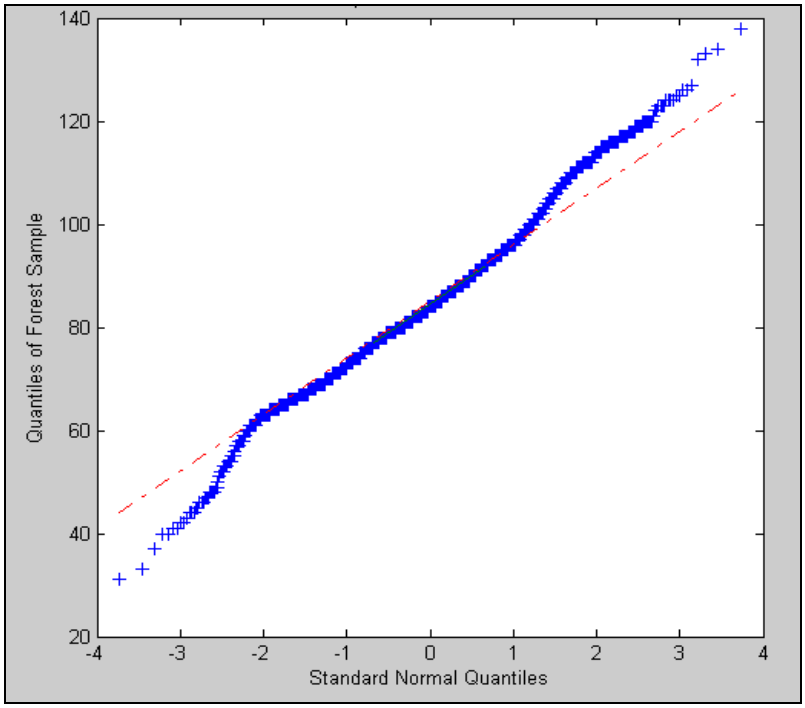

Figure 3.13. Nonforest training data 4th band QQ plot

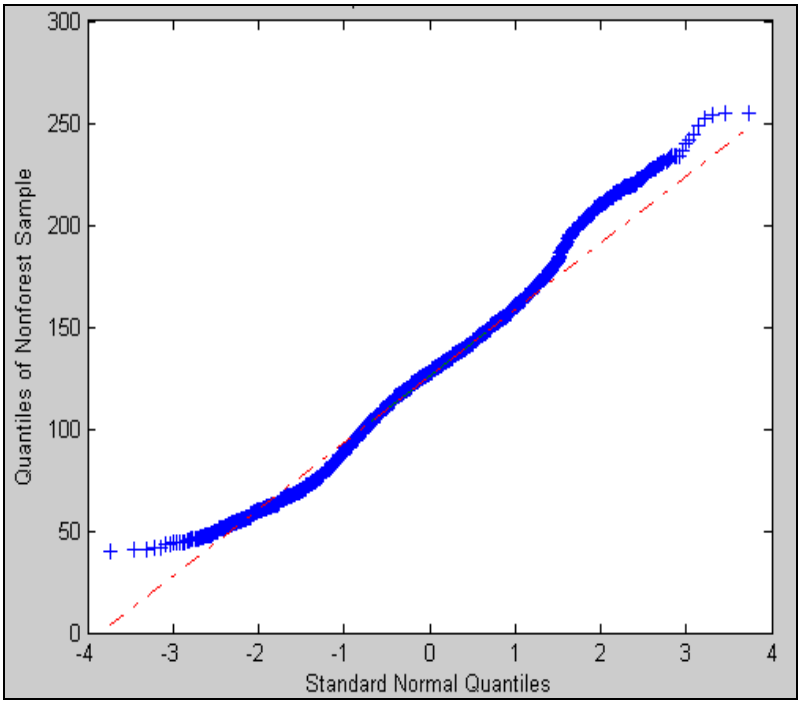

Figure 3.14. Nonforest training data 3rd band QQ plot

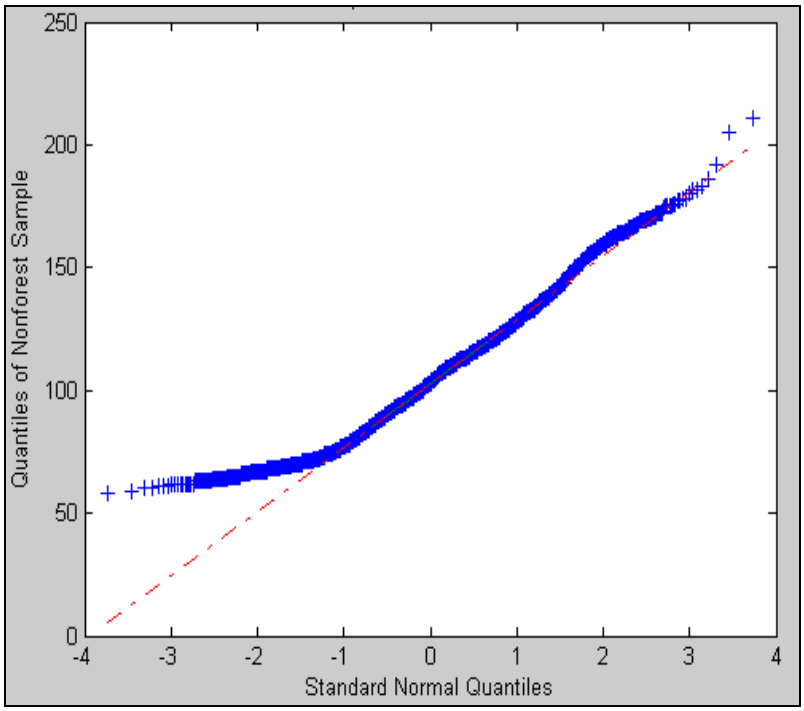

Figure 3.15. Nonforest training data 2nd band QQ plot

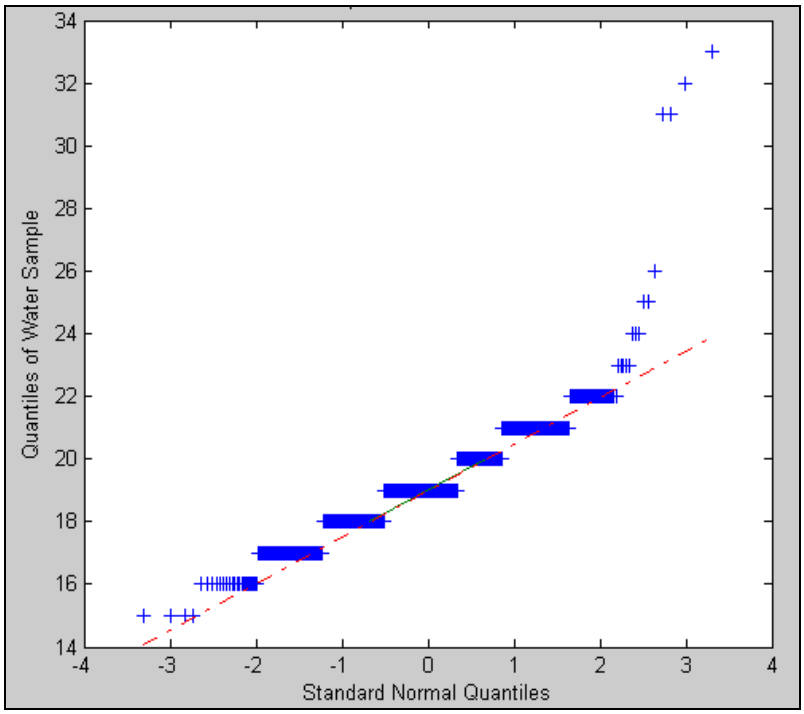

Figure 3.16. Water training data 4th band QQ plot

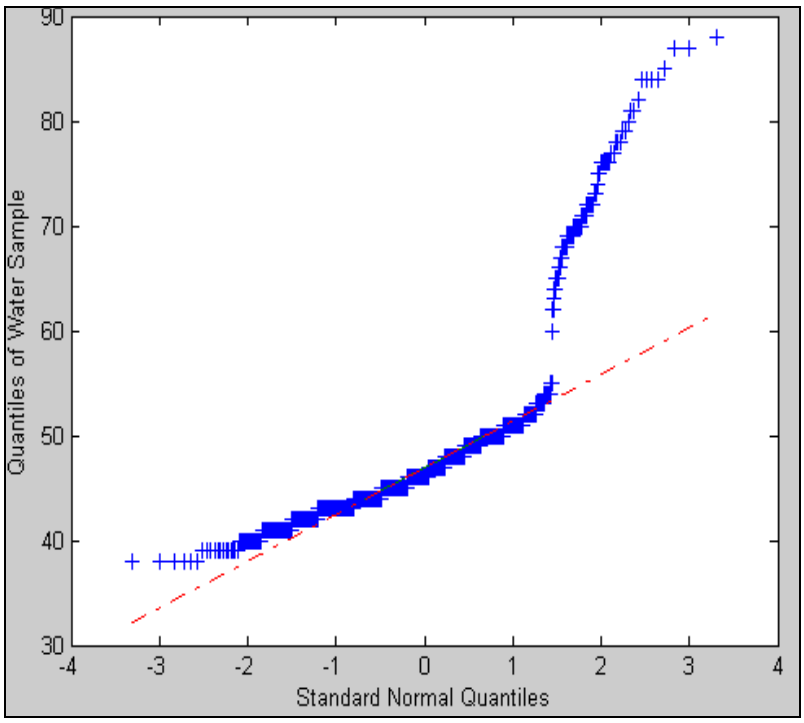

Figure 3.17. Water training data 3rd band QQ plot

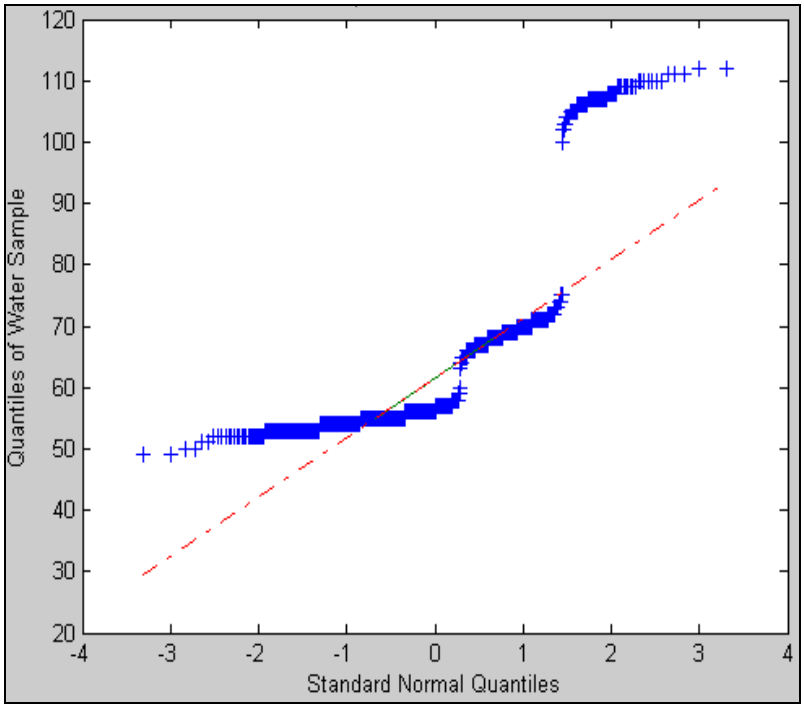

Figure 3.18. Water training data 2nd band QQ plot

3.6 Band selection

Usually, in satellite image classification, 3 bands are selected out of the maximum number of bands, which is 8 in the case of ETM and TM images. The main reason for this is ease of viewing. Computer can only display 3 color channels, red, green and blue. Therefore, true-color computer images consist of 3 bands (channels) only. The intensity of each band is represented with 8 bits (256 different possible values). Therefore, 3 bands take up 24 bits (2^{24} different possible values). When these 3 bands are assigned to red, green and blue channels, the computer can display all the different colors detectable by the human eye plus many more. Trying to display a fourth channel would result in colors that the human eye cannot detect. Remote sensing software that displays more than 3 bands does so by merging values from all channels into the three visible channels. This results in loss of real color information for the viewer. 24-bit (3 channel) image formats also have the advantage that they can be opened and handled by any computer program.

There is also a need to retain only a small number of useful and good features (bands) for the classification algorithm (Wiley, 1973, in Raudys and Jain, 1991). It can happen that as the number of bands is increased, the classification error also increases. This is called the curse of dimensionality. So, only the most relevant bands should be selected.

In Pereira, 2006, the band combinations (RGB;432) and (RGB;543) are recommended for detecting forest areas. Both band combinations were tested with their accuracy. No significant difference was found between the accuracies for the two suggestions, so the 432 combination was selected arbitrarily. Moreover, 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, 7^{th} bands are used for classification and compared with 432 and the result is more or less the same. 6^{th} Band is not selected because it decreases accuracy of classification.

3.7. Addition of DEM data

In addition to 3 selected bands, DEM information can also be considered in the classification algorithm as an extra (fourth) band. A digital elevation model (DEM)

is a digital representation of ground surface topography or terrain. In Pereira (2006), it is explained that DEM's represent the shape of Earth's surface, and each pixel represents an elevation measurement rather than a brightness value.

According to Campbell (2002), in Pereira (2006), data for DEM's can be compiled using one of the three alternatives:

- Contours from maps can be digitized, converted to vector files and tagged with elevation values producing an elevation map,
- Photogrammetric methods derive the elevation map from stereoscopic images,
- Elevation maps are constructed based on survey data but they are very expensive and time consuming.

One source of DEM data is Shuttle Radar Topographic Mission (SRTM). According to Gorokhovic and Voustianiouk (2006), SRTM has created an unparalleled data set of global elevations that is freely available, and can be easily downloaded from the Internet (URL4 and URL5). Almost 80% of the Earth surface is available from SRTM. This is of great value for scientists dealing with terrain analysis. The data were collected over the 11-day mission in February 2000. Since then, they have been described in detail (Farr and Kobrick, 2000; Rabus et al., 2003; Werner, 2001, in Gorokhovic and Voustianiouk, 2006). It is mentioned that SRTM data with 90 m spatial resolution is available globally, however, SRTM data with 30 m spatial resolution is only available for USA territory. SRTM data has been used for vegetation cover studies in Kellndorfer et al., 2004 (in Gorokhovic and Voustianiouk, 2006).

In Haapanen et al. (2004), several studies have found that stratification by DEM or other factors that drive vegetative gradients can also improve classification.

According to Pereira (2006), vegetation patterns are closely related to topography. Therefore, when working with vegetation patterns, elevation data is crucial for mapping vegetation patterns with digital data. With DEM data, it is possible to incorporate, manipulate, classify and display elevation data to improve the

classification task. DEM data can be used in image classification by adding the DEM as an additional band of data for classification, to improve classification accuracy. Additional data like DEM improve classification accuracy, as shown in Nangendo et al. (2007).

Whether topographic correction of TM bands and adding the digital elevation model (DEM) as additional band improves the accuracy of Landsat TM-based forest mapping was studied by Dorren et al. (2003). Here, different classification schemes were applied to Landsat TM images with and without the DEM as additional band. This technique has been called the “logical channel approach” (Strahler et al., 1978; Hutchinson, 1982, in Dorren et al., 2003). The classification results were assessed with error matrices and kappa statistics. It was found that classification with the DEM as additional band increases the accuracy of Landsat TM-based forest maps.

3.8. Flowcharts of KNN and Maximum Likelihood Classifications Applications

KNN and Maximum Likelihood Applications, which are used in this study, are summarized as the flowcharts in Figure 3.19 and Figure 3.20.

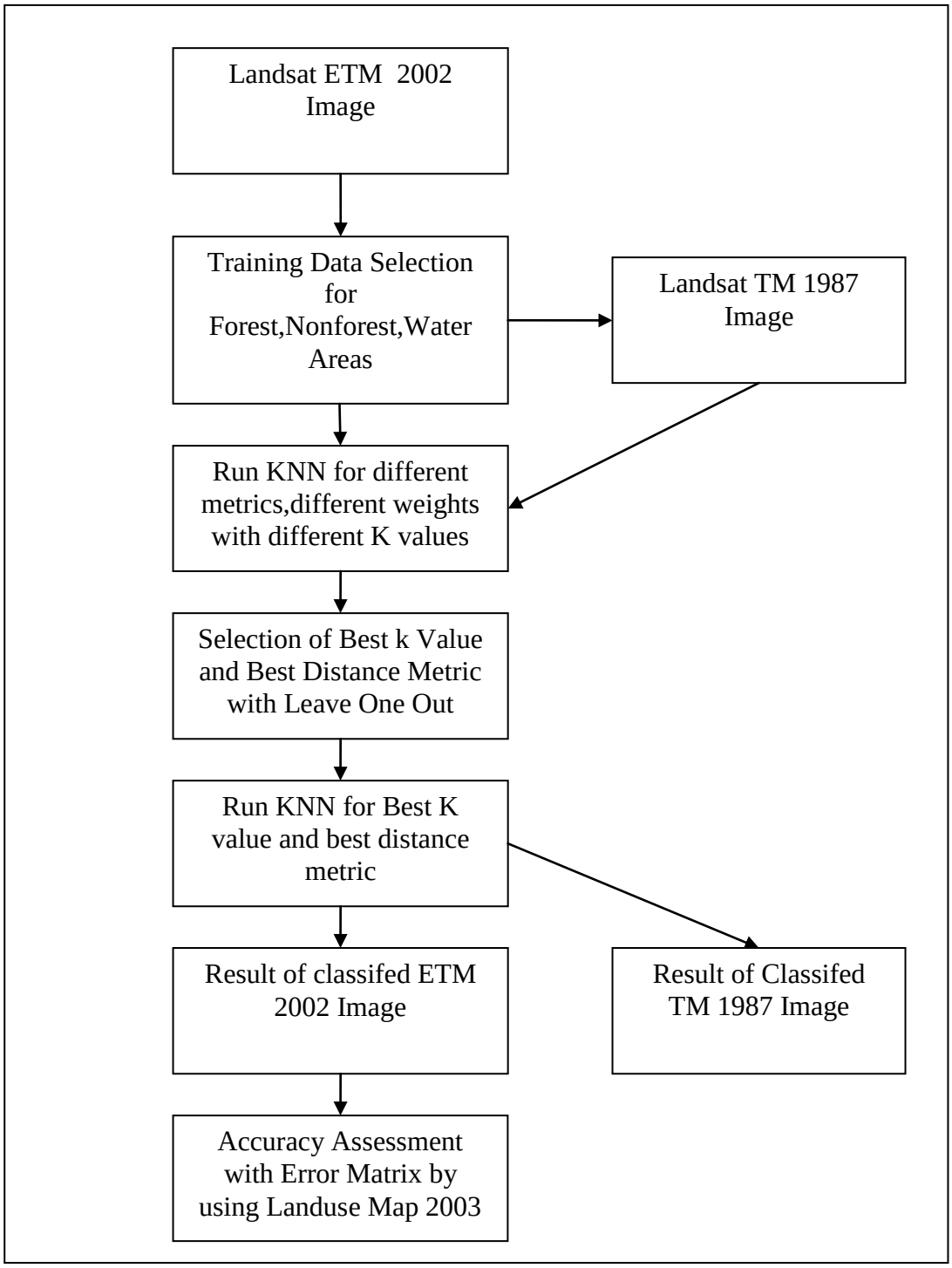

Figure 3.19. KNN Classification application flowchart

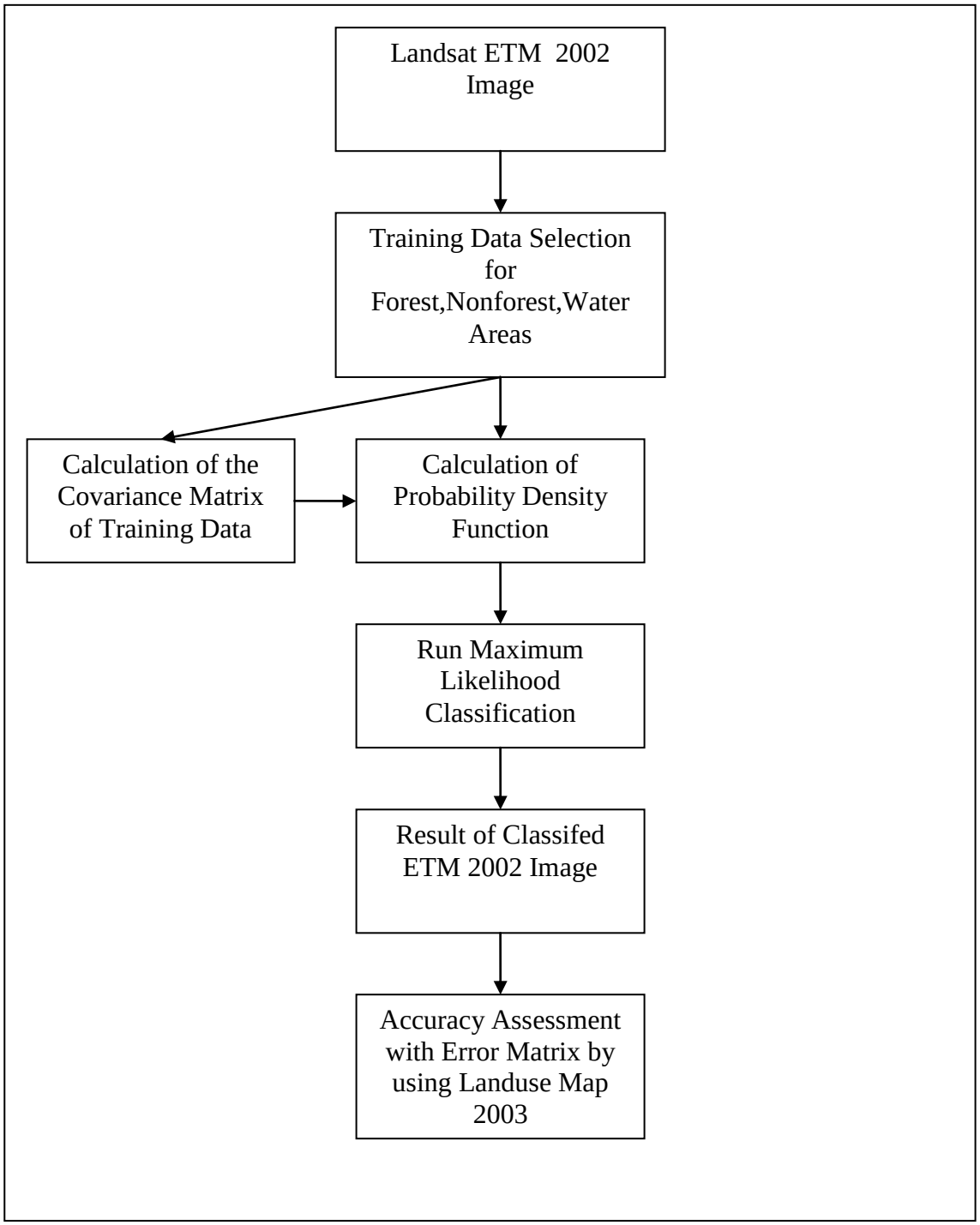

Figure 3.20. Maximum Likelihood Classification application flowchart

CHAPTER 4

ANALYSES

In this section, the analyses done with the Maximum Likelihood and KNN algorithms are explained. The complexity and the performance are compared. Implementation problems are discussed.

4.1. KNN Algorithm

The KNN algorithm was analyzed by examining various literatures on this topic. This algorithm is increasingly being used in land use classification, often to estimate forest areas. Initially, MATLAB was used to implement the algorithm, due to its clear and expressive syntax.

The KNN algorithm can be seen in Figure 4.1. There is no need to implement a complex probability distribution equation, or calculate statistical values like covariance. Since it is distribution free, KNN algorithm is non-parametric. The most complex part of the code was in the CalculateDistance() function. This is where all 3 color dimensions are taken into consideration. This function and the FindStrongestClass() function will be explained later. Note that in KNN, there is no need to make any assumptions about the image to be classified.

The inputs to this algorithm are the value of k, the three sets of training data and the image to be classified, all saved as 24-bit bitmap files. The output is a three-color (three-class) classified image, also saved as a 24-bit bitmap file (C# can only save 24-bit bitmaps). The value of k can have a significant effect on the accuracy of the algorithm. How to choose the best k is discussed in Chapter 2.

```
function KNN

k := 14

f := ReadForestTrainingData
nf := ReadNonforestTrainingData
w := ReadWaterTrainingData

sat := ReadSatelliteImage
classified := new Image

for each column in sat
      for each row in sat

         testPixel := sat[column, row]
         distanceList := new List()

         for each trainingPixel in f
           distance := CalculateDistance(testPixel,trainingPixel)
           AddToList(distanceList, trainingPixel, distance)
         for each trainingPixel in nf
           distance := CalculateDistance(testPixel,trainingPixel)
           AddToList(distanceList, trainingPixel, distance)
         for each trainingPixel in w
           distance := CalculateDistance(testPixel, trainingPixel)
           AddToList(distanceList, trainingPixel, distance)

         SortByDistance(distanceList)
         neighborList := distanceList[1..k]
         class := FindStrongestClass(neighborList)
         classified[column, row] := DefaultColorPixel(class)

SaveImage(classified)
```

Figure 4.1. The pseudocode of KNN Classification

It was noticed that the algorithm ran extremely slow, compared with the maximum likelihood algorithm. The reason is that most of the time-consuming parts are delayed until the moment of decision, that is, repeatedly calculated for each pixel combination. In addition to CalculateDistance(), the SortByDistance() function cost the highest performance. This problem is one of the weaknesses of the KNN algorithm. It could be possible to improve performance by using spatial indices such as R-tree or R*-tree. These would help retrieve data items quickly according to their spatial location (Gutmann, 1984). The output of KNN classification with Euclidean distance metric is given in Figure 4.2.

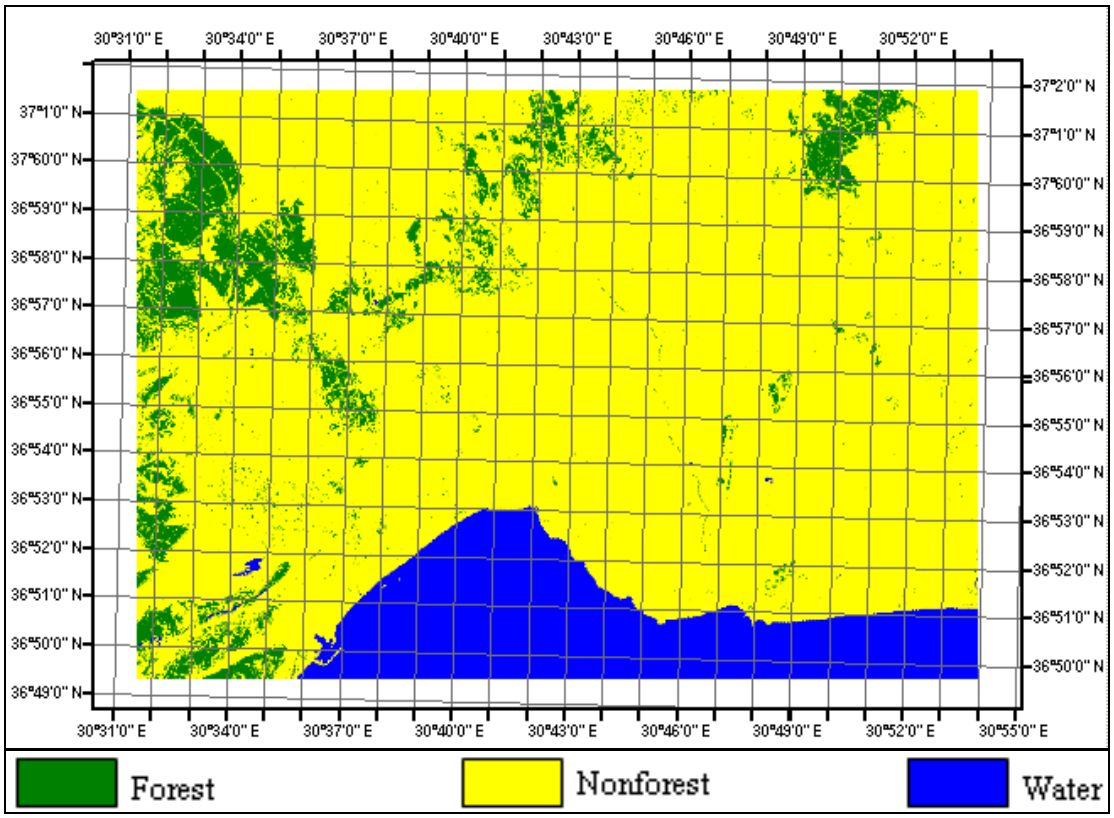

Figure 4.2. Landsat ETM (RGB; 432) with Euclidean distance metric.

4.1.1. Leave One Out Algorithm

This algorithm is used to estimate the accuracy of a classification algorithm. It can be used in repeated runs to find the optimum algorithm or the optimum parameters, such as the value of k.

Initially, MATLAB was used to implement the algorithm, due to its clear and expressive syntax.

```
function LOO
f := ReadForestTrainingData
nf := ReadNonforestTrainingData
w := ReadWaterTrainingData

trainingData := {f, nf, w}

pixelEstimates := new List

for each trainingPixel in trainingData
     RemovePixel(trainingData, trainingPixel)
     estimate := ClassifyPixel(trainingData, trainingPixel)
     AddToList(pixelEstimates, estimate)
     RestorePixel(trainingData, trainingPixel)

CalculateError(pixelEstimates, trainingData)
```

Figure 4.3. The pseudocode of Leave One Out Algorithm

The algorithm can be seen in Figure 4.3. The most complex parts here are the ClassifyPixel() function, which calls the classification algorithm which is being measured, and the CalculateError() function, which executes the error formula (Equation 2.10) on the results. The inputs to this algorithm are the classification algorithm and the three sets of training data. The output is the error values based on the classification. The algorithm runs pretty fast, especially due to the low size of the training data.

4.1.2. Distance functions

The CalculateDistance() function in KNN algorithm can implement one of the following distance measurements:

Euclidean,

Manhattan,

DiagonalMahalanobis,

Mahalanobis,

By using Leave One Out cross validation, error values were calculated, for different values of k and for different distance metrics. As it is seen in Table 4.1, they were sorted to understand the lowest value which gave the best result. The values were plotted in Figure 4.4. The best result was obtained when the distance metric was selected as Euclidean. The results of the classification are depicted in Figures 4.5-4.7 for Manhattan, Diagonal Mahalanobis and Mahalanobis distance metrics respectively.

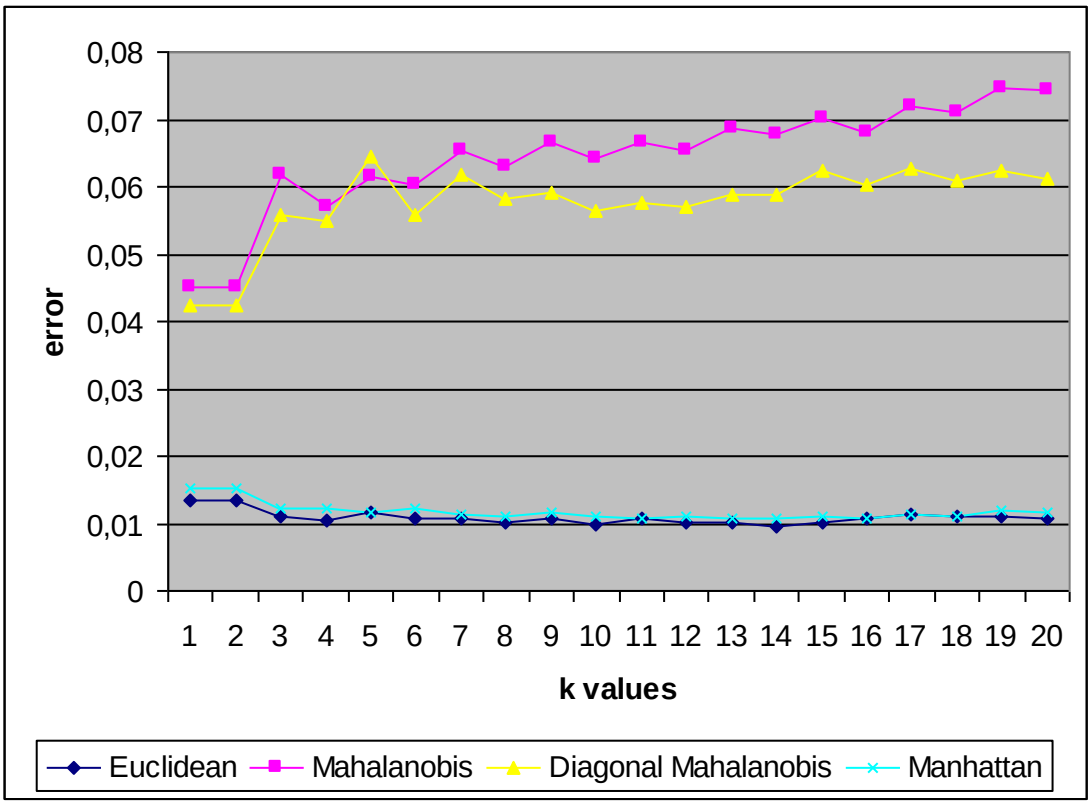

Figure 4.4. 2002 Landsat ETM satellite imagery KNN classification error plot with different metrics (RGB; 432).

Table 4.1. 2002 Landsat ETM satellite imagery (RGB; 432) KNN classification error values

k	Euclidean	Manhattan	Diagonal Mahalanobis	Mahalanobis
1	0,013578	0,015291	0,042446	0,045015
2	0,013578	0,015291	0,042446	0,045015
3	0,011131	0,012232	0,055902	0,061774
4	0,010398	0,01211	0,054801	0,057003
5	0,011498	0,011743	0,064465	0,061407
6	0,010765	0,01211	0,05578	0,060306
7	0,010887	0,011376	0,061651	0,065443
8	0,010275	0,011009	0,058104	0,062997
9	0,010765	0,011621	0,05896	0,066544
10	0,009908	0,011009	0,056269	0,06422
11	0,010642	0,010765	0,057492	0,066667
12	0,010275	0,011131	0,056881	0,065443
13	0,010031	0,010887	0,058716	0,068746
14	0,009419	0,010765	0,058716	0,067768
15	0,010153	0,011009	0,062263	0,070092
16	0,010887	0,010642	0,060183	0,068135
17	0,011376	0,011376	0,06263	0,071804
18	0,011009	0,011131	0,06104	0,07107
19	0,011009	0,011865	0,062263	0,074495
20	0,010642	0,011743	0,061162	0,074373

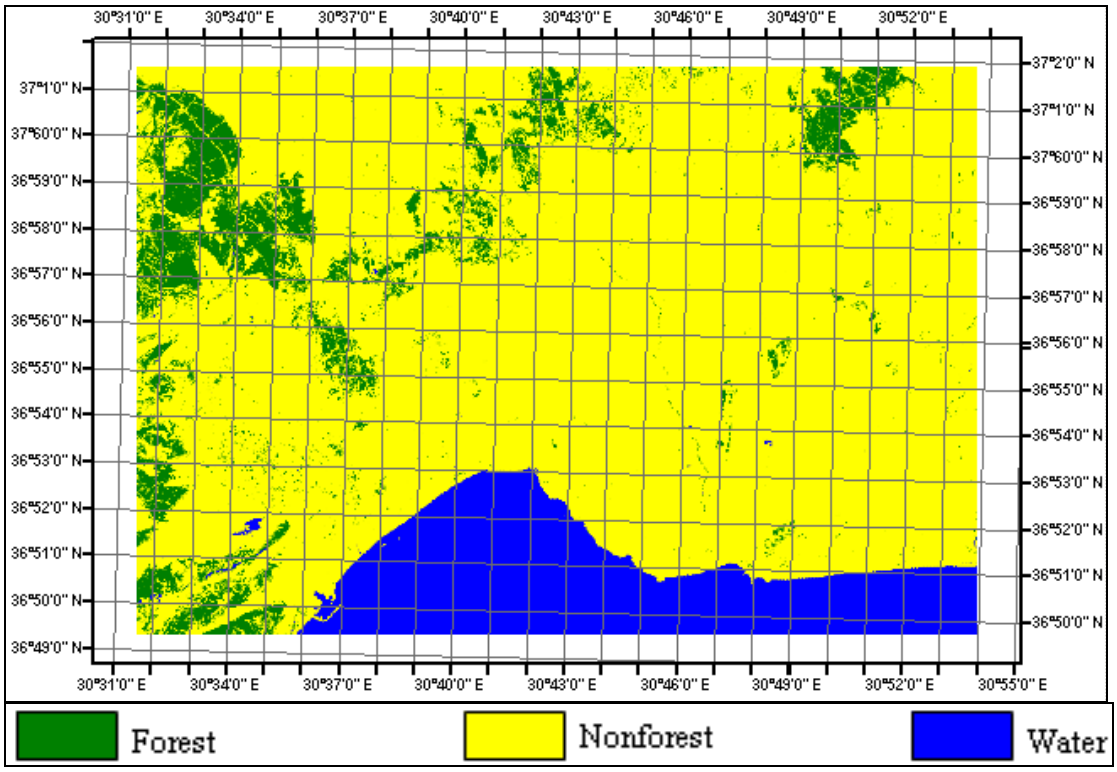

Figure 4.5. Landsat ETM (RGB; 432) with Manhattan distance metric.

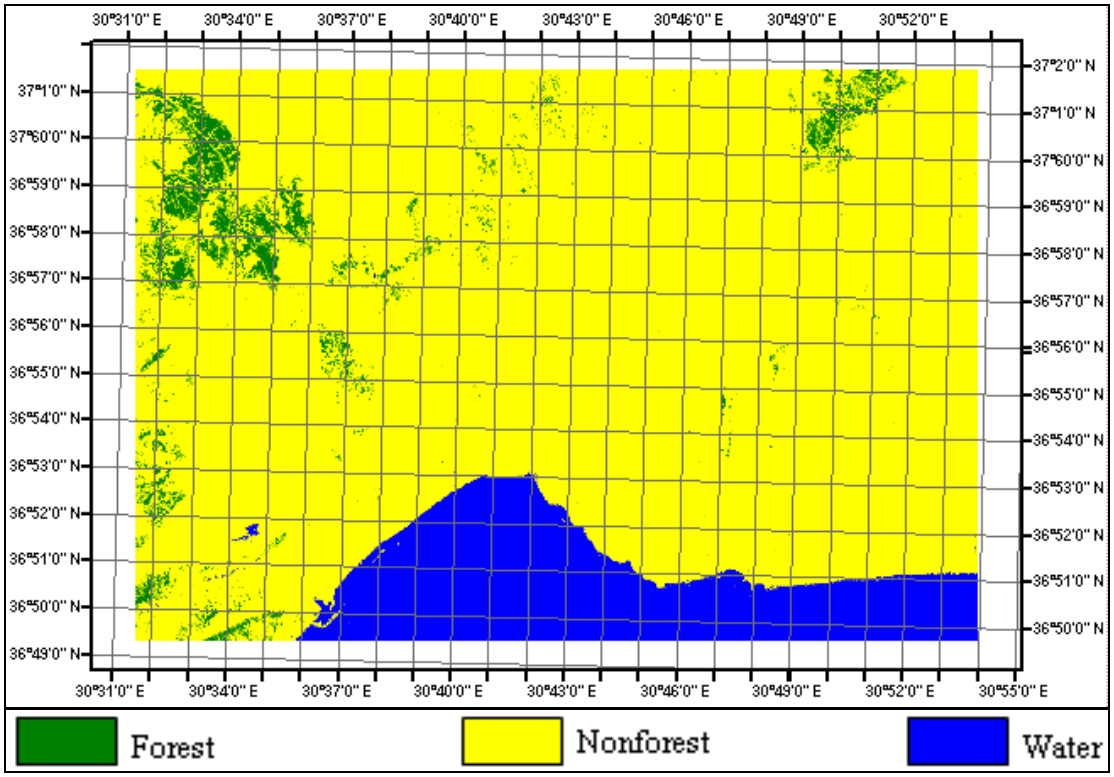

Figure 4.6. Landsat ETM (RGB; 432) with Diagonal Mahalanobis distance metric.

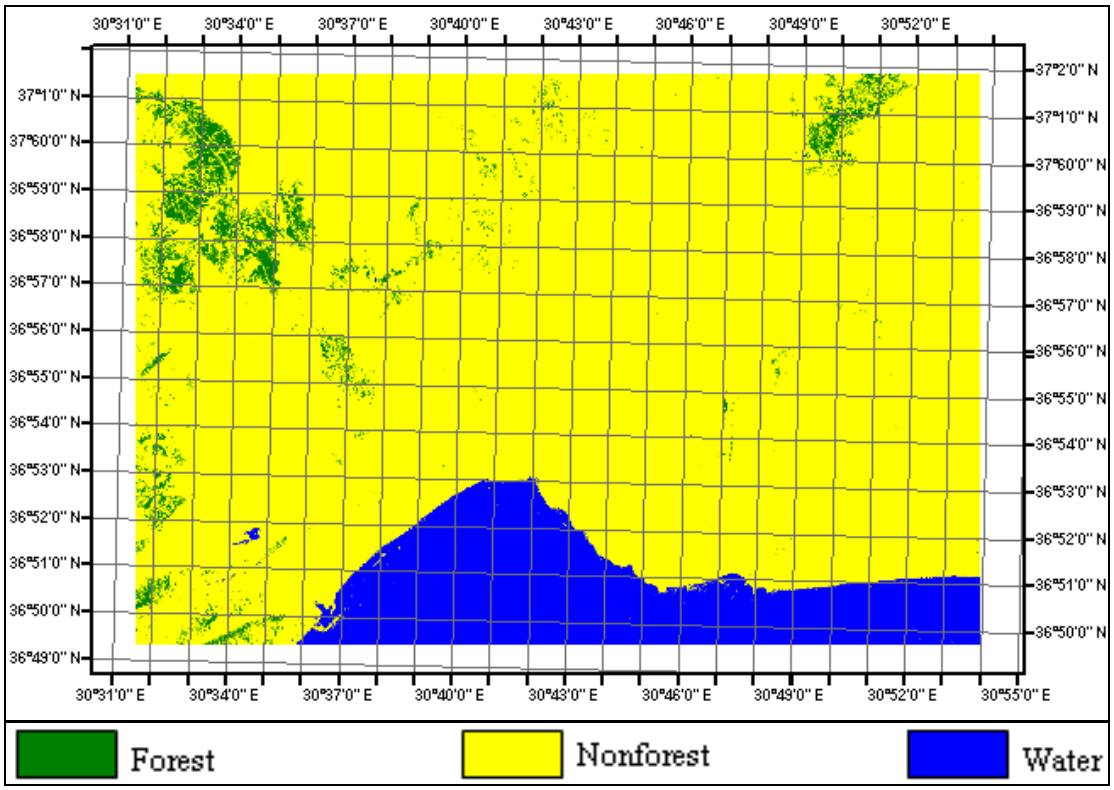

Figure 4.7. Landsat ETM (RGB; 432) with Mahalanobis distance metric.

4.1.3. Weight functions

The FindStrongestClass() function in KNN algorithm can implement one of the following weight functions:

None (Equal weight),

Fraction,

Stairs,

InverseDistance,

InverseSquareDistance

Note that it is not possible to omit the Sqrt() method when calculating the distance, if the weight function is dependent on the distance.

Table 4.2. 2002 Landsat ETM satellite imagery Euclidean and Weighted Euclidean KNN classification error values (RGB; 432).

k	Euclidean	Euclidean Inverse Distance	Euclidean Inverse Square Distance	Euclidean Stairs	Euclidean Fraction
1	0,013578	0,013578	0,013578	0,013578	0,013578
2	0,013578	0,013578	0,013578	0,013578	0,013578
3	0,011131	0,011254	0,011865	0,013578	0,013578
4	0,010398	0,010887	0,011254	0,010398	0,010398
5	0,011498	0,011131	0,01052	0,011009	0,010765
6	0,010765	0,01052	0,010398	0,011009	0,01052
7	0,010887	0,010765	0,010642	0,010275	0,010887
8	0,010275	0,010398	0,01052	0,010398	0,010765
9	0,010765	0,010642	0,010642	0,010642	0,011131
10	0,009908	0,009786	0,010398	0,010398	0,01052
11	0,010642	0,010153	0,010275	0,010275	0,01052
12	0,010275	0,010031	0,009908	0,01052	0,01052
13	0,010031	0,009541	0,009664	0,010398	0,010398
14	**0,009419**	0,009174	0,009541	0,01052	0,010153
15	0,010153	0,009908	0,009541	0,010275	0,010153
16	0,010887	0,010275	0,009664	0,010275	0,010275
17	0,011376	0,010398	0,010031	0,010275	0,010153
18	0,011009	0,010275	0,009908	0,010031	0,010031
19	0,011009	0,010031	0,009908	0,010153	0,010031
20	0,010642	0,010275	0,009908	0,010153	0,010275

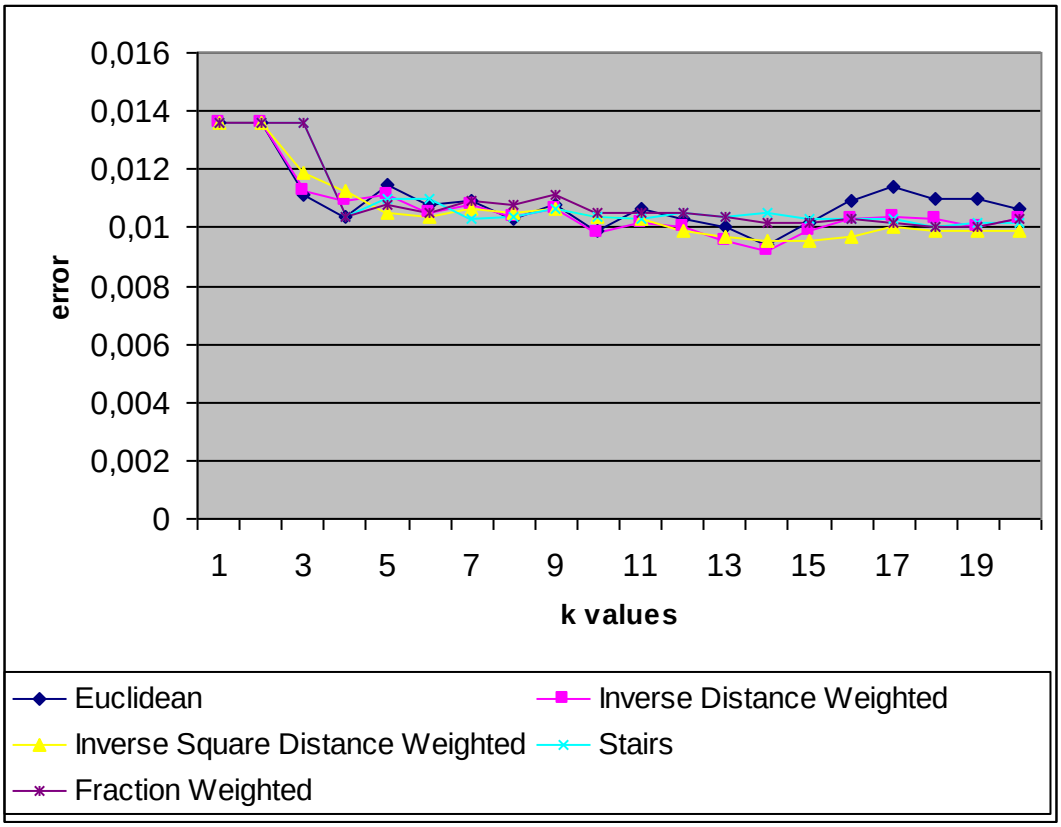

Figure 4.8. 2002 Landsat ETM satellite imagery Euclidean and Weighted Euclidean KNN classification error plot (RGB; 432).

After the selection of the Euclidean distance metric as the best one, the different weight functions were taken into consideration. Again, Leave One Out cross-validation was performed for each weight function and different values of k, and the error values were calculated. The best four weight functions were “Inverse Distance”, “Inverse Square Distance”, “Stairs” and “Fraction”. The results can be seen in Table 4.2 and Figure 4.8. It is also possible to compare the results with those obtained when no weight function is used (all neighbors have equal weight). The results are sorted to understand the lowest value which gives the best result.

4.1.4. Selecting the number of neighbors (k value selection)

After the selection of the Euclidean distance metric and the Inverse Distance Function as the best one, different values of k were taken into consideration. Again, Leave One Out cross-validation was performed for different values of k, and the error values were calculated. As it is seen in Table 4.3, the results are sorted to

understand the lowest value which gives the best result. Best result is obtained when k = 14. Then, all satellite images are classified with Inverse Distance Metric.

Table 4.3. 2002 Landsat ETM satellite imagery Inverse Square Distance Euclidean KNN classification error values (RGB; 432).

k	Euclidean Inverse Distance
1	0,013578
2	0,013578
3	0,011254
4	0,010887
5	0,011131
6	0,01052
7	0,010765
8	0,010398
9	0,010642
10	0,009786
11	0,010153
12	0,010031
13	0,009541
14	**0,009174**
15	0,009908
16	0,010275
17	0,010398
18	0,010275
19	0,010031
20	0,010275

4.1.5. Results of different KNN parameters

Non-weighted Euclidean distance metric were applied to both ETM (RGB; 432) and ETM (RGB; 321) and the results were presented respectively in Figure 4.2 and Figure 4.9. Then, Inverse Distance weighted KNN, which gave the lowest error, was applied to ETM (RGB; 432), ETM (RGB; 321), ETM (7 bands), TM (RGB; 432) satellite images which were classified by using the same training data respectively given in Figure 4.10-4.13. Subsequently, ETM (RGB; 432) was classified by using DEM as the 4^{th} band given in Figure 4.14.

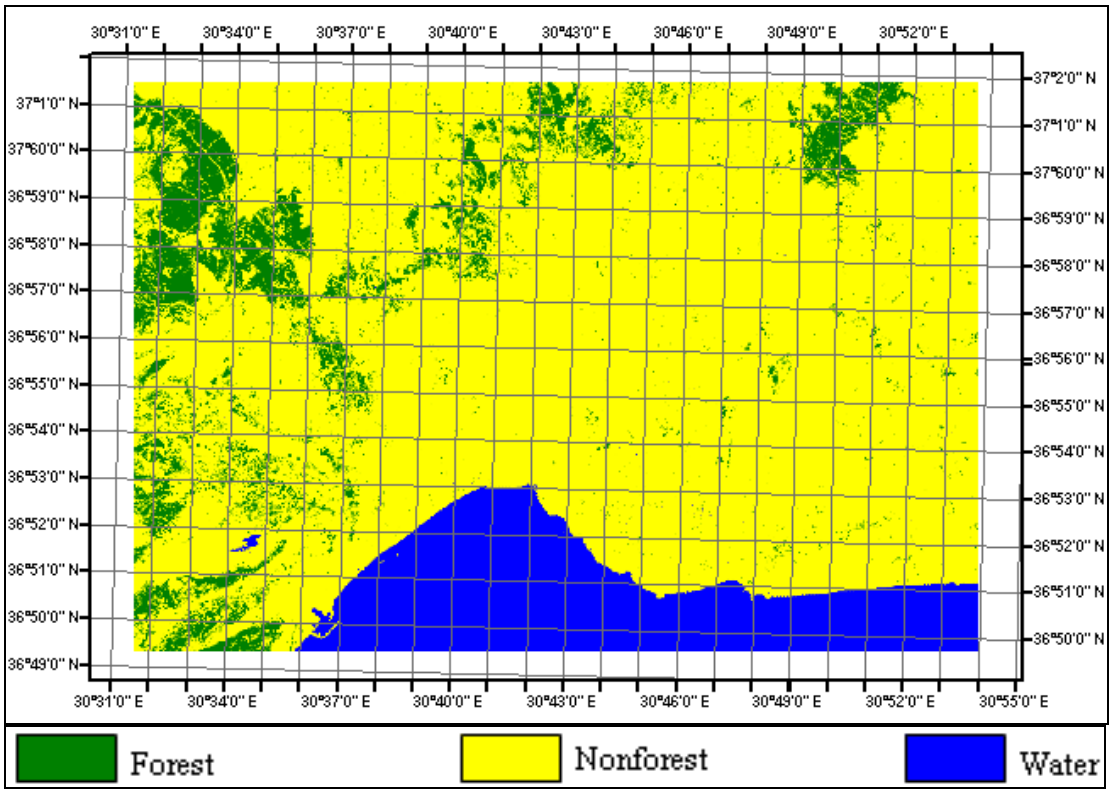

Figure 4.9. Landsat ETM (RGB; 321) with Euclidean distance metric (k=14).

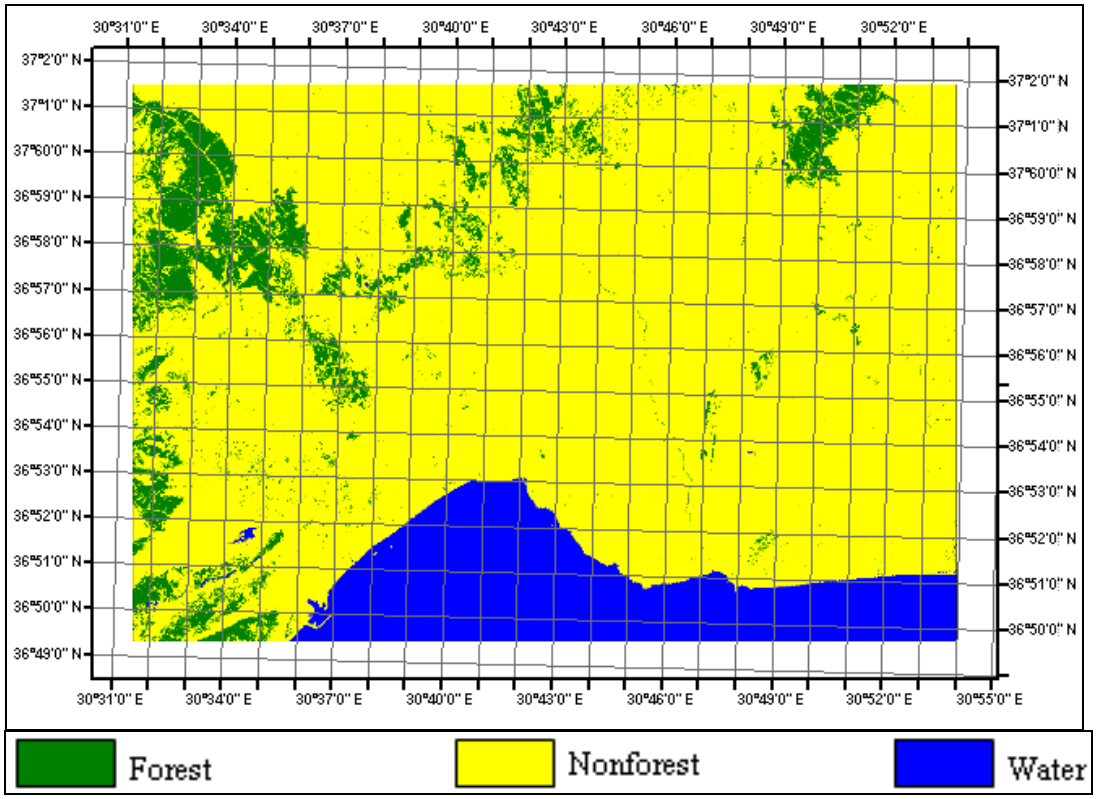

Figure 4.10. Landsat ETM (RGB; 432) with Inverse Distance Weighted Euclidean distance metric (k=14).

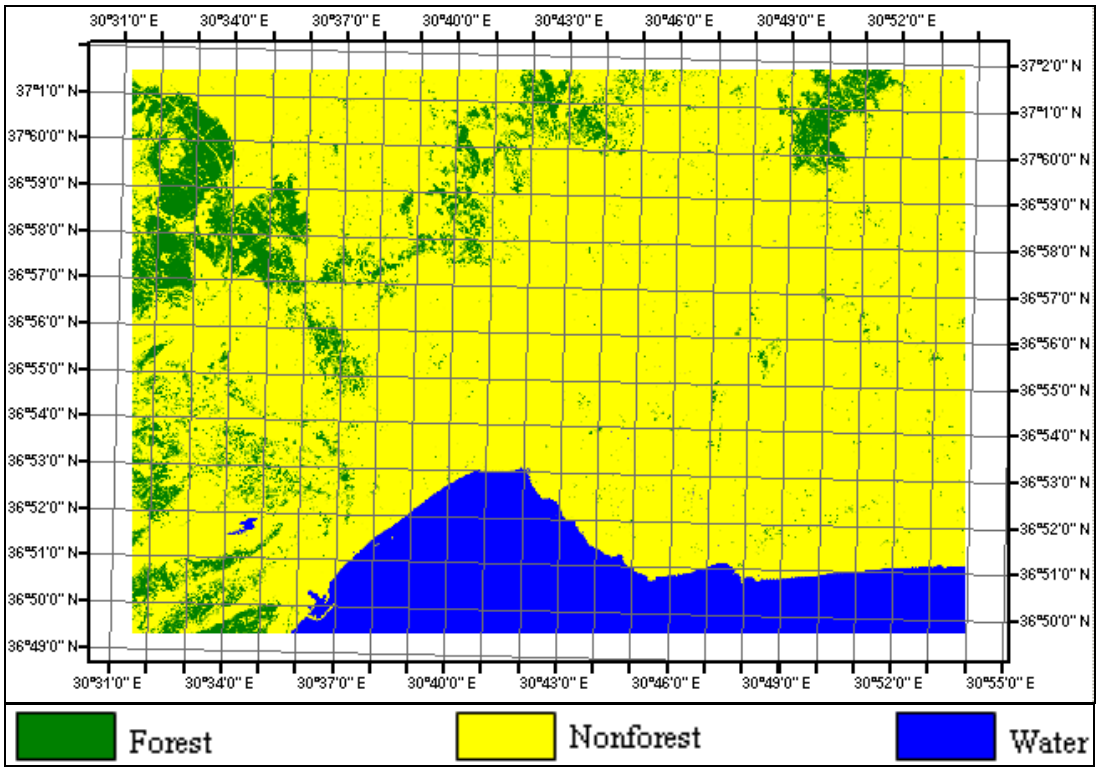

Figure 4.11. Landsat ETM (RGB; 321) with Inverse Distance Weighted Euclidean distance metric (k=14).

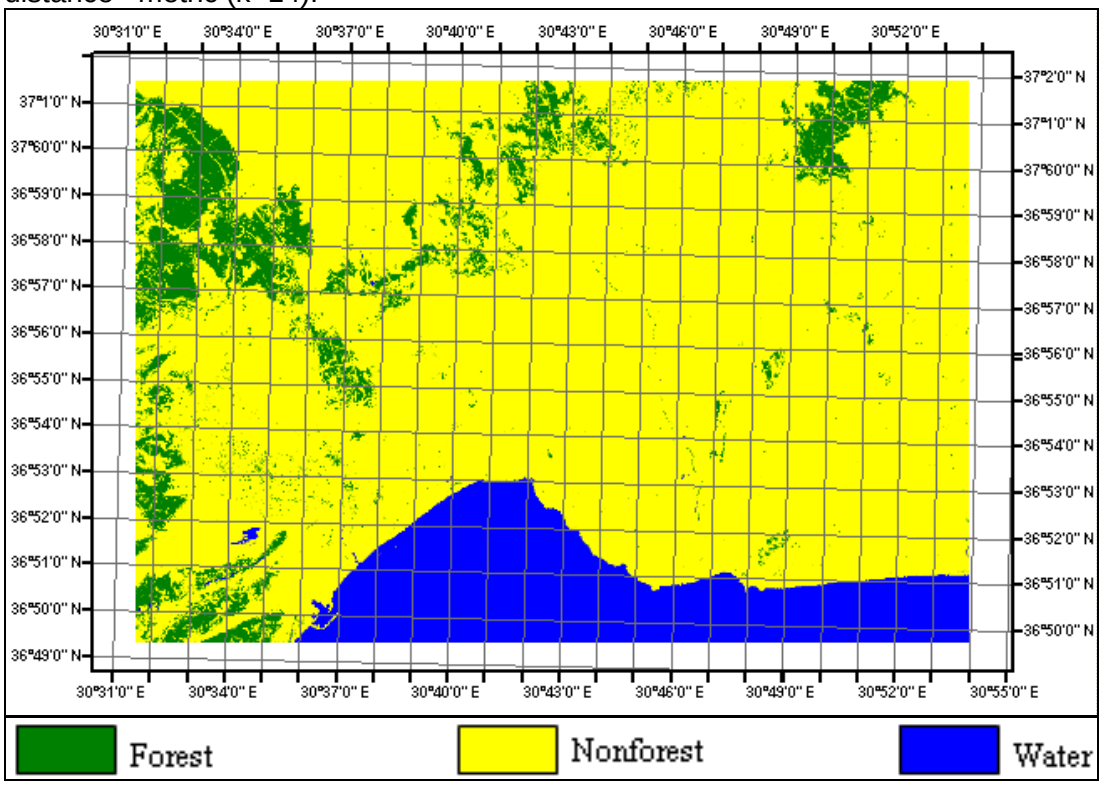

Figure 4.12. Landsat ETM (1;2;3;4;5;7) with Inverse Distance Weighted Euclidean distance metric (k=14).

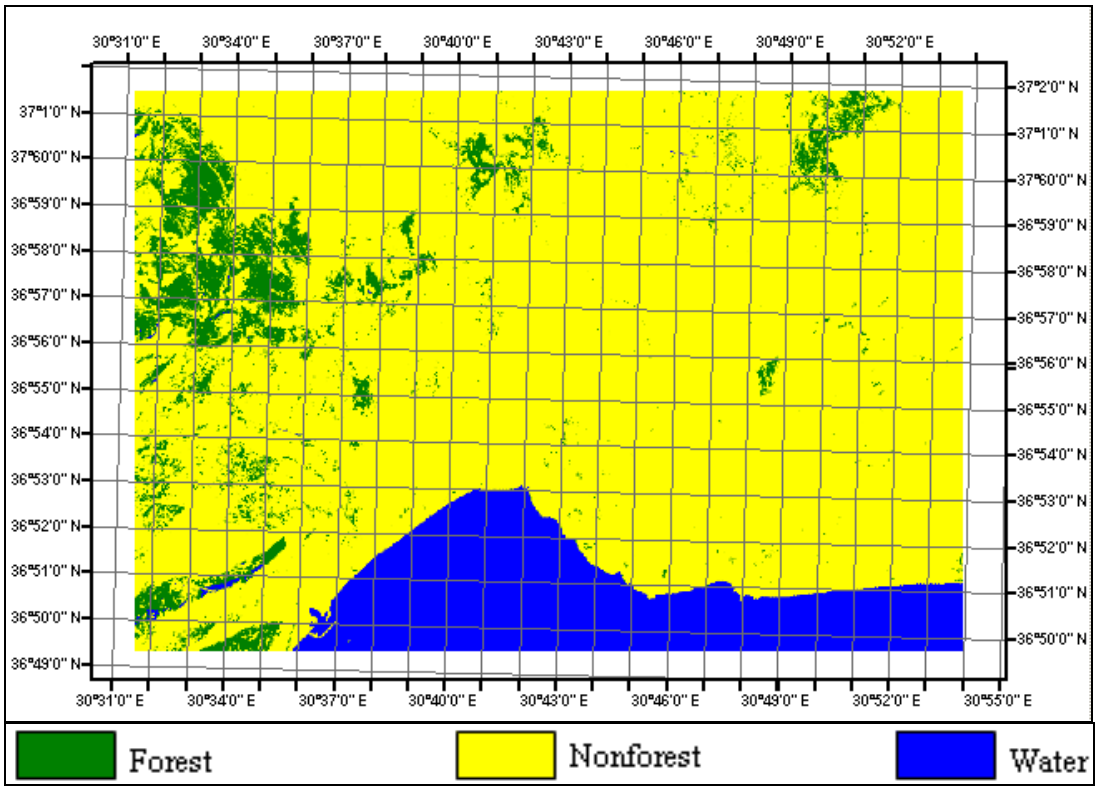

Figure 4.13. Landsat TM (RGB; 432) with Inverse Distance Weighted Euclidean distance metric (k=14).

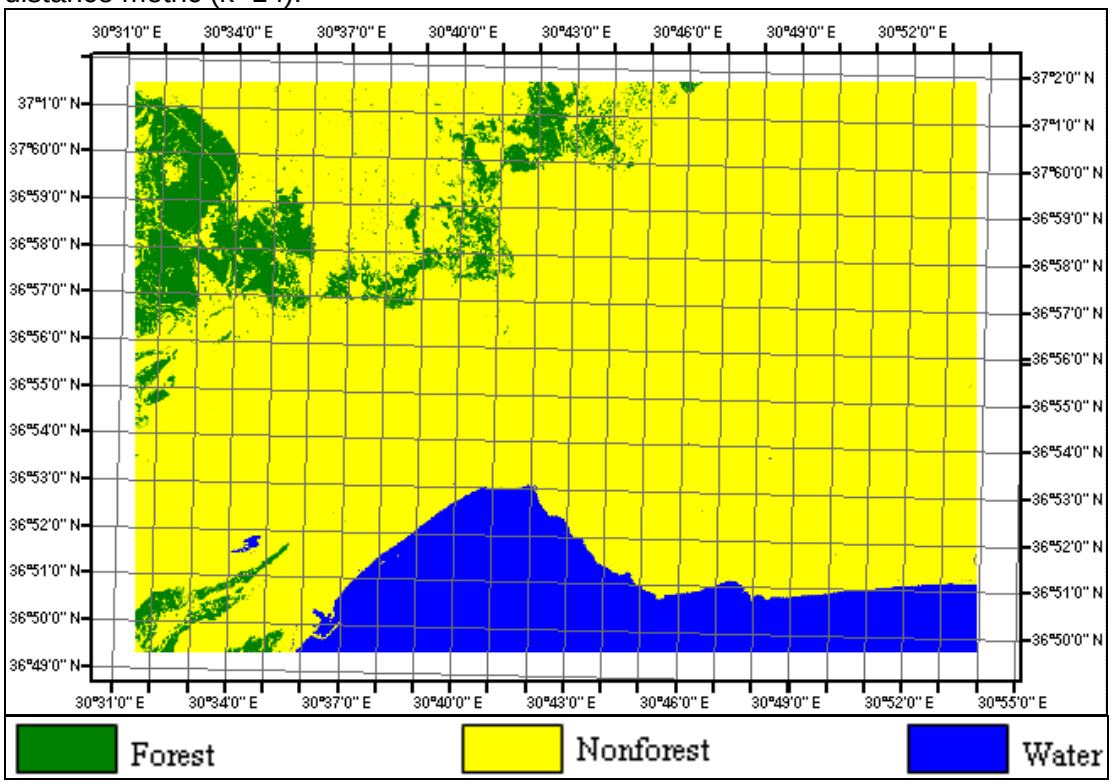

Figure 4.14. Landsat ETM (RGB; 432 and DEM) with Inverse Distance Weighted Euclidean distance metric (k=14).

4.1.6. Design and performance improvements for KNN

To obtain more computing performance, the function was converted to C#, and the following improvements were made:

- All code relating to the algorithm was encapsulated in its own class, called KNN. The input parameters were made configurable.

- The algorithm, which can be seen as pseudocode in Figure 4.1, was modified so that it would run with any number of classes, and therefore, with any number of training data sets. This may be useful in future studies, where a finer classification of forest or non-forest areas is desired.

- Since C# does not contain useful matrix functions like MATLAB by default, a freeware matrix library called “CSML” was installed and used (URL2). It was noticed that the performance of this library was insufficient for distance calculations in KNN. Therefore, the matrix operations were implemented by hand, using primitive arithmetic operations.

- Memory usage was improved. Whenever possible, large lists of objects were not duplicated, but the calculations were performed directly on the existing lists. For example, instead of creating a distance list for each band and then combining them into yet another list, a single distance list was used for all bands. In addition, the length of the distance list was limited to k instead of being equal to the size of the training data.

- It was found that it is noticeably faster to work with the “int” data type instead of the higher precision “double” data type, at times where the higher accuracy is not required. When calculating Mahalanobis distance, or when using weight functions, high-precision values are required, therefore, “double” variables have to be used.

- The Sqrt() function costs performance, so one should try to avoid it. However, it is used in the CalculateDistance() function. In cases where the

FindStrongestClass() does not need the exact distances, it was found that one can omit the Sqrt() function and just return the square distance. The SortByDistance() function will work just as well with square distances. This way, since square distances are always whole numbers, it is possible to use the faster “int” data type as well.

- The size of the training data had a huge impact on the computing performance. It was found that any increase in the training data size caused two problems:
    - For each test pixel, that many more distances had to be calculated.
    - The distance list was that much longer, causing the sort function to work longer.

  The performance could be improved by using spatial indices such as R-tree.

- The SortByDistance() function was the biggest performance bottleneck. If the number of distances is huge, the sort function is very slow. C# uses the quicksort algorithm, which is one of the fastest sorting algorithms, but it is still not fast enough. In our case, we preferred to keep the distance list sorted during AddToList(), and omit the SortByDistance() function altogether. Actually, only the first k members of the distance list were of interest, so only these members needed to be sorted. In AddToList(), a training pixel was added to the neighbor list only if its distance was less than any of the first k members of the neighbor list. Since the value of k is usually very low, this improvement brought a significant performance increase.

- The most significant speed increase came from keeping a history of classification results. For the same algorithm (with the same parameters) and the same training data, the same pixel is always classified to the same class. Therefore, in a classification run, it is possible to keep a history of all pixels that have already been classified. If a new pixel is encountered which has already been classified, then one can simply trust the corresponding class from the history. There is no need to run the KNN algorithm on the

pixel. It was noticed that the history can become quite large, often around one fifth of the number of pixels in the image. This shows that on average, each pixel is repeated five times in different places in the satellite image. In any case, due to the huge size of the history cache, it was necessary to find a fast method to access an individual pixel in it. This was accomplished by using a hash table, where the hashing function was based on the three color values (RGB) in the pixel. Another possibility would be to use a three-level tree, where each level has 256 branches, for all possible values of one color dimension. A completely filled tree would have 256x256x256 nodes in its lowest level. After successful implementation, this history cache was used for the maximum likelihood algorithm as well.

- The possibility of keeping a history of distances between two pixels also presented itself. However, after some experiments, it was found that this did not bring any speed increases. The reason was that the time required to search for a pair of pixels in a huge history hash table was more than the time required to calculate the distance between them from scratch.

Note that most of the performance optimizations focus on the innermost loop, where most of the calculation takes place.

Performing multiple experiments with the KNN algorithm was only possible after these performance optimizations. In its original form, it would have taken several hours to execute a single KNN run. The final form is about 10 times as fast. In addition, the process of finding the k nearest neighbors of a given instance can be improved further by using spatial indices such as R-tree, which will help retrieve data items quickly according to their spatial location. For future work, the search process can be facilitated by using a spatial index.

4.1.7. Design and performance improvements for Leave One Out cross validation

To obtain more computing performance, the function was converted to C#, and the following improvements were made:

- All code relating to the algorithm, which can be seen as pseudocode in Figure 4.3, was encapsulated in its own class. The input parameters were made configurable. This means that any classification algorithm can be evaluated using this class.

- Components of error were calculated during the loop, which prevented having to go through the pixel estimates a second time.

- Note that history caching is impossible here, since each classification runs on a different training data.

- The performance advantage of the maximum likelihood algorithm was lost, since each classification runs on a different training data. Therefore, the mean and covariance has to be recalculated before each classification.

4.2. Maximum Likelihood Classification

The maximum likelihood algorithm was analyzed by examining various literatures on this topic. This algorithm is often used in land use classification, often to estimate forest areas. Therefore, it is the best candidate to compare with the KNN algorithm, in terms of estimation and computing performance. Initially, MATLAB was used to implement the algorithm, due to its clear and expressive syntax.

The maximum likelihood algorithm can be seen in Figure 4.15. It was found that the most complex part of the code was the NormalDistributionProbability() function. Here, the normal distribution function as described under the Maximum Likelihood Algorithm section was implemented. Since, in our case, each pixel is 3-dimensional (RGB), a modified version of the function for multidimensional feature sets was used.

It was noticed that the algorithm ran pretty fast, even for larger images. The reason is that most of the time-consuming parts are calculated in advance, such as the mean and the covariance matrix, and only once. It is also possible to calculate parts of the NormalDistributionProbability() function in advance, to increase

performance further. The output of the Maximum Likelihood classification is given in Figure 4.16.

```
function ML

f := ReadForestTrainingData
nf := ReadNonforestTrainingData
w := ReadWaterTrainingData

fMean := CalculateMean(f)
nfMean := CalculateMean(nf)
wMean := CalculateMean(w)

fCovar := CalculateCovarianceMatrix(f)
nfCovar := CalculateCovarianceMatrix(nf)
wCovar := CalculateCovarianceMatrix(w)

sat := ReadSatelliteImage
classified := new Image

for each column in sat
      for each row in sat

            testPixel := sat[column, row]
            fProbability :=NormalDistributionProbability(testPixel,
                        fMean, fCovar)
            nfProbability :=NormalDistributionProbability(testPixel,
                         nfMean, nfCovar)
            wProbability :=NormalDistributionProbability(testPixel,
                        wMean, wCovar)
            class := Max(fProbability, nfProbability, wProbability)
            classified[column, row] := DefaultColorPixel(class)

SaveImage(classified)
```

Figure 4.15. The pseudocode of Maximum Likelihood Classification

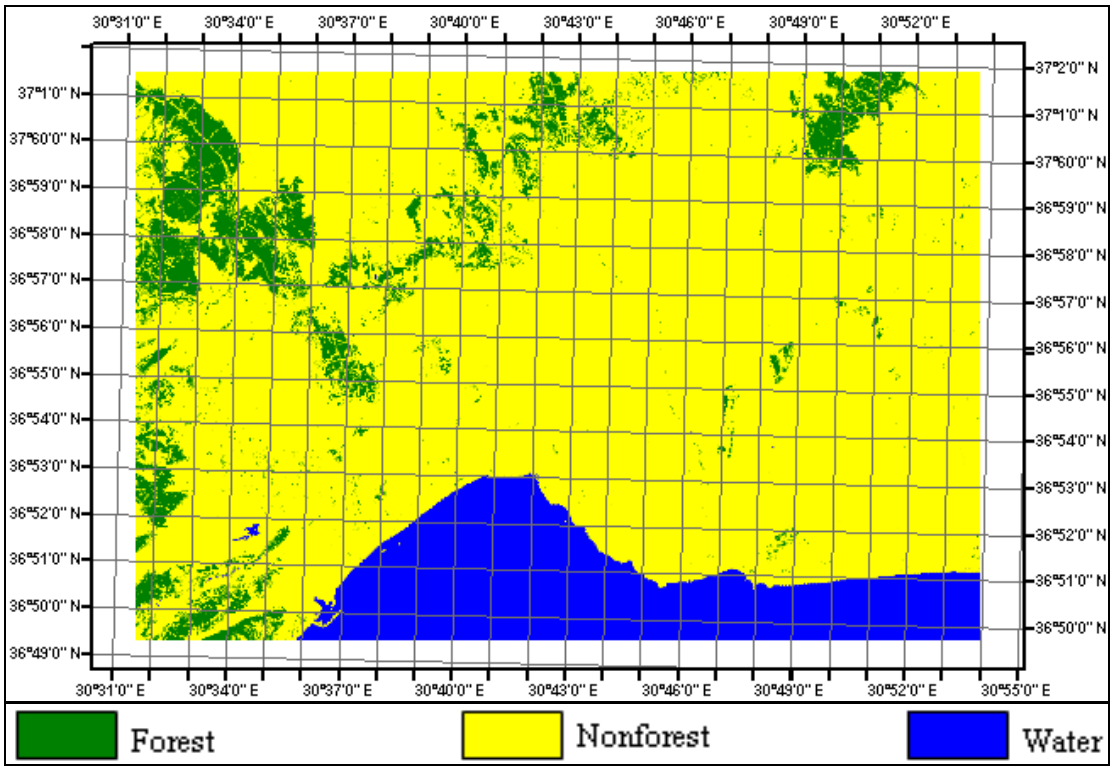

Figure 4.16. Landsat ETM (RGB; 432) with Maximum Likelihood Classification

4.2.1. Design and Performance Improvements for Maximum Likelihood Classification

To obtain more computing performance (as was necessary for the KNN algorithm), the function was converted to C#, and the following improvements were made:

- All code relating to the algorithm, which can be seen as Figure 4.15, was encapsulated in its own class, called Ml. The input parameters were made configurable.

- The algorithm was modified so that it would run with any number of classes, and therefore, with any number of training data sets. This may be useful in future studies, where a finer classification of forest or non-forest areas is desired.

- Some parts of the normal distribution equation were calculated in advance.

- Since C# does not contain useful matrix functions like MATLAB by default, a freeware matrix library called “CSML” was installed and used (URL2). Still, it is possible to increase performance further by implementing necessary matrix operations by hand, instead of using a generic library.

4.3. Other remarks

C# was chosen as language because of its following features:

- Object-orientation is very easy,
- There is automatic memory management, which guards against programmer errors,
- With JIT compilation, performance is high. With just-in-time compilation, the program code is completely translated into native machine code and cached before being executed. MATLAB, on the other hand, reinterprets each line or operand each time it is met. Machine code always runs faster than interpreted code.
- It is easy to find comfortable development environments,
- There are lots of useful math libraries that are built-in, although not as rich as MATLAB.
- There is lots of documentation, example code and free third party libraries available.

After implementing two classifier algorithms, it was noticed that they shared a lot of common functionality. This common code was factored out to a base class called Classifier. This may be useful in future studies, where other classifiers can be plugged in to this base class.

Although C# can only read images containing 3 bands (24 bits), the code was modified in a special way so that it became possible to read any number of bands. For this purpose, the user could prepare several bitmap files, each containing a different 3-band combination selected out of the total number of bands in the satellite image. The program would then read all of these bitmap files separately, and combine them in memory into a single image containing all bands. For example, 2 different bitmaps would contain a total of 6 bands.

Also, as the number of bands increases, the computational performance drops significantly, due to the following reasons:

1) The distance function in the innermost loop becomes more complex,
2) History caching is no longer useful. The probability that two pixels are exactly the same in 8 bands is much less than the probability that they are exactly the same in 3 bands. Therefore, most history searches fail. Also, because of having so many different values, the hash table becomes too large, or the search tree becomes too deep, making history searches unfeasible.

The classifier algorithms can be easily modified so that run in parallel. For example, in a dual CPU system, one CPU can work on classifying the upper half of the image, whereas the other CPU can work on classifying the lower half. For a test pixel, one CPU can calculate distances (or probabilities) from one half of the training data, whereas the other CPU can calculate distances (or probabilities) from the other half. Both CPU's can access the common training data, satellite image, classified image, neighbor list and classification history in memory. Matrix operations can also be easily parallelized. This concept can be scaled to any number of CPU's. For improving performance, parallelization is becoming more and more important, since the clock speeds of current CPU's are nearing the theoretical limit and manufacturers are focusing on multi-core architectures.

The 24-bit classified bitmaps generated by the classifiers were then saved as 256 color bitmaps using Paint.NET, to make accuracy assessment easier in ERDAS IMAGINE. In the whole classification process, it is important not to use lossy image file formats such as JPEG, since these may cause loss of color information.

CHAPTER 5

DISCUSSION OF THE RESULTS

K nearest neighbor algorithm with different metrics, different weights and different k values are applied to the Landsat ETM satellite image acquired in 2002. Band combination is selected for this reference image as (RGB; 432) because it is one of the most commonly used band combinations in the studies of forestry remote sensing. First, the best distance metric is selected with error results of Leave One Out cross validation technique by using bands (RGB; 432). The result is clearly Euclidean distance metric. After selecting the distance metric, the weights and the best k values are selected. With the same distance metrics, weights and k values of bands (RGB;432), The 6 bands which are 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, 7^{th} bands of Landsat ETM are selected as reference image for comparison. The 6^{th} band is not selected because thermal band decreases the accuracy. Additionally, Digital Elevation Model (DEM) data is used as the 4^{th} band and then as ancillary data to select training data better because, in the literature, it is expressed that the DEM data may increase the accuracy in some cases. The results of them are compared to understand which pixels are found with addition of DEM data.

The TM satellite image from the year 1987 is then selected as reference image to understand the change in forest percentage of both years. The bands are selected for this reference image as (RGB; 432).

Due to being the most commonly used supervised algorithm, Maximum Likelihood Algorithm, which is another supervised classification method, is applied to Landsat ETM (RGB; 432) to understand the proximity of the results between the KNN and Maximum Likelihood Classification methods.

5.1. Selecting the distance metric with LEAVE ONE OUT cross validation

The error values were calculated by using Leave One Out cross validation, for different values of k and for different distance metrics in Chapter 4. The best distance metric was selected as Euclidean. The weights, which are fraction, stairs, inverse distance and inverse square distance, were applied to Euclidean distance metric for several k values. It was seen that the best results was achieved by inverse distance weighted KNN with k value equal to 14.

In Franco-Lopez et al. (2001), KNN is applied by using Euclidean and Mahalanobis distance metrics for the estimation of the forest stand, type and volume. In that study, it is seen that Euclidean distance metric gives lower RMSE values than Mahalanobis distance for any number of neighbors. Therefore, the subsequent trials are based on Euclidean distance. It is also mentioned that the results contradict with the results found by Nilsson (1997), in Franco-Lopez et al. (2001).

Similarly, in this study, the Euclidean distance metric was found as the best metric, and it was taken into consideration for further classification tasks. This agrees with Franco-Lopez et al. (2001), Pereira (2006) and Cabaravdic (2007). However, it contradicts with the results found by Nilsson (1997), in Franco-Lopez et al. (2001), and Lorenzo et al. (2002), both of whom found Mahalanobis distance to be better.

The Mahalanobis distance metric would be expected to perform better when bands are strongly correlated. The bad performance of this metric in this study is an indication that the bands 4, 3 and 2 are not strongly correlated. This argument can also explain the higher performance of the Mahalanobis distance metric found by Lorenzo et al. (2002).

Diagonal (Class-Dependent) Mahalanobis distance metric performed better than Mahalanobis distance. This can be explained by the fact that this metric does not take band correlation into account. However, it still exhibited very low performance. The reason can be understood by noting that the distance of a test pixel to a training pixel is inversely proportional to the standard deviation of the training data which the training pixel comes from. This means that if a training data of a certain

class has a high standard deviation, then its pixels will have especially low distances to the test pixel. The underlying assumption is that the test pixel would fit a wide and flat distribution more easily than a narrow and thin distribution. In our case, the training data for the water and forest classes both have low standard deviation, but the training data for the non-forest class have an especially high standard deviation, since they are collected from widely different land features. Therefore, the closest neighbors to a test pixel tend to be chosen from non-forest pixels, due to their low distances. The non-forest class tends to overwhelm forest and water classes in the list of neighbors, resulting in an unfair advantage for the non-forest class. Many forest and water test pixels are mistakenly classified as non-forest, which decreases the accuracy.

The Manhattan distance metric gave pretty good results. The difference to the Euclidean distance metric is as follows: In a 3-band combination, if one of the bands of a training pixel has a high distance to the same band of the test pixel, then that training pixel is severely penalized by the Euclidean distance. A band with a high difference cannot be offset by another band with a low difference. On the other hand, Manhattan distance allows a band with a high difference to be offset by another band with a low distance. Even if there is a band with a high difference, Manhattan distance accepts the overall distance to be low, if the other bands have a low difference. For the Euclidean distance, all bands have to be low in order for the distance to be low. In other words, the closeness has to be consistent across all bands. Since the Euclidean distance metric gave the best results in our case, it was observed that it was better to expect a training pixel to be close to the test pixel in all of the bands 4, 3 and 2, in order to identify it as a neighbor. A high distance in one of the bands had to be penalized severely, even if the other bands had really low distances. Consistency in distance was important.

On the other hand, the Manhattan distance metric consists entirely of additions and subtractions, whereas the Euclidean distance metric includes slow-running multiplications and square roots. Therefore, in cases where computing performance must be maximized, the Manhattan distance is recommended over the Euclidean distance, considering its high accuracy.

5.2. Weighted KNN Application

The best result is obtained when the weight function is “Inverse Distance.” This holds true for every distance metric, but the focus here is on the Euclidean distance metric, which gives the highest accuracy. The superiority of “Inverse Distance” is contrary to the results obtained by Franco-Lopez et al. (2001), who observed that it was best to use no weights (equal weights). In their experiments, each test pixel had several close neighbors in most cases, with very little distance between them. Therefore, a distance-dependent weight function did not generate different results than assigning equal weights to all neighbors. In such cases, using fraction or stairs weight function is recommended. These functions are not dependent on the distance. Rather, the weight of a neighbor drops as its order increases among the list of neighbors, without exception.

It should be noted that the error drops steeply as k increases from 1 to 4. For subsequent values of k, the error drop is more gradual and irregular. This rapid early decrease, followed by diminishing marginal drops is mentioned by Franco-Lopez et al. (2001) as well.

It was also observed that there was very little difference between “Inverse Distance” and “Inverse Square Distance” weight functions. This agrees with the observations made by Haapanen (2004).

Although there were good and bad weight functions, it was usually not possible for an algorithm with a bad distance function to become superior to an algorithm with a good distance function, even if it used the best weight function.

In order to demonstrate the role of the distance value inside the weight function more clearly, it is necessary to observe the accuracy results for different values of k and different weight functions under Mahalanobis distance, which is given in Figure 5.1. This is a bad distance function and tends to collect misleading neighbors. Therefore, for all weight functions, the error increases as the value of k increases. Higher values of k cause neighbors that are really far away to be taken into consideration, which confuse the classifier even more. If there is no weight

function (all neighbors have the same weight), then the error increase (accuracy drop) is very steep. However, if the weight is calculated from the distance, the error increase is negligible, because if the weights of misleading neighbors are correspondingly low, then the confusion is also low. Therefore, a weight function that is inversely proportional to the distance is definitely recommended.

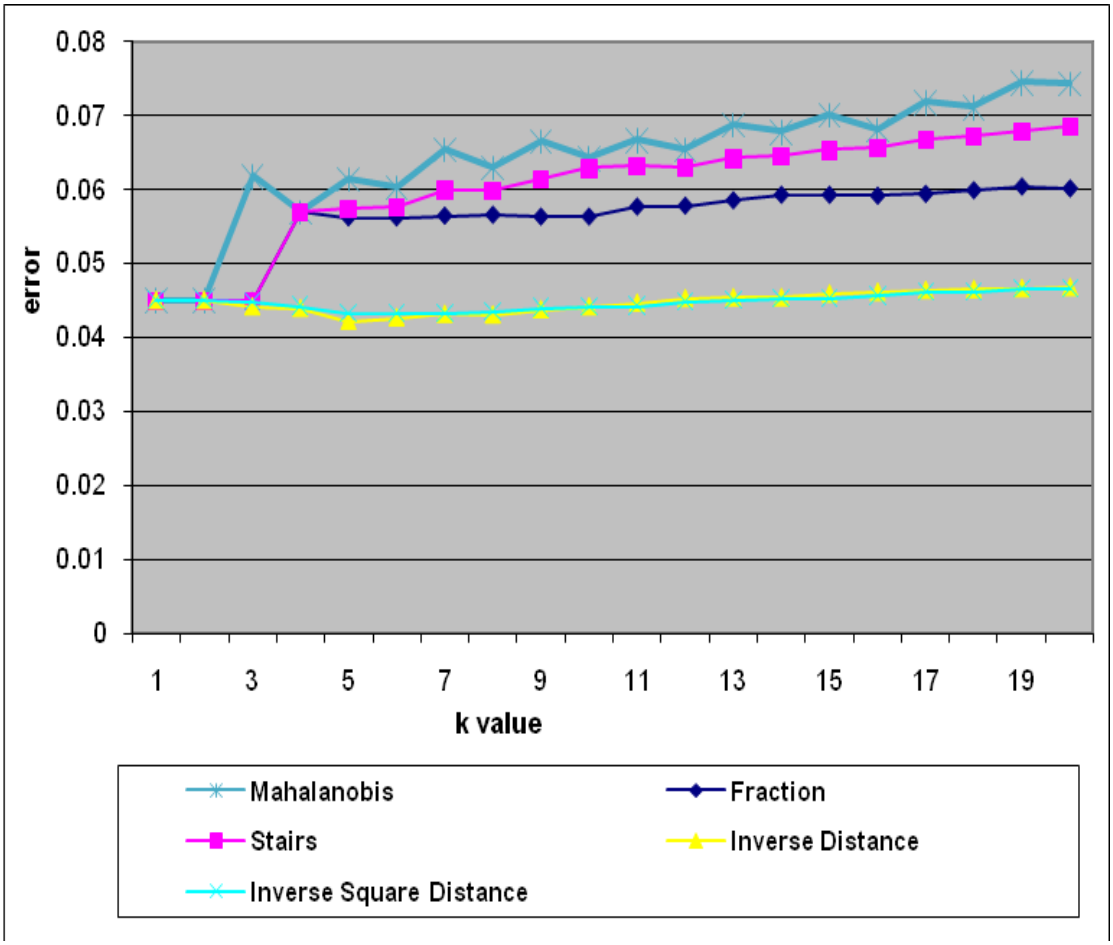

Figure 5.1. 2002 Landsat ETM satellite imagery Mahalanobis and Weighted Mahalanobis KNN classification error plot (RGB; 432).

For “Inverse Distance,” the k value with the minimum error was found to be 14. This agrees with Haapanen (2004), where it is mentioned that typically, the value of k employed in forest inventory studies has ranged from 1 to 15. For example, in Franco-Lopez et al. (2001), it is cited that Nilsson (1997) and Tokola et al. (2006) reported a stability point between 10 and 15 neighbors. Note that if a bad distance function is used, and there is no weight function, higher values of k can be very dangerous to the accuracy of the classifier.

In all measurements, the error for k = 2 was exactly equal to the error for k = 1. The reason is that the first neighbor's class always carries more weight than the second neighbor's class when a weight function is used. Also, in our version of the algorithm, if equal weights are used, the first neighbor's class overrides the second neighbor's class in case of a tie. Therefore, the second neighbor is completely ineffective.

With the optimum parameters, the Leave One Out error for KNN was 0.009174, whereas the same error for the Maximum Likelihood algorithm was found to be 0.0168. It was seen that KNN achieves better performance than Maximum Likelihood when classifier parameters are carefully selected.

5.3. DEM addition to KNN classification process

It is specified in section 3.7; the ancillary data can be also used as additional information in the KNN classification. Therefore, ETM (RGB; 432) was classified by using DEM as the 4^{th} band firstly.

In Figure 5.2, it can be seen that the classification of Landsat ETM (RGB; 432) with Inverse Distance Weighted Euclidean metric with DEM data failed to find the forest areas inside the black circle. The green pixels indicate areas additionally classified as forest, and red pixels indicate areas no longer classified as forest. This is due to the restrictive nature of the forest training data. The forest training data had been taken from areas where elevation was high. Therefore, KNN tended towards expecting more forest in high areas, and less forest in low areas. The algorithm was able to find forest near the mountaintops on the lower left successfully. It also found more forests towards the top left, where elevation is high. However, it ignored the area in the black circle, where elevation is low. The accuracy is still very high, because the random points did not include many pixels from the wrongly-classified areas. This shows that accuracy assessment can be misleading due to an unfortunate distribution of sample points. Because of this wrong classification, new forest training area was taken from this region for detecting the

forests in areas which had varying elevation values. The new classification results were given in Figure 5.3 and Figure 5.4.

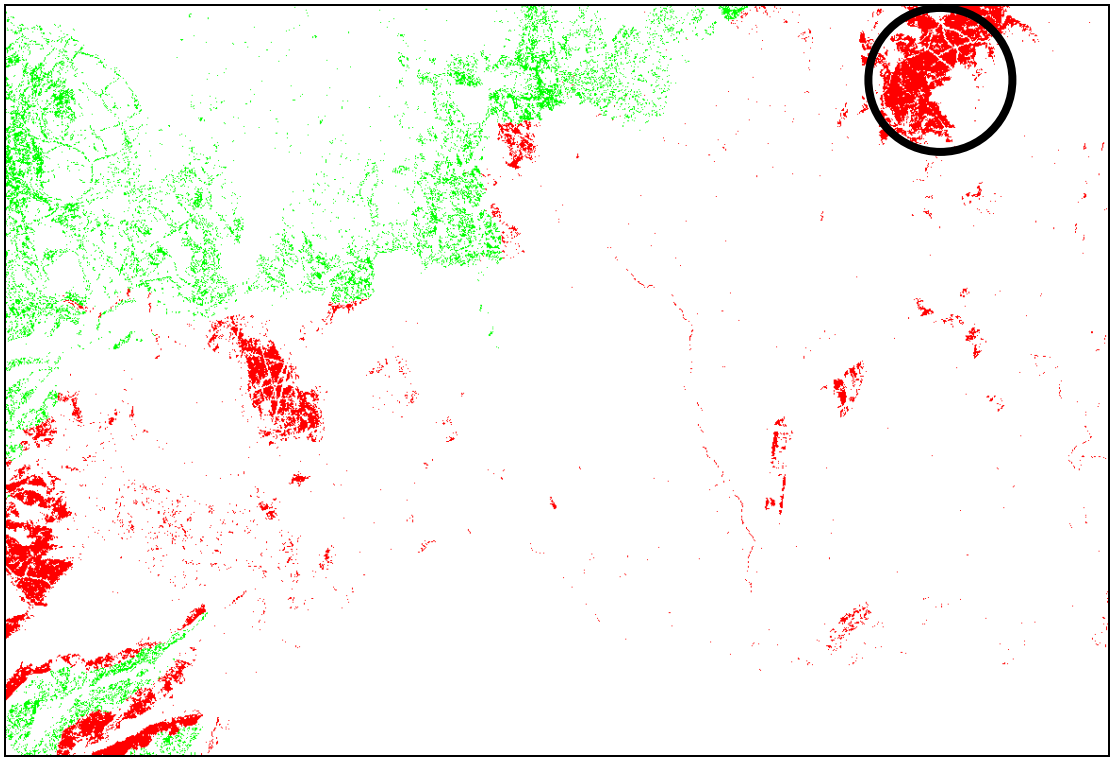

Figure 5.2. The changes in pixels with addition of DEM data as an extra band.

It was observed that this time, classification with DEM was able to find forests in the black circle successfully, which is shown in Figure 5.5. But it was also seen that some non-forest, low elevation areas were mistakenly classified as forest this time. Without the DEM band, the KNN algorithm had seen that the spectral values of these areas were sufficiently far from forest pixels. The DEM, used as an extra band, gave an additional hint that the areas were closer, because their low elevation matched that in the forest training data. As a result, they were classified as forest, and the accuracy decreased. It is seen that using DEM as an extra band only confuses the reliable information obtained from the spectral values. In areas where the different classes are strongly dependent on elevation such as different vegetation types, or special types of forest, the DEM band can be useful. However, both forest and non-forest classes were found on varying elevation levels in our study, which caused the DEM band to be impractical.

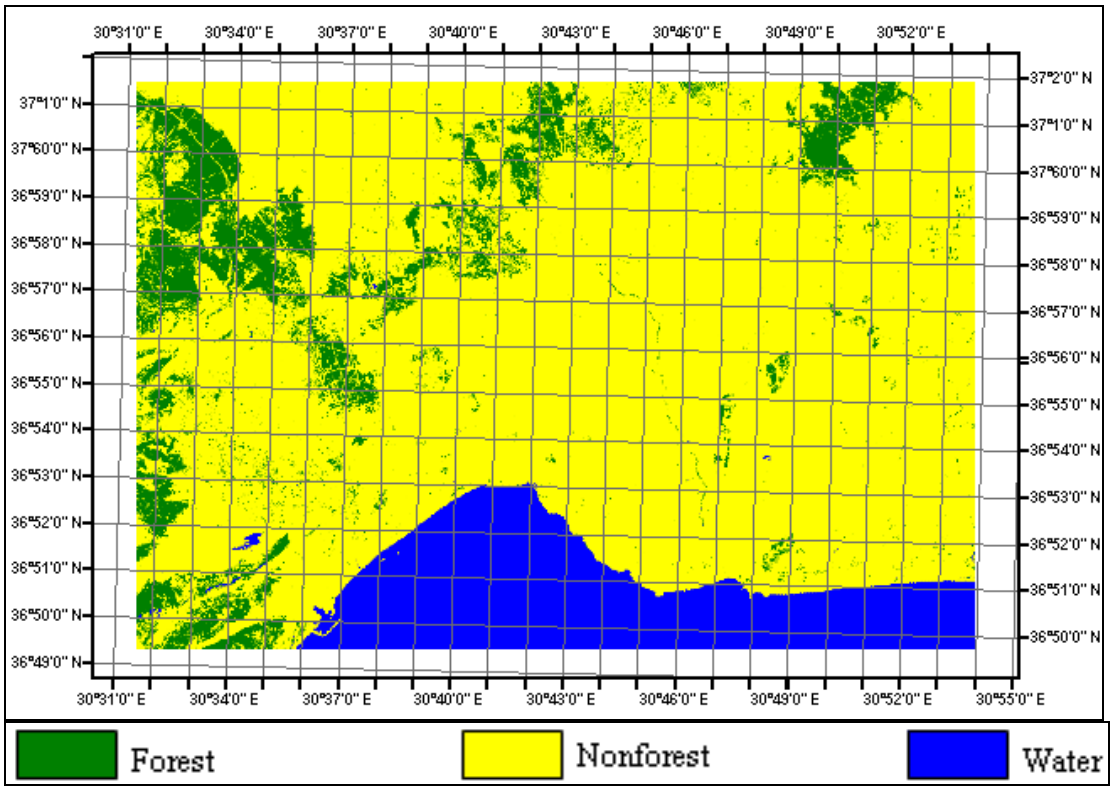

Figure 5.3. Landsat ETM (RGB; 432) with Inverse Distance Weighted Euclidean distance metric (k=14) by using new forest training data.

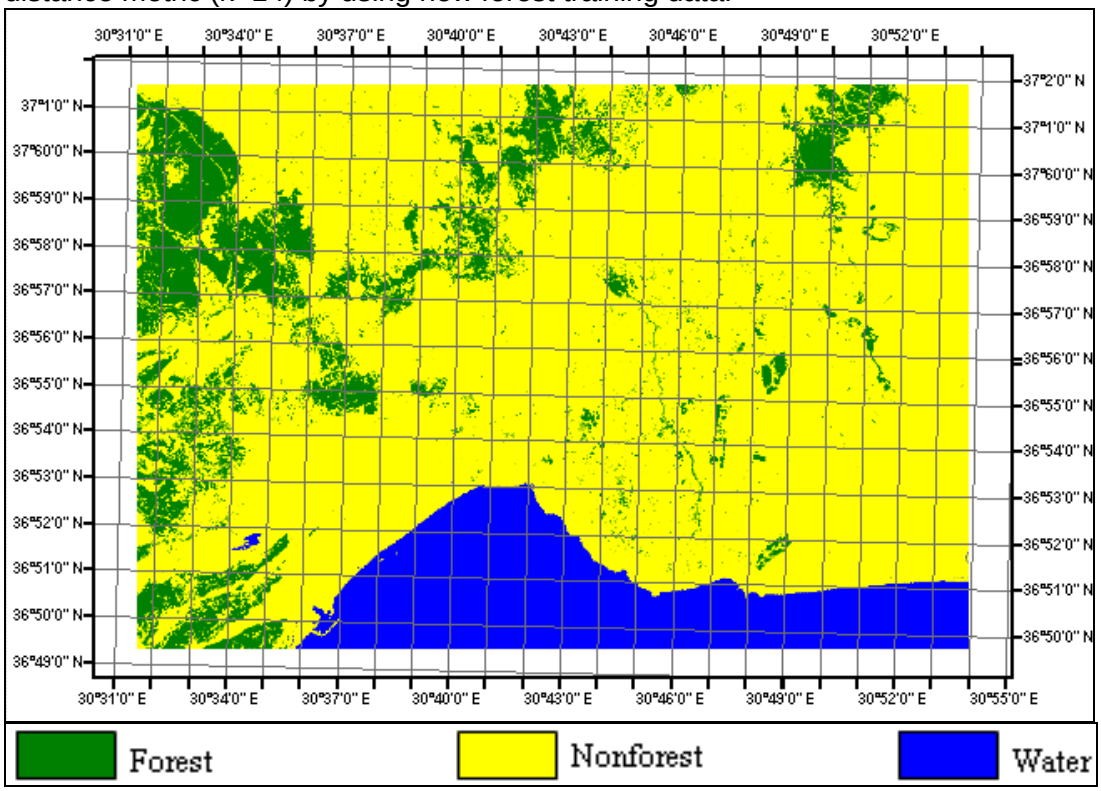

Figure 5.4. Landsat ETM (RGB; 432 and DEM) with Inverse Distance Weighted Euclidean distance metric (k=14) by using new forest training data.

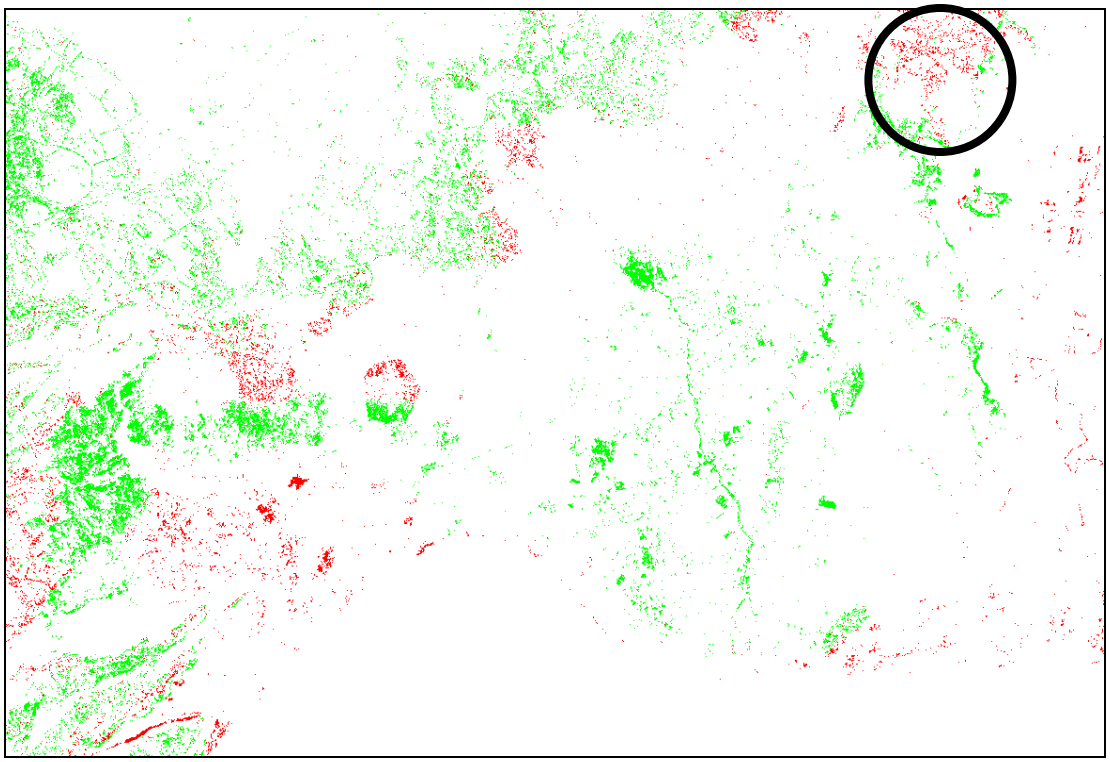

Figure 5.5. The changes in pixels with addition of DEM data as an extra band by using new forest training areas.

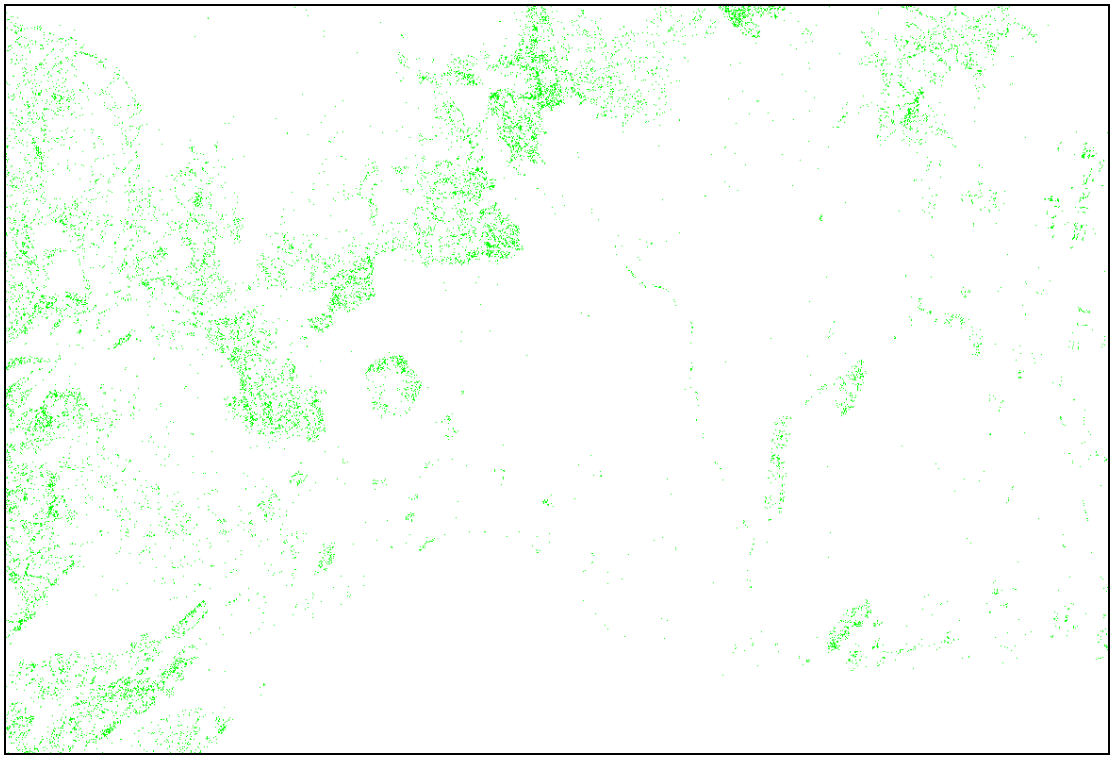

Figure 5.6. The changes in pixels with addition of DEM data as an ancillary data to select training data using new forest training areas.

On the other hand, the classification results using the new training data were better than the results using the old training data. When DEM was not used as 4th band, there was no pixel which could not be found as forest compared to other classified images. The extra forest pixels that were found using the new training data can be seen in Figure 5.6. This can demonstrate that by selecting the training data from different elevations (by observing the DEM band as auxiliary data), the accuracy can be increased. This indicates that forest areas reflect slightly different spectral values at different elevations, and these variations should be taken into account when selecting the training data.

5.4. Forest change from 1987 to 2002

Change detection is a tool to make decisions about development strategies. Forest change is a result of natural and man-caused changes. In this study, Landsat TM image was acquired in 1987 and Landsat ETM was acquired in 2002. There has been probably a change in forest areas in the study region during this fifteen-year period. As it was explained in section 3.1, histogram matching as a radiometric correction technique was applied to Landsat TM to classify both satellite images by using the same training data. KNN was applied to both of them by using Inverse Distance Weighted Euclidean distance metric with k equal to 14. The different pixels in forest areas were compared in Figure 5.7.

The red pixels in the pink and brown circles show the forest pixels which exist in 1987 but not in 2002. The pink circle indicates the forest areas in 1987 which became agricultural areas in 2002. The brown circles show the forest areas in 1987 that are apparently still forest in 2002, since these areas are forest in land use data. The pixels are red because the digital values of these pixels in 2002 are very different from forest training pixels that represent most forest areas in the satellite images. Forest training data have not been collected from the pixels in brown circles. Two conclusions can be reached from here. First, the training data should consist of all types of forest at the same time for better accuracy. Second, the land use data should give correct information. There are some regions which are seen as forest in land use, but these regions obviously do not have forest pixels in satellite images. It can be seen that the land use data are not very reliable

for accuracy assessment. Field survey and sampling plots, which consist of training data, are recommended for better accuracy.

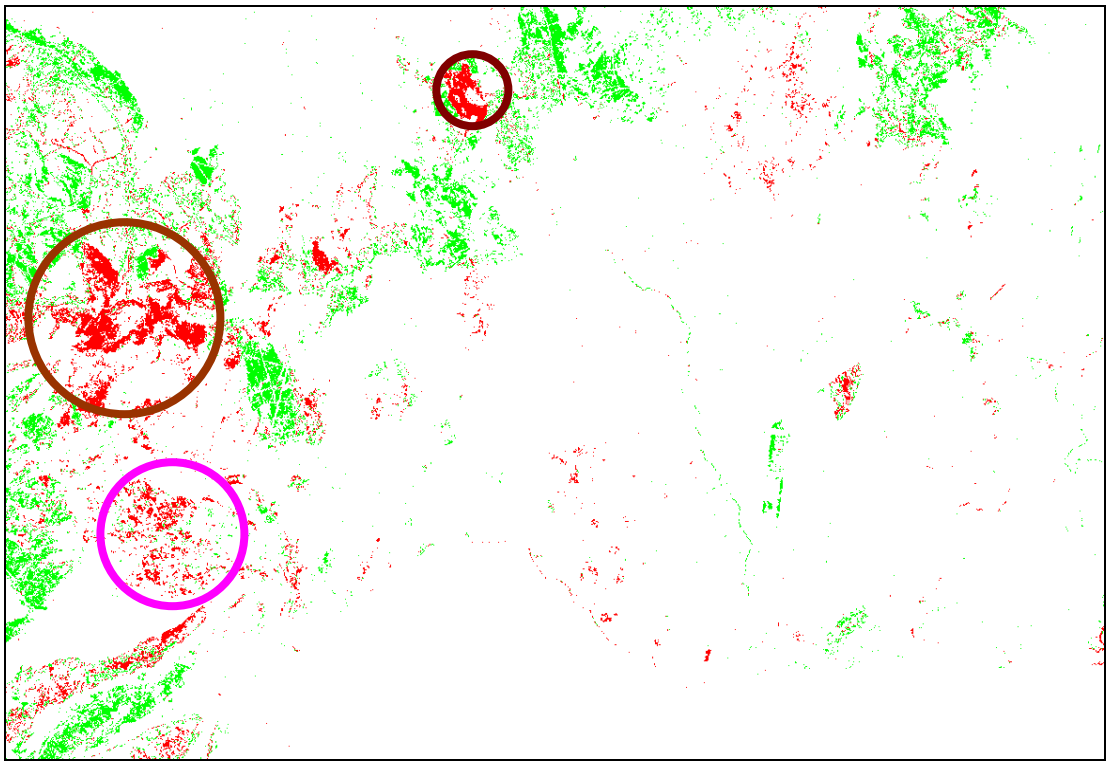

Figure 5.7. Change in forest pixels from 1987 to 2002.

Table 5.1. Class distribution (%) of TM and ETM classified images

	TM	**ETM**
Forest	7.07	8.61
Nonforest	80.57	79.05
Water	12.36	12.34

Forest areas increase from 7.07% to 8.61%. This means that the change in forest areas is approximately 21%. Although there is a rapid urbanization in Antalya, the increase in forest areas is an indicator of sustainable development.

5.5. KNN and Maximum Likelihood Classification Comparison

It can be seen in Figure 5.8 that there are some pixels in KNN different from Maximum Likelihood Classification. The green pixels are found as forest by Maximum Likelihood classification and not found as forest by KNN classification.

Similarly, red pixels are found as forest by KNN classification and not found as forest by Maximum Likelihood classification. There is not much difference between the two results. KNN found slightly more forest pixels in the top left and lower left areas. This is in agreement with the land use data, which shows these areas as forest. The spectral values of the pixels in these areas deviate from typical forest pixels. This shows that KNN is more successful in classifying such outlying pixels. Maximum Likelihood, on the other hand, found slightly more pixels in the middle left and top areas. The land use data shows these areas as forest as well. The spectral values of pixels from these areas are typical forest pixels. This indicates that Maximum Likelihood is more successful in classifying pixels that are closer to the mean.

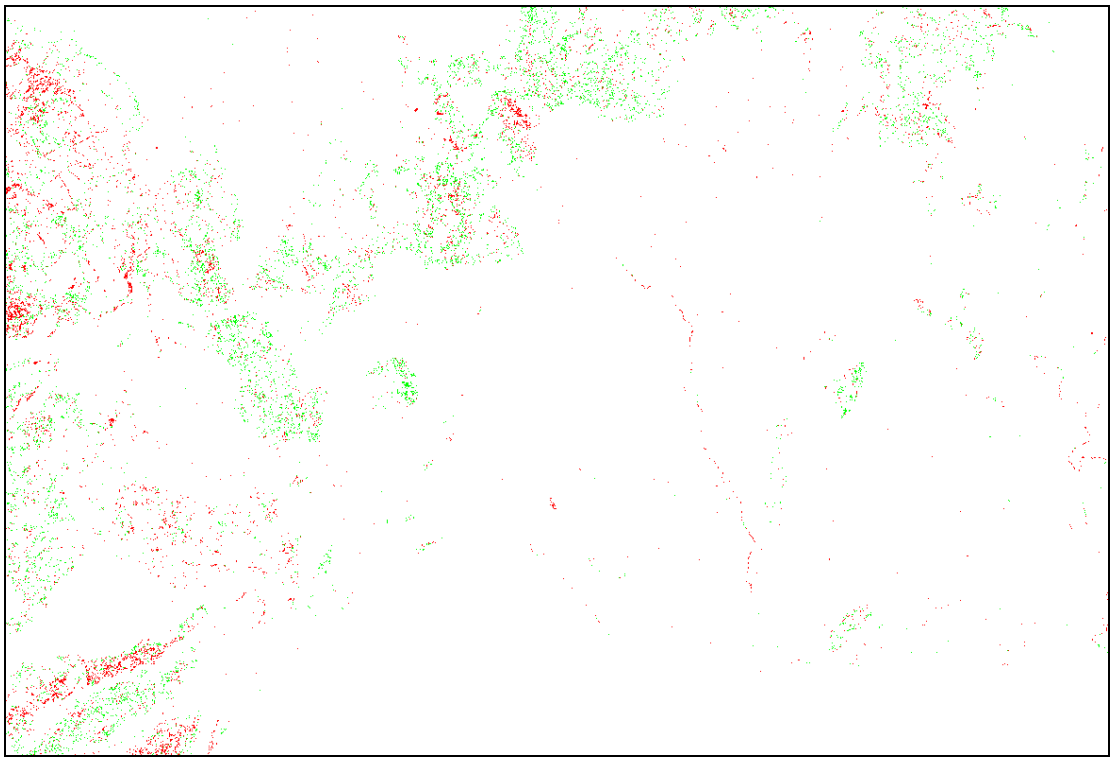

Figure 5.8. Different forest pixels in images classified by KNN and Maximum Likelihood classifications.

The overall accuracies of KNN and Maximum Likelihood classification are more or less the same for the random points in the accuracy assessment part of the study.

5.6. Accuracy Assessment

The accuracy assessment was performed with ERDAS Imagine by comparing the land use data with classified images. The two images were geometrically corrected so that they would overlap perfectly. Then, stratified random points calculated as 280 points were selected automatically on the classified image, and these were manually classified by referring to the land use data. The stratified random points can be seen in Figure 5.9.

Figure 5.9. Stratified random sample points used in accuracy assessment.

The error matrix and other accuracy values were automatically computed by using ERDAS. The highest accuracy values were obtained with the band selection RGB; 432 when using the KNN algorithm, with Euclidean distance and inverse distance weights, with k = 14.

The accuracy assessment reports of ETM (RGB; 432) KNN with Inverse Distance Euclidean and ETM (RGB; 321) KNN with Inverse Distance Euclidean can be seen in Figure 5.10 and Figure 5.11 show that overall, using (RGB; 432) bands give higher accuracy than using (RGB; 321) bands.

The reason for this is that band 4 is more significant than band 1 for classifying vegetation. Band 4 is “near-infrared.” Agricultural vegetation and forest vegetation reflect clearly different spectral values under infrared band. Therefore, it becomes easier for the classifier to detect the difference between agriculture (non-forest) and forest pixels. The accuracy reports of ETM (1;2;3;4;5;7) with KNN with Inverse Distance can be seen in Figure 5.12. Selecting all available spectral bands gave the same accuracy. It was decided that it is not worth to perform extra effort and computing performance for these bands. The selection of a small number of useful bands, such as (RGB; 432), was sufficient. The accuracy reports of DEM data added to ETM (RGB; 432) with KNN with Inverse Distance Euclidean can be found in Figure 5.13. Figure 5.14 and Figure 5.15 contain accuracy reports using the new forest training data selected by taking elevation values into consideration. These results were discussed in section 5.3. Figure 5.16 contains the ETM (RGB; 432), Maximum Likelihood classification accuracy reports. It was seen that there is slight difference between accuracies of KNN algorithm and maximum likelihood algorithm. Overall accuracies of the methods followed in this study can be seen in Table 5.2.

Table 5.2. Overall accuracies of different classification, band and ancillary data combinations.

Bands	**Ancillary Data**	**Classification**	**Overall Accuracy (%)**
RGB; 321	Landuse	Inverse Distance Weighted KNN	85,36
RGB; 432	Landuse	Inverse Distance Weighted KNN	86,07
1,2,3,4,5,7	Landuse	Inverse Distance Weighted KNN	86,07
RGB; 432	Landuse	Maximum Likelihood	87,14
RGB; 432 and DEM	Landuse& DEM	Inverse Distance Weighted KNN	87,14
RGB; 432 and DEM	Landuse	Inverse Distance Weighted KNN	87,50
RGB; 432	Landuse& DEM	Inverse Distance Weighted KNN	87,86

ERROR MATRIX

	Reference Data		
Classified Data	Forest	Nonforest	Water
Forest	20	2	0
Nonforest	35	186	1
Water	0	1	35
Column Total	55	189	36

ACCURACY TOTALS

Class Name	Reference Totals	Classified Totals	Number Correct	Producers Accuracy	Users Accuracy
Forest	55	22	20	36,36%	90,91%
Nonforest	189	222	186	98,41%	83,78%
Water	36	36	35	97,22%	97,22%
Totals	280	280	241		

Overall Classification Accuracy = **86,07%**

KAPPA (K^) STATISTICS

Overall Kappa Statistics = **0,6782**

Conditional Kappa for each Category.

Class Name	Kappa
Forest	0,8869
Nonforest	0,5010
Water	0,9681

Figure 5.10. ETM (RGB; 432) KNN with Inverse Distance Weighted Euclidean distance metric classification Accuracy Assessment Report.

ERROR MATRIX

	Reference Data		
Classified Data	Forest	Nonforest	Water
Forest	20	4	0
Nonforest	35	184	1
Water	0	1	35
Column Total	55	189	36

ACCURACY TOTALS

Class Name	Reference Totals	Classified Totals	Number Correct	Producers Accuracy	Users Accuracy
Forest	55	24	20	36,36%	83,33%
Nonforest	189	220	184	97,35%	83,64%
Water	36	36	35	97,22%	97,22%
Totals	280	280	239		

Overall Classification Accuracy = **85,36%**

KAPPA (K^) STATISTICS

Overall Kappa Statistics = **0,6644**

Conditional Kappa for each Category.

Class Name	Kappa
Forest	0,7926
Nonforest	0,4965
Water	0,9681

Figure 5.11. ETM (RGB; 321) KNN with Inverse Distance Weighted Euclidean distance metric classification Accuracy Assessment Report.

ERROR MATRIX

	Reference Data		
Classified Data	Forest	Nonforest	Water
Forest	20	2	0
Nonforest	35	186	1
Water	0	1	35
Column Total	55	189	36

ACCURACY TOTALS

Class Name	Reference Totals	Classified Totals	Number Correct	Producers Accuracy	Users Accuracy
Forest	55	22	20	36,36%	90,91%
Nonforest	189	222	186	98,41%	83,78%
Water	36	36	35	97,22%	97,22%
Totals	280	280	241		

Overall Classification Accuracy = 86,07%

KAPPA (K^) STATISTICS

Overall Kappa Statistics = 0,6782

Conditional Kappa for each Category.

Class Name	Kappa
Forest	0,8869
Nonforest	0,5010
Water	0,9681

Figure 5.12. ETM (Bands:1;2;3;4;5;7) with Inverse Distance Weighted Euclidean distance metric classification Accuracy Assessment Report

ERROR MATRIX

Classified Data	Reference Data: Forest	Nonforest	Water
Forest	22	0	0
Nonforest	33	188	1
Water	0	1	35
Column Total	55	189	36

ACCURACY TOTALS

Class Name	Reference Totals	Classified Totals	Number Correct	Producers Accuracy	Users Accuracy
Forest	55	22	22	40,00%	100,00%
Nonforest	189	222	188	99,47%	84,68%
Water	36	36	35	97,22%	97,22%
Totals	280	280	245		

Overall Classification Accuracy = **87,50%**

KAPPA (K^) STATISTICS

Overall Kappa Statistics = **0,7112**

Conditional Kappa for each Category.

Class Name	Kappa
Forest	1,0000
Nonforest	0,5288
Water	0,9681

Figure 5.13. DEM data added to ETM (RGB; 432) with Inverse Distance Weighted Euclidean distance metric classification Accuracy Assessment Report

ERROR MATRIX

	Reference Data		
Classified Data	Forest	Nonforest	Water
Forest	26	3	0
Nonforest	29	185	1
Water	0	1	35
Column Total	55	189	36

ACCURACY TOTALS

Class Name	Reference Totals	Classified Totals	Number Correct	Producers Accuracy	Users Accuracy
Forest	55	29	26	47,27%	89,66%
Nonforest	189	215	185	97,88%	86,05%
Water	36	36	35	97,22%	97,22%
Totals	280	280	246		

Overall Classification Accuracy = **87,86%**

KAPPA (K^) STATISTICS

Overall Kappa Statistics = **0,7270**

Conditional Kappa for each Category.

Class Name	Kappa
Forest	0,8713
Nonforest	0,5707
Water	0,9681

Figure 5.14. ETM (RGB; 432) KNN with Inverse Distance Weighted Euclidean distance metric classification (by using new forest training data) Accuracy Assessment Report.

ERROR MATRIX

	Reference Data		
Classified Data	Forest	Nonforest	Water
Forest	27	6	0
Nonforest	28	182	1
Water	0	1	35
Column Total	55	189	36

ACCURACY TOTALS

Class Name	Reference Totals	Classified Totals	Number Correct	Producers Accuracy	Users Accuracy
Forest	55	33	27	49,09%	81,82%
Nonforest	189	211	182	96,30%	86,26%
Water	36	36	35	97,22%	97,22%
Totals	280	280	244		

Overall Classification Accuracy = **87,14%**

KAPPA (K^) STATISTICS

Overall Kappa Statistics = **0,7153**

Conditional Kappa for each Category.

Class Name	Kappa
Forest	0,7737
Nonforest	0,5771
Water	0,9681

Figure 5.15. DEM data added to ETM (RGB; 432) with Inverse Distance Weighted Euclidean distance metric classification (by using new forest training data) Accuracy Assessment Report.

ERROR MATRIX

	Reference Data		
Classified Data	Forest	Nonforest	Water
Forest	23	2	0
Nonforest	32	186	1
Water	0	1	35
Column Total	55	189	36

ACCURACY TOTALS

Class Name	Reference Totals	Classified Totals	Number Correct	Producers Accuracy	Users Accuracy
Forest	55	25	23	41,82%	92,00%
Nonforest	189	219	186	98,41%	84,93%
Water	36	36	35	97,22%	97,22%
Totals	280	280	244		

Overall Classification Accuracy = 87,14%

KAPPA (K^) STATISTICS

Overall Kappa Statistics = 0,7064

Conditional Kappa for each Category.

Class Name	Kappa
Forest	0,9004
Nonforest	0,5364
Water	0,9681

Figure 5.16. ETM (RGB; 432) Maximum Likelihood Classification Accuracy Assessment Report.

CHAPTER 6

CONCLUSIONS and RECOMMENDATIONS

The application of image classification algorithms to satellite images makes it possible to map land cover types of huge areas automatically. In the particular field of forest area detection, appropriate classification methods help improve forest planning. A lot of research on the area of image classification focuses on improving classification accuracy and thus increasing its applicability for practical use. This study detailed the KNN algorithm in particular, which is commonly used in the detection of forest areas.

6.1. Conclusions

The study area was the Center District of Antalya, which has undergone high urbanization over recent years. Using different parameters, the accuracy of the KNN algorithm was evaluated by means of Leave One Out cross validation. Among the four different distance functions that were used in the KNN algorithm, the best function turned out to be the simple Euclidean distance, followed by the Manhattan distance. There was a very clear accuracy weakness with the Mahalanobis distance. This showed that the Landsat bands 4, 3 and 2 are not strongly correlated. Diagonal Mahalanobis, which did not take band correlation into account, performed better. However, it was still unsatisfactory, due to the high standard deviation of non-forest training data. If more granular classes were used for non-forest areas, Diagonal Mahalanobis could have performed better.

Among the different weight factors (functions) that were used in the KNN algorithm, the best function turned out to be the inverse distance, followed by the inverse square distance. Both were better than using no weight factors.

It was noticed that the choice of distance function had a significant effect on the resulting accuracy. The choice of weight function did not have such a strong impact. Therefore, it is crucial to select a good distance function first, before choosing a good weight function.

In addition, for the best distance and weight functions, the best value for k was found to be 14.

It was observed that choosing bands 4, 3 and 2 resulted in higher accuracies than choosing bands 3, 2 and 1. This was due to the importance of band 4 in detecting vegetation. As expected, using all available bands did not increase the accuracy, so just 3 bands were sufficient. Also, it was observed that adding DEM as an extra band did not improve classification accuracy. DEM would be much more useful for classifying forest or vegetation types that are closely dependent on elevation. Alternatively, it can increase accuracy when it is observed as ancillary data to specifically select training pixels from varying elevations. This indicates that performing topographic normalization using DEM data could have increased accuracy results.

As a result, the best flavor of KNN can be achieved by taking Euclidean distances, inverse distance weights, k equal to 14, with bands 4, 3 and 2. With these optimizations, it was found that KNN has an advantage over the Maximum Likelihood algorithm, which is also a classification method commonly used for forest area detection.

In the accuracy assessment done by comparing the classified image with official land use data, it was found that there is slight difference between optimized KNN result and the Maximum Likelihood result. Also, it was observed that KNN was more successful in classifying pixels that deviate significantly from the corresponding training data, whereas Maximum Likelihood was more successful in classifying pixels that are closer to the mean of the corresponding training data. In the literature, the KNN method has been applied while using grids of sample plots, with successful results. However, the land use data available for this study was not as precise as the plots used in those cited studies. If more precise and detailed fieldwork had been performed, then it would have been possible to select better

training data, and perform accuracy assessment against a more correct reference image. Thus, the accuracy results of KNN could have been better.

It was seen that both algorithms were straightforward to implement in code, but one needs a good design in order to make the algorithms configurable. Also, it was observed that the computing performance of KNN is low in comparison to the Maximum Likelihood algorithm. It was seen that the running time of KNN was highly dependent on the number of training samples. Increasing the number of bands, using Mahalanobis distance function, and adding weight calculation all increased the running time. Ways to improve this performance have been discussed. The optimized source code has been included, which can be easily manipulated to modify classification parameters or add new image classifiers. Thus, it can act as a high-performance library for future studies.

A secondary objective of this study was to classify two satellite images from two different years, and detect the change percentage of forest areas. This required applying radiometric correction to the images. Histogram matching was used for this purpose. After radiometric correction, it was observed that the optimized KNN had no problems classifying both images based on the training data taken from one of them. However, it was not possible to measure the accuracy of the second classified image due to the lack of land use data.

When performing change detection between the TM and ETM images, it was found that the forest areas actually increased approximately 21% during the urbanization phase of Antalya over the years. This shows that in spite of massive urbanization, reforestation strategies in the Center District of Antalya have been somewhat successful. This is valuable data for decision makers who want to plan future strategies for forest management.

As a result, the practical applicability of the KNN algorithm for mapping forest areas over large areas was demonstrated, with acceptable accuracy and computing performance. It is anticipated that progress in this field will continue, especially in other possible modifications of the classifier. The final objective is the operational application of classifiers in creating and maintaining forest inventories.

6.2. Recommendations

For forest/non-forest/water stratification, the KNN algorithm is recommended due to its advantages listed in Chapter 2 and its high accuracy demonstrated in Chapter 5.

In classification using KNN, Euclidean distance should be used for bands that are not strongly correlated, such as bands 4, 3 and 2. If computing performance is more important than maximum accuracy, then Manhattan distance can also be used. For strongly correlated bands, Mahalanobis distance should be used. In that case, care should be taken to use granular classes, so that the standard deviation of a single class does not become too large. When selecting training data, it is important to include pixels from all variations of a class.

For highest accuracy in KNN during forest/non-forest/water stratification, it is strongly recommended to have neighbor weights decrease in inverse proportion to their distances from the test pixel. This will also help reduce the accuracy problems that can arise if the value of k is selected too big. However, if each test pixel has several close neighbors in most cases, with very little distance between them, then functions that are not dependent on the distance are recommended, such as fraction or stairs. Also, the value of k should be selected around 14, and the bands 4, 3 and 2 should be used.

When classes are not dependent on elevation, it is not recommended to use DEM as a fourth band; however, it can increase accuracy when used as ancillary data during training data selection or in topographic correction.

For maximum reliability when selecting training data and performing accuracy assessment, it is important to have precise and up-to-date land use data.

For reducing the running time of the KNN algorithm, it is recommended to keep the size of the training data below an acceptable limit. Also, the search process of k nearest neighbors can be facilitated by using a spatial index such as R-tree for future work.

REFERENCES

American Institute of Biological Sciences, http://www.actionbioscience.org/environment/nilsson.html, last accessed date: 10.04.2008. (URL1)

Berrueta, L. A., Alonso-Salces, R. M. and Heberger, K., 2007. Supervised pattern recognition in food analysis. Journal of Chromatography A, Vol. 1158, pp.196–214.

Cabaravdic, A., 2007. Efficient Estimation of Forest Attributes with *k* NN, Albert-Ludwigs-Universität.

College of Natural Resources, University of California, Berkeley, http://nature.berkeley.edu/~bingxu/UU/spatial/Week15/HistogramAdjust.pdf, last accessed date: 10.04.2008. (URL3)

Congalton, R. G. and Green, K., 1999. Assessing the Accuracy of Remotely Sensing Data: Principles and Practices. CRC Press, Florida.

Department of Electrical and Computer Engineering, American University of Beirut, http://webfea-lb.fea.aub.edu.lb/dsaf/labs/projectv1.1.pdf, last accessed date: 12.01.2008. (URL8)

Dorren, L. K. A., Maier, B. and Seijmonsbergen, A. C., 2003. Improved Landsat-based forest mapping in steep mountainous terrain using object-based classification. Forest Ecology and Management, Vol. 183, pp. 31–46.

Franco-Lopez, H., Ek, A. R. and Bauer, M. E., 2001. Estimation and mapping of forest stand density, volume, and cover type using the k-nearest neighbors method. Remote Sensing of Environment, Vol. 77, pp. 251–274.

Gjertsen, A., 2007. Accuracy of forest mapping based on Landsat TM data and a kNN-based method. Remote Sensing of Environment, Vol. 110, pp. 420–430.

Gorokhovic, Y. and Voustianiouk, A., 2006. Accuracy assessment of the processed SRTM-based elevation data by CGIAR using field data from USA and Thailand and its relation to the terrain characteristics. Remote Sensing of Environment, Vol. 104, pp. 409–415.

Guttman, A., 1984. R-tree: A dynamic index structure for spatial searching. In Proceedings of the ACM International Conference on Management of Data, pp.47-58.

Haapanen, R., Ek, A. R., Bauer, M. E. and Finley, A. O., 2004. Delineation of forest/nonforest land use classes using nearest neighbor methods. Remote Sensing of Environment, Vol. 89, pp. 265–271.

Intelligent Sensor Systems, Wright State University, http://research.cs.tamu.edu/prism/lectures/iss/iss_l12.pdf, last accessed date: 11.01.2008. (URL6)

Katila, M. and Tomppo, E., 2001. Selecting estimation parameters for the Finnish multisource National Forest Inventory. Remote Sensing of Environment, Vol. 76, pp. 16–32.

Lorenzo, B., Gherardo, C., Piermaria, C. Marco, M., Fabio, M. and Carlo, R., 2002. Stand Basal Area Extension Over Forestland by Different Classification Algorithms Applied to Landsat ETM+ Imagery. ForestSAT Symposium, Heriot Watt University, Edinburgh, August 5th-9th, 2002.

Mäkelä, H. and Pekkarinen, A., 2001. Estimation of timber volume at the sample plot level by means of image segmentation and Landsat TM imagery. Applied Radiation and Isotopes, Vol. 77, pp. 66–75.

Mäkelä, H. and Pekkarinen, A., 2004. Estimation of forest stand volumes by Landsat TM imagery and stand-level field-inventory data. Forest Ecology and Management, Vol. 196, pp. 245– 255.

McRoberts, R. E. and Tomppo, E. O., 2007. Remote sensing support for national forest inventories. Remote Sensing of Environment, Vol. 110, pp. 412–419.

Nangendo, G., Skidmore, A. K. and van Oosten, H., 2007. Mapping East African tropical forests and woodlands - A comparison of classifiers. ISPRS Journal of Photogrammetry & Remote Sensing, Vol. 61, pp. 393–404.

National Aeronautics and Space Administration, ftp://e0srp01u.ecs.nasa.gov, last accessed date: 01.05.2008. (URL4)

Nelson, T., Wilson, H. G., Boots, B. and Wulder, M. A., 2005. Use of ordinal conversion for radiometric normalization and change detection. International Journal of Remote Sensing, Vol. 26, No. 3, pp. 535–541.

Ozdemir, I., 2004. Orman Envanterinde Uydu Verilerinden Yararlanma Olanakları. Süleyman Demirel Üniversitesi Orman Fakültesi Dergisi, Seri: A, Sayı: 1, pp. 84–96.

Pereira, C. R., 2006. Estimating and Mapping Forest Inventory Variables Using the K-NN Method: Mocuba District Case Study – Mozambique, Universita Tuscia, pp. 42.

Raudys, S. J. and Jain, A. K., 1991. Small Sample Size Effects in Statistical Pattern Recognition: Recommendations for Practitioners. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13. No. 3, pp. 252–264.

Sevik, O., 2006. Application of Sleuth Model in Antalya, METU, pp. 35.

Tampere International Center for Signal Processing, http://sigwww.cs.tut.fi/TICSP/Bio_Seminar/Files/19_09_01_Nicorici.pdf, last accessed date: 12.01.2008. (URL7)

The Code Project, http://www.codeproject.com/KB/cs/CSML.aspx, last accessed date: 10.03.2008. (URL2)

The National Map Seamless Server, U.S. Geological Survey, http://seamless.usgs.gov, last accessed date: 01.05.2008. (URL5)

Thessler, S., Sesnie, S., Bendaña, Z. S. R., Ruokolainen, K., Tomppo, E. and Finegan, B., 2008. Using k-nn and discriminant analyses to classify rain forest types in a Landsat TM image over northern Costa Rica. Remote Sensing of Environment, not yet published.

Tomppo, E. and Halme, M., 2004. Using coarse scale forest variables as ancillary information and weighting of variables in k-NN estimation: a genetic algorithm approach. Remote Sensing of Environment, Vol. 92, pp. 1–20.

Tsiriga, V. and Virvou, M., 2003. Dynamically initializing the student model in a Web-based language tutor. Third IEEE International Conference on Advanced Learning Technologies (ICALT'03), Vol. 1, pp. 138–143.

Wulder, M., M. Gillis, J. Luther, A. Dyk, 2001; A guide to the Estimation of Canada's Forest Inventory Attributes from Landsat TM Data (Natural Resources Canada, Canadian Forest Service, Pacific Forestry Center, Working paper, 84 p.)

Yang, C. Y. and Chou, J. J., 2005. A comparative evaluation approach for the classification of rotifers with modified non-parametric kNN. Image and Vision Computing, Vol. 23, pp. 427–439.