

Air Force Institute of Technology

**AFIT Scholar**

---

Theses and Dissertations

Student Graduate Works

---

8-12-2008

## Multi-Class Classification for Identifying JPEG Steganography Embedding Methods

Benjamin M. Rodriguez II

Follow this and additional works at: <https://scholar.afit.edu/etd>



Part of the [Information Security Commons](#)

---

### Recommended Citation

Rodriguez, Benjamin M. II, "Multi-Class Classification for Identifying JPEG Steganography Embedding Methods" (2008). *Theses and Dissertations*. 2641.  
<https://scholar.afit.edu/etd/2641>

This Dissertation is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact [AFIT.ENWL.Repository@us.af.mil](mailto:AFIT.ENWL.Repository@us.af.mil).



**MULTI-CLASS CLASSIFICATION FOR IDENTIFYING JPEG  
STEGANOGRAPHY EMBEDDING METHODS**

DISSERTATION

Benjamin M. Rodriguez II

AFIT/DEE/ENG/08-20

**DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY**

**AIR FORCE INSTITUTE OF TECHNOLOGY**

---

---

**Wright-Patterson Air Force Base, Ohio**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/DEE/ENG/08-20

**MULTI-CLASS CLASSIFICATION FOR IDENTIFYING JPEG  
STEGANOGRAPHY EMBEDDING METHODS**

DISSERTATION

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Benjamin M. Rodriguez II, BS, MS

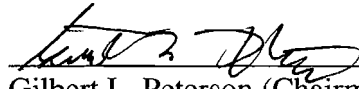
September 2008

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED


**MULTI-CLASS CLASSIFICATION FOR IDENTIFYING JPEG  
STEGANOGRAPHY EMBEDDING METHODS**

Benjamin M. Rodriguez II, BS, MS

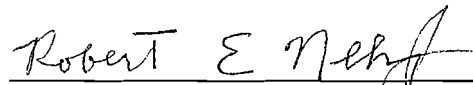
Approved:

  
\_\_\_\_\_  
Gilbert L. Peterson (Chairman)

8 AUG 08  
Date


  
\_\_\_\_\_  
Kenneth W. Bauer Jr. (Member)

8 AUG 08  
Date

  
\_\_\_\_\_  
Robert E. Neher Jr., Lt Col, USAF (Member)

8 AUG 08  
Date

Accepted:

  
\_\_\_\_\_  
M. U. Thomas  
Dean, Graduate School of  
Engineering and Management

12 Aug 08  
Date

## **Abstract**

Over 725 steganography tools are available over the Internet, each providing a method for covert transmission of secret messages. This research presents four steganalysis advancements that result in an algorithm that identifies the steganalysis tool used to embed a secret message in a JPEG image file. The algorithm includes feature generation, feature preprocessing, multi-class classification and classifier fusion. The first contribution is a new feature generation method which is based on the decomposition of discrete cosine transform (DCT) coefficients used in the JPEG image encoder. The generated features are better suited to identifying discrepancies in each area of the decomposed DCT coefficients. Second, the classification accuracy is further improved with the development of a feature ranking technique in the preprocessing stage for the kernel Fisher's discriminant (KFD) and support vector machines (SVM) classifiers in the kernel space during the training process. Third, for the KFD and SVM two-class classifiers a classification tree is designed from the kernel space to provide a multi-class classification solution for both methods. Fourth, by analyzing a set of classifiers, signature detectors, and multi-class classification methods a classifier fusion system is developed to increase the detection accuracy of identifying the embedding method used in generating the steganography images. Based on classifying stego images created from research and commercial JPEG steganography techniques, F5, JP Hide, JSteg, Model-based, Model-based Version 1.2, OutGuess, Steganos, StegHide and UTSA embedding methods, the performance of the system shows a statistically significant increase in classification accuracy of 5%. In addition, this system provides a solution for identifying steganographic fingerprints as well as the ability to include future multi-class classification tools.

## **Acknowledgments**

I would like to express my sincere gratitude to my advisor, Dr. Gilbert L. Peterson. I would also like to thank my committee members Dr. Kenneth W. Bauer Jr. and Lt Col Robert E. Neher Jr., for their input and advice. I would like to thank June Rodriguez, Mei-Ching Chen and Okan Caglayan for their countless hours in helping me edit my document. My AFIT friends, Maj Steve Oimoen, Lt Col Matt Sambora, Capt Nate Leap and Capt Mike Turnbaugh, for their help in all of our classes. Finally, special thanks goes to my wife and two daughters for their understanding and support during the dissertation process and my time at the Air Force Institute of Technology.

## Table of Contents

	Page
Abstract.....	iv
Acknowledgments.....	v
Table of Contents.....	vi
List of Figures.....	xi
List of Tables.....	xiv
I. Introduction.....	1
1.1 Background.....	4
1.1.1 Introduction to Steganography.....	4
1.1.2 Difference between Steganography and Cryptography.....	5
1.1.3 Differences between Steganography and Watermarking.....	6
1.1.4 Steganalysis.....	6
1.2 Problem Statement.....	8
1.2.1 Feature Generation.....	10
1.2.2 Feature Improvement.....	11
1.2.3 Classification.....	13
1.2.4 Classifier Fusion.....	14
1.3 Methodology.....	15
1.3.1 DCT Feature Generation.....	15
1.3.2 Feature Improvement.....	16
1.3.3 Classification.....	16
1.3.4 Classifier Fusion.....	17
1.3.5 Results.....	17



1.4 Summary.....	18
II. Literature Review .....	20
2.1 JPEG Image Representation Background .....	21
2.2 Feature Generation for JPEG Images .....	26
2.2.1 Wavelet Statistical Model .....	27
2.2.2 DCT Features .....	32
2.3 Feature Preprocessing.....	37
2.3.1 Data Preparation.....	37
2.3.2 Outlier Removal.....	39
2.4 Feature Extraction .....	41
2.4.1 Principal Component Analysis (PCA) .....	42
2.4.2 Kernel PCA .....	43
2.5 Feature Ranking/Selection.....	44
2.5.1 Bhattacharyya Distance .....	44
2.5.2 Fisher's Linear Discriminant Ratio (FDR/F-Score) .....	45
2.5.3 Signal-to-Noise Feature Selection .....	47
2.5.4 Kernel Fisher's Recursive Feature Elimination.....	48
2.5.5 Zero-Norm Feature Ranking.....	49
2.6 Classification .....	50
2.6.1 Expectation Maximization (EM) .....	51
2.6.1.1 Mixture Models.....	53
2.6.1.2 Bayes Classifier .....	55

2.6.2 $k$ -Nearest Neighbors.....	56
2.6.3 Kernel Fisher's Discriminant (KFD) .....	57
2.6.4 Parzen Window .....	60
2.6.5 Probabilistic Neural Networks (PNN) .....	62
2.6.6 Support Vector Machine (SVM).....	64
2.7 Multi-Class Classification .....	67
2.7.1 One-Against-One .....	68
2.7.2 One-Against-All.....	69
2.8 Classifier Fusion.....	69
2.8.1 Boosting .....	70
2.8.2 Bayes Network for Model Averaging.....	72
2.8.3 Probabilistic Neural Network (PNN) Fusion.....	73
2.9 Summary.....	73
III. Methodology .....	75
3.1 Feature Generation .....	77
3.1.1 DCT Representation.....	78
3.1.2 Arrangement of Decomposed DCT Coefficients.....	80
3.1.2.1 Frequency and Directional Coefficient Vectors .....	80
3.1.2.2 Block Shifted Coefficient Vectors .....	83
3.1.2.3 Neighboring Coefficient Matrices .....	89
3.1.3 Metrics Calculation.....	92

3.1.3.1 Mean Difference between DCT Coefficients and Neighboring Coefficients .....	92
3.1.3.2 Least Squares Linear Regression .....	93
3.1.4 Statistics Calculation.....	95
3.1.5 Features .....	96
3.2 Feature Ranking/Selection.....	97
3.2.1 SVM-Kernel Feature Ranking (KFR).....	97
3.2.2 Kernel Fisher's Discriminant Classifier Kernel Feature Ranking .....	105
3.3 Learning Decision Trees using Kernel mapping for creating Multi-class Classification from two-class KFD and SVM Classifiers .....	108
3.4 Fusion of Multi-Class Classification Systems.....	112
3.4.1 AdaBoost Boosting .....	113
3.4.2 Bayes Network for Model Averaging.....	114
3.4.3 Probabilistic Neural Network (PNN) Fusion.....	116
3.5 Summary.....	117
IV. Analysis and Results.....	118
4.1 Confirming and Validating the Analysis.....	119
4.2 Feature Generation Method Comparison .....	121
4.2.1 Wavelet Feature Generation (Lyu and Farid, 2004) .....	122
4.2.2 DCT Feature Generation (Pevny and Fridrich, 2006) .....	123
4.2.3 DCT Directional and Frequency Decomposition .....	125
4.2.4 Combined Features .....	126

4.2.5 Summary of Feature Generation Methods .....	128
4.3 Results for Individual Multi-class Detection Systems .....	130
4.3.1 Expectation Maximization .....	130
4.3.2 $k$ -Nearest Neighbors ( $k$ -NN) .....	132
4.3.3 Probabilistic Neural Networks (PNN) .....	134
4.3.4 Parzen window .....	135
4.3.5 Kernel Fisher's Discriminant (KFD) with Multi-class Tree .....	136
4.3.6 Support Vector Machines (SVM) with Multi-class Tree .....	138
4.3.7 StegoWatch .....	140
4.3.8 Summary of Steganalysis Multi-Class results .....	141
4.4 Fusion .....	143
4.4.1 AdaBoost .....	143
4.4.2 Bayes Fusion .....	144
4.4.3 PNN Fusion .....	145
4.4.4 Summary of Multi-class Steganalysis Fusion Techniques .....	146
4.5 Summary .....	148
V. Conclusion and Recommendations .....	149
5.1 Application of Results .....	149
5.2 Recommendations for Future Work .....	151
5.3 Conclusion .....	152
Appendix A .....	154
Bibliography .....	157
Vita .....	173

## List of Figures

	Page
Figure 1.1. Prisoner's Problem, Schematic of the Principles of Steganography. ....	2
Figure 1.2. Steganalysis Classification System in Training Stage.....	9
Figure 2.1. Basic Detection System. ....	20
Figure 2.2. Block Diagrams of Sequential JPEG Encoder. ....	22
Figure 2.3. Typical quantization matrix.....	24
Figure 2.4. DCT decomposition zig-zag structure for an 8×8 block, a) zig-zag pattern b) coefficient ordering sequence .....	25
Figure 2.5. Low-pass and high-pass quadrature mirror filter frequency. ....	28
Figure 2.6. Wavelet Structure a) Simple image with vertical, horizontal and diagonal lines b) 2 level wavelet decomposed c) 3 level wavelet decomposition.....	29
Figure 2.7. Feature generating structure. ....	33
Figure 2.8. Expectation Maximization using mixture models with Decision Boundary..	54
Figure 2.9. <i>k</i> -NN Decision Boundary. ....	56
Figure 2.10. KFD Decision Boundary using RBF Kernel.....	60
Figure 2.11. Parzen Density Estimator with RBF window with Decision Boundary.....	62
Figure 2.12. Probabilistic Neural Network Classification Structure. ....	63
Figure 2.13. PNN with Decision Boundary.....	64
Figure 2.14. SVM with Optimal Hyperplane.....	66
Figure 3.1. Detection System.....	76
Figure 3.2. General Feature Generation System.....	77

Figure 3.3. DCT decomposition a) zig-zag scan pattern b) low, medium and high frequency distributions c) vertical, diagonal and horizontal directions d) $8 \times 8$ image with a horizontal edge between pixels e) $8 \times 8$ image with a diagonal edge between pixels f) $8 \times 8$ image with a vertical edge between pixels g) 2-D DCT representation of horizontal image h) 2-D DCT representation of diagonal image i) 2-D DCT representation of vertical image. ....	79
Figure 3.4. DCT Coefficient Locations and Separations a) DCT Coefficient Location after Zig-Zag Scan b) Coefficient Locations of Vertical, Diagonal and Horizontal directions c) Coefficient Locations of Low, Mid and High Frequencies d) $8 \times 8$ block Coefficient Separation of both frequencies and directions. ....	81
Figure 3.5. Original block and right shifted pixel locations .....	84
Figure 3.6. Original block and down shifted pixel locations.....	86
Figure 3.7. Original block and diagonal shifted pixel locations. ....	88
Figure 3.8. One dimensional mapping of Equation (3.19) when the strong ranked feature is removed. ....	99
Figure 3.9. One dimensional mapping of Equation (3.19) when the weak ranked feature is removed.....	100
Figure 3.10. One dimensional mapping of Equation (3.20) when the highest ranked feature is removed. ....	103
Figure 3.11. One dimensional mapping of Equation (3.20) when the lowest ranked feature is removed. ....	104

Figure 3.12. One dimensional mapping of Equation (3.19) when the top 25% of the ranked features are kept. ....	105
Figure 3.13. Decision tree for a 10-class classification problem with 10 leaf nodes, 9 parent nodes and a maximum depth of 6. ....	108
Figure 3.14. Distance values $\hat{D}_{C_k}$ for a 10 class problem.....	111
Figure 3.15. Detection structure for 8 classification models. ....	114
Figure 3.16. Probabilistic Neural Network Classification Structure. ....	116
Figure 4.1. 5-fold cross-validation with 5 runs consisting of 80% of the data for training the classification model and 20% for testing the training model. ....	120
Figure 4.2. Decision tree for a 10-class classification problem with 10 leaf nodes, 9 parent nodes and a maximum depth of 6. ....	137
Figure 4.3. Decision tree for a 10-class classification problem with 10 leaf nodes, 9 parent nodes and a maximum depth of 6. ....	139
Figure 5.17. Detection system. ....	150

## List of Tables

	Page
Table 2.1. All 23 distinguishing functionals.....	36
Table 2.2. Parameterization for mixture models.....	54
Table 3.1. Test statistics for generating features.....	96
Table 3.2. Distribution of the image types.....	114
Table 4.1. Classification accuracy for wavelet feature generation. ....	122
Table 4.2. Classification accuracy for wavelet feature generation using feature ranking. .....	122
Table 4.3. <i>t</i> -test; paired two samples for means between Tables 4.1 and 4.2.....	123
Table 4.4. Classification accuracy for DCT feature generation.....	123
Table 4.5. Classification accuracy for DCT feature generation using feature ranking...	124
Table 4.6. <i>t</i> -test; paired two samples for means between Tables 4.4 and 4.5.....	124
Table 4.7. Classification accuracy for DCT directional and frequency feature generation. .....	125
Table 4.8. Classification accuracy for DCT directional and frequency feature generation using feature ranking.....	125
Table 4.9. <i>t</i> -test; paired two samples for means between Tables 4.7 and 4.8.....	126
Table 4.10. Classification accuracy for combined feature generation using feature ranking.....	126
Table 4.11. <i>t</i> -test: paired two samples for means of wavelet features with feature ranking vs. combined features with feature ranking. ....	127



Table 4.12. <i>t</i> -test: paired two samples for means of DCT features with feature ranking vs. combined features with feature ranking. ....	127
Table 4.13. <i>t</i> -test: paired two samples for means of DCT directional and frequency features with feature ranking vs. combined features with feature ranking. ....	128
Table 4.14. Classification accuracy summary for the individual feature generation and combined features when feature ranking is used. ....	129
Table 4.15. Number of features used from each of the feature generation method in feature combination. ....	129
Table 4.16. Classification accuracy for 10-class expectation maximization classifier...	131
Table 4.17. Classification accuracy for 10-class <i>k</i> -NN classifier. ....	133
Table 4.18. Classification accuracy for 10-class PNN classifier. ....	134
Table 4.19. Classification accuracy for 10-class Parzen window classifier. ....	135
Table 4.20. Classification accuracy for 10-class KFD classifier. ....	138
Table 4.21. Classification accuracy for 10-class SVM classifier. ....	140
Table 4.22. Classification accuracy for StegoWatch detection system. ....	141
Table 4.23. Classification accuracy for multi-class detection system. ....	142
Table 4.24. <i>t</i> -test: paired two samples for means. ....	142
Table 4.25. Classification accuracy for AdaBoost fusion. ....	144
Table 4.26. Classification accuracy for Bayes fusion. ....	145
Table 4.27. Classification accuracy for PNN fusion. ....	146
Table 4.28. Classification accuracy comparisons between the best individual results and the three fusion methods. ....	147

Table 4.29. t-test: paired two samples for means of classification accuracy between PNN fusion and SVM. ....	147
---	-----

# MULTI-CLASS CLASSIFICATION FOR IDENTIFYING JPEG STEGANOGRAPHY EMBEDDING METHOD

## I. Introduction

*Steganography* plays an important role in information security, i.e., any form of covert communication. Literally, the meaning of steganography originated from the ancient Greek words is “covered writing” (The Oxford English dictionary, 1933). It puts emphasis on perceptual unobservable/undetectable data hiding, i.e., the inability to prove that a cover file contains hidden data. In order to hide secret information, three components in steganography are the stego message, cover file and embedding method. *Stego message* is the covert message that a sender wishes to remain confidential, such as text, picture, audio, etc. A *clean file* is a file that has not been modified from its original characteristics while a *cover file/carrier* is a file in which a message will be hidden within. After using an embedding method, the stego system results in *stego/dirty files* that are digital files containing the hidden information with the cover file and the stego message as input, i.e., files have been manipulated by an embedding method by hiding information. In the embedding and decoding procedures, a parameter, *stego key*, shared by the sender and the receiver is used to limit the authority of extracting the stego message from the stego file.

The classic model for steganography proposed by Simmons (1984) is the prisoners’ problem. Figure 1.1 illustrates a scenario of the problem that Alice and Bob are arrested for a crime and thrown in two different cells. They want to develop an escape plan, but the warden Wendy monitors all communications between the two prisoners. She will not let them communicate through encryption and if she notices any suspicious communication, she will place them in solitary confinement and thus suppress the exchange of all messages. Hence, both parties must communicate invisibly in order to avoid arousing Wendy’s suspicion; they have to set up a subliminal channel. A practical

way to do so is to hide meaningful information in some harmless message: Alice could, for instance, use a digital photo of an aircraft and send this image to Bob. Wendy has no idea that the binary value representation of the image transmits a secret escape plan (stego message). After receiving the stego file, Bob reconstructs the message with a key he shares with Alice.

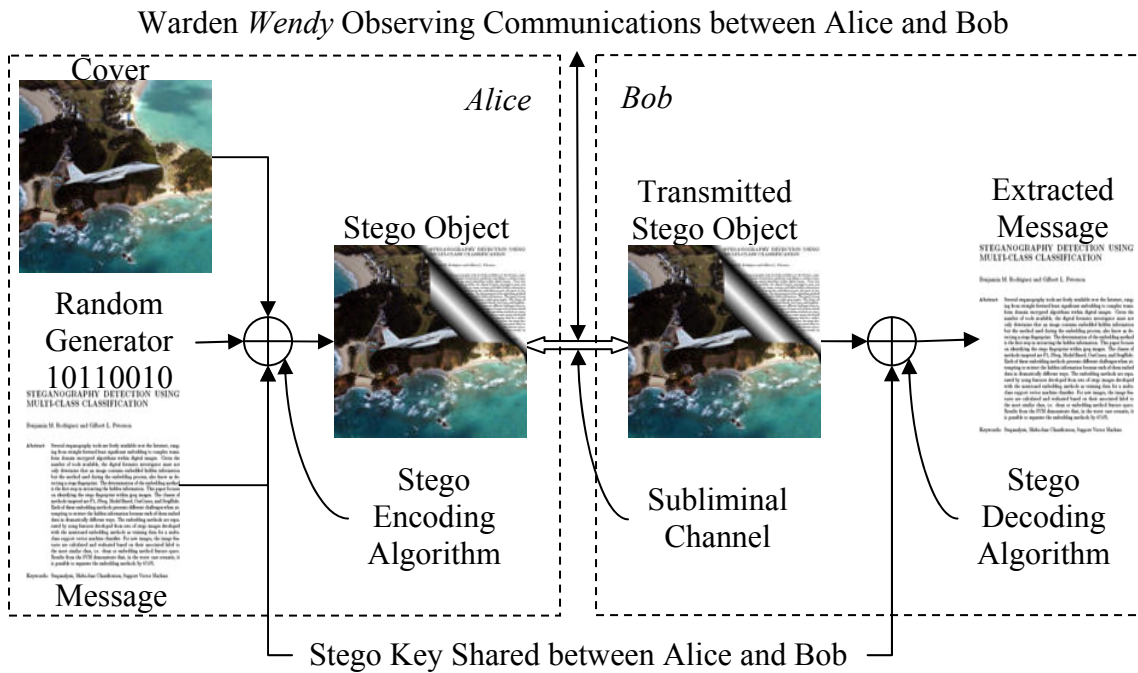


Figure 1.1. Prisoner's Problem, Schematic of the Principles of Steganography.

Contrary to steganography, *steganalysis*, the main research in this dissertation, is the process for identifying a file containing steganography and/or extract the stego message. Steganalysis has progressed from the simple case of determining whether an image contains hidden information to the more complex problem of extracting the hidden information. With over 725 steganography tools available over the Internet (Backbone Security, 2008) this is an escalating problem. From a digital forensics standpoint, it is important to extract the hidden data. A step in the process for doing this is identifying the embedding algorithm used to create the stego file. Stego method identification however,

is not trivial with so many tools available. A step towards algorithm identification requires determining the class of steganography algorithm used during embedding. This identification requires developing a steganalysis system.

This research focuses on building up a multi-class steganalysis system for detecting the secret in compressed images. The system includes generating the features from inputs, which are the characteristics of the JPEG images. The generated raw features are sent through a set of preprocessing steps; feature ranking, feature selection and feature extraction, which are used to eliminate redundancies within features. The preprocessed features are input to SVM or KFD classifiers using the presented multi-class tree with the selected and the fusion of classifiers. The performance of the system is based on the classification accuracy on input images, determining clean or stego images of which in the following JPEG embedding methods are used: F5, JP Hide, JSteg, Model-based, Model-based Version 1.2, OutGuess, Steganos, StegHide and UTSA.

In the following section, a background of steganalysis is given which includes the definition of steganography, a brief history, comparisons with cryptography and watermarking along with a definition of steganalysis. Following this, a section devoted to the problem statement for a multi-class classification system is outlined. The problems which are encountered in the development of multi-class systems such as the generation of features, selection of the best set of features, classification with classifier selection and the fusion of multi-class classification methods are also discussed. The methodology section gives an overview of the multi-class steganalysis system including the generation of features for JPEG images, the multi-class tree structure for classification, selection of the most relevant features and the multi-class classification fusion system. The last section concludes with the summary of the topics discussed within this chapter.

## **1.1 Background**

In this section steganography, one of the information hiding techniques, is defined with respect to current multimedia formats. A brief history of steganography (Kahn, 1996) is given beginning with ancient Greece (Littlebury, 1737a; 1737b; Rawlinson, 1862; 1875; 1880; 1889) to the current classical model. In addition, a comparison is made between steganography, cryptography and watermarking, which are the other data hiding techniques, followed by a definition of steganalysis.

### **1.1.1 Introduction to Steganography**

Communication systems have long been used to send and receive secret messages. In many of these systems the messages may be transmitted through public communication channels either open to be viewed or concealed from an outside observer. Stego messages are the ones that have been hidden within innocent looking cover files creating a stego file. Even though data hiding terminology is fairly modern due to the popularity of multimedia, the roots of steganography can be traced back to ancient Greece (Littlebury, 1737a; 1737b; Rawlinson, 1862; 1875; 1880; 1889). A history of steganography was written by Kahn (1996) providing specific steganography events. Herodotus, the father of history, gives several cases (Littlebury, 1737a; 1737b; Rawlinson, 1862; 1875; 1880; 1889). A man named Harpagus wanted to send a secret message so he killed a hare and hid a message inside its body. He sent it with a messenger who pretended to be a hunter (Littlebury, 1737a, pp. 80-81; Rawlinson, 1889, pp. 201). In another instance (Littlebury, 1737a, pp. 19; Rawlinson, 1862, pp. 197), Histaieus wished to inform his friends that it was time to begin a revolt against the Medes and the Persians. He shaved the head of one of his trusted slaves, tattooed the message on the head, waited till his hair grew back, and sent him along. It worked; the message successfully reached his intended recipients in Persia and the revolt succeeded. Things worked more slowly in the days before faxes, e-mail and the Internet. Herodotus also tells of a man named Demeratus who wanted to

report from the Persian court back to his friends in Greece that Xerxes the Great was about to invade Greece (Littlebury, 1737b, pp. 278-279; Rawlinson, 1880, pp. 187). Messages in those days were sent via writing tablets made of two pieces of wood, hinged as a book, with each face covered with wax. One wrote on the wax; the recipient melted the wax and reused the tablet. Demeratus removed the wax of the tablet, concealed a message on the wood itself and recovered the tablet with wax. He then sent the apparently blank tablets to Greece. At first nobody could figure out what they meant. Then a woman named Gorgo guessed that maybe the wax was concealing something. She removed it and became the first woman cryptanalyst (Kahn, 1996). Unfortunately, her ingenuity had fatal consequences for her husband Leonidas, the king of Sparta; he died with band of Greeks holding off the Persians at Thermopylae (Littlebury, 1737, pp. 270-271; Rawlinson, 1880, pp. 178).

### **1.1.2 Difference between Steganography and Cryptography**

An alternative method to steganography in secure communication is cryptography. An important point to note is that both steganography and cryptography provide secure communications and may be used concurrently. Steganography and cryptography differ in execution. In cryptography, the secret message which is the transmitted file itself cannot be recovered without the secret key; however, the encrypted file is identified as being sent. It helps to protect confidentiality but protection vanishes after decryption. In steganography the existence of the stego message is concealed in a cover file in a way that does not allow an enemy to observe that there is a message present (Petitcolas et al., 1999). The stego message can be extracted with stego key as long as the stego file is identified by which embedding method is used.

### **1.1.3 Differences between Steganography and Watermarking**

Except for steganography, watermarking has been the other data hiding technique broadly used for authentication. Both of them share many common rules but the objectives for these techniques are different. In watermarking, the important information is the cover media. The embedded data is inserted solely for the protection of the cover media. In steganography, the cover media is not important. It typically serves as a diversion from the embedded data. Steganographic communications are usually between a sender and single receiver while watermarking techniques are usually between a sender and many receivers (Katzenbeisser and Petitcolas, 2000). Digital watermarking may be thought of as a commercial application of steganography, being used to trace, identify and locate digital media across networks (Johnson and Jajodia, 1998A; 1998B). The encoding/decoding part of steganographic systems is similar to watermarking. However, steganography has reduced robustness requirements allowing a higher embedding rate.

### **1.1.4 Steganalysis**

Steganalysis is the science of detecting hidden information within a cover file, i.e., to identify a file as containing stego and/or extract the stego message. An investigator using steganalysis techniques is known as a steganalyst, such as Wendy in the prisoner's problem. Steganalysis is a relatively young research discipline with few articles appearing before the late-1990s (Kessler, 2004). The science of steganalysis was initially intended to detect or estimate the existence of stego information based on observing some data transfer, while having no assumptions of the steganography algorithm applied (Chandramouli, 2002). In digital image steganalysis an analyst has three goals, first determine if an embedded message exists, next determine the embedding method used to create the stego image, and finally extract the hidden message. This research focuses on the second goal, that is, to identify the embedding technique used to create the steganography image. Several detection systems currently exist, so the identification



problem becomes one of determining which detection system has correctly identified the embedding method. Most steganalysis today is signature-based, similar to anti-virus and intrusion detection systems. In this type of application, the known embedding algorithms provide fingerprints that are added to a steganographic fingerprint database in which the analyst creates a message and uses a known stego tool to create a stego file. This known stego file is then analyzed to determine patterns for later use against other stego files (Silman, 2001). Steganography detection and extraction is generally sufficient if the purpose is evidently gathering related to a past crime. Although, disable the hidden message so that the recipient cannot extract it and/or alter the hidden message to send misinformation to the recipient might also be legitimate law enforcement goals during an on-going investigation of criminal or terrorist groups (Jackson, 2003). The law enforcement community does not always have the luxury of knowing when and where steganography has been used or the algorithm that has been employed. Generic detection tools generated from emerging research capable of detecting and classifying steganography are becoming available, including research prototypes (Fridrich, 2004; Lyu and Farid, 2004; Shi et al., 2005; Pevny and Fridrich, 2007; Rodriguez and Peterson, 2007; Wang and Moulin, 2007) and commercially-available tools (e.g., ILook Investigator, Inforenz Forager, StegalyzerSS, SecureStego, StegDetect (Provos, 2004) and WetStone's Stego Suite).

The following definitions were introduced by Johnson and Jajodia (1998B) and are frequently used by the steganalysis community:

- Stego-only attack: The stego file is the only item available for analysis.
- Known cover attack: The cover and stego file are both available for analysis.
- Known message attack: The hidden message is known.
- Chosen stego attack: The stego file and tool are both known.
- Chosen stego message attack: The steganalyst generates stego files from a known steganography tool using a chosen stego message.
- Known stego attack: The cover file, stego file and stego tool are known.

## **1.2 Problem Statement**

There is an estimated 725 steganography methods available on the Internet with the majority being used for hiding messages in digital images (Backbone Security, 2008). Several are downloadable for free and have user friendly graphical user interfaces (GUIs) (Higgins, 2007). While these tools have been used to hide various forms of information for privacy, these tools have also been used for criminal activity and malicious intent. Documented examples of this have occurred, including an incident involving an engineer sending an email with two attached images that turned out to be a set of stego files containing intellectual property (Radcliff, 2002). Other crimes involving the use of steganography include child pornography where the stego files are used to hide a predator's location when posting digital pictures on Web sites or sending them through email (Astrowsky, 2000). Steganography may also be used to allow communication between affiliates of an underground community, such as terrorist organizations (Kelley, 2001). To combat these image stego tools, an initial step requires determining if an observed image contains a stego message. If an image is identified as being a stego file, the second step is determining the embedding method. This step of identifying the steganography method enables the steganalyst to then target the steganography method and extract the hidden information in a final step.

Identifying the tool used to create the stego image will help in the extraction process of removing the hidden message. Therefore, a system must be designed to identify which stego tool is used. Several factors must be addressed in the steganalysis multi-class classification system including feature generation, feature improvement, classifier selection and fusion as shown in Figure 1.2.

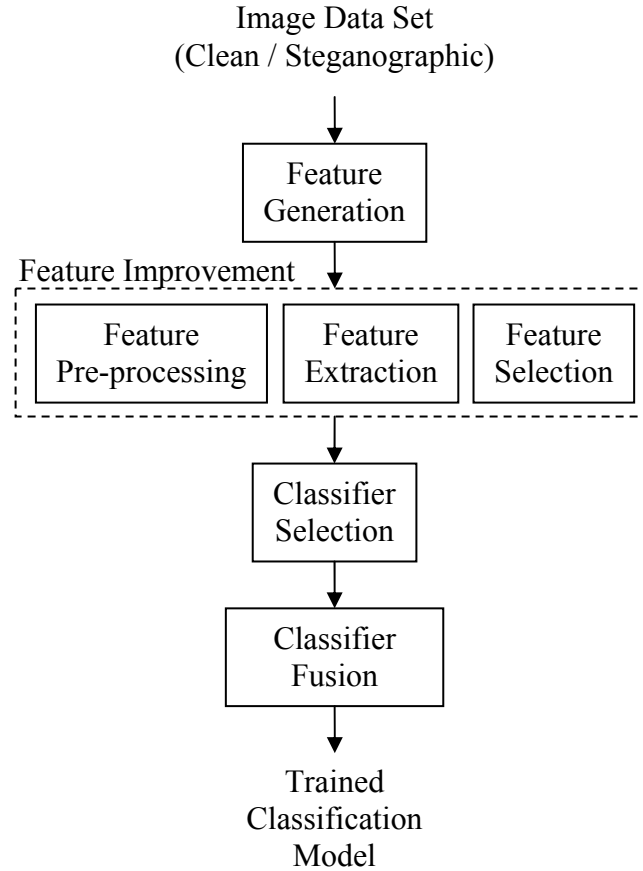


Figure 1.2. Steganalysis Classification System in Training Stage.

As in the training stage, given clean and steganographic image datasets, the system with all these procedures is trained to find out the suitable parameters used for classification. The trained classification model as the output in this stage contains parameters for feature improvement, the classifier parameters, and parameters for classifier fusion. Once the model is set, the testing stage in Figure 1.3 indicates the output of the model is which stego method is used, either none, F5, JP Hide, JSteg, Model-based, Model-based Version 1.2, OutGuess, Steganos, StegHide and UTSA.

Several detection systems are available from research tools (Lyu and Farid 2002; 2004; Fridrich, 2004; Lie and Lin, 2005; Shi et al., 2005; Xuan et al., 2005; Fu et al., 2006; Pevny and Fridrich, 2006; 2007; Rodriguez and Peterson, 2007; Wang and Moulin, 2007)

to commercially available systems (ILook Investigator © toolsets, Inforenz Forager®, SecureStego (Air Force Research Laboratory, Rome, NY), StegDetect (Provos, 2004), WetStone Stego Suite™). Each of the available systems has certain advantages over each other. A steganalyst should use as many of these tools as possible when analyzing a set of images. A problem arises when each detection system used potentially returns different class labels representing different embedding techniques. In the event each of the detection systems identifies a different stego tool, the analyst must then properly determine the correct method from the different set of identified stego labels. The solution described in this research fuses the results of each detection systems to get better detection accuracy and alleviate the steganalyst from having to make this assessment. The remainder of this section introduces the basic concept of feature generation, feature improvement, classifier selection, multi-class classification and the fusion of classifier systems.

### **1.2.1 Feature Generation**

The basic concept of generating features is to transform a given image, which contains an extensive number of data values in a two dimensional matrix, into a new set of features. If the transform is suitably chosen the transform domain features can exhibit high information properties about the original input image in a compact vector form. This means that most of the classification related information is compressed in a relatively small number of values leading to a reduced feature space (Theodoridis and Koutroumbas, 2006). For example, consider a grayscale image that is of  $512 \times 512$  pixels. This image would contain 262,144 pixel values, mapping the image into a new domain with the use of a transfer function can potentially represent the image with a significantly smaller number of values. The basic reasoning behind transform-based features is that an appropriate chosen transform can exploit and remove redundancies that usually exist in digital images (Theodoridis and Koutroumbas, 2006). Consider the problem of steganalysis, an input image that has been manipulated by an embedding method will

contain changes that are not visible to the human eye. In the case of JPEG images a compression technique is used which is based on the discrete cosine transform (DCT). Generating features for discriminating between a clean image (an original cover file) and a stego image (stego file) using the DCT will eliminate redundant pixel information. When generating features derived from calculating the DCT, most of the energy lies in the frequency bands of the coefficients providing important information for class discrimination. This however leads to a large number of features, which for classification accuracy must be reduced.

### **1.2.2 Feature Improvement**

With the raw features, feature improving before classification is vital. The goal for improving the input features is to select a subset of feature and/or extract the most feasible features able to categorize the inputs.

**Feature Ranking and Selection** - The major task in feature selection, given a large number of features, is to select the most important features and reduce the dimensionality while retaining class discriminatory information. This procedure is important when determining which features are to be used to train the classification model. If features with little discrimination power are selected the subsequent classification model will lead to poor classification performance. On the other hand, if information rich features are selected the design of the classifier can be greatly simplified. In a more quantitative description, feature selection leads to large between-class distances and small within-class distances in the feature space. That is, features should separate different classes by a large distance, and should have small distance values between objects in the same class. Several methods are available to identify individual features with linear separation, a few ranking and selection methods include; divergence measure (Fukunaga, 1990; Theodoridis and Koutroumbas, 2006), Bhattacharyya distance (Bhattacharyya, 1943; Fukunaga, 1990) and Fisher's linear discriminant ratio (Fisher, 1936; 1943; Dillon and

Goldstein, 1984; Fukunaga, 1990; van der Heijden et al., 2004; Bishop, 1995, 2006; Theodoridis and Koutroumbas, 2006).

When measuring nonlinear class separability, care must be taken when using feature ranking methods. Ranking methods developed for specific classifiers are often best suited for determining the best set of ranked features. For neural network classifiers features are ranking and selected based on a saliency metric (Ruck et al., 1990; Belue and Bauer, 1995) and signal-to-noise ratio (Bauer et al., 2000). For kernel based classifiers, such as kernel Fisher's discriminant and support vector machines, method-specific techniques are best suited for ranking. These techniques include recursive feature elimination (Guyon et al., 2002; Guyon, 2007), zero-norm feature ranking (Weston et al., 2003), gradient calculations using recursive feature elimination (Rakotomamonjy, 2003), and kernel Fisher's discriminant using recursive feature elimination (Louw and Steel, 2006).

**Feature Extraction** - Another approach to reducing the dimension of the input features is to use a transformed space instead of the original feature space. For example using a transformation  $\phi(\cdot)$  that maps the data points  $\mathbf{x}$  of the input space,  $\mathbb{R}^n$ , into a reduced dimensional space  $\mathbb{R}^p$ , where  $n > p$ , creates features in a new space that may have better discriminatory properties. Classification is based on the new feature space rather than the input feature space. The advantage of feature extraction over feature selection is that no information from any of the elements of the measurement vector is removed. In some situations feature extraction is easier than feature selection. A disadvantage of feature extraction is that it requires the determination of a suitable transformation  $\phi(\cdot)$ . Some methods include principal component analysis (Hotelling, 1933; Dillon and Goldstein, 1984) and kernel principal component analysis (Scholkopf et al., 1998; Bishop, 2006). If the transformation chosen is too complex, the ability to generalize from a small data set will be poor. On the other hand, if the transformation chosen is too simple, it may constrain the decision boundaries to a form that is inappropriate to discriminate between

classes. Another disadvantage is that all features are used, even if some of them have noise like characteristics. This might be unnecessarily expensive in term of computation (van der Heijden et al., 2004). It should be noted that the transformation used for the input features in the training of the classification model should also be used for the testing features.

### **1.2.3 Classification**

Given an input sample the training of a classification model may consist of supervised or unsupervised learning. In supervised learning the input sample includes an identification of its class membership. In unsupervised learning the class of the input sample is not known (Jain et al., 2000). This research concentrates on supervised learning. Supervised learning can be further broken down into subcategories of classification models. These models include but not limited to the following classifier types (Duda and Hart, 1973; Fukunaga, 1990; Theodoridis and Koutroumbas, 2006);

- Classifiers based on Bayes decision theory include; Bayesian networks, discriminant functions, and mixture models, specifically, expectation maximization. Linear classifiers include; Bayes linear classifier, Fisher's linear discriminant, and the perceptron algorithm.
- Nonlinear classifiers include; decision trees, kernel Fisher's discriminant, multi-layer perceptron, radial basis neural networks, and nonlinear support vector machines.
- Nonparametric classifiers include; locally weighted regression, and Parzen window.

These classifiers are predominantly two-class classifiers while some can be either two-class or multi-class classifiers. In this research the concentration is on multi-class classification. The specific problem addressed is how to design discriminant functions which are able to separate more than two classes (Duda and Hart, 1973; Platt et al., 2000; Schwenker, 2000; Tax and Duin, 2002; Rifkin and Klautau, 2004; Eibl and Pfeiffer,

2005; Wang and Casasent, 2005; Liu and Zheng, 2005; Bishop, 2006; Middelmann et al., 2006; Theodoridis and Koutroumbas, 2006; Yang et al., 2006).

#### **1.2.4 Classifier Fusion**

As noted, there is a large pool of different classifiers. In the literature, classifier fusion has been proposed for improving classification performance by exploiting the individual advantages of each of the classifiers (Woods et al., 1997; Duin and Tax, 2000; Ruta and Gabrys, 2001; Shipp and Kuncheva, 2002; Jaeger, 2004; Kuncheva, 2004; Leap et al., 2004; Theodoridis and Koutroumbas, 2006). The success of classifier fusion depends on two factors defined by Goebel and Yan (2004); first, the proper selection of a pool of diverse individual classifiers to be fused, and second, the proper method of fusing individual classifiers. A third factor should also be considered, that is the subspace of the classifiers being fused. Identifying the appropriate classifier for a particular problem is not trivial. Selecting the single best performing classifier on the training data and applying it to the testing data is the easiest method. While this approach is the simplest the most advantageous performance may not be guaranteed. An increase of performance can possibly be obtained by increasing the available dataset. When this is not an option, the most reliable strategy is to evaluate as many different classifier designs as possible and subsequently select the best performing model. The difficulty is that such a wide evaluation is computationally complex. In relation to classifier fusion, selection identifies the answers to which classifier and how many classifiers to select in order to obtain an increased performance. In certain situations, a problem arises when the outputs of the individual classifiers are of different types, either discrete values or posterior probabilities. Hence, the proper classifier fusion technique has to be used for a specific problem.



### **1.3 Methodology**

The following sections describe the multi-class JPEG image steganalysis system. Each subsection introduces the main advancements this research provides in the area of steganalysis, specifically, feature generation, feature selection, classifier selection, and multi-class classifier fusion.

#### **1.3.1 DCT Feature Generation**

This section describes the novel JPEG steganalysis feature generation method used in the classification system. In this method the DCT coefficients are separated into vertical, diagonal and horizontal orientation as well as low, medium and high frequencies. This is known as DCT decomposition (Rao and Yip, 1990). Each of the  $8 \times 8$  blocks is divided into nine DCT decompositions represented by both the frequency distributions and directions. The coefficients of interest are within the vertical, diagonal and horizontal orientation of the low and medium frequency bands. The predictors are used to estimate modifications made to an image by an embedding method. In this research four different predictor methods are used. The first is a distance measure in which the distance between neighboring coefficients is calculated and averaged. The second method used to calculate the predictors is a least squares linear regression technique on the DCT neighboring coefficients for JPEG images originally proposed by Farid (2002). In the final method the predictors are calculated by shifting the  $8 \times 8$  blocks by one pixel in the spatial domain followed by recompressing the pixels using the JPEG properties. To measure the coefficients, neighboring coefficients and shifted coefficients, 180 features are generated from higher-order statistics that aid in the assessment of changes made to the image by an embedding method. As more features are created, the problem becomes one of relevancy to the actual classification problem.

### **1.3.2 Feature Improvement**

The feature ranking/selection method used for improving identification accuracy is designed for two kernel based classifiers, the kernel Fisher's discriminant (KFD) and the support vector machines (SVM). The benefit of this feature selection method is that the classification algorithms being used assists in discriminating between important features and noise features by ranking features in the kernel space. The ranking method consists of; first, removing one feature at a time from the input space and transforming the remaining features into the kernel space, second, identifying the alpha vectors and support vectors, and third, assigning a ranking value to the removed feature using the alpha vectors and support vectors with a new derived ranking measure. The selection is based on the percentage of features necessary to increase classification performance, and is termed SVM-Kernel Feature Ranking (SVM-KFR). This however, does not resolve the need to discriminate between several classes.

### **1.3.3 Classification**

For detecting stego messages in various embedding methods, a fusion of classifiers is used to increase classification accuracy. Prior to the fusion process, the selection of classifiers is vital. One approach is to first heuristically pick a number and types of classifiers while ensuring a diverse output. Another approach is choosing classifiers from a large pool to achieve classification performance as close to an error rate of zero as possible. This should be accomplished while avoiding the exhaustive evaluation of all possible classifier combinations. The classifiers are multi-class classifiers, including Bayes decision theory method and expectation maximization (EM); the nonlinear classifier probabilistic neural network (PNN); and nonparametric classifiers,  $k$ -nearest neighbors and Parzen windows. Two nonlinear kernel based methods are also used, the support vector machine (SVM) and kernel Fisher's discriminant (KFD). These two methods however are two-class classifiers. In this methodology, the focus is to solve

multi-class classification for identifying various stego embedding methods. In order to solve the KFD and SVM two-class problem, a new multi-class classification tree is designed specifically for the KFD and SVM where two-class classifiers reside at each node of the tree. This tree is designed by separating classes into two groups at each node. The classes are grouped according to the smallest distances between classes. This tree is gradually expanded by adding a node each time a set of two or more classes is identified. The smallest distance between a set of classes represents a low value in classification accuracy. The distance measure is based on the kernel transform.

#### **1.3.4 Classifier Fusion**

The output labels of the multi-class classifiers expectation maximization (EM),  $k$ -nearest neighbors ( $k$ -NN), Parzen window and probabilistic neural networks along with the output labels of the new KFD and SVM multi-class classifiers are fused to increase classification accuracy. Along with the six multi-class detection systems two commercial tools, StegAlyzerSS and StegoSuite, are also fused. In this work, the individual detection systems are fused using three fusion methods; the first method used for fusion is boosting, specifically AdaBoost (Freund and Schapire, 1995); the second method is Bayesian networks for model averaging (Murphy, 2001); the final method is probabilistic neural networks.

#### **1.3.5 Results**

The simulation of the methodology is done by 5-fold cross validation having both training and testing. With feature preprocessing, an average increase in classification accuracy is achieved for the individual multi-class classifiers, EM,  $k$ -nearest neighbors, Parzen window, PNN by as much as 22% in comparison to no features preprocessing. A multi-class classification system for KFD and SVM is created by using a multi-class tree. With the use of the tree structure the classification accuracy of this new system by

applying the feature preprocessing in the individual nodes, an increase in classification accuracy is achieved by 10% than without feature preprocessing. With the use of the classifier fusion, the overall accuracy by 5% over the best individual best classifier is attained. Furthermore, the performance of the methodology shows statistical difference between the newly fused system in comparison to the individual detection systems.

## **1.4 Summary**

This chapter defined steganography, provided its brief history and how steganography is used with current multimedia formats was given. A definition of steganalysis was also given followed by a section devoted to the problem statement for a multi-class classification system. The specific problems encountered in the development of multi-class systems in this chapter are generation of features, selection of the best set of features, classification selection and the fusion of multi-class classification methods. The methodology for this research was introduced in Section 1.3 which included the generation of features for identifying JPEG stego and clean images, selection of the most relevant features, the design of a multi-class classification system for both KFD and SVM and the fusion of multi-class classifiers.

Chapter 2 provides the necessary background and literature review in solving the complex problem of identifying the embedding method used. In Chapter 3, the methodology is described in detail in which the full detection system is developed. This involves the generation of features, the ranking and selection of features, the design of the classification tree and the fusion of classifiers to solve the multi-class problem. In Chapter 4, the results are based on a twelve class dataset which contains a set of clean images (one class) and steganography images (seven classes). The embedding methods targeted in this paper are F5 (Westfeld, 2001; 2003), JP Hide (Latham, 1999), JSteg (Upham, 1993), Model-base (Sallee, 2003; 2006), Model-based Version 1.2 (Sallee, 2008a), OutGuess (Provos, 2004), Steganos (2008), StegHide (Hetzl, 2003) and UTSA

(Agaian et al., 2006). The classification results are provided from EM,  $k$ -nearest neighbors, Parzen Window and probabilistic neural networks multi-class classifiers, new multi-class tree with KFD and SVM, commercial tool and fusion of all the multi-class systems. The results also show the classification of the embedding methods with the new feature generation methods compared with the wavelet based features (Lyu and Farid 2002) and the DCT based features (Pevny and Fridrich, 2007). These results show four techniques that improve classification accuracy; first, the new feature generation method, second, the multi-class tree allows the KFD and SVM to be used as multi-class classifier, third, the selection of features at each node for the KFD and SVM classifiers, and the final technique is the fusion of the various classifiers. Finally, Chapter 5 provides a conclusion, contribution to DoD and future directions that may be considered in expanding the steganalysis multi-class system.

## II. Literature Review

This chapter presents related work relevant to the development of a steganalysis system. There are several sub-components to this research, including JPEG image representation, feature generation, feature preprocessing, feature extraction, feature selection, classification, multi-class classification and classifier fusion. Figure 2.1 shows the basic structure of the detection system and its primary components discussed in this chapter.

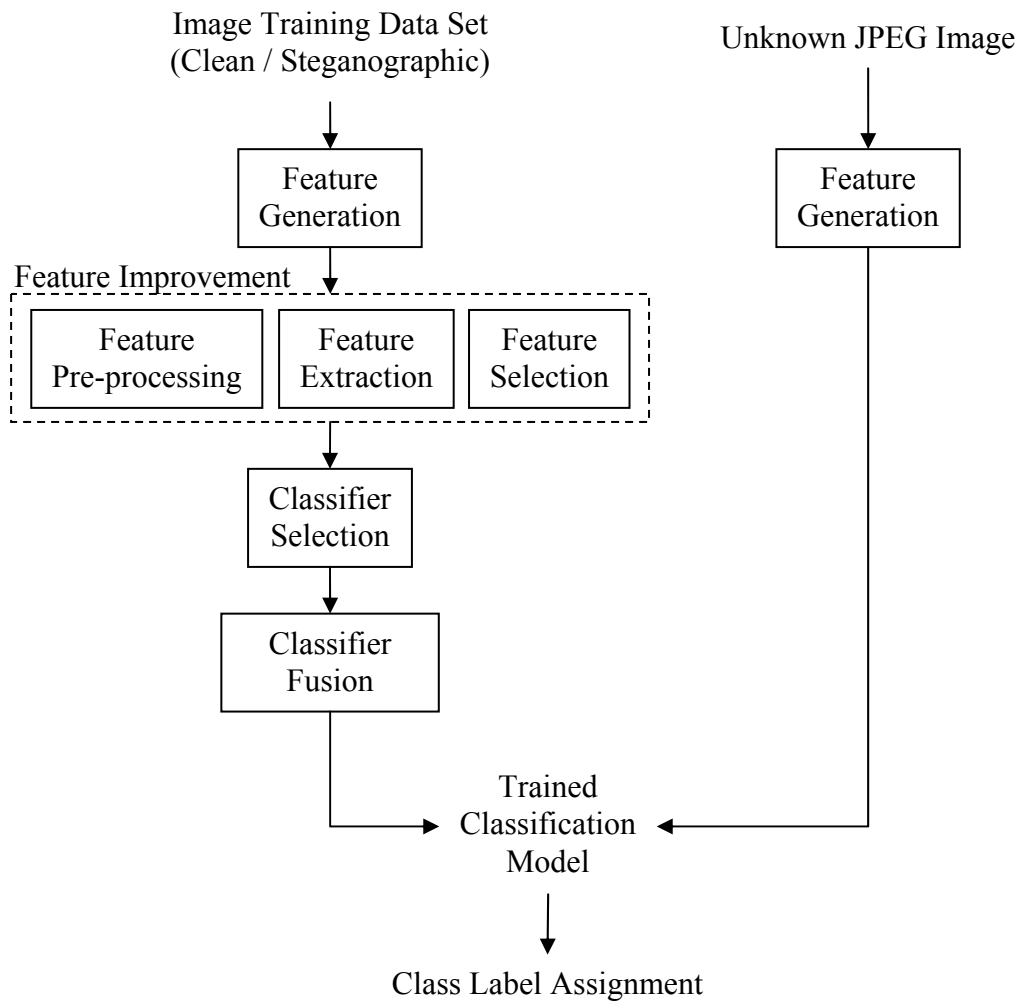


Figure 2.1. Basic Detection System.

Related work on each of these topics is presented in this order in the following sections.

- Image Representation: The JPEG image format is described along with a basic description of the areas within a JPEG image that are manipulated by an embedding method.
- Feature Generation: Using statistical measures to identify changes made to a JPEG image by an embedding method, two transform based methods in this chapter generate one dimensional feature vectors from a matrix image representation.
- Feature Extraction: The methods in this chapter map a set of feature vectors to a lower dimensional space.
- Feature Ranking/Selection: A subset of features is chosen according to feature ranking, noise features and class separability (means and variances).
- Classification: Six classification methods are described, i.e., expectation maximization,  $k$ -nearest neighbors, kernel Fisher's discriminant, Parzen window probabilistic neural networks and support vector machines.
- Multi-class Classification: The multi-class methods include true multi-class classifiers and the combination of two-class classifiers.
- Classifier Fusion: Three fusion methods are described; AdaBoost (Freund and Schapire, 1995), Bayesian networks for model averaging (Murphy, 2001) and probabilistic neural networks.

## **2.1 JPEG Image Representation Background**

In this section, the basic structure of the JPEG image format and the steps in the compression process are described. This is followed by a brief introduction of JPEG image embedding methods.

The Joint Photographic Experts Group (JPEG) format uses lossy compression to achieve high levels of compression on images with many colors (Elysium Ltd., 2004). JPEG is

an international standard for still image compression, and is widely used for compressing gray scale and color images. JPEG images are commonly used for storing digital photos, and publishing Web graphics; tasks for which slight reductions in the image quality are barely noticeable. Due to the loss of quality during the compression process, JPEGs should be used only where image file size is important (Murry and vanRyper, 1994; Brown and Shepherd, 1995).

The JPEG encoder, shown in Figure 2.2, performs compression with the following sequential steps: image preprocessing (divides the input image into  $8 \times 8$  blocks), forward DCT of each  $8 \times 8$  block, quantization with scaling factor, separation of DC and AC coefficients, prediction of the DC coefficient and zig-zag scan the AC coefficients and Huffman encoder (there is a separate encoder for the DC and AC coefficients).

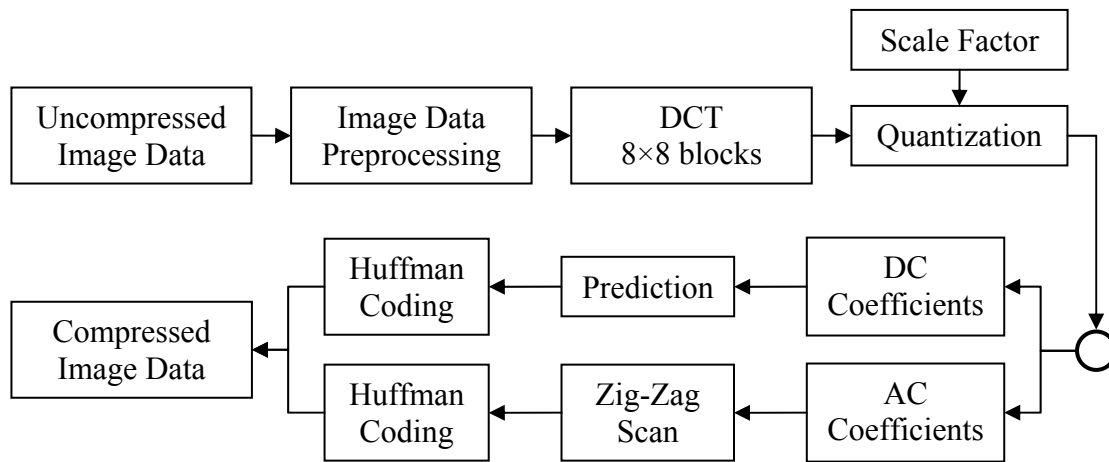


Figure 2.2. Block Diagrams of Sequential JPEG Encoder.

In JPEG decoding, all steps from the encoding process are reversed. The following procedure is a short description of the JPEG baseline systems.



**Preprocessing block** - Subdivides the image into blocks of  $8 \times 8$  pixels and level-shift the original pixel values from the range  $[0, 225]$  to the range  $[-128, +127]$  by subtracting 128. The shifting procedure is a preprocessing step for the DCT calculation.

**Forward DCT block** - Perform a two dimensional discrete cosine transform (DCT) on each level-shifted block  $B$  from the Preprocessing block step. The two dimension DCT is defined as

$$C(n_1, n_2, k_1, k_2) = \begin{cases} \frac{1}{\sqrt{N_1 N_2}} & k_1 = k_2 = 0 \\ \frac{2}{\sqrt{N_1 N_2}} \cos\left(\frac{\pi(2n_1 + 1)k_1}{2N_1}\right) \cos\left(\frac{\pi(2n_2 + 1)k_2}{2N_2}\right) & \begin{matrix} 1 \leq k_1 \leq N_1 - 1 \\ 1 \leq k_2 \leq N_2 - 1 \end{matrix} \end{cases} \quad (2.1)$$

where  $0 \leq n_1 \leq N_1 - 1$  and  $0 \leq n_2 \leq N_2 - 1$ . In the JPEG encoding process,  $N_1 = N_2 = 8$ . The transform is performed on the two dimensional matrix  $B$  as  $CBC^T$ .

The transform helps to remove data redundancy by mapping data from a spatial domain to the frequency domain. No compression has been achieved in this stage, but by changing representation of the information contained in the image block it makes the data more suitable for compression.

**Quantization** - Quantize the DCT coefficients block obtained from the previous step using the quantization table  $Q$ . The quantization table is a matrix used to divide the transformed block for compression purpose by reducing the amplitude of the DCT coefficient values and increasing the number of zero valued coefficients. The Huffman encoder takes advantage of these quantized values. When  $Qs$  is represented the value  $s$  is a scalar multiple, called the scale (or quality) factor, which defines the amount of compression within the image. Higher values of  $s$  yield higher compression. Figure 2.3 shows an instance of  $Qs$ .

$$Q_s = s \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Figure 2.3. Typical quantization matrix.

A set of four quantization tables are specified by the JPEG standard (Independent JPEG Group, 1998). After quantization, most of the DCT coefficients in the 8×8 blocks are truncated to zero values. It is the principal of lossiness in the JPEG transform-based encoder.

**DC Coefficient Coding** - The first coefficient, coefficient 1 (upper left) in Figure 2.4 b) is called the “DC coefficient”, short for the *direct current* coefficient, and represents the average brightness (intensity) of the component block. To encode the DC coefficient, the JPEG standard utilizes a Huffman difference code table that categorizes the value according to the number of  $k$  bits that are required to represent its magnitude. The value of the element is encoded with  $k$  bits.

**AC Coefficients Coding** - The remaining 63 coefficients are the “AC coefficients”, short for the *alternating current* coefficients. The Huffman code assigns short (binary) codewords to each AC coefficient. The AC coefficient encoding scheme is slightly more elaborate than the one for the DC coefficient. For each AC array, a run-length of 0 elements is recorded. When encountering a non-zero element, the length of 0s is recorded and the number of  $k$  bits to represent the magnitude of the element is determined. The

run-length and  $k$  bits are used as a category in the JPEG default Huffman table for assigning a code.

Using a zig-zag run encoder converts the  $8 \times 8$  array of DCT coefficients into a column vector of length  $k$  (zig-zag goes from left to right and top to bottom). The “zig-zag” scan attempts to trace the DCT coefficients according to their significance, shown in Figure 2.

4.

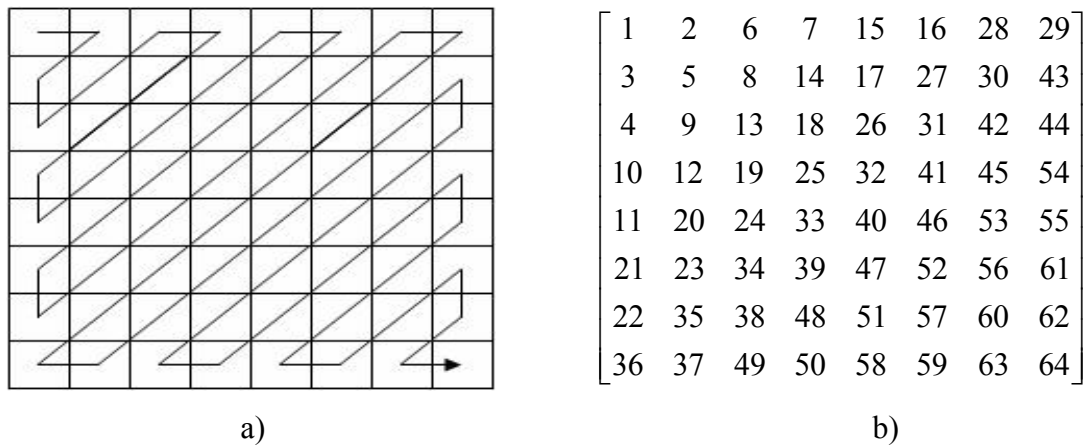


Figure 2.4. DCT decomposition zig-zag structure for an  $8 \times 8$  block, a) zig-zag pattern b) coefficient ordering sequence

The Huffman encoding reduces the number of bits needed to store each of the 64 integer coefficients. For example, when a true color uncompressed image of size  $512 \times 512$  pixels is stored the file size is 769 kilobytes. However, this same image store as a JPEG at a quality factor of 75, the image is stored in 200 kilobytes or smaller. The Huffman encoding tables for the DC and AC coefficients can be found in Gonzalez and Woods (1992, 2002, 2007), Elysium Ltd. (2004), Independent JPEG Group (1998), and JPEG (1994).

One of the primary reasons using image embedding methods for creating stego files is due to the number of redundant portions within a digital image. The vast number of JPEG

images on the Internet makes them ideal cover images for hiding secret data and transmitting them as stego images. In JPEG steganography, the stego message is converted to binary values and embedded into DC and AC coefficients prior to Huffman encoding. By embedding at this stage, the stego message can be extracted without losing the message. The embedding methods range from simple embedding techniques that alter the least significant bits (LSB) of the coefficients such as JP Hide (Latham, 1999) and JSteg (Upham, 1993) to more complicated embedding techniques that maintain natural histograms of the coefficients such as; F5 (Westfeld, 2001; 2003), JP Hide (Latham, 1999), JSteg (Upham, 1993), Model-base (Sallee, 2003; 2006), Model-based Version 1.2 (Sallee, 2008a), OutGuess (Provos, 2004), Steganos (2008), StegHide (Hetzl, 2003) and UTSA (Agaian et al., 2006). The six tools selected provide a set of embedding methods that differ in embedding strategy. Investigation of these methods has provided an insight into six different and unique embedding capacities, embedding patterns and the appearance of the individual feature spaces. Another reason for selecting these particular tools is in previous research and existing steganalysis tools, these 6 embedding methods have been used for analysis (Provos and Honeyman, 2003; Lyu and Farid, 2004; Kharrazi et al., 2005; Shi et al., 2005; Xuan et al., 2005; Fu et al., 2006; Pevny and Fridrich, 2007).

In summary, a useful property of JPEG is that the degree of lossiness can be varied by adjusting the quality factor  $s$  (scale of the quantization table), shown in Figure 2.3. The ease of file sharing with JPEG images and its popularity over the internet has made JPEG image format a desirable cover file for many stego methods. Each embedding method leaves a signature that can be identified by various statistical measures. The next section describes feature generation methods used to identify changes made to a JPEG image.

## **2.2 Feature Generation for JPEG Images**

Several steganalysis feature generation methods used to identify changes made to a JPEG image have been published (Lie and Lin, 2005; Shi et al., 2005; Xuan et al., 2005; Fu et

al., 2006; Wang and Moulin, 2007). In this section two well known methods are discussed. The first method developed by Lyu and Farid (2002; 2004) is a wavelet based method in which features are generated from the wavelet coefficient using various statistics. The second method is a DCT based feature generation method in which the features are developed with the use of functions for the difference between DCT coefficients of input image and of the predicted image (Fridrich, 2004; Pevny and Fridrich, 2006).

The JPEG image coefficients are extracted using a transform, i.e., DCT or wavelet transform, where the wavelet is calculated over the spatial domain not the transform domain. These coefficients represent the image characteristics in a raw format, e.g., low, mid and high frequencies for the DCT and vertical, horizontal and diagonal for the wavelet transforms. The *predictors* which are the estimates of where the stego message is hidden within an image are based on the feature generation method. Lyu and Farid (2002, 2004) use a regression technique to develop the weights associated with the coefficients to produce the predictors. Fridrich (2004) crops an input image and re-expands the image to develop the predictors. The features are finally generated by calculating statistics from the coefficients and the predictors.

### 2.2.1 Wavelet Statistical Model

The image decomposition employed here is based on separable quadrature mirror filters (Lyu and Farid, 2002, 2004). In digital signal processing, a quadrature mirror filter is a filter bank which splits an input signal into two bands, low-pass and high-pass frequencies. The low-pass and high-pass filters are related by the following equation:

$$\left| \hat{h}(\xi) \right|^2 + \left| \hat{h}\left(\xi + \frac{\pi}{2}\right) \right|^2 = 1 \quad (2.2)$$

where  $\xi$  is the frequency, and the sampling rate is normalized to  $2\pi$ , as shown in Figure 2.5.

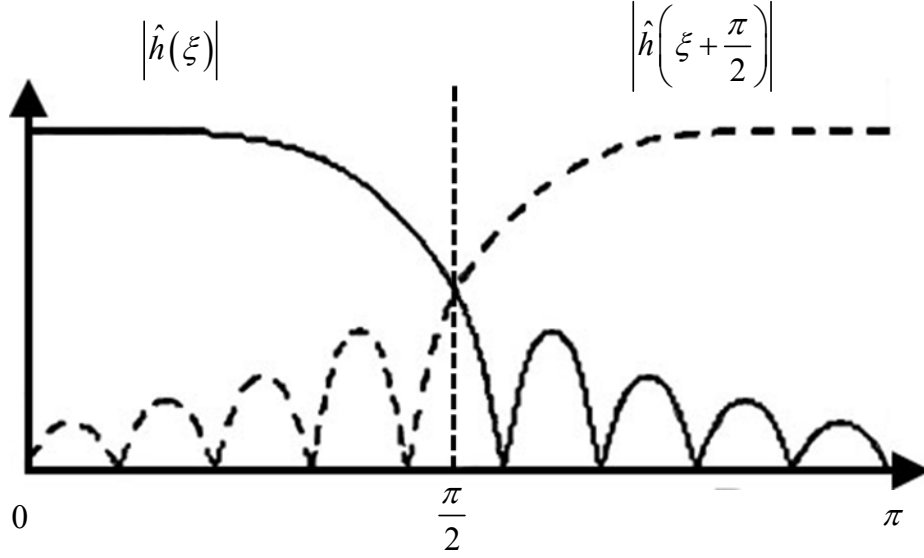


Figure 2.5. Low-pass and high-pass quadrature mirror filter frequency.

Orthogonal wavelets such as the Haar wavelets and related Daubechies wavelets are generated by scaling functions which, with the wavelet, satisfy a quadrature mirror filter relationship (Addison, 2002; Gonzalez and Woods, 2004). Farid (2002) uses a variety of wavelets but in this related work the symmetric quadrature mirror filters (Simoncelli and Adelson, 1990) are used. A wavelet is a mathematical function used to divide a given function into different frequency components and study each component with a resolution that matches its scale. A wavelet transform is the representation of a function by wavelets. The wavelets are scaled and translated copies (known as daughter wavelets) of a finite-length or fast-decaying oscillating waveform (known as the mother wavelet). Wavelet transforms have advantages over traditional Fourier transforms for representing functions that have discontinuities and sharp peaks (Gonzalez and Woods, 2002; Addison, 2002).

The following explanation of the feature generation method is from Lyu and Farid (2002, 2004). The mapping from the spatial domain to the wavelet transform domain,  $f(x, y) \rightarrow V(x, y)$ ,  $H(x, y)$ , and  $D(x, y)$  is a decomposition that splits the frequency space into multiple orientations and scales. For a grayscale image, the vertical, horizontal and diagonal subbands at scale  $i$  are denoted as  $V_i(x, y)$ ,  $H_i(x, y)$ , and  $D_i(x, y)$ , respectively. In Figure 2.6b,  $i$  is equal to 1 for the first level wavelet decomposition and the second level decomposition is represented in Figure 2.6c. For a color (RGB) image, the decomposition is applied independently to each color channel. The resulting subbands are denoted as  $V_i^c(x, y)$ ,  $H_i^c(x, y)$ , and  $D_i^c(x, y)$ , where  $c \in \{r, g, b\}$ .

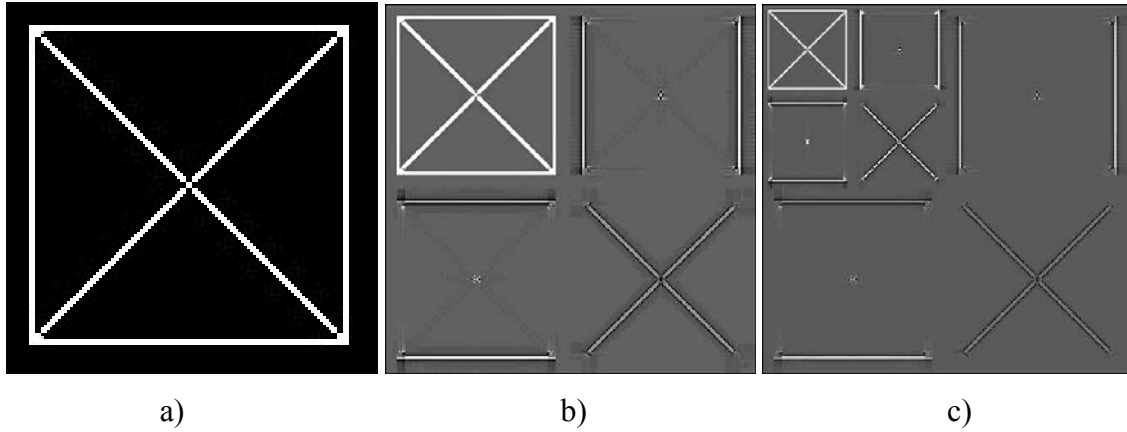


Figure 2.6. Wavelet Structure a) Simple image with vertical, horizontal and diagonal lines b) 2 level wavelet decomposed c) 3 level wavelet decomposition.

Given the decomposed image, the statistical model is composed of the mean  $\mu$ , variance  $\sigma^2$ , skewness  $\gamma_1$  and kurtosis  $\gamma_2$  of the subband coefficients at each orientation, scale and color channel. In order to capture higher-order statistical correlations, a second set of statistics are collected that are based on the errors in a linear predictor of coefficient magnitude. For the purpose of illustration, consider a vertical band of the green channel at scale  $i$ ,  $V_i^g(x, y)$ . A linear predictor for the magnitude of these coefficients in a subset of all possible spatial, orientation, scale, and color neighbors is given by:

$$\begin{aligned}
|V_i^g(x, y)| = & w_1 |V_i^g(x-1, y)| + w_2 |V_i^g(x+1, y)| + w_3 |V_i^g(x, y-1)| \\
& + w_4 |V_i^g(x, y+1)| + w_5 |V_i^g(x/2, y/2)| + w_6 |D_i^g(x, y)| \\
& + w_7 |D_i^g(x/2, y/2)| + w_8 |V_i^r(x, y)| + w_9 |V_i^b(x, y)|
\end{aligned} \tag{2.3}$$

where  $|\cdot|$  denotes absolute value and  $w_k$  are the weights. This linear relationship can be expressed more compactly in matrix form as:

$$\vec{v} = Q\vec{w} \tag{2.4}$$

where  $\vec{v}$  contains the coefficient magnitudes of  $V_i^g(x, y)$  strung out into a column vector (to reduce sensitivity to noise, only magnitudes greater than 1 are considered), the columns of the matrix  $Q$  contain the neighboring coefficient magnitudes as specified in Equation (2.4), and  $\vec{w} = (w_1 \dots w_9)^T$ . The weights  $\vec{w}$  are determined by minimizing the following quadratic error function:

$$E(\vec{w}) = [\vec{v} - Q\vec{w}]^2 \tag{2.5}$$

Using regression techniques the error function is minimized by differentiating with respect to  $\vec{w}$ :

$$\frac{\partial E(\vec{w})}{\partial \vec{w}} = 2Q^T (\vec{v} - Q\vec{w}) \tag{2.6}$$

setting the result equal to zero, and solving for  $\vec{w}$  to yield the following solution:

$$\vec{w} = (Q^T Q)^{-1} Q^T \vec{v} \tag{2.7}$$



Given the large number of constraints (one per pixel) and nine unknowns, it is generally assumed that the  $9 \times 9$  matrix  $Q^T Q$  will be invertible.

Given the linear predictor, the log error between the actual coefficient and the predicted coefficient magnitudes is:

$$\vec{p} = \log(\vec{v}) - \log(|Q\vec{w}|) \quad (2.8)$$

where the  $\log(\cdot)$  is computed point-wise on each vector component. The  $\log(\cdot)$  is used to scale the values of the coefficients. Note, if data standardization is used on the generated features after the statistics are calculated the  $\log(\cdot)$  operation may be omitted. It is from this error that additional statistics are collected namely the mean, variance, skewness and kurtosis. This process is repeated for scales  $i = 1, \dots, n$ , and for the subbands  $V_i^r$  and  $V_i^b$ , where the linear predictors for these subbands are of the form:

$$\begin{aligned} |V_i^r(x, y)| &= w_1 |V_i^r(x-1, y)| + w_2 |V_i^r(x+1, y)| + w_3 |V_i^r(x, y-1)| \\ &\quad + w_4 |V_i^r(x, y+1)| + w_5 |V_i^r(x/2, y/2)| + w_6 |D_i^r(x, y)| \\ &\quad + w_7 |D_i^r(x/2, y/2)| + w_8 |V_i^g(x, y)| + w_9 |V_i^b(x, y)| \end{aligned} \quad (2.9)$$

and

$$\begin{aligned} |V_i^b(x, y)| &= w_1 |V_i^b(x-1, y)| + w_2 |V_i^b(x+1, y)| + w_3 |V_i^b(x, y-1)| \\ &\quad + w_4 |V_i^b(x, y+1)| + w_5 |V_i^b(x/2, y/2)| + w_6 |D_i^b(x, y)| \\ &\quad + w_7 |D_i^b(x/2, y/2)| + w_8 |V_i^r(x, y)| + w_9 |V_i^g(x, y)| \end{aligned} \quad (2.10)$$

A similar process is repeated for the horizontal and diagonal subbands. As an example, the predictor for the green channel takes the form:

$$\begin{aligned}
|H_i^g(x, y)| &= w_1 |H_i^g(x-1, y)| + w_2 |H_i^g(x+1, y)| + w_3 |H_i^g(x, y-1)| \\
&+ w_4 |H_i^g(x, y+1)| + w_5 |H_i^g(x/2, y/2)| + w_6 |D_i^g(x, y)| \\
&+ w_7 |D_i^g(x/2, y/2)| + w_8 |H_i^r(x, y)| + w_9 |H_i^b(x, y)|
\end{aligned} \tag{2.11}$$

$$\begin{aligned}
|D_i^g(x, y)| &= w_1 |D_i^g(x-1, y)| + w_2 |D_i^g(x+1, y)| + w_3 |D_i^g(x, y-1)| \\
&+ w_4 |D_i^g(x, y+1)| + w_5 |D_i^g(x/2, y/2)| + w_6 |H_i^g(x, y)| \\
&+ w_7 |V_i^g(x, y)| + w_8 |D_i^r(x, y)| + w_9 |D_i^b(x, y)|
\end{aligned} \tag{2.12}$$

For the horizontal and diagonal subbands, the predictor for the red and blue channels are determined in a similar way as was done for the vertical subbands, Equations (2.9) and (2.10). For each oriented, scale and color subband, a similar error metric, Equation (2.11), and error statistics are computed.

For a multi-scale decomposition with scales  $i = 1, \dots, s$ , the total number of basic coefficient statistics is  $36(s - 1)$  ( $12(s - 1)$  per color channel), and the total number of error statistics is also  $36(s - 1)$ , yielding a grand total of  $72(s - 1)$  statistics. These statistics form the feature vectors to be used to discriminate between images with and without hidden messages. The set of 72 features representing an input image are used in Chapter 4 as a subset of 526 features for the steganalysis detection system in this research.

### 2.2.2 DCT Features

In this method two types of features are calculated over an image, i.e., first order features and second order features. The following explanation of the generated features in the DCT and spatial domains are from Fridrich, (2004). A vector functional  $F$  is applied to the stego JPEG image  $J_1$ . The stego image  $J_1$  is de-compressed to the spatial domain, cropped by 4 pixels in each direction, and recompressed with the same quantization table

used in decompressing  $J_1$  to obtain  $J_2$ , as shown in Figure 2.7. The vector functional  $F$  is then applied to  $J_2$ . The  $L_1$  norm is defined for a vector/ matrix as a sum of absolute values of all vector/matrix elements. The final feature  $f$  is obtained as an  $L_1$  norm of the difference in the vector functional between the original and modified image as follows:

$$f = \|F(J_1) - F(J_2)\|_{L_1} \quad (2.13)$$

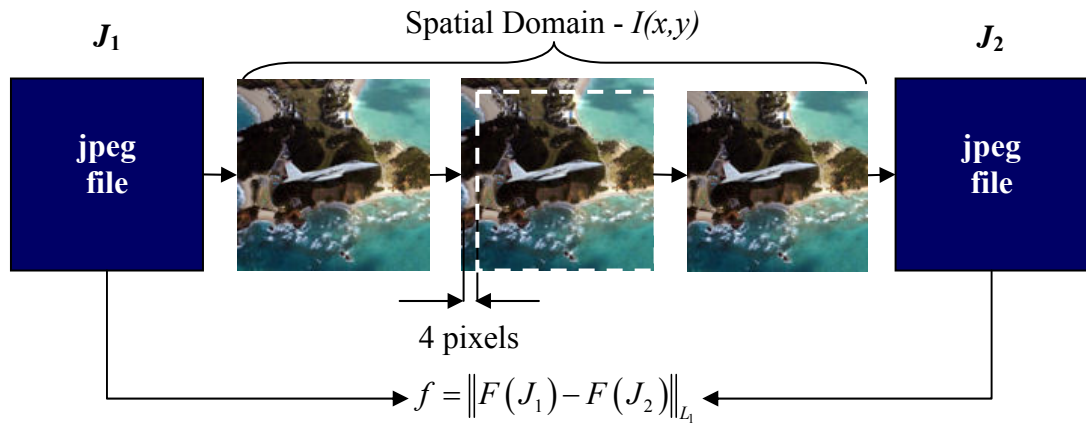


Figure 2.7. Feature generating structure.

**First Order Features** - The simplest first order statistic of DCT coefficients is their histogram. Representing the JPEG image with a DCT coefficient array  $d_k(u,v)$  and a quantization matrix  $Q(u,v)$ , where  $u,v = 1, \dots, 8$ ,  $k = 1, \dots, B$ . The symbol  $d_k(u,v)$  denotes the  $u,v^{\text{th}}$  quantized DCT coefficient in the  $k^{\text{th}}$  block, there are total of  $B$  blocks. The global histogram of all 64  $k$  DCT coefficients is denoted as  $H_r$ , where  $r = L, \dots, R$ ,  $L = \min_{k,i,j} d_k(u,v)$  and  $R = \max_{k,i,j} d_k(u,v)$ .

There are steganographic programs that preserve the histogram. Thus, individual histograms for low frequency DCT modes are added to the set of functionals. For a fixed DCT mode  $(u,v)$ , let  $r = L, \dots, R$ , denote the individual histogram of values  $d_k(u,v)$ ,  $k = 1, \dots, B$ . Only histograms of low frequency DCT coefficients are used because histograms

of coefficients from medium and higher frequencies are usually statistically unimportant due to the small number of non-zero coefficients.

To provide additional first order macroscopic statistics to the set of functionals, dual histograms have been included. For a fixed coefficient value  $d$ , the dual histogram is an  $8 \times 8$  matrix  $g_{uv}^d$

$$g_{uv}^d = \sum_{k=1}^B \delta(d, d_k(u, v)) \quad (2.14)$$

where  $\delta(d, d_k(u, v)) = 1$  if  $u=v$  and 0 otherwise.

**Second Order Features** - Let  $I_r$  and  $I_c$  denote the vectors of block indices while scanning the image “by rows” and “by columns”, respectively. The first functional capturing inter-block dependency is the “variation”  $V$  defined as

$$V = \frac{\sum_{u,v=1}^8 \sum_{k=1}^{|I_r|-1} |d_{I_r(k)}(u, v) - d_{I_r(k+1)}(u, v)| + \sum_{u,v=1}^8 \sum_{k=1}^{|I_c|-1} |d_{I_c(k)}(u, v) - d_{I_c(k+1)}(u, v)|}{|I_r| + |I_c|}. \quad (2.15)$$

Most steganographic techniques in some sense add entropy to the array of quantized DCT coefficients and thus are more likely to increase the variation  $V$  than decrease.

Embedding changes also increase the discontinuities along the  $8 \times 8$  block boundaries. Two blockiness measures  $B_\alpha$ ,  $\alpha = 1, 2$ , have been included to the set of functionals. The blockiness is calculated from the decompressed JPEG image (spatial domain) and thus represents an integral measure of inter-block dependency over all DCT modes over the whole image:

$$B_\alpha = \frac{\sum_{x=1}^{\lfloor (M-1)/8 \rfloor} \sum_{y=1}^N |I(8x, y) - I(8x+1, y)|^\alpha + \sum_{x=1}^{\lfloor (N-1)/8 \rfloor} \sum_{y=1}^M |I(x, 8y) - I(x, 8y+1)|^\alpha}{N \lfloor (M-1)/8 \rfloor + M \lfloor (N-1)/8 \rfloor}. \quad (2.16)$$

In the expression above,  $M$  and  $N$  are image dimensions and  $I(x, y)$  are grayscale values of the decompressed JPEG image.

The final three functionals are calculated from the co-occurrence matrix of neighboring DCT coefficients. Recalling the notation,  $L \leq d_k(u, v) \leq R$ , the co-occurrence matrix  $C$  is a square  $D \times D$  matrix,  $D = R - L + 1$ , defined as follows

$$C_{st} = \frac{\sum_{k=1}^{|I_r|-1} \sum_{u,v=1}^8 \delta(s, d_{I_r(k)}(u, v)) \delta(t, d_{I_r(k+1)}(u, v))}{|I_r| + |I_c|} + \frac{\sum_{k=1}^{|I_c|-1} \sum_{u,v=1}^8 \delta(s, d_{I_c(k)}(u, v)) \delta(t, d_{I_c(k+1)}(u, v))}{|I_r| + |I_c|}. \quad (2.17)$$

The co-occurrence matrix describes the probability distribution of pairs of neighboring DCT coefficients. It usually has a sharp peak at (0,0) and then quickly falls off. Let  $C(J_1)$  and  $C(J_2)$  be the co-occurrence matrices for the JPEG image  $J_1$  and its calibrated version  $J_2$ , respectively. Due to the approximate symmetry of  $C_{st}$  around  $(s, t) = (0, 0)$ , the differences  $C_{st}(J_1) - C_{st}(J_2)$  for  $(s, t) \in \{(0, 1), (1, 0), (-1, 0), (0, -1)\}$  are strongly correlated. The same is true for the group  $(s, t) \in \{(1, 1), (-1, 1), (1, -1), (-1, -1)\}$ . For practically all steganographic schemes, the embedding changes to DCT coefficients make perturbations by some small value. Thus, the co-occurrence matrix for the embedded image can be obtained as a convolution  $CP(q)$ , where  $P$  is the probability distribution of the embedding distortion, which depends on the relative message length  $q$ . This means that the values of

the co-occurrence matrix  $CP(q)$  will be more “spread out”. To quantify this spreading, the following three quantities are taken as features:

$$N_{00} = C_{0,0}(J_1) - C_{0,0}(J_2)$$

$$N_{01} = C_{0,1}(J_1) - C_{0,1}(J_2) + C_{1,0}(J_1) - C_{1,0}(J_2) + C_{-1,0}(J_1) - C_{-1,0}(J_2) + C_{0,-1}(J_1) - C_{0,-1}(J_2)$$

$$N_{11} = C_{1,1}(J_1) - C_{1,1}(J_2) + C_{1,-1}(J_1) - C_{1,-1}(J_2) + C_{-1,1}(J_1) - C_{-1,1}(J_2) + C_{-1,-1}(J_1) - C_{-1,-1}(J_2) .$$

The final set of 20 vector functionals used in this method is summarized in Table 2.1. Three additional features are listed in the bottom of Table 2.1.

Table 2.1. All 23 distinguishing functionals.

Functional/Feature Name	Functional $F(\cdot)$
Global Histogram	$H / \ H\ _{L_1}$
Individual Histogram for 5 DCT Modes	$\frac{h^{21}}{\ h^{21}\ _{L_1}}, \frac{h^{31}}{\ h^{31}\ _{L_1}}, \frac{h^{12}}{\ h^{12}\ _{L_1}}, \frac{h^{22}}{\ h^{22}\ _{L_1}}, \frac{h^{13}}{\ h^{13}\ _{L_1}}$
Dual Histograms for 11 DCT Values (-5,...,5)	$\frac{g^{-5}}{\ g^{-5}\ _{L_1}}, \frac{g^{-4}}{\ g^{-4}\ _{L_1}}, \dots, \frac{g^4}{\ g^4\ _{L_1}}, \frac{g^5}{\ g^5\ _{L_1}}$
Variation	$V$
$L_1$ and $L_2$ Blockiness	$B_1, B_2$
Co-occurrence	$N_{00}, N_{01}, N_{11}$ (features not functionals)

The features in Table 2.1 are extended from 23 to 193 by analyzing DCT coefficients in the range of -5 to 5 (Pevny and Fridrich, 2007). Apply the cropping technique in Figure 2.7 with a Markov process an additional 81 features are created for a total of 274 features (Pevny and Fridrich, 2007). The set of 274 features representing an input image are used in Chapter 4 as a subset of 526 features for the steganalysis detection system in this research.

## 2.3 Feature Preprocessing

After features are generated it is necessary to preprocess the features that are to be used for classification. In many practical situations the classification model may receive input features whose values lie within different dynamic ranges. Thus, features with large values may inadvertently influence classification over features with small values. Another problem arises when a particular sample is not within the same area as the other features. To resolve these issues the feature preprocessing methods used in this research are data normalization (Theodoridis and Koutroumbas, 2006, pp. 214-215), data standardization (Dillon and Goldstein, 1984, pp. 12-13) and outlier removal (Barnett and Lewis, 1994). The training vectors in this section are represented by  $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\ell] \in \mathbb{R}^n$  with a dimension of  $n$  and the number of sample defined as  $\ell$ .

### 2.3.1 Data Preparation

Data preparation scales the features so that they have similar magnitudes. Some of the procedures used for data preparation are feature standardization (Dillon and Goldstein, 1984, pp. 12-13), feature min-max normalization (Theodoridis and Koutroumbas, 2006, pp. 214-215), min-max global normalization (Guyon et al., 2006, pp. 254), sigmoid normalization (Theodoridis and Koutroumbas, 2006, pp. 214-215) and softmax scaling (Theodoridis and Koutroumbas, 2006, pp. 214-215). We use zero-mean normalization (feature standardization) and min-max normalization (feature normalization) and describe them in more details.

**Min-max normalization** performs a linear scaling on the original data. The normalization is calculated by estimates of the minimum and maximum of the values. The normalization technique is defined for the  $\ell$  available data samples and the  $k^{\text{th}}$  feature as:

$$\hat{x}_{ik} = \frac{x_{ik} - \min(x_k)}{\max(x_k) - \min(x_k)}((b-a) + a), \quad k = 1, 2, \dots, n \quad (2.18)$$

where  $a$  and  $b$  are scaling factors. When  $a = 0$  and  $b = 1$  the individual feature values are in the range of  $[0,1]$ . In the event that the denominator of Equation (2.18) is equal to zero that feature is removed, avoiding the potential of normalizing a feature of constants.

**Z-score normalization (Standardization)** is based on the mean and standard deviation of each feature. Each feature in this method is separately standardized by subtracting its mean and dividing by the standard deviation as follows:

$$\hat{x}_{ik} = \frac{x_{ik} - \mu_k}{\sigma_k}, \quad k = 1, 2, \dots, \ell \quad (2.20)$$

where  $\mu_k$  and  $\sigma_k$  are defined as:

$$\mu_k = \frac{1}{\ell} \sum_{i=1}^{\ell} x_{ik} \quad (2.21)$$

$$\sigma_k^2 = \frac{1}{\ell - 1} \sum_{i=1}^{\ell} (x_{ik} - \mu_k)^2 \quad (2.22)$$

and  $\ell$  is the number of samples. In the event that the standard deviation of a particular feature is zero (e.g., each element of the observed feature is a constant value), the feature is discarded.



### 2.3.2 Outlier Removal

An outlier is defined as a sample that is inconsistent with the existing sample distribution. The inconsistency is defined by the analyst observing the input data. Outliers can be discarded if the number of samples is small in comparison with the remaining samples, e.g., one or two samples. Various guides are provided by Barnett and Lewis (1994) to determine a small number of outliers. When a large number of outliers exist, care must be taken by the analyst. In this case, the classification model may have to be trained to accommodate the presence of outliers, e.g., expectation maximization can be trained using ellipsoids. Two outlier removal techniques are used in the case of multivariate outliers in this section. The first is a technique in which the mean is used to identify an upper and lower boundary of a confidence interval to identify an outlier and remove the sample (Barnett and Lewis, 1994). The second is a multivariate outlier technique presented by Wilks (1963).

**Confidence Interval Outlier Removal** – In confidence interval outlier removal, any sample outside of the confidence interval is considered an outlier. This method assumes the data is normally distributed and generates a confidence interval for each feature. The first step identifies an upper and lower limit means from the global mean as follows:

$$\mu_{upper} = \frac{1}{S_{upper}} \sum_{i \in S_{upper}} \mathbf{x}_i, \text{ for } \mathbf{x}_i > \mu \quad (2.23)$$

$$\mu_{lower} = \frac{1}{S_{lower}} \sum_{i \in S_{lower}} \mathbf{x}_i, \text{ for } \mathbf{x}_i < \mu \quad (2.24)$$

where  $\mu$  is the global mean vector

$$\mu = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{x}_i \quad (2.25)$$

where  $\mathcal{L}$  is the number of samples,  $S_{upper}$  and  $S_{lower}$  are the number of samples meeting the criteria  $\mathbf{x}_i > \mu$  and  $\mathbf{x}_i < \mu$ , respectively. The term  $i \in S_{lower}$  and  $i \in S_{upper}$  indicate the indices when the criteria  $\mathbf{x}_i < \mu$  and  $\mathbf{x}_i > \mu$  are met. This now leads to the confidence interval defined as:

$$\left[ \mu_{lower} - \left( \alpha (\mu - \mu_{lower}) \right), \mu_{upper} + \left( \alpha (\mu_{upper} - \mu) \right) \right] \quad (2.26)$$

where  $\alpha$  is the parameter set by the user. A good starting point is  $\alpha = 0.5$  allowing the parameter to be adjusted based on the data set being analyzed. The terms multiplied by  $\alpha$  in Equation (2.26) can be replaced by the critical of the  $t$ -distribution as described by Barnett and Lewis (1994, page 74) providing robustness of validity for the confidence interval. Another alternate modification to Equation (2.26) is to simply replace  $(\mu - \mu_{lower})$  by the standard deviation of  $\mu_{lower}$  and  $(\mu_{upper} - \mu)$  by the standard deviation of  $\mu_{upper}$  allowing the standard deviation to determine the confidence interval.

**Wilks' Outlier Removal** – Wilks' outlier removal technique uses an upper bound for detection of a single outlier from a set of normal multivariate samples in which the maximum squared Mahalanobis distance (Equation (2.27)) approaches an  $F$  distribution (Wilks, 1963).

$$D_i^2 = (\mathbf{x}_i - \mu) \Sigma^{-1} (\mathbf{x}_i - \mu)^T \quad (2.27)$$

In multivariate outlier detection the normality between samples is assessed. A partial mathematical description is provided by Rencher (2002, pp. 101-104) and expanded in application by Trujillo-Ortiz, et al. (2008).

Determining the threshold is defined by the  $F$  distribution critical value (inverse of  $F$  cumulative distribution function) with  $n$  and  $(\ell-n-1)$  degrees of freedom using the Bonferroni correction (Bonferroni, 1935; 1936). The final critical value is defined by:

$$C_v = \frac{n(\ell-1)^2}{\ell(\ell-n-1) + (\ell n F)} \quad (2.28)$$

The index of an outlier(s) is identified by the following criteria:

$$D_i^2 \geq C_v \quad (2.29)$$

This method is provided in full detail by Trujillo-Ortiz, et al. (2008).

## 2.4 Feature Extraction

Feature extraction maps the input samples,  $\mathbf{x}$ , from the input feature space  $\mathbf{x} \in \mathbb{R}^n$  to a new feature space  $\mathbf{z} \in \mathbb{R}^p$ , where  $n > p$ , features are extracted. In this case, the classification is based on the samples in the new feature space,  $\mathbf{z}$ , rather than on the input feature space. The advantage of feature extraction over feature selection is that no information from any of the elements of the input feature is lost. In certain situations feature extraction may be easier to calculate than feature selection. In this section two feature extraction methods are discussed, principal component analysis (PCA) where the new feature space  $\mathbf{z} \in \mathbb{R}^m$  and kernel PCA where the feature space  $\hat{\mathbf{z}} \in \mathbb{R}^p$ .

### 2.4.1 Principal Component Analysis (PCA)

The idea of feature extraction using PCA (Hotelling, 1933) is to represent a new space in a way to extract mutually uncorrelated features from the current space. The new features are known as the principal components after transform mapping. The dimensionality assessment is accomplished by extracting the principal components from the correlation matrix and retaining only the factors described in Kaiser's criterion (eigenvalues:  $\lambda \geq 1$ ) (Kaiser, 1960). The criterion is used as a guide line to determine the number of principal components to retain by calculating the correlation matrix of the input features. Each observed variable contributes one unit of variance to the total variance in the data set. Hence, any principal component that has an eigenvalue,  $\lambda$ , greater than one accounts for a greater amount of variance than had been contributed by one variable. Additionally, a principal component that displays an eigenvalue less than one indicates less variance than had been contributed by one variable. The covariance matrix,  $\Sigma$ , is used to extract eigenvectors,  $\mathbf{e}$ , retaining only the number of principal components corresponding to Kaiser's criterion.

The basic concept of feature extraction using PCA is to map  $\mathbf{x}$  onto a new space capable of reducing the dimensionality of the input space. The data is partitioned by variance using a linear combination of 'original' factors. To perform PCA, let  $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L] \in \mathbb{R}^n$  be a set of training vectors from the  $n$ -dimensional input space  $\mathbb{R}^n$ . The set of vectors  $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L] \in \mathbb{R}^m$  is a lower dimensional representation of the input training vectors  $\mathbf{x}$  in the  $m$ -dimensional space  $\mathbb{R}^m$ . The vectors  $\mathbf{z}$  are obtained by the linear orthonormal projection

$$\mathbf{z} = \mathbf{A}^T (\mathbf{x} - \mu) \quad (2.30)$$

where  $\mathbf{A}$  is an  $[n \times m]$  matrix containing the top  $m$  eigenvectors and  $\mu$  is the mean of the each set of features from  $\mathbf{x}$ .

### 2.4.2 Kernel PCA

The Kernel Principal Component Analysis (Kernel PCA) is the non-linear extension of the ordinary linear PCA (Scholkopf et al., 1998). The input training vectors  $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\ell] \in \mathbb{R}^n$  are mapped by a nonlinear transformation  $\phi(\cdot): X \rightarrow F$  to a new dimensional feature space  $F \in \mathbb{R}^\ell$ . The mapping  $\phi(\cdot)$  is represented in the kernel PCA by a kernel function  $K(\cdot, \cdot)$  which defines an inner product in  $\mathbb{R}^\ell$ . This yields a non-linear (kernel) projection of data which has a general definition as

$$\hat{\mathbf{z}} = \hat{\mathbf{A}}^T K(\mathbf{x}_i, \mathbf{x}_j) + \mathbf{b} \quad (2.31)$$

where  $\hat{\mathbf{A}}$  is an  $[\ell \times p]$  matrix containing the top  $p$  values,  $\mathbf{b}$  is a bias vector and  $\hat{\mathbf{z}} \in \mathbb{R}^p$  is the vector of extracted features. The eigenvectors are not computed directly from the kernel matrix  $K(\cdot, \cdot)$ . The kernel matrix must be centered as follows:

$$\mathbf{K}_c = K(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{1}_{[\ell \times \ell]} K(\mathbf{x}_i, \mathbf{x}_j) - K(\mathbf{x}_i, \mathbf{x}_j) \mathbf{1}_{[\ell \times \ell]} + \mathbf{1}_{[\ell \times \ell]} K(\mathbf{x}_i, \mathbf{x}_j) \mathbf{1}_{[\ell \times \ell]} \quad (2.32)$$

where  $\mathbf{1}_{[\ell \times \ell]}$  is a  $[\ell \times \ell]$  matrix in which every value is  $1/\ell$ . The eigenvalues,  $\lambda$ , and eigenvectors,  $\mathbf{e}$ , are determined with the use of  $\mathbf{K}_c$ . The bias vector  $\mathbf{b}$  is computed as:

$$\mathbf{b} = \hat{\mathbf{A}}^T \left( \mathbf{1}_{[\ell \times \ell]} K(\mathbf{x}_i, \mathbf{x}_j) \mathbf{1}_\ell - K(\mathbf{x}_i, \mathbf{x}_j) \mathbf{1}_{[\ell \times \ell]} \right) \quad (2.33)$$

where  $\mathbf{1}_\ell$  is an  $[\ell \times 1]$  vector with each element equal to  $1/\ell$ .

## 2.5 Feature Ranking/Selection

When a decision problem has an extremely large number of features, often a classification algorithm has difficulty identifying the best features to use for classification. For this reason one step in the classification process is the identification of features that retain as much class discriminatory information as possible. This procedure is known as feature ranking/selection or reduction. A first step in feature ranking/selection is to look at each of the feature independently and test its discriminatory capability for the problem. Although looking at the features independently is far from optimal, this procedure helps to discard features that do not separate the classes. In this section, five ranking methods are described which are used in this research, Bhattacharyya distance, Fisher's discriminant ratio, signal to noise ratio, kernel Fisher's discriminant feature ranking and zero-norm feature ranking. The selection of vital features for each of these methods is determined by the user based on either a ranking value threshold or the classification accuracy of a selected subset of top ranked features.

### 2.5.1 Bhattacharyya Distance

The Bhattacharyya distance is used as a class separability measure. For two-class normal distributions the Bhattacharyya distance is defined as:

$$B = \frac{1}{8} (\mu_{-1} - \mu_{+1})^T \left( \frac{\Sigma_{-1} + \Sigma_{+1}}{2} \right)^{-1} (\mu_{-1} - \mu_{+1}) + \frac{1}{2} \ln \frac{|\frac{\Sigma_{-1} + \Sigma_{+1}}{2}|}{\sqrt{|\Sigma_{-1}| |\Sigma_{+1}|}} \quad (2.34)$$

where  $|\cdot|$  denotes the determinant of the respective matrix. The Bhattacharyya distance corresponds to the optimum Chernoff bound when  $\Sigma_{-1} = \Sigma_{+1}$ . It is readily seen that in this case the Bhattacharyya distance becomes proportional to the Mahalanobis distance between the means. It should be noted that the Bhattacharyya distance consists of two terms. The first term gives the class separability due to the mean difference and disappears when  $\mu_{-1} = \mu_{+1}$ . The second term gives the class separability due to the covariance difference and disappears when  $\Sigma_{-1} = \Sigma_{+1}$  (Fukunaga, 1990).

The Bhattacharyya distance for the multi-class case is represented as:

$$B_{ij} = \frac{1}{8} (\mu_i - \mu_j)^T \left( \frac{\sigma_i + \sigma_j}{2} \right)^{-1} (\mu_i - \mu_j) + \frac{1}{2} \ln \left( \frac{\sigma_i^2 + \sigma_j^2}{2\sigma_i\sigma_j} \right), i \neq j \quad (2.35)$$

where  $i, j \in \mathbb{Z}$  in this case corresponding to the classes  $\mathbf{C} = C_j = [C_1, C_2, \dots, C_c]$ ,  $j = 1, 2, \dots, c$ . In this case for each feature an individual class is compared to the remaining classes based on distance. The features are assigned a ranking value according to the greatest distance between classes.

### 2.5.2 Fisher's Linear Discriminant Ratio (FDR/F-Score)

The FDR is used to quantify the separability capabilities of individual features (Fisher, 1936). FDR is a simple technique which measures the discrimination of sets of real numbers. The within-class scatter matrix is defined as

$$S_w = \sum_{\mathbf{C}} P_{\mathbf{C}} S_{\mathbf{C}} \quad (2.36)$$

where  $S_{\mathbf{C}}$  is the covariance matrix for class  $\mathbf{C} \in \{-1, +1\}$

$$S_C = \sum_{\substack{i=1 \\ i \in C}}^{\ell_C} (\mathbf{x}_i - \mu_C)(\mathbf{x}_i - \mu_C)^T \quad (2.37)$$

and  $P_C$  is the *a priori* probability of class  $C$ . That is,  $P_C \approx \ell_C/\ell$ , where  $\ell_C$  is the number of samples in class  $C$ , out of a total of  $\ell$  samples. The between-class scatter matrix is defined as

$$S_B = \sum_C P_C (\mu_C - \mu)(\mu_C - \mu)^T \quad (2.38)$$

where  $\mu$  is the global mean vector

$$\mu = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{x}_i \quad (2.39)$$

and the class mean vectors  $\mu_C$  is defined as

$$\mu_C = \frac{1}{\ell_C} \sum_{\substack{i=1 \\ i \in C}}^{\ell_C} \mathbf{x}_i. \quad (2.40)$$

These criteria take a special form in the one-dimensional, two-class problem. In this case, it is easy to see that for equiprobable classes  $|S_W|$  is proportional to  $\sigma_{-1}^2 + \sigma_{+1}^2$  and  $|S_B|$  proportional to  $(\mu_{-1} - \mu_{+1})^2$ . Combining  $S_B$  and  $S_W$ , the Fisher's Discriminant ratio results in the following equation

$$\mathbf{FDR} = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}. \quad (2.41)$$



$FDR$  is sometimes used to quantify the separability capabilities of individual features. For the multi-class case, averaging forms of  $FDR$  can be used. One possibility is

$$FDR = \sum_i^M \sum_{j \neq i}^M \frac{(\mu_i - \mu_j)^2}{\sigma_i^2 + \sigma_j^2} \quad (2.42)$$

where the subscripts  $i, j$  refer to the mean and variance corresponding to the feature under investigation for the classes  $C_i, C_j$ , respectively.

For the one-dimensional multi-class case, the Fisher's discriminant ratio is modified as:

$$FDR_{ij} = \frac{(\mu_i - \mu_j)^2}{\sigma_i^2 + \sigma_j^2} \quad (2.43)$$

### 2.5.3 Signal-to-Noise Feature Selection

One method for neural networks feature selection uses a signal-to-noise ratio ( $SNR$ ) saliency measure (Bauer et al., 2000). This measure directly compares the saliency of a feature to that of an injected noise feature. The  $SNR$  saliency measure is computed using the following:

$$SNR_i = 10 \log_{10} \left( \frac{\sum_{j=1}^J (w_{i,j}^1)^2}{\sum_{j=1}^J (w_{N,j}^1)^2} \right) \quad (2.44)$$

where  $SNR_i$  is the value of the  $SNR$  saliency measure for feature  $i$ ,  $J$  is the number of hidden nodes,  $w_{i,j}^1$  is the first layer weight from node  $i$  to node  $j$ , and  $w_{N,j}^1$  is the first layer

weight from the injected noise node  $N$  to node  $j$ . The weights,  $w_{N,j}^1$  for the noise feature are initialized and updated in the same fashion as the weights,  $w_{i,j}^1$  emanating from the other features in the first layer. The injected noise feature is created such that its distribution follows that of a Uniform (0,1) random variable. The *SNR* screening method potentially requires only a single training run, because the *SNR* saliency measure appears highly robust relative to the effects of weight initialization. For the classification method probabilistic neural network described in Chapter 3, this method is used to determine the appropriate subset of features.

#### 2.5.4 Kernel Fisher's Recursive Feature Elimination

The SVM-RFE (Guyon et al., 2002) discussed in Section 2.1 is extended to the kernel Fisher's discriminant (KFD) for feature ranking. The method in this subsection starts with all  $n$  available features, and performs KFD on the kernel space alpha vectors  $\alpha$  (Louw and Steel, 2006). The feature ranking value for the kernel Fisher's recursive feature elimination (KF-RFE) is calculated as

$$R_m = \frac{\alpha^T M^{(m)} \alpha}{\alpha^T N^{(m)} \alpha} \quad (2.45)$$

where

$$\begin{aligned} M^{(m)} &= \left( M_{-1}^{(m)} - M_{+1}^{(m)} \right) \left( M_{-1}^{(m)} - M_{+1}^{(m)} \right)^T \\ M_{-1}^{(m)} &= \frac{1}{\ell_{-1}} \sum_{\substack{j=1 \\ j \in C_{-1}}}^{\ell_{-1}} K^{(m)}(\mathbf{x}_i, \mathbf{x}_j) \\ M_{+1}^{(m)} &= \frac{1}{\ell_{+1}} \sum_{\substack{j=1 \\ j \in C_{+1}}}^{\ell_{+1}} K^{(m)}(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \quad (2.46)$$

and

$$N^{(m)} = \sum_{\substack{j=1 \\ j \in \mathbf{C}}}^{\ell_{\mathbf{C}}} K^{(m)}(\mathbf{x}_i, \mathbf{x}_j) \left( I - 1/\ell_{\mathbf{C}} \right) K^{(m)}(\mathbf{x}_i, \mathbf{x}_j)^T \quad (2.47)$$

where  $\mathbf{C} = \{C_{-1}, C_{+1}\} = \{-1, +1\}$ . The KF-RFE algorithm consists of the following steps:

1. Calculate the alpha values as:

$$\alpha = (M_{-1} - M_{+1}) / N$$

2. For the number of input features  $n$  initialize the feature dimensionality as  $\tilde{n} = n$ , perform steps 3 through 6,  $n$  times.
3. For the number of input features  $\tilde{n}$  perform steps 3 through 5,  $\tilde{n}$  times.
4. Assign ranking values  $R_m$  by calculating Equation (2.45), removing one feature at a time at location  $m$ .
5. Sort the ranking values  $R_m$  removing the highest ranked feature, storing the index of the removed feature and assign the new dimension as  $\tilde{n} \leftarrow \tilde{n} - 1$ .

### 2.5.5 Zero-Norm Feature Ranking

Weston, et al. (2003) proposed a zero-norm feature ranking method capable of identifying features that are close to linear separation. This method was extended to the nonlinear case by using support vector machines with kernels capable of separating nonlinear features. The nonlinear feature selection method calculates ranking values ( $R_m$ ) as follows:

$$R_m = \sum_{k,j} \alpha_k \alpha_j y_k y_j \left( K(\mathbf{x}_k, \mathbf{x}_j) \bullet K^{(m)}(\mathbf{x}_k, \mathbf{x}_j) \right) \quad (2.48)$$

where  $(\bullet)$  in this method is a point by point multiplication of the two kernel matrices. The zero-norm feature ranking algorithm consists of the following steps:

1. Initialize the weights,  $\hat{\mathbf{w}}$ , to ones.
2. Weight  $\mathbf{x}$  by the weights  $\hat{\mathbf{w}}$ ,  $\mathbf{x} \leftarrow \mathbf{x} \cdot \hat{\mathbf{w}}$ .
3. Using the selected SVM model identify the alpha values,  $\alpha$ , and support vectors,  $\mathbf{x}_k$ .
4. Calculate Equation (2.48).
5. Calculate the new weights  $\hat{\mathbf{w}} \leftarrow \hat{\mathbf{w}} |\max(R_m) - R_m|^T$ .
6. Sort the weights  $\hat{\mathbf{w}}$ , identify the weights that are less than a set threshold, remove the features corresponding to the identified weights, and store the index of the ranked feature.
7. Repeat steps 2 through 6 until the maximum number of iterations is met or all of the features have been ranked.

The threshold used in step 6 is set to the maximum  $\hat{\mathbf{w}}$  divided by  $10^3$  (Weston et al., 2006). The remaining weights  $\hat{\mathbf{w}}$  for the nonlinear case should be normalized between zero and one to avoid an unnecessary feature increase in step 2. The maximum number of iterations, 20 (Weston et al., 2006), in step 7 avoids calculating  $n$  number of SVM models in step 3.

## 2.6 Classification

Machine learning for a classification task involves training over a set of samples  $\mathbf{x} = [x_1, x_2, \dots, x_\ell]^T \in \mathbb{R}^n$ . Each sample in the training set contains one target value  $\mathbf{C} = C_j = [C_1, C_2, \dots, C_c]$ ,  $j = 1, 2, \dots, c$ , (known as the class labels  $y_i \in \mathbf{C}$ ,  $i = 1, 2, \dots, \ell$ ) which describes the class to which the sample is a member of. The objective is to separate the data into their classes such that the degree of association is strong between the data sets of the same class and weak between members of different classes. From the class separation, an unseen sample  $\mathbf{x}_0 \in \mathbb{R}^n$  can then be appropriately classified. In this section six classification methods are presented, expectation maximization with mixture models (EM),  $k$ -nearest neighbors ( $k$ -NN), kernel Fisher's discriminant (KFD), Parzen window, probabilistic neural networks (PNN) and support vector machines (SVM).

### 2.6.1 Expectation Maximization (EM)

The idea behind the EM algorithm (Dempster et al., 1977) is that even though the data values of  $\mathbf{x}$ , feature vectors  $\mathbf{x} \in \mathbb{R}^n$ , are unknown/incomplete the distribution  $f(\mathbf{x}|p)$  can be used to determine an estimate for the maximum likelihood (Tomasi, 2006). In maximum likelihood estimation, the estimate to be modeled is the parameter(s) for which the observed data are the most likely. This is done by iteratively estimating the data parameters, then using the data to update the estimated parameters, until a desired convergence is met. The two major steps of the EM algorithm are the expectation step (E-Step) and the maximization step (M-Step).

The EM algorithm consists of choosing initial parameters for the means,  $\mu_k^{(j)}$ , standard deviations,  $\sigma_k^{(j)}$ , and mixing probabilities,  $p^{(j)}(k|\mathcal{L})$ , for a user defined number of clusters,  $k$ , then performing the E-Step and M-Step successively until convergence, where  $i$  is the current iteration and  $n$  is the number of samples. The convergence criteria is determined by examining when the parameters quit changing, i.e., when  $|\mu_k^{(j)} - \mu_k^{(j+1)}| < \varepsilon$  &  $|\sigma_k^{(j)} - \sigma_k^{(j+1)}| < \varepsilon$  &  $|p^{(j)}(k|\mathcal{L}) - p^{(j+1)}(k|\mathcal{L})| < \varepsilon$  for some epsilon ( $\varepsilon$ ) and distance calculation (Euclidian distance). The maximum likelihood estimation is a method of estimating the parameters of the distributions based upon the observed data.

The expectation step (E-Step) calculates the membership probabilities,  $p(k|\mathcal{L})$  (Tomasi, 2006). The mixing probabilities  $p_k$  are viewed as the sample mean of the membership probabilities  $p(k|\mathcal{L})$  assuming a uniform distribution over all the data points. The Gaussian function,  $g(\mathbf{x}; \mu_k^{(i)}, \sigma_k^{(i)})$ , is used to compute mixture of Gaussian functions as shown in the denominator of  $p(k|\mathcal{L})$ .

$$p^{(j)}(k|\mathcal{L}) = \frac{p_k^{(j)} g(\mathbf{x}; \mu_k^{(j)}, \sigma_k^{(j)})}{\sum_{k=1}^K p_k^{(j)} g(\mathbf{x}; \mu_k^{(j)}, \sigma_k^{(j)})} \quad (2.49)$$

$$g(\mathbf{x}; \mu_k^{(j)}, \sigma_k^{(j)}) = \frac{1}{(\sqrt{2\pi}\sigma_k)^n} e^{-\frac{1}{2}\left(\frac{\|\mathbf{x}-\mu_k\|}{\sigma_k}\right)^2} \quad (2.50)$$

The maximization step (M-Step) uses the data from the expectation step as if it were measured data to determine the maximum likelihood estimate of the parameter (Tomasi, 2006). This estimated data is often referred to as the “imputed” data. This step is dependent upon the membership probabilities  $p(k|\mathcal{L})$  which are computed in the E-Step. The EM algorithm consists of iterating the mean, standard deviation, and mixing probabilities until convergence. The mixing probabilities are the sample mean of the conditional probabilities  $p(k|\mathcal{L})$  assuming a uniform distribution over all the data points.

$$\mu_k^{(j+1)} = \frac{\sum_{i=1}^{\mathcal{L}} p^j(k|i) \mathbf{x}_i}{\sum_{i=1}^{\mathcal{L}} p^j(k|i)} \quad (2.51)$$

$$\sigma_k^{(j+1)} = \sqrt{\frac{1}{D} \frac{\sum_{i=1}^{\mathcal{L}} p^j(k|i) \|\mathbf{x}_i - \mu_k^{(j+1)}\|^2}{\sum_{i=1}^{\mathcal{L}} p^j(k|i)}} \quad (2.52)$$

$$p_k^{(j+1)} = \frac{1}{\mathcal{L}} \sum_{i=1}^{\mathcal{L}} p^j(k|i) \quad (2.53)$$

### 2.6.1.1 Mixture Models

In mixture models, also known as model-based Gaussian clustering, the multivariate Gaussian normal is used as a density function similarly described in Equation (2.50). The general multivariate normal density for  $n$  dimensions is

$$g(\mathbf{x}; \mu_k, \Sigma_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right)}{(\sqrt{2\pi})^n |\Sigma_k|^{1/2}}. \quad (2.54)$$

The geometric characteristics (size, shape and orientation) of the clusters are determined by the covariance matrix  $\Sigma_k$  which is generated in terms of eigenvalue decomposition described in Martinez and Martinez (2002). The decomposition of the covariance matrix  $\Sigma_k$  is used as a suitable model for the geometric characteristics of the cluster. The structure of the covariance matrix is as follows:

$$\Sigma_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T \quad (2.55)$$

where  $\lambda_k$  is a scalar,  $D_k$  is the orthogonal matrix of eigenvectors and  $A_k$  is a diagonal matrix whose elements are proportional to the eigenvalues of  $\Sigma_k$ . Note that in EM the values  $p_k$ ,  $\mu_k$ , and  $\sigma_k$  are updated after each iteration and in the mixture models  $\sigma_k$  is replaced by  $\Sigma_k$  to represent the geometric characteristics of the clusters.

The eigenvalue decomposition can be modeled as various clustering arrangements. Celeux and Govaert (1995), describe in detail fourteen models based on the eigenvalue decomposition. Allowing for variations in the orientation, volume, shape and size of the clusters; six of these models are shown in Table 2.2 (Martinez and Martinez, 2002).

Table 2.2. Parameterization for mixture models.

Model	$\Sigma_k$	Geometric Shape	Volume	Shape	Orientation
1	$\lambda \mathbf{I}$	Spherical	Equal	Equal	NA
2	$\lambda_k \mathbf{I}$	Spherical	Variable	Equal	NA
3	$\lambda \mathbf{D} \mathbf{A} \mathbf{D}^T$	Ellipsoid	Equal	Equal	Equal
4	$\lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k^T$	Ellipsoid	Variable	Variable	Variable
5	$\lambda \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T$	Ellipsoid	Equal	Equal	Variable
6	$\lambda_k \mathbf{D}_k \mathbf{A} \mathbf{D}_k^T$	Ellipsoid	Variable	Equal	Variable

The eigenvalue decomposition can be modeled as various clustering arrangements, i.e., spheres, ellipsoids and rotations of ellipsoids. Allowing the orientation, volume, shape and size of the clusters define the various models used. Figure 2.8 shows the mixture model using rotated ellipsoids (Model 4) to generate the decision boundary around each class.

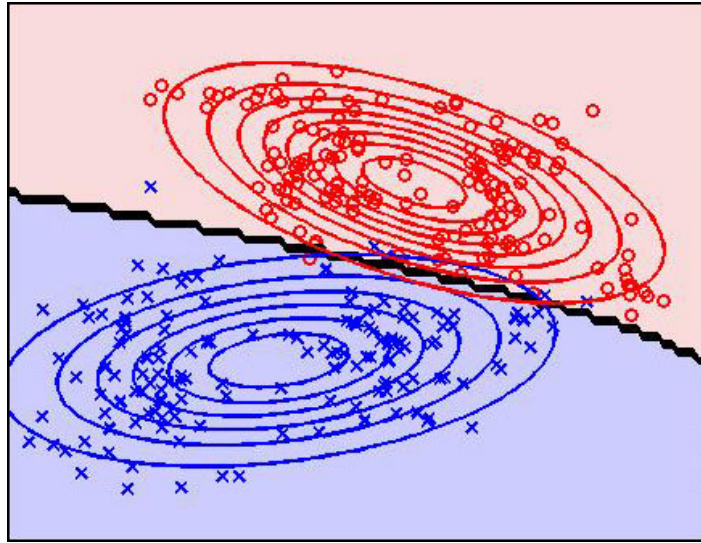


Figure 2.8. Expectation Maximization using mixture models with Decision Boundary.



### 2.6.1.2 Bayes Classifier

The EM algorithm can be used to find a class label for an input sample. Classification uses input samples described by feature vectors  $\mathbf{x}_0 \in \mathbb{R}^n$  to assign the samples to a given class  $\mathbf{C} = C_j = [C_1, C_2, \dots, C_c]$ ,  $j = 1, 2, \dots, c$ . The Bayes classifier extends a general multivariate normal case where the covariance matrix  $\Sigma_j$  for each class is different. For the multi-class classifier each class must have individual conditional probability densities where the densities are modeled as normal distributions. The classes  $C_j$  are defined as normal distributions centered about the mean vector  $\mu_j$ . The mean vector,  $\mu_j$ , and the covariance matrix,  $\Sigma_j$ , are calculated using the EM algorithm. The vector  $\mathbf{x}_0$  is a  $n$ -dimensional vector of the observed data, and  $|\Sigma_i|$  and  $\Sigma_i^{-1}$  are the determinants and inverse covariance matrix of the given class. The posterior probability of class membership can be calculated by Bayes rule if  $C_j$  is defined as the event of belonging to population  $j$ . Using the density function  $g(\mathbf{x}; \mu_k^{(i)}, \sigma_k^{(i)})$  (Tomasi, 2006), the Bayes classifier can be expressed in terms of the prior probabilities,  $P(C_i)$ , and posterior probability of class membership as follows:

$$P(C_j | \mathbf{x}_0) = \frac{P(C_j) \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} \exp\left[-\frac{1}{2}(\mathbf{x}_0 - \mu_j)^T \Sigma_j^{-1} (\mathbf{x}_0 - \mu_j)\right]}{\sum_{i=1}^c P(C_i) \frac{1}{\sqrt{(2\pi)^n |\Sigma_i|}} \exp\left[-\frac{1}{2}(\mathbf{x}_0 - \mu_i)^T \Sigma_i^{-1} (\mathbf{x}_0 - \mu_i)\right]} \quad (2.56)$$

where the a priori probabilities  $P(C_j)$  are the estimates of belonging to a class and under the assumption that  $\Sigma_j = \Sigma$  for  $\forall j$ .

### 2.6.2 $k$ -Nearest Neighbors

$k$ -Nearest Neighbors, Figure 2.9, is a lazy learning approach that compares new samples with all of the samples in the training set, looking for the  $k^{\text{th}}$  nearest (Cover and Hart, 1967; Duda et al., 2001; Bishop, 2006).

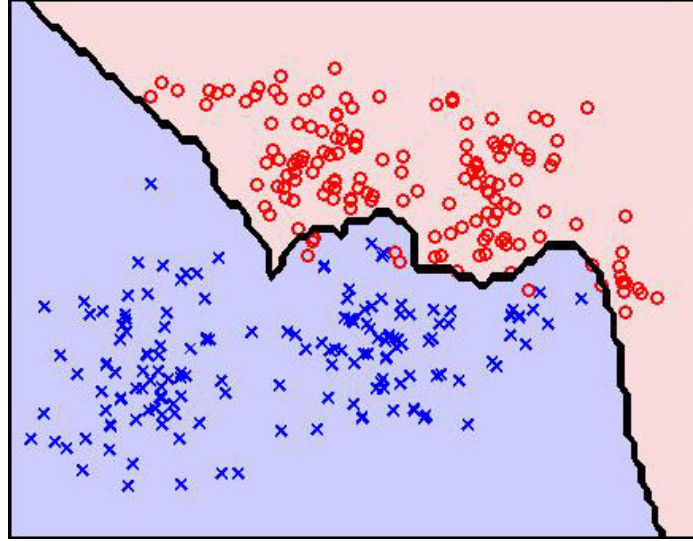


Figure 2.9.  $k$ -NN Decision Boundary.

Let the vectors  $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\ell]^T \in \mathbb{R}^n$  and class labels  $y_i \in \mathbf{C} = [C_1, C_2, \dots, C_c]$ ,  $c \in \mathbb{Z}$ ,  $i = 1, 2, \dots, \ell$ , be a set of training vectors. Given an unknown feature vector  $\mathbf{x}_0$  and a distance measure, the algorithm for the  $k$ -nearest neighbor rule is as follows (Theodoridis and Koutroumbas, 2006):

- Out of the  $\ell$  training vectors  $\mathbf{x}$ , identify the  $k$ -nearest neighbors, irrespective of class label.  $k$  is chosen to be odd for a two-class problem, and in general not to be a multiple of the number of classes.
- Out of the  $k$  samples, identify the number of vectors,  $k_j$ , that belong to class  $\mathbf{C}$ , where  $\sum_j k_j = k$ .

- Assign  $\mathbf{x}_0$  to the class  $\mathbf{C}$  with the maximum number of  $k_j$  of samples.

The distance measures used from the feature  $\mathbf{x}_i$  to each of its  $k$ -nearest neighbors include the Euclidian and Mahalanobis. The advantage of  $k$ -nearest neighbor is the simplicity of the assignment procedure. The disadvantage of the method lies in the necessity to store all samples and compare each with an unknown sample (Fukunaga, 1990).

### 2.6.3 Kernel Fisher's Discriminant (KFD)

The kernel Fisher discriminant is the non-linear extension of the linear FLD (Jaakola and Haussler, 1998; Mika et al., 1999; Scholkopf and Smola, 2002). In the linear case, Fisher's discriminant is computed by maximizing the coefficients of the following equation

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} \quad (2.57)$$

To use the Fisher's discriminant for nonlinearly separable data Mika, et al. (1999) map the input feature space with the use of a kernel. The input space is represented by a training set  $\mathbf{x}_i$  of vectors with a feature dimensionality of  $n$ . The corresponding class labels are represented as  $y_i \in \mathbf{C}$ , where  $\mathbf{C} = [C_{-1}, C_{+1}] = [-1, +1]$ ,  $i = 1, 2, \dots, \ell$  and  $\ell$  is the training set size. The basic idea is to first map the input features from the input space to the kernel space via a kernel function and then perform linear FLD. The aim is to find a direction  $\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i)$  from the feature space to the kernel space given by alpha vectors  $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_\ell]^T$  (Mika et al., 1999). Using the definitions of  $S_B$  and  $S_W$  the Fisher's linear discriminant in the mapped feature space can be defined as

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}^T M \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T N \boldsymbol{\alpha}} \quad (2.58)$$

where  $M = (M_{-1} - M_{+1})(M_{-1} - M_{+1})^T$  is a  $[\ell \times \ell]$  matrix,

$$\begin{aligned} M_{-1} &= \frac{1}{\ell_{-1}} \sum_{\substack{j=1 \\ j \in C_{-1}}}^{\ell_{-1}} K(\mathbf{x}_i, \mathbf{x}_j) \\ M_{+1} &= \frac{1}{\ell_{+1}} \sum_{\substack{j=1 \\ j \in C_{+1}}}^{\ell_{+1}} K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \quad (2.59)$$

and

$$N = \sum_{\substack{j=1 \\ j \in \mathbf{C}}}^{\ell_{\mathbf{C}}} K(\mathbf{x}_i, \mathbf{x}_j) \left( I - \frac{1}{\ell_{\mathbf{C}}} \right) K(\mathbf{x}_i, \mathbf{x}_j)^T \quad (2.60)$$

where  $\mathbf{C} = \{C_{-1}, C_{+1}\} = \{-1, +1\}$ .

In (Mika et al., 1999), *numerical issues* and *regularization* are discussed regarding the calculation of (2.60). This is resolved by simply adding a multiple of the identity matrix to  $N$  defined as:

$$N_{\mu} = N + \mu I. \quad (2.61)$$

The next step is to use the alpha vectors and the kernel matrix to project the  $n$ -1 dimensional input feature space into a one dimensional space as follows:

$$\hat{\mathbf{x}} = K(\mathbf{x}_i, \mathbf{x}_j) \boldsymbol{\alpha}. \quad (2.62)$$

The projection in (2.61) now becomes the space that is to be solved using an optimization solution to maximize the margin of separation between classes as shown in Figure 2.10. In (Mika et al., 1999), the Matlab Optimization Toolbox (2004) is used to solve the optimization problem (Scholkopf and Smola, 2002) with the projected space calculated in (2.62). In this research the one dimensional SMO (Franc and Hlavac, 2007) is used as the optimization solution. This results in the non-negative alpha vectors  $\hat{\alpha}_i = (\hat{\alpha}_1, \dots, \hat{\alpha}_\ell)$  with an upper bound  $\hat{C}$ ,  $\hat{C} \geq \hat{\alpha}_\ell \geq 0$ . The support vectors for the KFD trained model are  $\mathbf{x}_k = \mathbf{x}_i$  and the decision function of the KFD classifier is written as  $\text{sign}(f(\mathbf{x}))$  where  $f(\mathbf{x})$  is defined by:

$$f(\mathbf{x}) = \mathbf{w} \phi(\mathbf{x}) + b = \sum_{i=1}^{\ell} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + b. \quad (2.63)$$

This is equivalent to the maximal margin hyperplane in the input space defined by the kernel (Cristianini and Shawe-Taylor, 2000). The goal of the KFD is to solve for  $\alpha$  and the bias  $b$ . To compute the bias  $b$ , Equation (2.63) is rewritten as follows:

$$\sum_{k=1}^{\ell_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_i) + b = y_i. \quad (2.64)$$

Therefore, the bias is calculated by obtaining the average as (Scholkopf and Smola, 2002):

$$b = \frac{1}{\ell} \sum_{i=1}^{\ell} \left( y_i - \sum_{k=1}^{\ell_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_i) \right) \quad (2.65)$$

In order to reduce the number of false positives and false negatives the optimal bias in (2.65) can be adjusted accordingly. In this case the bias is a threshold (Scholkopf and Smola, 2002).

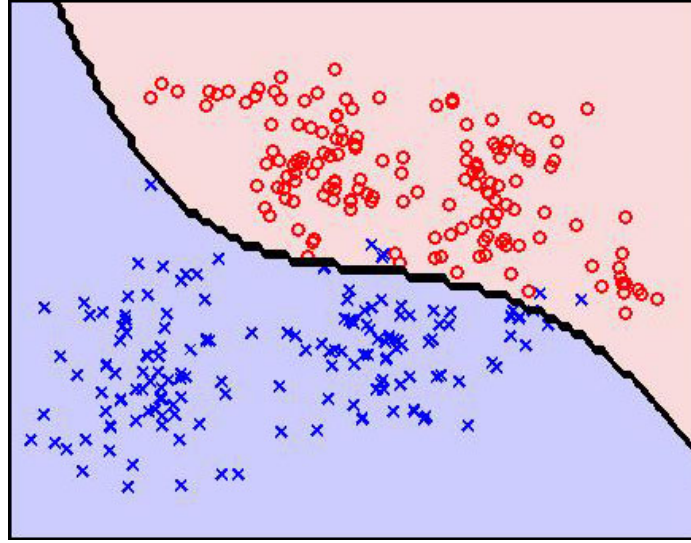


Figure 2.10. KFD Decision Boundary using RBF Kernel.

#### 2.6.4 Parzen Window

Parzen estimation is a refinement of histogramming (Parzen, 1962; Fukunaga, 1990; Duda et al., 2001; Bishop, 2006). The basic idea behind Parzen window estimation is that the knowledge gained by each training sample  $\mathbf{x}$  of the input space,  $\mathbb{R}^n$ , is represented by a function centered at  $\mathbf{x}$  in the feature space. The functions themselves are represented with the use of a distance measure or a *kernel* estimator. The final class estimation is derived by summing the results from the kernel functions of each training sample:

$$p_k = \frac{1}{\ell_k} \sum_{i=1}^{\ell_k} K(\mathbf{x}, \mathbf{x}_i). \quad (2.66)$$

For example, the Parzen window density model is optimized by maximizing the likelihood of the training data with the use of a Gaussian window surrounding each input data point. The Gaussian window can be represented with the use of a kernel function  $K(\mathbf{x}, \mathbf{x}_i)$  as an interpolation function which defines an inner product between the individual training sample. The Radial Basis kernel function uses a window width parameter,  $\sigma$ , which is also known as the spread of the function:

$$p_k = \frac{1}{\ell_k} \sum_{i=1}^{\ell_k} \frac{1}{\sqrt{(2\pi)^{\ell_k} \sigma^{\ell_k}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right). \quad (2.67)$$

This results in a sum of small multivariate Gaussian probability distributions centered at each training sample  $\mathbf{x}$ , an example is shown in Figure 2.11. As the density of the training samples and their respective Gaussian distributions increase the estimation of the probabilities approach the true probability density function (PDF) of the training samples. The estimation for classification for a data cluster is then based on a threshold set for the combined posterior probability from all samples. The classification decision assigns the samples to the class with maximal posterior probability according to the inequality:

$$\frac{1}{\ell_k} \sum_{i=1}^{\ell_k} \frac{1}{\sqrt{(2\pi)^{\ell_k} \sigma^{\ell_k}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right) > \frac{1}{\ell_j} \sum_{i=1}^{\ell_j} \frac{1}{\sqrt{(2\pi)^{\ell_j} \sigma^{\ell_j}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right), \forall k \neq j \quad (2.68)$$

This method requires a reasonably large training data set and is computationally inexpensive during training but is computationally expensive for testing. During testing the kernel function must be computed for each of the training samples making a comparison between the new sample  $\mathbf{x}_0$  and all of the existing training samples  $\mathbf{x}$ . Several kernel approaches have been proposed in literature (Fukunaga, 1990; Wand and Jones, 1995). The kernels were originally presented by Parzen (1962).

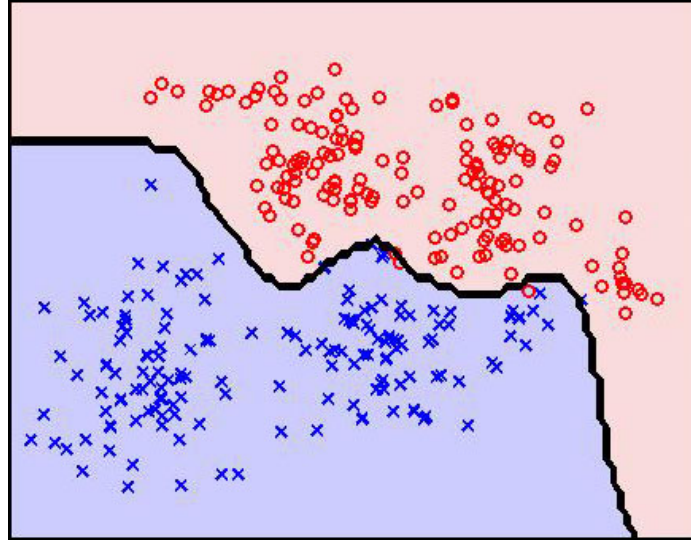


Figure 2.11. Parzen Density Estimator with RBF window with Decision Boundary.

### 2.6.5 Probabilistic Neural Networks (PNN)

The classification frame work of the probabilistic neural network is shown in Figure 2.12 (Specht, 1998; 1990). There are a few decisions that have to be made regarding training of the neural network. First, the number of training samples and number of classes are selected for the pattern layer; this defines the structure of the network. For example, the set of input training samples is represented as  $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\ell]^T \in \mathbb{R}^n$  and a class label  $y_i \in \mathbf{C} = [C_1, C_2, \dots, C_c]$ ,  $i = 1, 2, \dots, \ell$ . This will result in  $c$  groups with each group in the pattern layer containing  $\ell$  neurons. Second, for the summation layer the smoothing parameter,  $\sigma$ , in the nonlinear operation  $f(\mathbf{z}_i)$  of the neural network must be determined. As a general guideline the value of the smoothing parameter,  $\sigma$ , should chosen as a function of the dimension of the problem,  $n$ , and the number of training samples,  $\ell$  (Specht, 1990). The structure of the probabilistic neural network classifier has three layers as shown in Figure 2.12, pattern layer, summation layer and the decision layer. The pattern layer forms a dot product of the input features,  $\mathbf{x}$ , with the weight vectors,  $\mathbf{w}_i$ ,



resulting in  $\mathbf{z}_i = \mathbf{x} \bullet \mathbf{w}_i$ . A nonlinear operation  $f(\mathbf{z}_i)$  on  $\mathbf{z}_i$  is preformed prior to outputting the activation to the summation level.

$$f(\mathbf{z}_i) = \exp\left(\frac{\mathbf{z}_i - 1}{\sigma^2}\right) \quad (2.69)$$

The summation layer sums the inputs from the pattern layer that corresponds to the class from which the training patterns were selected. The output layer returns the summation values for each of the  $c$  classes, a two-class example is shown in Figure 2.13. Each output values  $P_1, \dots, P_c$  is the posterior probability that the sample belongs to that particular class,

where  $\sum_{j=1}^c P_j = 1$ .

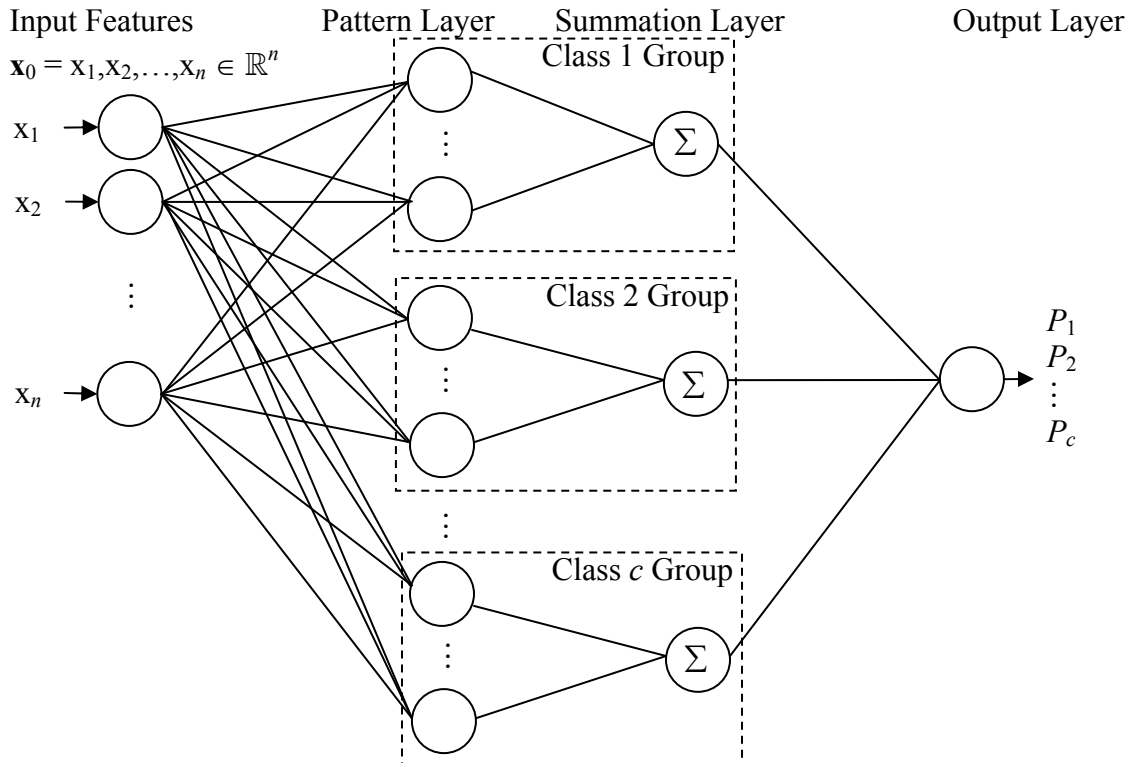


Figure 2.12. Probabilistic Neural Network Classification Structure.

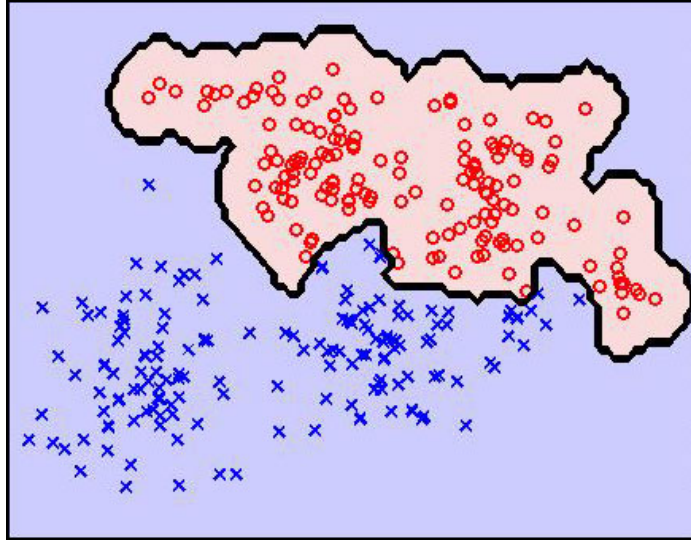


Figure 2.13. PNN with Decision Boundary.

### 2.6.6 Support Vector Machine (SVM)

SVM performs pattern recognition for two-class problems by determining the separating hyperplane that maximizes the distance between the closest points of each class in the training set (Scholkopf et al., 1998; 1999; 2002; Burgers, 1998; Vapnik, 1998; Platt, 2000; Hsu et al., 2006). These closest points are called support vectors. In finding the hyperplane, the SVM performs a nonlinear separation in the input space by using a nonlinear transformation  $\phi(\mathbf{x}_i)$  that maps the data points  $\mathbf{x}_i$  of the input space,  $\mathbb{R}^n$ , into a potential higher dimensional space, called kernel space  $\mathbb{R}^\ell$  ( $\ell > n$ ). The mapping  $\phi(\mathbf{x}_i)$  is represented in the SVM classifier by a kernel function  $K(\mathbf{x}_i, \mathbf{x}_j)$  that defines an inner product in  $\mathbb{R}^\ell$ .

The optimal hyperplane is the one with the maximal distance (in space  $\mathbb{R}^p$ ) to the closest points  $\phi(\mathbf{x}_i)$  of the training data, an example is shown in Figure 2.14. Determining the hyperplane requires maximizing the following function with respect to  $\alpha$

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (2.70)$$

under the constraints  $\sum_j \alpha_j y_j, i = 1, \dots, \ell$ . The non-negative Lagrangian multipliers are  $\alpha_k = (\alpha_1, \dots, \alpha_{\ell_s})$  with an upper bound  $\hat{C}, \hat{C} \geq \alpha_{\ell_s} \geq 0$ . The Lagrangian multipliers are also known as the alpha vectors.

With the given support vectors  $\mathbf{x}_k$  and class labels  $y_k$ , the decision function of the SVM classifier can be written as  $\text{sign}(f(\mathbf{x}))$  where  $f(\mathbf{x})$  is defined by:

$$f(\mathbf{x}) = \mathbf{w} \phi(\mathbf{x}) + b = \sum_{k=1}^{\ell_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}) + b \quad (2.71)$$

This is equivalent to the maximal margin hyperplane in the input space defined by the kernel (Cristianini and Shawe-Taylor, 2000). The goal of the SVM is to solve for  $\boldsymbol{\alpha}$ , the bias  $b$  and the support vectors  $\mathbf{x}_k$ . To compute the bias  $b$ , Equation (2.71) is rewritten as follows:

$$\sum_{k=1}^{\ell_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_i) + b = y_i \quad (2.72)$$

Therefore, the bias is calculated by obtaining the average as (Scholkopf and Smola, 2002):

$$b = \frac{1}{\ell} \sum_{i=1}^{\ell} \left( y_i - \sum_{k=1}^{\ell_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_i) \right) \quad (2.73)$$

In order to reduce the number of false positives and false negatives the optimal bias in (2.73) can be adjusted accordingly. In this case the bias is a threshold (Scholkopf and Smola, 2002).

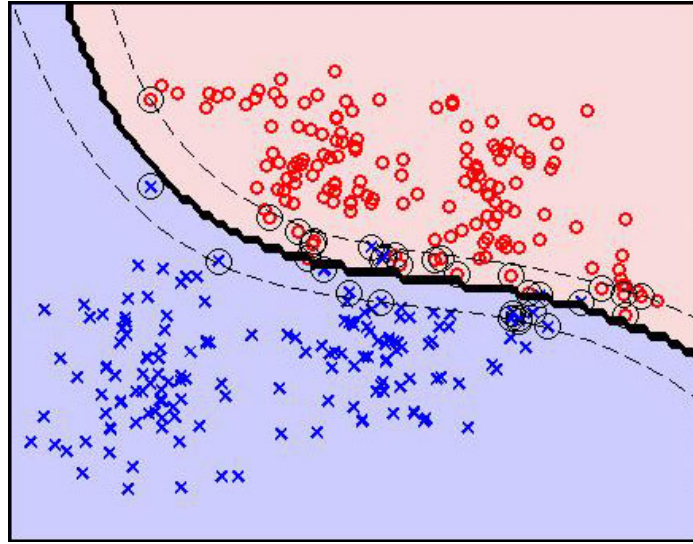


Figure 2.14. SVM with Optimal Hyperplane.

Solving Equation (2.70) is a dual quadratic programming (QP) problem. There are several methods used to solve the quadratic programming problem, including Kernel-Adatron (Friess et al., 1998; Cristianini and Shawe-Taylor, 2004), LOQO (Vanderbei and Shanno, 1999) and sequential minimal optimization (SMO) (Cristianini and Shawe-Taylor, 2000; Franc and Hlavac, 2007; Mak, 2000; Platt, 2000).

Several solutions are available as complete SVM systems to include LIBSVM (Chang and Lin, 2001), Matlab Optimization Toolbox (2007) and SVMlight (Joachims, 1998, 2007). Each of these methods has individual advantages and disadvantages that are beyond the scope of this research.

## 2.7 Multi-Class Classification

In the previous section two-class classifiers were described. However, in many real world problems there are cases where more than two classes exist. The classification methods EM,  $k$ -NN, KFD, Parzen window, probabilistic neural network and SVM can all be modified for a multi-class solution. EM can be used to determine the mean and covariance for each of the classes individually and classified using the Bayes classifier. This however, has the disadvantage of producing inaccurate results when the class distributions are not normally distributed or linearly separable.  $k$ -NN can also be trained to solve a multi-class problem. Selecting  $k$ -nearest neighbors of the input vector  $\mathbf{x}$  a count of the training samples from each of the classes can be used to determine the class label of  $\mathbf{x}$ . The multi-class case performs better with a larger number of input training vectors  $\mathbf{x}$  but has the disadvantage of determining the number of nearest neighbors. Unlike the two-class case where better performance is achieved for large  $k$ , for the multi-class classification this is not always true. The KFD is a two-class classifier by design. It could be converted into a multi-class system in a similar manner as the BSVM (Hsu et al., 2002). In this research only the two-class KFD will be used. For the Parzen window density estimator a multi-class solution can be achieved. As with the two-class case, this method is easily trained but computationally expensive. The expense is in terms of its processing time and memory allocation when the number of samples is large. For a multi-class solution the larger the number of training samples per class the better the performance is achieved. In the multi-class case of the SVM two methods are used in which the margins of separation are determined in the kernel space, BSVM (Hsu and Lin, 2002) and BSVM 2.0 (Hsu et al., 2002). BSVM 2.0 solves the multi-class classification problem for the solution of large classification and regression problems. It includes three methods

- Multi-class classification by solving a single optimization problem using a bound-constrained formulation.

- Multi-class classification using Crammer and Singer's formulation (Crammer and Singer, 2000; Crammer and Singer, 2001).
- Regression using a bound-constrained formulation

While each of these methods can be used for multi-class classification they each have disadvantages when compared to their two-class counterparts. In this research the two-class SVM is used since experimentation has shown that the BSVM 2.0 begins to provide a reduction in classification accuracy when more than 5 classes are used for the clean and stego image data sets.

In several multi-class classification methods two-class classifiers are combined using one-against-one and one-against-all (Fukunaga, 1990; Duda et al., 2001; Tax and Duin, 2002; Lin et al., 2003; Bishop, 2006; Theodoridis and Koutroumbas, 2006). Learning architectures are used to combine several two-class classifiers in order to create a multi-class classifier. In these methods training is done by comparing one class against each of the other classes or by training one class against the remaining classes. This produces several classifiers in which a winner take all approach is used. The winner take all assigns the class label based on a majority vote wins. In this section the following multi-class approaches are presented: one-against-one and one-against-all methods.

### **2.7.1 One-Against-One**

In one-against-one each class is trained against each of the others. The goal is to train the multi-class rule based on the majority vote strategy. The majority votes based multi-class classifier assigns the test input vector  $\mathbf{x}_0$  into class  $C = [C_1, C_2, \dots]$  having the majority of the votes. This is a fairly reliable method assuming that the feature space is separable from one class to the other. Problems arise when a large number of classes are being trained; the resulting system becomes computationally expensive as the number of classifiers increases factorially. The one-against-one approach constructs  $k(k-1)/2$

classifiers from two different classes for each one of the training data sets. This is for training data from the  $i^{\text{th}}$  and the  $j^{\text{th}}$  classes which has  $k$  classes. As an example consider a case with 10 classes,  $k = 10$ . This will require 45 classifiers to be trained. In most classification systems a voting strategy is used. In binary classification the voting strategy votes are cast for all data points  $\mathbf{x}$  where the majority number of votes for a class wins, “Max Wins”. This may lead to a situation where two classes have the same number of votes. One approach to resolving this conflict is to select the class with the smallest index (Hsu et al., 2002).

### **2.7.2 One-Against-All**

Several articles have been written on one-against-all training methods (Liu and Zheng, 2005). The one-against-all method trains the multi-class problem as a series of  $C_i$  two-class subtasks that can be trained by any two-class classifier. If there is  $k > 2$ -class exemplars,  $k$  2-class classifiers will be constructed which separate one class from all other classes. To get  $k$ -classifiers it is common to construct a set of binary classifiers each trained to separate an individual class from the remaining classifiers. One disadvantage of this method is with a significant number of classifiers a large number of two-class classifiers will need to be compared. When grouping all of the classes together the classification may become more difficult as separating the one from all of the rest may not lead to a separation between the classes, and lead to poor classification performance.

## **2.8 Classifier Fusion**

To improve the classification accuracy for the multi-class classification, combining classifiers, classifier fusion, may prove useful on the overall performance of the classification system. The main focus of recent research in classifier fusion has been on establishing the relationship between the diversity of the classifiers and their resulting accuracy/performance. The paradigms of the different models differ on the assumptions

about classifier dependencies, type of classifier outputs, aggregation strategy either global or local, aggregation procedure such as a function, a neural network or an algorithm, etc. (Kittler et al., 1998; Duin and Tax, 2000; Ruta and Gabrys, 2000; Duin, 2002, Kittler, 2002). Three methods of combining classifiers are described which included boosting, Bayes networks and probabilistic neural network combiners.

### 2.8.1 Boosting

Boosting is a powerful technique for combining an ensemble of base classifiers to produce a form of committee whose performance can be significantly increased over any of the single classifiers. The most widely used form of boosting is AdaBoost, developed by Freund and Schapire (1995). Boosting provides good results even if the base classifiers, are weak learners, and have a performance that is only slightly better than random (Freund and Schapire, 1999).

The primary difference between boosting and bagging is that the base classifiers are trained in sequence, and each base classifier is trained using a weighted form of the data set in which the weighting coefficient associated with each data point depends on the performance of the previous classifiers. In particular, points that are misclassified by one of the base classifiers are given greater weight when used to train the next classifier in the next sequence. Once all the classifiers have been trained, their predictions are then combined through a weighted majority voting scheme. AdaBoost calls a given weak or base learning algorithm repeatedly in a series of rounds,  $y_i = 1, \dots, \ell$ . The precise form of the AdaBoost algorithm is given below:

#### AdaBoost Algorithm (Bishop, 2006, pp. 658)

1. The data weighting coefficients  $\{w_i\}$  are initialized as  $w_i^{(1)} = 1/\ell$  for  $i = 1, \dots, \ell$ .
2. For  $k = 1, \dots, K$ :



(a) Fit a classifier  $M_k(\mathbf{x})$  to the training data by minimizing the weighted error function

$$J_k = \sum_{i=1}^{\ell} w_i^{(k)} I(M_k(\mathbf{x}_i) \neq y_i) \quad (2.71)$$

where  $I(M_k(\mathbf{x}_i) \neq y_i)$  is the indicator function and equals 1 when  $M_k(\mathbf{x}_i) \neq y_i$  and 0 otherwise.

(b) Evaluate the quantities

$$\varepsilon_k = \frac{\sum_{i=1}^{\ell} w_i^{(k)} I(M_k(\mathbf{x}_i) \neq y_i)}{\sum_{i=1}^{\ell} w_i^{(k)}} \quad (2.72)$$

and then use these to evaluate

$$\alpha_k = \ln \left\{ \frac{1 - \varepsilon_k}{\varepsilon_k} \right\} \quad (2.73)$$

(c) Update the data weighting coefficients

$$w_i^{(k+1)} = w_i^{(k)} e^{\alpha_k I(M_k(\mathbf{x}_i) \neq y_i)} \quad (2.74)$$

3. Making a prediction using the final trained model for an input image sample  $\mathbf{x}_0$  is given by

$$f(\mathbf{x}_0) = \sum_{k=1}^K \alpha_k M_k(\mathbf{x}_0) \quad (2.75)$$

The first base classifier  $M_1(\mathbf{x})$  is trained using weighting coefficients  $w_i^{(1)}$  that are all equal, which corresponds to the usual procedure for training a single classifier. In Step 2(c), subsequent iterations in the weighting coefficients  $w_i^{(k)}$  are increased for data points that are misclassified and decreased for data points that are correctly classified. Successive classifiers are forced to place greater emphasis on points that have been misclassified by previous classifiers, and data points that continue to be misclassified by successive classifiers receive even greater weight. The quantities  $\varepsilon_k$  represents the

weighted measures of the error weights of each of the base classifiers on the data set. Therefore, in Step 2(b) the weighing coefficients  $\alpha_k$  give greater weights to the more accurate classifiers when computing the overall output given by Step 3 (Bishop, 2006, pp. 658).

### 2.8.2 Bayes Network for Model Averaging

Bayes model averaging merges together several multi-class classifiers by combining the probabilistic density estimation of each classifier's classification accuracy as a mixture of Gaussians (Hoeting et al., 1999; Murphy, 2001). Murphy's (2001) Bayes Net Toolbox (BNT) for Matlab was used in the analysis to facilitate the computations in the model averaging. The probabilistic density estimation specifies the local *conditional probability distributions* (CPD) for a classification model,  $M_k$ , where  $k$  is one of the  $K$  classifiers, and  $M$  is the set of all classifiers. The CPD of each model  $M_k$  is  $p(M_k|T)$ . This represents for each class, the probability of what a classification model will classify a target instance  $T$  as. In this research the implementation uses confusion matrices which represent the correct and incorrect classification for each multi-class classifier providing the probabilistic density estimation for each classifier.

The fusion process uses the classifications from the classification models ( $\mathbf{M}$ ), in conjunction with Bayes Rule, to compute the posterior probability for each target classification  $T = c$ :

$$p(T = c | \mathbf{M}) = \eta \prod_{k=1}^K p(M_k | T = c) p(T = c) \quad (2.76)$$

where  $\eta$  is a normalizing constant.

The final classification is then the target classification,  $T = c$ , with the highest probability. The prior probability of  $p(T)$  is calculated from the number of targets.

### 2.8.3 Probabilistic Neural Network (PNN) Fusion

The fusion method in this work is an extension from the two-class fusion investigated by Leap et al., (2007) to a multi-class system fusion. In this method the outputs of individual classification systems are treated as input features to train a probabilistic neural network (Specht, 1990) for fusion. The key is to use the class labels from each of the systems as posterior probability estimates and employing them as features in the neural network. It should be noted that one of the posterior probabilities from the input classifier should be removed. For example, if  $K$  three-class classifiers are used, then each of the classification models,  $M_k$ , will contribute two inputs for training the PNN. The fusion method treats the posterior probabilities from individual detection systems as features to the neural network and outputs an overall posterior probability of a sample as being in a given class. This fusion does not impose any independence assumptions on the input systems.

## 2.9 Summary

This chapter presented the key elements necessary to solve the steganalysis multi-class classification system for identifying JPEG steganography embedding methods. JPEG image representation was described by introducing the discrete cosine transform and the JPEG image format and its compression steps. The feature generation methods described in this chapter were a wavelet based method and a discrete cosine transform method. In feature preprocessing outlier removal, data normalization and data standardization were presented. For feature extraction, PCA and Kernel PCA were described. The feature ranking/selection method presented in this chapter were the Bhattacharyya distance, Fisher's linear discriminant ratio, signal to noise ratio, kernel Fisher's discriminant recursive feature elimination and the zero-norm feature ranking. In classification both

two-class and multi-class classification method used in this research were described; the six methods used are EM,  $k$ -NN, KFD, Parzen window, probabilistic neural networks and SVM. A section was devoted to improving classification performance with classifier fusion covering boosting, Bayes networks and probabilistic neural networks.

In this chapter several methods have been described that are essential in making a comparison with the proposed overall detection method described in Chapter 3. Some of the methods described in this chapter are modified to accommodate the needs of the proposed method. In other cases, the methods in this chapter are incorporated into the system.

### III. Methodology

This chapter presents a multi-class fusion system for classification of steganographic methods. This detection method classifies JPEG images based on generated image features whereby previously unseen images are associated with exactly one element of the label set, i.e., clean or type of stego image. The stego image consists of one of seven targeted embedding methods, F5 (Westfeld, 2001; 2003), JP Hide (Latham, 1999), JSteg (Upham, 1993), Model-base (Sallee, 2003; 2006), Model-based Version 1.2 (Sallee, 2008a), OutGuess (Provos, 2004), Steganos (2008), StegHide (Hetzl, 2003) and UTSA (Agaian et al., 2006).

Figure 3.1 shows the classification system developed in this research. The image set consists of clean and stego images that have data embedded using one of nine methods. Features are generated from each image and each feature set is assigned a class label identifying the embedding method used. The features are used in three components of the multi-class system. See Figure 3.1. The first component is *Multi-class Detection for EM/k-NN/Parzen/PNN*. The existing feature improvement methods and classifiers are used to create four multi-class detection systems that each return a class label assigned to the input sample (Rodriguez and Peterson, 2008a). The second component is *Multi-class Detection for KFD/SVM*. It contains a new feature ranking method along with a new multi-class tree to generate a multi-class classification label with the combination of two-class classifiers. The third component, *Commercial Detection Systems*, has two commercial steganalysis tools that return class labels for a variety of stego methods. The assigned class labels for 8 multi-class systems are fused shown as *Classifier Fusion* in the figure and a final class label is assigned.

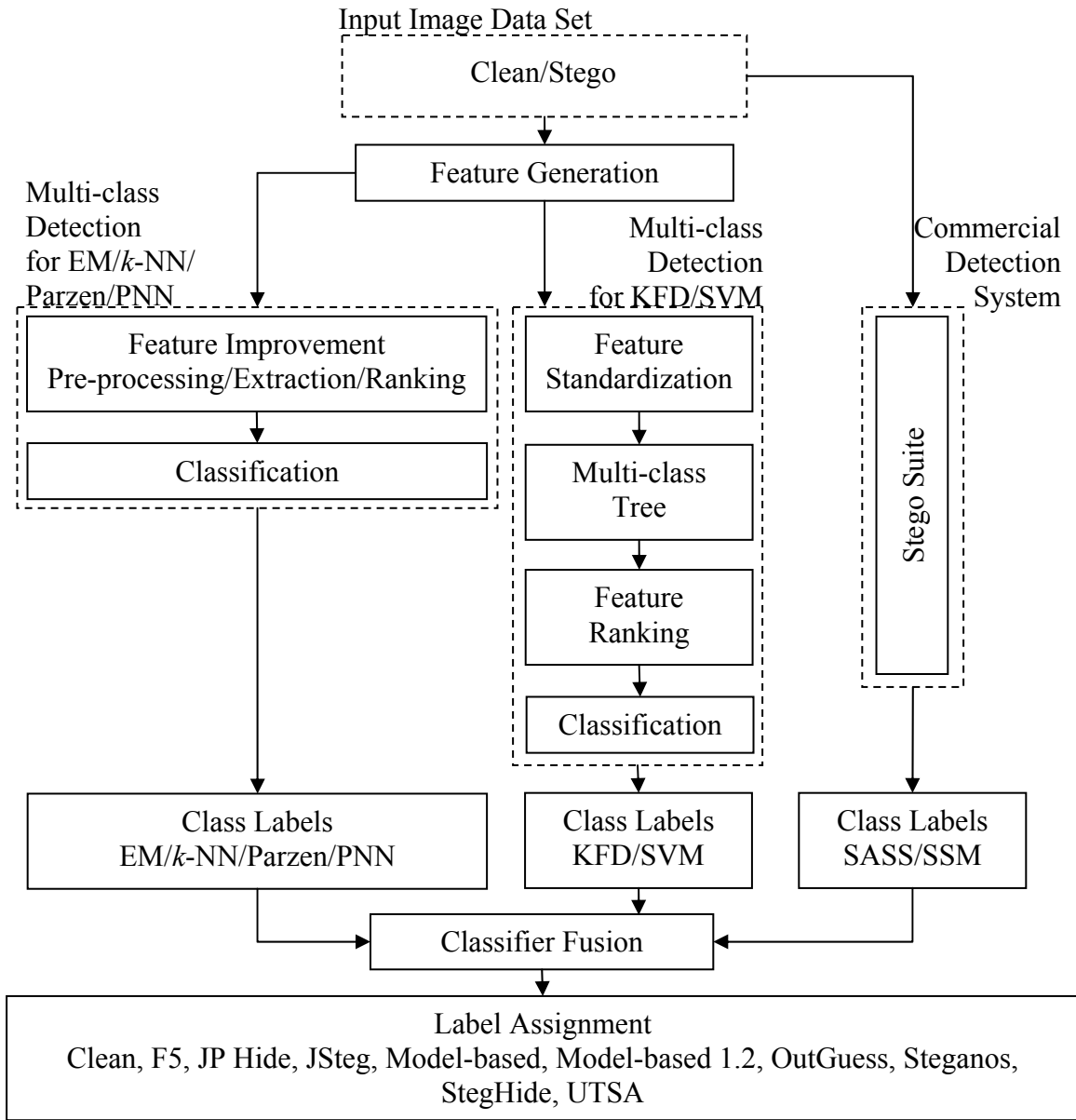


Figure 3.1. Detection System.

This chapter presents four improvements to steganalysis pattern recognition. The first is the creation of new features generated from the frequency bands and directions of the Discrete Cosine Transform (DCT) coefficients of JPEG images. The second improvement is a new feature ranking method. From the original input feature set, it selects a subset of features specifically designed for the kernel Fisher's discriminant

(KFD) and the support vector machines (SVM). The third improvement is a multi-class classification tree designed for the KFD and SVM classifiers. The final contribution of this steganalysis classification system is the fusion of multi-class classifiers. These improvements are designed to increase the identification of embedding methods used to create stego images.

### 3.1 Feature Generation

This section details the novel DCT feature generation method. Figure 3.2 illustrates the main components of the novel feature generation method.

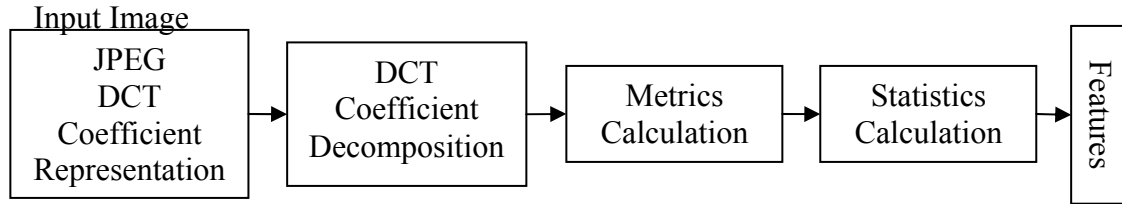


Figure 3.2. General Feature Generation System.

The first component builds on details of the DCT coefficient representation which is used in a decomposition. Two metrics are calculated on each  $8 \times 8$  block of the decomposed coefficients in a JPEG image. The first metric is a difference calculation that compares DCT coefficients with neighboring coefficients. The second metric is a least square linear regression metric that uses DCT coefficients, shifted coefficients and neighboring coefficients to calculate weights used in the regression model. Statistics (e.g., mean, variance, etc.) are calculated over the DCT coefficients, neighboring coefficients, shifted coefficients and the metrics. The last three set of statistics are then subtracted from the statistics of the DCT coefficients creating a set of 180 features used to identify clean and stego images.

### 3.1.1 DCT Representation

The standard DCT used in JPEG compression has two properties, i.e., the directional and frequency distributions of  $8 \times 8$  blocks within an image (Rao and Yip, 1990). In JPEG compression on a two dimensional (2-D) signal, the zig-zag scan shown in Figure 3.3a is used to take advantage of the frequency distributions of the DCT shown in Figure 3.3b (Brown and Shepherd, 1995, pp. 224). The DCT decomposition divides the coefficients into low, medium and high frequencies. Figure 3.3c shows the breakdown of the vertical, diagonal and horizontal directions of the coefficients. In this research both the frequencies and directions of the DCT are investigated to generate features. Figure 3.3d shows an  $8 \times 8$  image with a horizontal edge between black and white pixels. The corresponding 2-D DCT of Figure 3.3d is shown in Figure 3.3g which has coefficients that are prominent along the first column. In Figure 3.3e an image is shown with a diagonal edge between black and white pixels with a corresponding 2-D DCT shown in Figure 3.3h which has coefficients located along the diagonal. In Figure 3.3f an image is shown with a vertical edge between black and white pixels with a corresponding 2-D DCT shown in Figure 3.3i which has coefficients located along the first row.



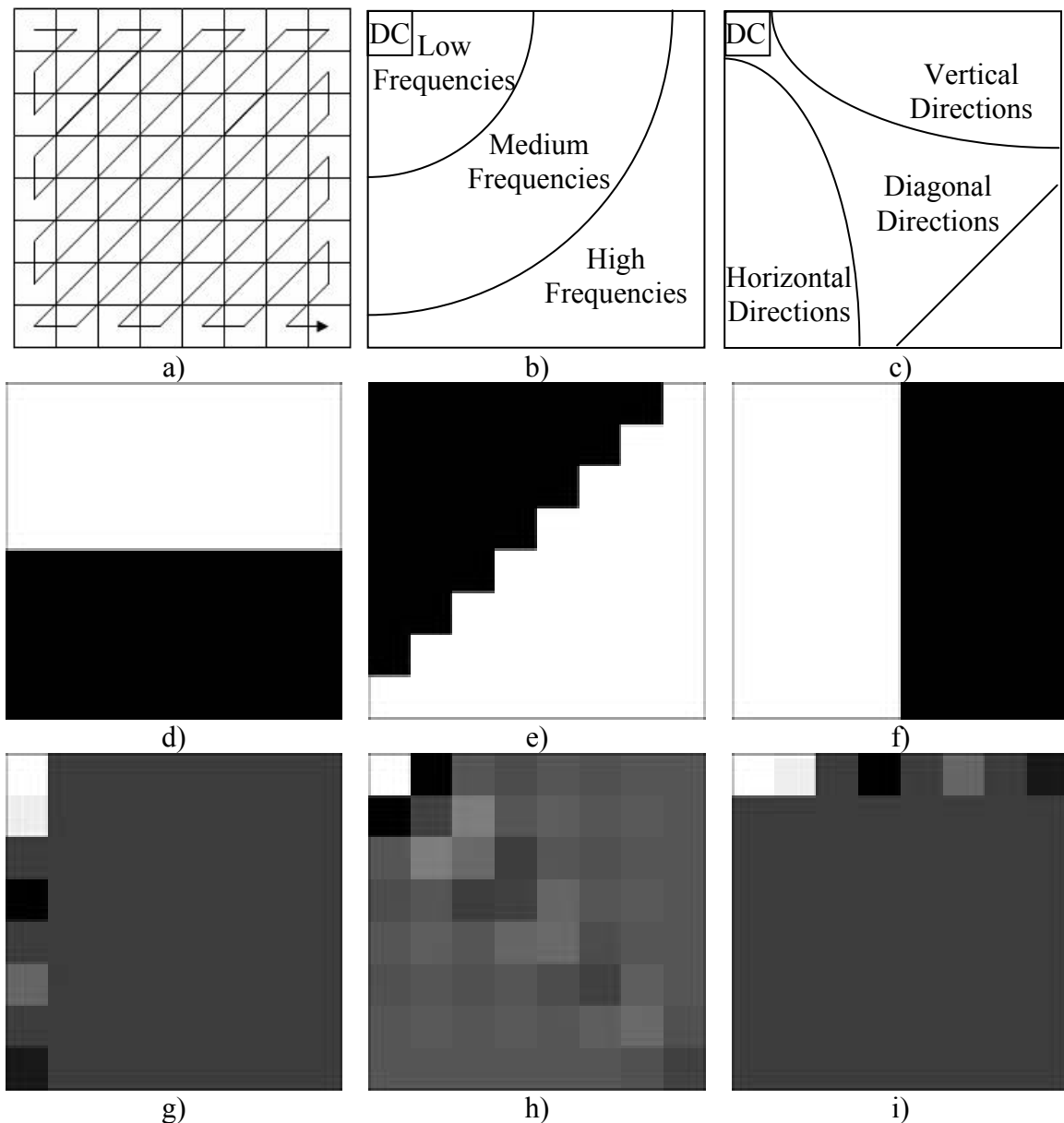


Figure 3.3. DCT decomposition a) zig-zag scan pattern b) low, medium and high frequency distributions c) vertical, diagonal and horizontal directions d) 8×8 image with a horizontal edge between pixels e) 8×8 image with a diagonal edge between pixels f) 8× 8 image with a vertical edge between pixels g) 2-D DCT representation of horizontal image h) 2-D DCT representation of diagonal image i) 2-D DCT representation of vertical image.

### 3.1.2 Arrangement of Decomposed DCT Coefficients

The calculation of the features requires rearranging the DCT coefficients in three different ways. The first, DCT decomposition separates coefficients into areas within the  $8 \times 8$  DCT block, three frequency bands as well as three directions. This results into 9 areas that the coefficients are decomposed into where 6 different areas are used. The second is a set of coefficients generated by shifting the  $8 \times 8$  pixel blocks in the spatial domain and recalculating the quantized DCT coefficients. The DCT decomposition feature method is then used over these shifted blocks. Three different shifting operations are used, shifting the  $8 \times 8$  block to the right by four pixels (*block shift right*), down by four pixels (*block shift down*), and diagonal by four pixels (*block shift diagonal*). The last arrangement of the DCT coefficients are sets of neighboring coefficients within an  $8 \times 8$  DCT block for a DCT coefficient of interest.

#### 3.1.2.1 Frequency and Directional Coefficient Vectors

The  $8 \times 8$  coefficient values are represented as  $d^b(u, v)$  where  $u = v = 1, \dots, 8$ ,  $b = 1, \dots, B$ , where  $B$  is the number of  $8 \times 8$  blocks within a color layer of an image. The zig-zag pattern shown in Figure 3.3a is used to translate the  $8 \times 8$  matrix into a vector. The vector is represented as  $\hat{d}_{\hat{k}}^b$ ,  $\hat{k} = 1, \dots, 64$ , and the locations of  $\hat{k}$  are shown in Figure 3.4.

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

a)

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

b)

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

c)

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

d)

Figure 3.4. DCT Coefficient Locations and Separations a) DCT Coefficient Location after Zig-Zag Scan b) Coefficient Locations of Vertical, Diagonal and Horizontal directions c) Coefficient Locations of Low, Mid and High Frequencies d) 8×8 block Coefficient Separation of both frequencies and directions.

The coefficient vector indices from the zig-zag method are shown in Figure 3.4a. The DC coefficient is at location 1 and locations 2 through 64 are the AC coefficients. Figure 3.4b shows the separations of vertical (red), diagonal (green) and horizontal (blue) DCT decompositions. The remaining coefficients correspond to the high frequencies and are normally zero due to the quantization compression of the JPEG method. For a typical compression of JPEG images the high frequencies correspond to the black cells in Figure 3.4c. The DCT decompositions of low (white), medium (gray) and high (black) frequencies coefficients are shown in Figure 3.4c. In this research the coefficients will be decomposed as shown in Figure 3.4d. As shown in Figure 3.4d the  $8 \times 8$  block is divided into eight DCT decompositions represented by both the frequency distributions and directions.

The coefficients are arranged as follows:

- The combination of vertical and low frequencies ( $VL$ ) is shown as red in Figure 3.4d. The vector  $\hat{d}_{VL}^b$  contains the DCT coefficients of block  $b$  after the zig-zag scan at locations 2, 6, 7 and 8 such that the vector  $D_{VL} = (\hat{d}_{VL}^b | b = 1, \dots, B)$ .
- The diagonal and low frequencies ( $DL$ ) are shown as green in Figure 3.4d. The vector  $\hat{d}_{DL}^b$  contains the DCT coefficients of block  $b$  at locations 5 and 13 such that  $D_{DL} = (\hat{d}_{DL}^b | b = 1, \dots, B)$ .
- The horizontal and low frequencies ( $HL$ ) are shown as blue in Figure 3.4 d. The vector  $\hat{d}_{HL}^b$  contains the DCT coefficients of block  $b$  at locations 3, 4, 9 and 10 such that  $D_{HL} = (\hat{d}_{HL}^b | b = 1, \dots, B)$ .
- The vertical and mid frequencies ( $VM$ ) are shown as dark red in Figure 3.4d. The vector  $\hat{d}_{VM}^b$  contains the DCT coefficients of block  $b$  after the zig-zag scan at locations 14, 15, 16, 17, 27, 28, 30, and 31 such that  $D_{VM} = (\hat{d}_{VM}^b | b = 1, \dots, B)$ .

- The diagonal and mid frequencies ( $DM$ ) are shown as dark green in Figure 3.4d. The vector  $\hat{d}_{DM}^b$  contains the DCT coefficients of block  $b$  after the zig-zag scan at locations 18, 19, 24, 25, 26, 32, 33, 39, 40 and 41 such that  $D_{DM} = (\hat{d}_{DM}^b | b = 1, \dots, B)$ .
- The horizontal and mid frequencies ( $HM$ ) are shown as dark blue in Figure 3.4d. The vector  $\hat{d}_{HM}^b$  contains the DCT coefficients of block  $b$  after the zig-zag scan at locations 11, 12, 20, 21, 22, 23, 34 and 35 such that  $D_{HM} = (\hat{d}_{HM}^b | b = 1, \dots, B)$ .

The remaining coefficients of Figure 3.4d shown in black are not analyzed with the decomposition since during JPEG compression they are often zero valued and typically not used to hide a stego message (Fridrich, 2004).

### 3.1.2.2 Block Shifted Coefficient Vectors

In this subsection the individual  $8 \times 8$  blocks of an input JPEG image are shifted in the spatial domain and recompressed using the JPEG compression technique. Three shifting techniques are used, shifting to the right, down and right and down each by four pixels. The coefficients from the shifted blocks are placed in vectors as in subsection 3.1.2.1.

The first set of shifted coefficients focuses on shifting the pixel values to the right by four pixels in the spatial domain as shown in Figure 3.5.

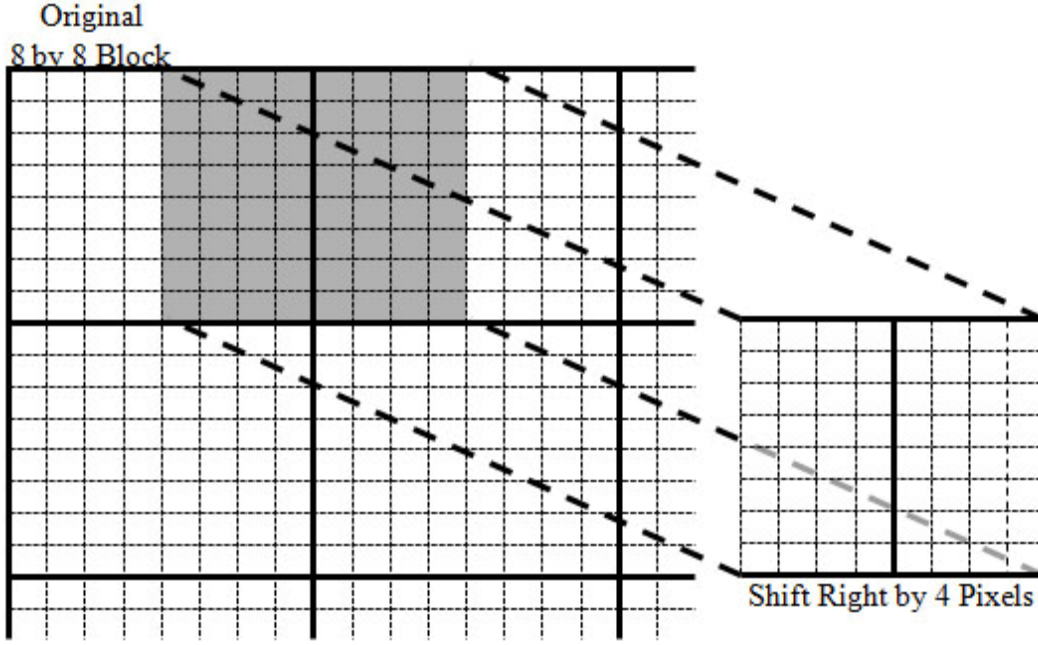


Figure 3.5. Original block and right shifted pixel locations

The original block containing the spatial domain pixels is transformed using the JPEG compression properties, e.g., the same quantization table used in compression. The last column of blocks has no neighboring blocks so the final four columns of pixels in the image are duplicated to ensure  $B$  shifted blocks exist.

Using the same vector representation of the DCT coefficients as in subsection 3.1.2.1 for the right shifted blocks results in the following vector representations:

- The combination of vertical and low frequencies ( $VL$ ) results in a vector  $\hat{s}_{VL,Right}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 2, 6, 7 and 8 such that the vector  $S_{VL,Right} = (\hat{s}_{VL,Right}^b \mid b = 1, \dots, B)$ .
- The diagonal and low frequencies ( $DL$ ) result in a vector  $\hat{s}_{DL,Right}^b$  containing the DCT coefficients of block  $b$  at locations 5 and 13 such that  $S_{DL,Right} = (\hat{s}_{DL,Right}^b \mid b = 1, \dots, B)$ .

- The horizontal and low frequencies (*HL*) result in a vector  $\hat{S}_{HL,Right}^b$  containing the DCT coefficients of block  $b$  at locations 3, 4, 9 and 10 such that  $S_{HL,Right} = (\hat{S}_{HL,Right}^b \mid b = 1, \dots, B)$ .
- The vertical and mid frequencies (*VM*) result in a vector  $\hat{S}_{VM,Right}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 14, 15, 16, 17, 27, 28, 30, and 31 such that  $S_{VM,Right} = (\hat{S}_{VM,Right}^b \mid b = 1, \dots, B)$ .
- The diagonal and mid frequencies (*DM*) result in a vector  $\hat{S}_{DM,Right}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 18, 19, 24, 25, 26, 32, 33, 39, 40 and 41 such that  $S_{DM,Right} = (\hat{S}_{DM,Right}^b \mid b = 1, \dots, B)$ .
- The horizontal and mid frequencies (*HM*) result in a vector  $\hat{S}_{HM,Right}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 11, 12, 20, 21, 22, 23, 34 and 35 such that  $S_{HM,Right} = (\hat{S}_{HM,Right}^b \mid b = 1, \dots, B)$ .

The second set of shifted coefficients focuses on shifting the pixel values down by four pixels in the spatial domain as shown in Figure 3.6.

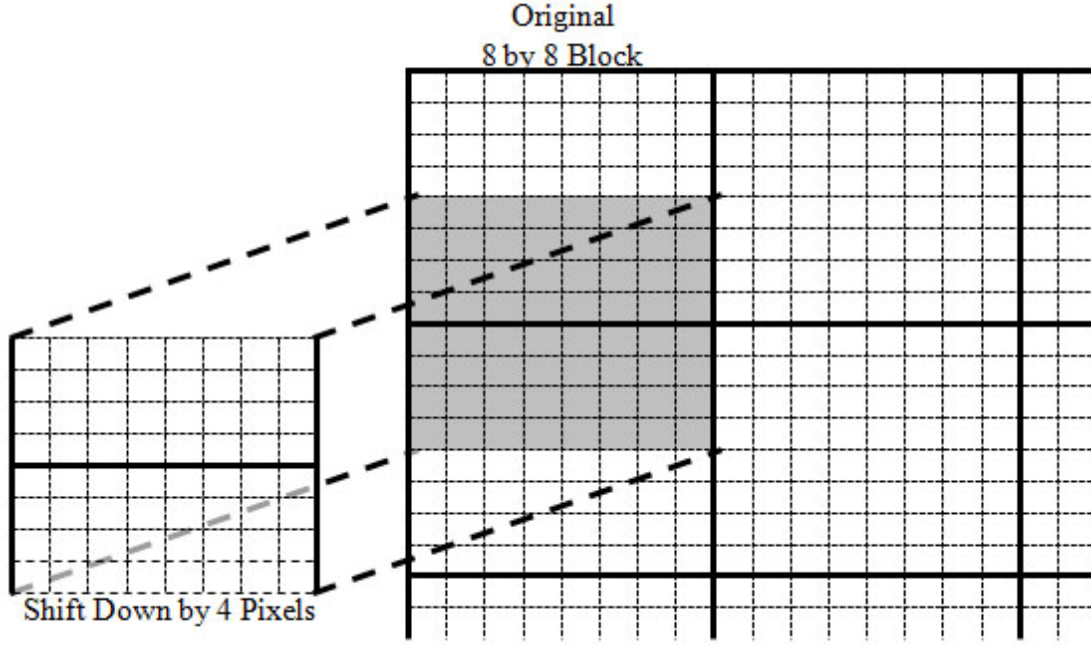


Figure 3.6. Original block and down shifted pixel locations

For this method the last row of blocks has no neighboring blocks so the final four rows of pixels in the image are duplicated to ensure  $B$  shifted blocks exist.

The down shifted blocks results in the following vector representations:

- The combination of vertical and low frequencies ( $VL$ ) results in a vector  $\hat{s}_{VL,Down}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 2, 6, 7 and 8 such that the vector  $S_{VL,Down} = (\hat{s}_{VL,Down}^b | b = 1, \dots, B)$ .
- The diagonal and low frequencies ( $DL$ ) result in a vector  $\hat{s}_{DL,Down}^b$  containing the DCT coefficients of block  $b$  at locations 5 and 13 such that  $S_{DL,Down} = (\hat{s}_{DL,Down}^b | b = 1, \dots, B)$ .



- The horizontal and low frequencies (*HL*) result in a vector  $\hat{s}_{HL,Down}^b$  containing the DCT coefficients of block  $b$  at locations 3, 4, 9 and 10 such that  $S_{HL,Down} = (\hat{s}_{HL,Down}^b | b = 1, \dots, B)$ .
- The vertical and mid frequencies (*VM*) result in a vector  $\hat{s}_{VM,Down}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 14, 15, 16, 17, 27, 28, 30, and 31 such that  $S_{VM,Down} = (\hat{s}_{VM,Down}^b | b = 1, \dots, B)$ .
- The diagonal and mid frequencies (*DM*) result in a vector  $\hat{s}_{DM,Down}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 18, 19, 24, 25, 26, 32, 33, 39, 40 and 41 such that  $S_{DM,Down} = (\hat{s}_{DM,Down}^b | b = 1, \dots, B)$ .
- The horizontal and mid frequencies (*HM*) result in a vector  $\hat{s}_{HM,Down}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 11, 12, 20, 21, 22, 23, 34 and 35 such that  $S_{HM,Down} = (\hat{s}_{HM,Down}^b | b = 1, \dots, B)$ .

The third set of shifted coefficients focuses on shifting the pixel values to the right by four pixels and down by four pixels in the spatial domain as shown in Figure 3.7.

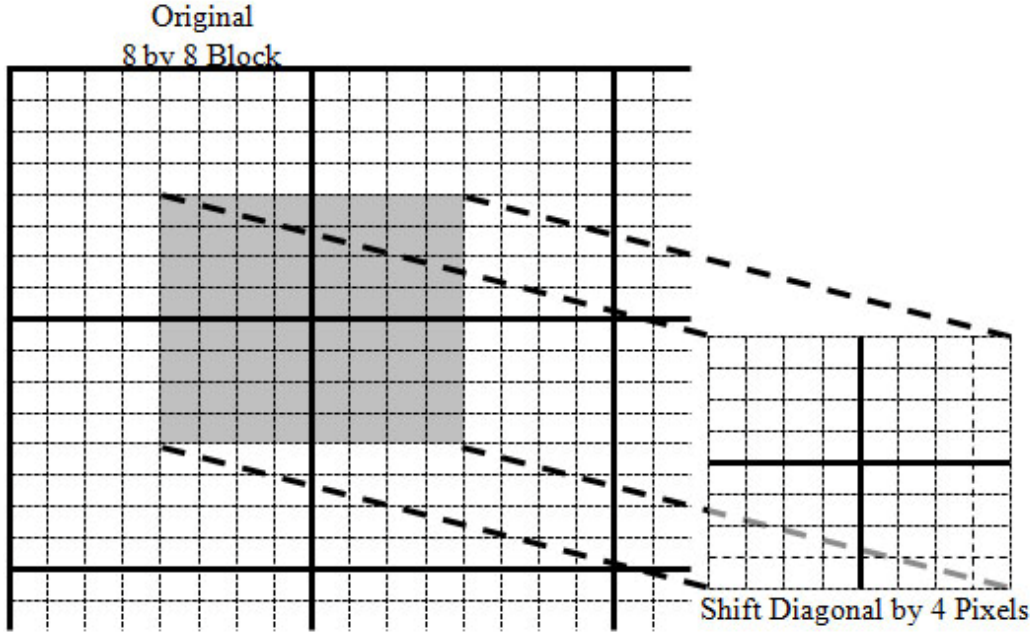


Figure 3.7. Original block and diagonal shifted pixel locations.

Shifting the blocks diagonally, the last row and column of blocks have no neighboring blocks so the final four rows and the final four columns of pixels in the image are duplicated to ensure  $B$  diagonally shifted blocks exist.

The diagonally shifted blocks results in the following vector representations:

- The combination of vertical and low frequencies ( $VL$ ) results in a vector  $\hat{s}_{VL,Diag}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 2, 6, 7 and 8 such that the vector  $S_{VL,Diag} = (\hat{s}_{VL,Diag}^b \mid b = 1, \dots, B)$ .
- The diagonal and low frequencies ( $DL$ ) result in a vector  $\hat{s}_{DL,Diag}^b$  containing the DCT coefficients of block  $b$  at locations 5 and 13 such that  $S_{DL,Diag} = (\hat{s}_{DL,Diag}^b \mid b = 1, \dots, B)$ .

- The horizontal and low frequencies (*HL*) result in a vector  $\hat{s}_{HL,Diag}^b$  containing the DCT coefficients of block  $b$  at locations 3, 4, 9 and 10 such that  $S_{HL,Diag} = (\hat{s}_{HL,Diag}^b \mid b = 1, \dots, B)$ .
- The vertical and mid frequencies (*VM*) result in a vector  $\hat{s}_{VM,Diag}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 14, 15, 16, 17, 27, 28, 30, and 31 such that  $S_{VM,Diag} = (\hat{s}_{VM,Diag}^b \mid b = 1, \dots, B)$ .
- The diagonal and mid frequencies (*DM*) result in a vector  $\hat{s}_{DM,Diag}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 18, 19, 24, 25, 26, 32, 33, 39, 40 and 41 such that  $S_{DM,Diag} = (\hat{s}_{DM,Diag}^b \mid b = 1, \dots, B)$ .
- The horizontal and mid frequencies (*HM*) result in a vector  $\hat{s}_{HM,Diag}^b$  containing the DCT coefficients of block  $b$  after the zig-zag scan at locations 11, 12, 20, 21, 22, 23, 34 and 35 such that  $S_{HM,Diag} = (\hat{s}_{HM,Diag}^b \mid b = 1, \dots, B)$ .

### 3.1.2.3 Neighboring Coefficient Matrices

Each DCT coefficient has a corresponding vector of neighboring coefficients. For a coefficient of interest in an  $8 \times 8$  block, the neighboring coefficients are defined as its surrounding coefficients. The six vectors representing the directional and frequency coefficients described in subsection 3.1.2.1 each have a matrix of neighboring coefficients.

The vectors and matrices of neighboring coefficients are as follows:

- For the vertical directions and low frequencies vector  $\hat{d}_{VL}^b$  when  $VL = 2$  the coefficient at location  $\hat{d}_2^b$  has corresponding neighboring coefficients 1, 6, 7, 8 and

14 represented by the vector  $\hat{n}_{2,k_{VL}}^b = [1 \ 6 \ 7 \ 8 \ 14]$ ,  $k_{VL} = 1, \dots, 5$ . The matrix of neighboring coefficients for  $\hat{d}_{VL}^b$  is as follows:

$$n_{VL,k_{VL}}^b = \begin{bmatrix} n_{2,k_{VL}}^b \\ n_{6,k_{VL}}^b \\ n_{7,k_{VL}}^b \\ n_{8,k_{VL}}^b \end{bmatrix} = \begin{bmatrix} 1 & 6 & 7 & 8 & 14 \\ 2 & 7 & 14 & 15 & 17 \\ 6 & 15 & 16 & 17 & 27 \\ 5 & 14 & 17 & 18 & 26 \end{bmatrix} \text{ such that } N_{VL} = (n_{VL,k_{VL}}^b \mid b=1, \dots, B)$$

- The matrix of neighboring coefficients for the horizontal directions and low frequencies vector  $\hat{d}_{DL}^b$  are represented as follows:

$$n_{DL,k_{DL}}^b = \begin{bmatrix} n_{5,k_{DL}}^b \\ n_{13,k_{DL}}^b \end{bmatrix} = \begin{bmatrix} 1 & 8 & 9 & 13 & 25 \\ 5 & 18 & 19 & 25 & 40 \end{bmatrix} \text{ such that } N_{DL} = (n_{DL,k_{DL}}^b \mid b=1, \dots, B)$$

- The matrix of neighboring coefficients for the horizontal directions and low frequencies vector  $\hat{d}_{HL}^b$  are represented as follows:

$$n_{HL,k_{HL}}^b = \begin{bmatrix} n_{3,k_{HL}}^b \\ n_{4,k_{HL}}^b \\ n_{9,k_{HL}}^b \\ n_{10,k_{HL}}^b \end{bmatrix} = \begin{bmatrix} 1 & 4 & 9 & 10 & 12 \\ 3 & 10 & 11 & 12 & 20 \\ 5 & 12 & 19 & 20 & 24 \\ 4 & 11 & 20 & 21 & 23 \end{bmatrix} \text{ such that } N_{HL} = (n_{HL,k_{HL}}^b \mid b=1, \dots, B)$$

- The matrix of neighboring coefficients for the vertical directions and the medium frequencies vector  $\hat{d}_{VM}^b$  are represented as follows:

$$n_{VM,k_{VM}}^b = \begin{bmatrix} n_{14,k_{VM}}^b \\ n_{15,k_{VM}}^b \\ n_{16,k_{VM}}^b \\ n_{17,k_{VM}}^b \\ n_{27,k_{VM}}^b \\ n_{31,k_{VM}}^b \end{bmatrix} = \begin{bmatrix} 5 & 8 & 17 & 18 & 26 & 27 & 31 \\ 6 & 7 & 16 & 17 & 27 & 28 & 30 \\ 7 & 15 & 27 & 28 & 29 & 30 & 43 \\ 8 & 14 & 26 & 27 & 30 & 31 & 42 \\ 14 & 17 & 30 & 31 & 42 & 43 & 44 \\ 18 & 26 & 41 & 42 & 44 & 45 & 54 \end{bmatrix} \text{ such that } N_{VM} = (n_{VM,k_{VM}}^b \mid b=1, \dots, B)$$

- The matrix of neighboring coefficients for the horizontal directions and medium frequencies vector  $\hat{d}_{DM}^b$  are represented as follows:

$$n_{DM,k_{DM}}^b = \begin{bmatrix} n_{18,k_{DM}}^b \\ n_{19,k_{DM}}^b \\ n_{24,k_{DM}}^b \\ n_{25,k_{DM}}^b \\ n_{26,k_{DM}}^b \\ n_{32,k_{DM}}^b \\ n_{33,k_{DM}}^b \\ n_{39,k_{DM}}^b \\ n_{40,k_{DM}}^b \\ n_{41,k_{DM}}^b \end{bmatrix} = \begin{bmatrix} 2 & 8 & 25 & 26 & 32 & 40 & 41 & 46 \\ 3 & 9 & 24 & 25 & 33 & 39 & 40 & 47 \\ 4 & 12 & 33 & 34 & 39 & 47 & 48 & 51 \\ 5 & 13 & 32 & 33 & 40 & 46 & 47 & 52 \\ 6 & 14 & 31 & 32 & 41 & 45 & 46 & 53 \\ 8 & 18 & 41 & 40 & 46 & 52 & 53 & 56 \\ 9 & 19 & 39 & 40 & 47 & 51 & 52 & 57 \\ 12 & 24 & 47 & 48 & 51 & 57 & 58 & 59 \\ 13 & 25 & 46 & 47 & 52 & 56 & 57 & 60 \\ 14 & 26 & 45 & 46 & 53 & 55 & 56 & 61 \end{bmatrix}$$

such that  $N_{DM} = (n_{DM,k_{DM}}^b \mid b=1, \dots, B)$

- The matrix of neighboring coefficients for the horizontal directions and medium frequencies vector  $\hat{d}_{HM}^b$  are represented as follows:

$$n_{HM,k_{HM}}^b = \begin{bmatrix} n_{11,k_{HM}}^b \\ n_{12,k_{HM}}^b \\ n_{20,k_{HM}}^b \\ n_{21,k_{HM}}^b \\ n_{23,k_{HM}}^b \\ n_{34,k_{HM}}^b \end{bmatrix} = \begin{bmatrix} 4 & 10 & 20 & 21 & 23 & 22 & 35 \\ 5 & 9 & 19 & 20 & 23 & 24 & 34 \\ 9 & 12 & 23 & 24 & 34 & 35 & 38 \\ 10 & 11 & 22 & 23 & 35 & 36 & 37 \\ 12 & 20 & 34 & 35 & 37 & 38 & 49 \\ 19 & 24 & 38 & 39 & 48 & 49 & 50 \end{bmatrix}$$

such that  $N_{HM} = (n_{HM,k_{HM}}^b \mid b=1, \dots, B)$

The arrangement of the coefficients into the vectors  $D$ ,  $S_{Right}$ ,  $S_{Down}$ ,  $S_{Diag}$  along with the matrices  $N$  will be used to calculate the metrics in the next sect and used to calculate statistics necessary for generating the features.

### 3.1.3 Metrics Calculation

In this subsection two metrics used to compare coefficients are described. The first is a difference calculation that compares DCT coefficients with neighboring coefficients. The second metric is a least square linear regression metric that uses DCT coefficients, shifted coefficients and neighboring coefficients to calculate weights used in the regression model.

#### 3.1.3.1 Mean Difference between DCT Coefficients and Neighboring Coefficients

The mean difference metric between the DCT coefficients in subsection 3.1.2.1 and the neighboring coefficients from subsection 3.1.2.3 are described in this subsection. Vectors are generated for the three directions and the frequencies.

The mean differences are calculated as follows:

- Vertical direction and low frequencies

$$\bar{d}_{VL}^b = \frac{1}{5} \sum_{k_{VL}=1}^5 \left( \hat{d}_{VL}^b - n_{VL,k_{VL}}^b \right) \text{ such that } \bar{D}_{VL} = \left( \bar{d}_{VL}^b \mid b=1, \dots, B \right) \quad (3.1)$$

- Diagonal direction and low frequencies

$$\bar{d}_{DL}^b = \frac{1}{5} \sum_{k_{DL}=1}^5 \left( \hat{d}_{DL}^b - n_{DL,k_{DL}}^b \right) \text{ such that } \bar{D}_{DL} = \left( \bar{d}_{DL}^b \mid b=1, \dots, B \right) \quad (3.2)$$

- Horizontal direction and low frequencies

$$\bar{d}_{HL}^b = \frac{1}{5} \sum_{k_{HL}=1}^5 \left( \hat{d}_{HL}^b - n_{HL,k_{HL}}^b \right) \text{ such that } \bar{D}_{HL} = \left( \bar{d}_{HL}^b \mid b=1, \dots, B \right) \quad (3.3)$$

- Vertical direction and medium frequencies

$$\bar{d}_{VM}^b = \frac{1}{7} \sum_{k_{VM}=1}^7 \left( \hat{d}_{VM}^b - n_{VM,k_{VM}}^b \right) \text{ such that } \bar{D}_{VM} = \left( \bar{d}_{VM}^b \mid b=1, \dots, B \right) \quad (3.4)$$

- Diagonal direction and medium frequencies

$$\bar{d}_{DM}^b = \frac{1}{8} \sum_{k_{DM}=1}^8 \left( \hat{d}_{DM}^b - n_{DM,k_{DM}}^b \right) \text{ such that } \bar{D}_{DM} = \left( \bar{d}_{DM}^b \mid b=1, \dots, B \right) \quad (3.5)$$

- Horizontal direction and medium frequencies

$$\bar{d}_{HM}^b = \frac{1}{7} \sum_{k_{HM}=1}^7 \left( \hat{d}_{HM}^b - n_{HM,k_{HM}}^b \right) \text{ such that } \bar{D}_{HM} = \left( \bar{d}_{HM}^b \mid b=1, \dots, B \right) \quad (3.6)$$

### 3.1.3.2 Least Squares Linear Regression

Regression analysis is used to assess the relationship between dependent variables and one or more independent variables. The independent variables are known as predictor variables. To avoid confusion in this chapter, the independent variables are the neighboring and shifted coefficients while the predictor variables are the DCT coefficients. The coefficients in this section are used to calculate the least square linear regression metric (Legendre, 1805, Gauss, 1809, pp. 205-224; Davis, 1809/1857, pp. 249-273; Dillon and Goldstein, 1984, pp. 209-250; Draper and Smith, 1998; Neter et al., 1996). The idea is to predict the mean value of the dependent variables (in this case DCT coefficients) on the basis of the fixed neighboring coefficients and shifted coefficients. The regression model with multiple variables in  $N$  is written as

$$\hat{D} = \beta_0 + \beta_1 N_1 + \beta_2 N_2 + \dots \quad (3.7)$$

where  $\beta_0$  is referred to as the intercept coefficient and the remaining  $\beta$ 's are the slope coefficients which gives the change in  $D$  with respect to  $N$ . The  $\beta$ 's are calculated as

$$\beta = \left( N^T N \right)^{-1} N^T D \quad (3.8)$$

The intercept coefficient  $\beta_0$  in Equation (3.7) cannot be calculated using Equation (3.8) for  $D$  and  $N$ . To solve this problem a column vector of 1's is added to the front of the matrix  $N$ . The column of 1's allows the regression model to contain the term  $\beta_0$ . If the  $\beta_0$  term is omitted from the regression model, the response of the model is zero when all of the predictor variables are zero. In a straight line regression model the line has a zero intercept when  $\beta_0 = 0$  resulting in a poor model (Draper and Smith, 1998).

The vectors for the regression metric in this subsection for the three directions and frequencies are calculated as follows:

- Vertical direction and low frequencies

$$\begin{aligned}\hat{N}_{VL} &= \begin{bmatrix} 1's & N_{VL} & S_{VL,Right} & S_{VL,Down} & S_{VL,Diag} \end{bmatrix} \\ \beta_{VL} &= \left( \hat{N}_{VL}^T \hat{N}_{VL} \right)^{-1} \hat{N}_{VL}^T D_{VL} \\ \hat{D}_{VL} &= \hat{N}_{VL} \beta_{VL}\end{aligned}$$

- Diagonal direction and low frequencies

$$\begin{aligned}\hat{N}_{DL} &= \begin{bmatrix} 1's & N_{DL} & S_{DL,Right} & S_{DL,Down} & S_{DL,Diag} \end{bmatrix} \\ \beta_{DL} &= \left( \hat{N}_{DL}^T \hat{N}_{DL} \right)^{-1} \hat{N}_{DL}^T D_{DL} \\ \hat{D}_{DL} &= \hat{N}_{DL} \beta_{DL}\end{aligned}$$

- Horizontal direction and low frequencies

$$\begin{aligned}\hat{N}_{HL} &= \begin{bmatrix} 1's & N_{HL} & S_{HL,Right} & S_{HL,Down} & S_{HL,Diag} \end{bmatrix} \\ \beta_{HL} &= \left( \hat{N}_{HL}^T \hat{N}_{HL} \right)^{-1} \hat{N}_{HL}^T D_{HL} \\ \hat{D}_{HL} &= \hat{N}_{HL} \beta_{HL}\end{aligned}$$

- Vertical direction and medium frequencies

$$\begin{aligned}\hat{N}_{VM} &= \begin{bmatrix} 1's & N_{VM} & S_{VM,Right} & S_{VM,Down} & S_{VM,Diag} \end{bmatrix} \\ \beta_{VM} &= \left( \hat{N}_{VM}^T \hat{N}_{VM} \right)^{-1} \hat{N}_{VM}^T D_{VM} \\ \hat{D}_{VM} &= \hat{N}_{VM} \beta_{VM}\end{aligned}$$



- Diagonal direction and medium frequencies

$$\hat{N}_{DM} = \begin{bmatrix} 1's & N_{DM} & S_{DM,Right} & S_{DM,Down} & S_{DM,Diag} \end{bmatrix}$$

$$\beta_{DM} = \left( \hat{N}_{DM}^T \hat{N}_{DM} \right)^{-1} \hat{N}_{DM}^T D_{DM}$$

$$\hat{D}_{DM} = \hat{N}_{DM} \beta_{DM}$$

- Horizontal direction and medium frequencies

$$\hat{N}_{HM} = \begin{bmatrix} 1's & N_{HM} & S_{HM,Right} & S_{HM,Down} & S_{HM,Diag} \end{bmatrix}$$

$$\beta_{HM} = \left( \hat{N}_{HM}^T \hat{N}_{HM} \right)^{-1} \hat{N}_{HM}^T D_{HM}$$

$$\hat{D}_{HM} = \hat{N}_{HM} \beta_{HM}$$

For each of the coefficients investigated in  $D$  a set of neighboring coefficients were selected based on experimental analysis and an understanding of both JPEG compression and how the embedding methods alter the coefficients. Determining the number of neighboring coefficients can be expanded to sequential selection used in regression, e.g., backward selection, forward selection and stepwise selection (Dillon and Goldstein, 1984).

### 3.1.4 Statistics Calculation

By using the metrics derived from the previous subsection, the statistics are calculated over the vectors in subsection 3.1.2 and 3.1.3 in order to generate the features. Table 3.1 lists five statistics: mean, standard deviation, skewness, kurtosis, and entropy along with their calculation.

Table 3.1. Test statistics for generating features.

Test Statistic	Statistical Function $F(\cdot)$
Mean	$F_{\mu}(D) = \mu(D) = \frac{1}{n} \sum_{i=1}^n D_i$
Standard Deviation	$F_{\sigma}(D) = \sigma(D) = \left( \frac{1}{n} \sum_{i=1}^n (D_i - \mu(D))^2 \right)^{1/2}$
Skewness	$F_{\gamma}(D) = \gamma(D) = \frac{\sum_{i=1}^n (D_i - \mu(D))^3}{\sigma(D)^3}$
Kurtosis	$F_{\kappa}(D) = \kappa(D) = \frac{\sum_{i=1}^n (D_i - \mu(D))^4}{\sigma(D)^4}$
Entropy	$F_E(D) = E(D) = - \sum_{i=1}^n (D_i) \log(D_i)$

### 3.1.5 Features

The new feature generation method produces a total of 180 features for an input image. By taking the differences between the calculated statistics, the number of features in the following is dependent on the DCT decomposition and the selected coefficients as described in subsection 3.1.2 through 3.1.4.  $D$  includes the coefficient vectors in 3.1.2.1,  $\hat{D}$  is the regression model described in 3.1.3.2,  $\bar{D}$  are the mean differences in 3.1.3.1,  $\bar{N}$  contains the average of the neighboring coefficients in 3.1.2.3,  $S_{Right}$ ,  $S_{Down}$  and  $S_{Diag}$  are block shifted coefficient vectors in 3.1.2.2, and the statistical calculation functions  $F(\cdot)$  are described in 3.1.4.

$F(D) - F(\hat{D})$  generates 30 features

$F(D) - F(\bar{D})$  generates 30 features

$F(D) - F(\bar{N})$  generates 30 features

$F(D) - F(S_{Right})$  generates 30 features

$F(D) - F(S_{Down})$  generates 30 features

$F(D) - F(S_{Diag})$  generates 30 features

These are denoted as raw features. The three detection systems which are going to be described in the following sections consider these features as inputs in order to achieve the goal in this research.

### 3.2 Feature Ranking/Selection

The previous section presents a feature generation method that results in 180 features that identify the difference between clean and stego images. Some of these features separate the clean from stego images better than others. In this section a new feature ranking method for two-class kernel Fisher's discriminant and support vector machines classifiers is described that identifies the best features to use for accurate classification (Rodriguez et al., 2008a).

#### 3.2.1 SVM-Kernel Feature Ranking (KFR)

SVM-KFR consists of a three-step feature ranking strategy to choose representative features and remove noisy features for a data set with multiple features. The first step is to remove one feature at a time from the training data set. Specifically remove feature  $m$  from  $\mathbf{x}_i$  denoted as  $\mathbf{x}_i^{(m)}$ , where  $(m)$  indicates the removed feature  $m$ . The second step is to solve Equation (2.70) to identify the support vectors,  $\mathbf{x}_k$ , and the non-negative alpha vectors,  $\hat{C} \geq \alpha_i \geq 0$ . Once the support vectors are identified the kernel matrix is calculated as:

$$K(\mathbf{x}_k^{(m)}, \mathbf{x}_j^{(m)}) = K^{(m)}(\mathbf{x}_k, \mathbf{x}_j). \quad (3.18)$$

The final step multiplies the kernel matrix with the  $m$  feature removed,  $K^{(m)}(\mathbf{x}_k, \mathbf{x}_j)$  by the alpha vectors,  $\alpha^{(m)}$ , and associated class labels  $y$ . By rewriting Equation (2.71) the multiplication results in the following solution:

$$f(\mathbf{x}_j^{(m)}) = \sum_{k=1}^{\ell_s} \alpha_k^{(m)} y_k K^{(m)}(\mathbf{x}_k, \mathbf{x}_j) + b. \quad (3.19)$$

This projection results in approximated class labels without the bias shown in Equation (2.70). In the event a feature with strong class separability is removed an incorrect estimate results. As an example, consider a nonlinearly separable set of 50 samples with 100 features and equal number of classes. Figure 3.8 shows the mixture of classes when a strongly separating feature is removed. The x-axis in Figure 3.8 represents the index,  $j$ , of sample  $\mathbf{x}_j^{(m)}$  and the y-axis represents the predicted class value,  $f(\mathbf{x}_j^{(m)})$ , for each sample after calculating Equation (3.19) where the alpha values are in the range of  $0 \leq \alpha_k^{(m)} \leq 6$ . The range is determined by the upper bound  $\hat{C}$  when solving Equation (2.70).

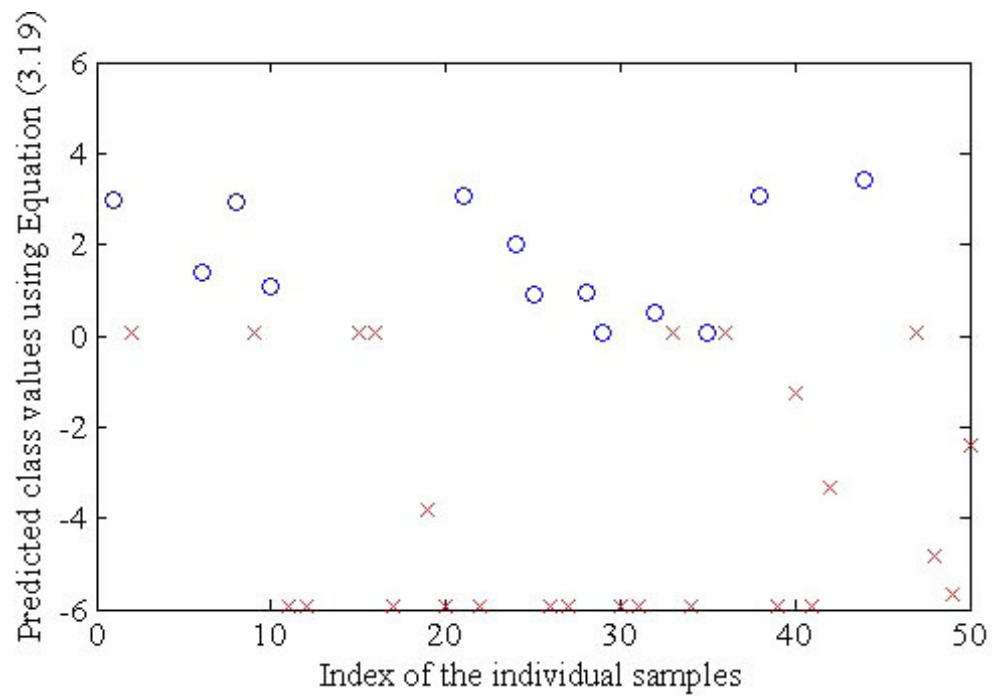


Figure 3.8. One dimensional mapping of Equation (3.19) when the strong ranked feature is removed.

On the other hand if a noise-like feature in class separability is removed the two classes show a separation. Figure 3.9 shows the result of removing a weak ranked feature.

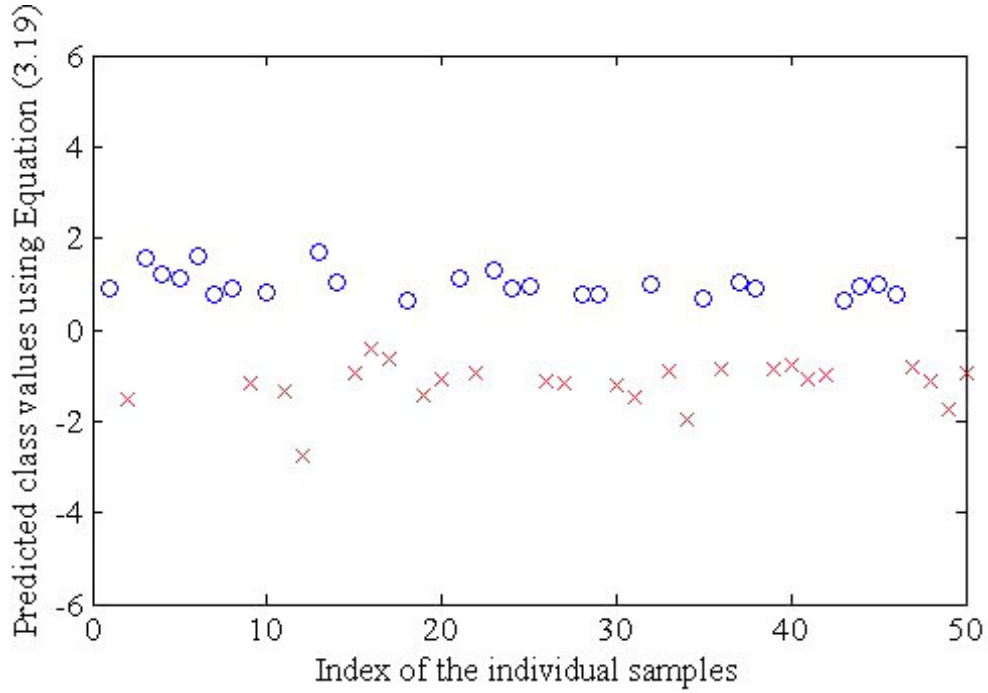


Figure 3.9. One dimensional mapping of Equation (3.19) when the weak ranked feature is removed.

For ranking purposes, the projection of the samples  $f(\mathbf{x}_j^{(m)})$  is summed. The problem arises when positive and negative values are summed resulting in potential cancelation of the results. Because of this, the labels  $y_i$  in Equation (3.19) are excluded from the decision function as

$$f(\mathbf{x}_j^{(m)}) = \sum_{k=1}^{\ell_s} \alpha_k^{(m)} K^{(m)}(\mathbf{x}_k^{(m)}, \mathbf{x}_j^{(m)}) + b. \quad (3.20)$$

The solution for the ranking can be defined as the summation of Equation (3.20) resulting in a ranking value for feature  $m$  as follows:

$$R_m = \sum_{j=1}^{\ell_s} \sum_{k=1}^{\ell_s} \alpha_k^{(m)} K^{(m)}(\mathbf{x}_k^{(m)}, \mathbf{x}_j^{(m)}) + b \quad (3.21)$$

where  $\alpha_k^{(m)}$  contains the weights for the support vectors. It is important to note that only the support vectors are used to calculate  $K^{(m)}$  during the ranking process implying that  $\mathbf{x}_k^{(m)} = \mathbf{x}_j^{(m)}$ . A normalizing factor of  $1/\ell$  can be applied to the ranking values  $R_m$  for the feature ranking criterion but is not required. The algorithm of this method is provided as follows:

1. For each of the  $n$  features perform steps 2, 3 and 4.
2. Remove the current feature  $m$  from the data set  $\mathbf{x}_i$  and train the SVM model, extracting the  $\alpha$ -vectors and the support vectors  $\mathbf{x}_k$ .
3. Calculate  $K^{(m)}$  using the support vectors  $\mathbf{x}_k$  from step 2
4. Assign a ranking value  $R_m$  according to Equation (3.21) and replace the feature  $m$ .
5. After completion of the loop, sort the ranking values  $R_m$  in descending order.
6. Select the  $r$  highest ranked features for training the SVM classification model.

Equation (3.20) estimates the effect of the optimization solution in Equation (2.70) by removing one feature at a time. The summation of the mapping  $\alpha_k^{(m)} K^{(m)}(\mathbf{x}_k^{(m)}, \mathbf{x}_j^{(m)})$  in Equation (3.21) seeks to maximize the distance between classes,  $\mathbf{C} = \{-1, +1\}$ . To explain the mathematical representation of the ranking criterion in Equation (3.21), it is necessary to re-examine  $f(\mathbf{x})$  from Equation (2.71) which denotes the solution for classification determined by the values of the vector  $\alpha$  and the bias  $b$  at a particular stage of the learning. Letting

$$E_j = f(\mathbf{x}_j) - y_j = \left( \sum_{k=1}^{\ell_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_j) + b \right) - y_j \quad (3.22)$$

be the error difference between the function output and the target value (Cristianini and Shawe-Taylor, 2000) on the training data  $\mathbf{x}$ , it is possible to show the relationship between Equations (2.71) and (3.21). For an ideal case the desired value of  $E_i$  would be 0. The goal is to retain the features that approximate the sum as follows

$$\sum_{j=1}^{\ell_s} \left( \sum_{k=1}^{\ell_s} \alpha_k^{(m)} K^{(m)}(\mathbf{x}_k^{(m)}, \mathbf{x}_j^{(m)}) + b \right) \approx \sum_j |y_j| \quad (3.23)$$

Setting the ideal situation of  $E_j$  equal to 0 the following equation is used

$$E_j = \left( \sum_{k=1}^{\ell_s} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_j) + b \right) - y_j = 0 \quad (3.24)$$

where  $j = 1, \dots, \ell$ . Using the absolute values of  $y_k$  and  $y_i$  results in the following equation:

$$\sum_{j=1}^{\ell_s} \left( \sum_{k=1}^{\ell_s} \alpha_k^{(m)} K^{(m)}(\mathbf{x}_k^{(m)}, \mathbf{x}_j^{(m)}) + b \right) = \sum_j |y_j| \quad (3.25)$$

which is similar to Equation (3.23). When a feature is removed, a larger ranking indicates a prediction farther away from the true class,  $y_i$ . The  $R_m$  criterion allows a view of how well the SVM model separates the space in the absence of the removed feature.

Figures 3.10 and 3.11 show the values of the decision function  $f(\mathbf{x}_j^{(m)})$  when the highest and lowest ranked features are removed. The axes of Figure 3.10 and 3.11 represent the index of sample  $\mathbf{x}_j^{(m)}$  on the x-axis and the y-axis represents the value of the sample after calculating Equation (3.21). In Figure 3.10 the top ranked feature is removed showing the



space between 0 and 6. When Equation (3.21) is calculated this results in a large ranking value.

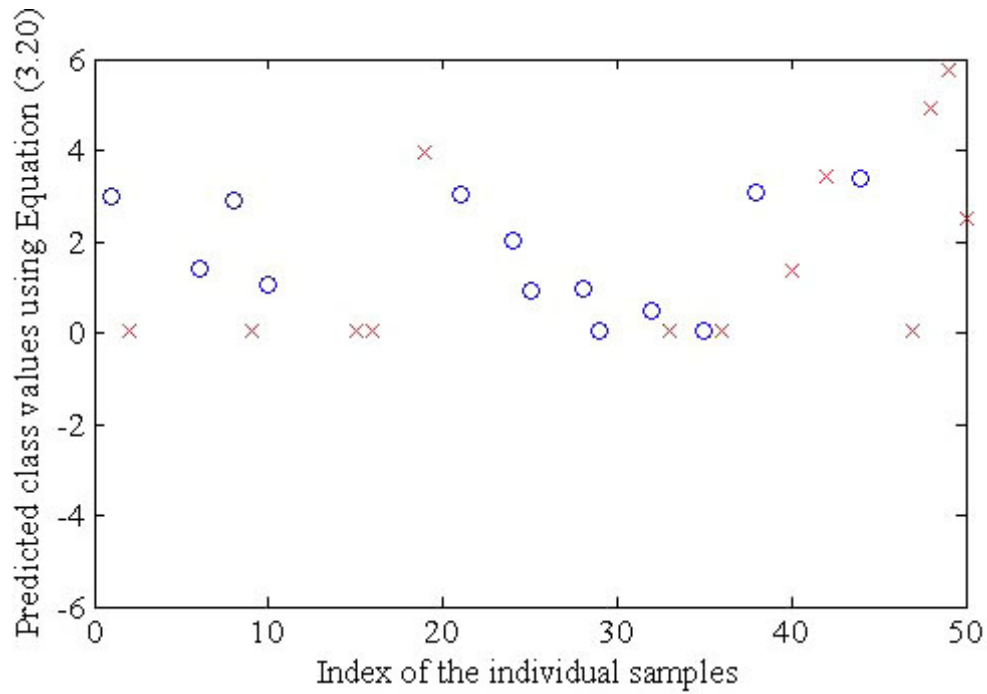


Figure 3.10. One dimensional mapping of Equation (3.20) when the highest ranked feature is removed.

In Figure 3.11 the lowest ranked feature is removed showing the space converging on 1. This will result in a ranking value approximately equal to  $|y_j|$  indicating a low ranking.

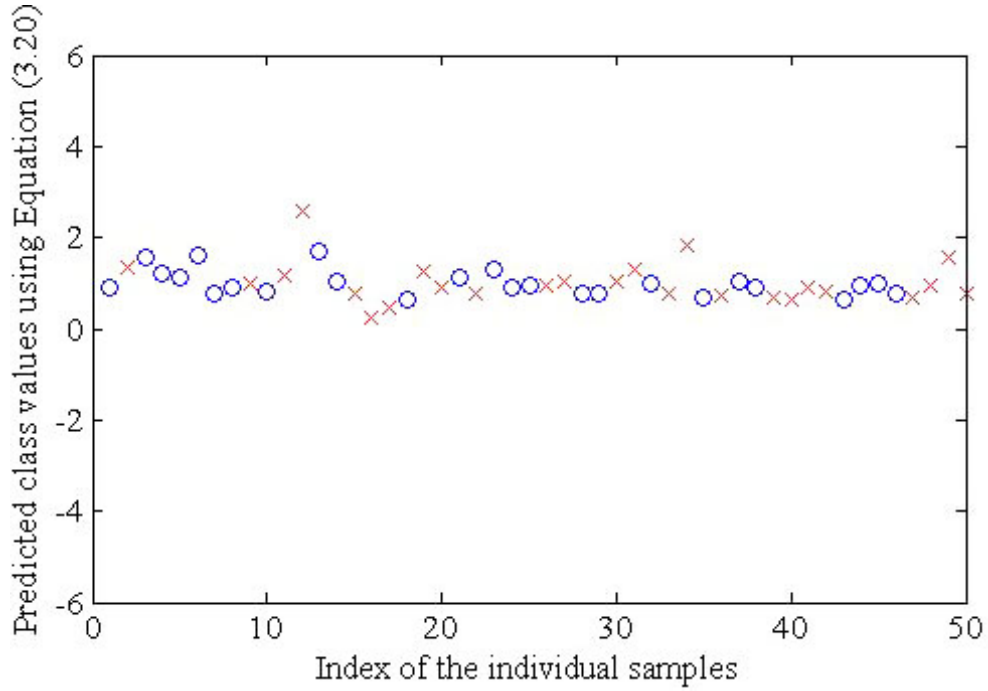


Figure 3.11. One dimensional mapping of Equation (3.20) when the lowest ranked feature is removed.

While the space is not perfectly separated in Figure 3.10 and 3.11, the reader should be aware of the fact that the figures do not show a mapping of  $\alpha_k^{(m)} K^{(m)}(\mathbf{x}_k, \mathbf{x}_j) + b$  with the top ranked features. The two figures are shown to give an insight of the effects a removed feature has on the mapping from the input space to the ranking space using Equation (3.21).

In Figure 3.12 the top 25% of the ranked features are kept. In this simple example Figure 3.12 shows that maintaining the top ranked features the error function in Equation (3.22) can be trained to zero.

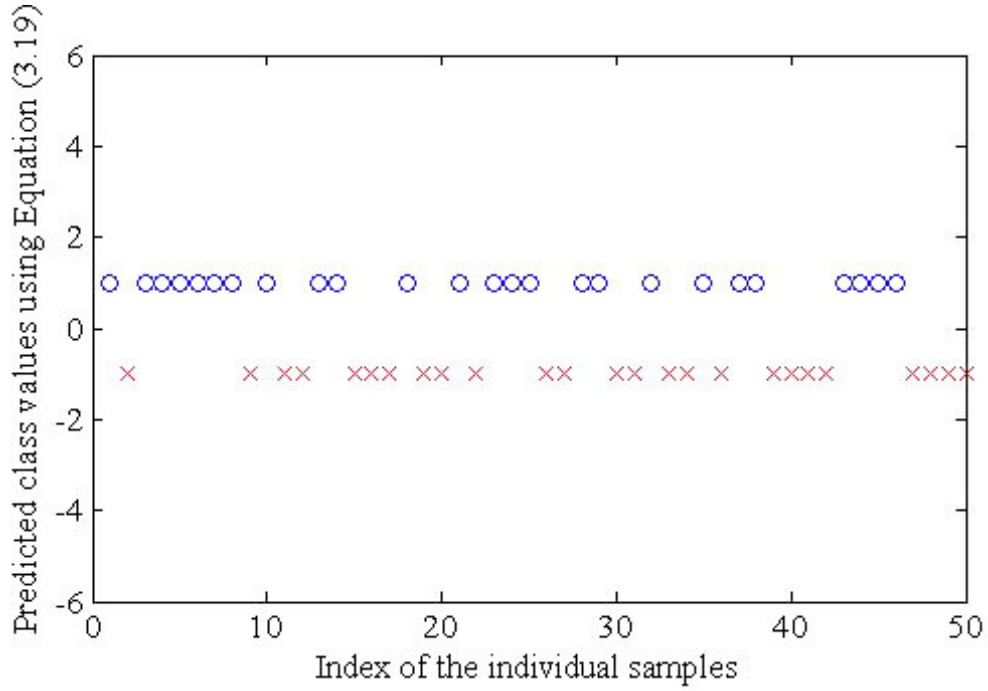


Figure 3.12. One dimensional mapping of Equation (3.19) when the top 25% of the ranked features are kept.

This method takes advantage of the classification decision function Equation (2.71). The simplicity of this method makes it ideal for inclusion in most kernel based classifiers with decision function similar to Equation (2.71). In the next subsection this ranking method is applied to the kernel Fisher's discriminant classifier.

### 3.2.2 Kernel Fisher's Discriminant Classifier Kernel Feature Ranking (KF-KFR)

The same application in Section 3.2 can be extended to ranking features for the KFD classifier. The first step is to calculate the initial alpha vectors as follows:

$$\alpha^{(m)} = \frac{M_{-1}^{(m)} - M_{+1}^{(m)}}{N_{\mu}^{(m)}} \quad (3.26)$$

where

$$\begin{aligned} M_{-1}^{(m)} &= \frac{1}{\ell_{-1}} \sum_{\substack{j=1 \\ j \in C_{-1}}}^{\ell_{-1}} K^{(m)}(\mathbf{x}_i, \mathbf{x}_j) \\ M_{+1}^{(m)} &= \frac{1}{\ell_{+1}} \sum_{\substack{j=1 \\ j \in C_{+1}}}^{\ell_{+1}} K^{(m)}(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \quad (3.27)$$

and

$$N^{(m)} = \sum_{\substack{j=1 \\ j \in \mathbf{C}}}^{\ell_{\mathbf{C}}} K^{(m)}(\mathbf{x}_i, \mathbf{x}_j) \left( I - \frac{1}{\ell_{\mathbf{C}}} \right) K^{(m)}(\mathbf{x}_i, \mathbf{x}_j)^T \quad (3.28)$$

where  $\mathbf{C} = \{C_{-1}, C_{+1}\} = \{-1, +1\}$ . Mika, et al. (1999) discuss *Numerical issues and Regularization* regarding the calculation of Equation (3.28). This is resolved by simply adding a multiple of the identity matrix to  $N$  defined as:

$$N_{\mu}^{(m)} = N^{(m)} + \mu I \quad (3.29)$$

The next step is to use the alpha vectors and the kernel matrix to project the  $n$ -1 dimensional input feature space into a one dimensional space as follows:

$$\hat{\mathbf{x}} = K^{(m)}(\mathbf{x}_i, \mathbf{x}_j) \alpha^{(m)}. \quad (3.30)$$

The projection in Equation (3.30) now becomes the space that is to be solved using an optimization solution. Mika, et al. (1999) use the Matlab Optimization Toolbox (Matlab, 2007) to solve the optimization problem with the projected space calculated in Equation (3.30). For the interested reader the optimization problem is described in detail on pp. 460-462 of (Scholkopf and Smola, 2002). In this paper the one dimensional SMO (Franc

and Hlavac, 2007) is used as the optimization solution. This results in the non-negative alpha vectors  $\hat{\alpha}_i = (\hat{\alpha}_1, \dots, \hat{\alpha}_\ell)$  with an upper bound  $\hat{C}, \hat{C} \geq \hat{\alpha}_\ell \geq 0$ . The support vectors for the KFD trained model are  $\mathbf{x}_k = \mathbf{x}_i$  and the decision function of the KFD classifier is written as  $\text{sign}(f(\mathbf{x}))$  where  $f(\mathbf{x})$  is defined by:

$$f(\mathbf{x}) = \mathbf{w} \phi(\mathbf{x}) + b = \sum_{i=1}^{\ell} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + b. \quad (3.31)$$

The bias  $b$  is calculated by obtaining the average as in Equation (2.65). The final step is to rewrite Equation (3.21) to calculate the ranking values as follows:

$$R_m = \sum_{j=1}^{\ell} \sum_{i=1}^{\ell} \hat{\alpha}_i^{(m)} K^{(m)}(\mathbf{x}_i, \mathbf{x}_j) + b. \quad (3.32)$$

The algorithm for the kernel Fisher's feature ranking method is as follows:

1. For each of the  $n$  features perform steps 2, 3 and 4.
2. Remove the current feature  $m$  from the data set  $\mathbf{x}_i$  training the KFD model using Equations (3.26) through (3.30) to obtain the alpha vectors, support vectors and bias.
3. Assign a ranking value  $R_m$  according to Equation (3.32) and replace the feature  $m$ .
4. After completion of the loop, sort the ranking values  $R_m$  in descending order.
5. Select the  $r$  highest ranked features for training the final KFD classification model.

The procedure is conducted for each feature and ranked in descending order where the largest value corresponds to the feature of most importance. It should be noted that the calculation of the alpha weights in Equation (3.26) is an important step when ranking the features.

### 3.3 Learning Decision Trees using Kernel mapping for creating Multi-class Classification from two-class KFD and SVM Classifiers

In this section a multi-class tree structure for performing multi-class classification with two-class KFD and SVM classifiers is described. The structure and learning of the tree is known as a learning decision tree (Russell and Norvig, 2003). Designing the structure of the tree, at each node a distance measure in the kernel space is calculated between three or more classes. A *branch* connects two nodes within the tree. Branches are added from each node, known as a *parent node*, so long as more than one class remains. A *leaf node* from a parent node specifies the class value when a single class is reached, that is, a node with no successor in the tree. The depth of the tree is determined by the number of nodes along a path from the top parent node to a leaf node. For example, Figure 3.13 shows the tree structure for a ten-class problem where the labels represent the individual classes, 1 = Clean, 2 = F5, 3 = JP Hide, 4 = JSteg, 5 = Model-based, 6 = Model-based Ver. 1.2, 7 = OutGuess, 8 = Steganos, 9 = StegHide, 10 - UTSA.

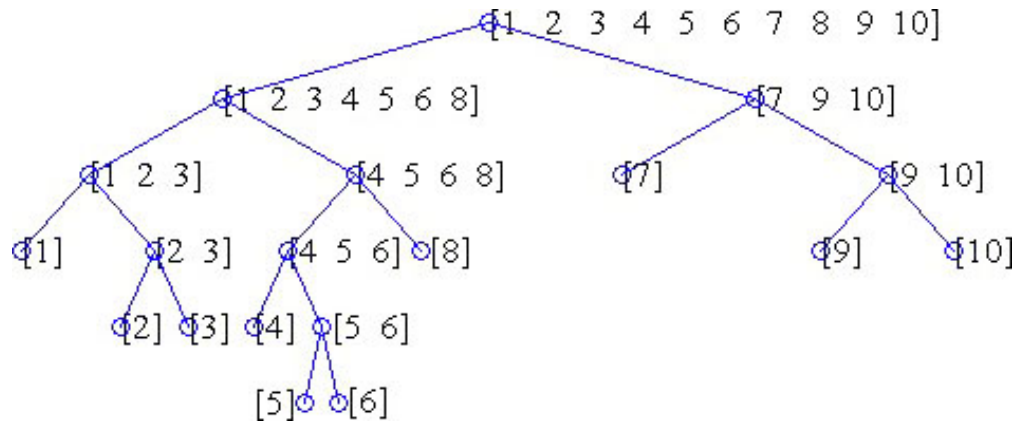


Figure 3.13. Decision tree for a 10-class classification problem with 10 leaf nodes, 9 parent nodes and a maximum depth of 6.

The top node labeled with [1 2 3 4 5 6 7 8 9 10] is at the first level of the tree and the parent node of nodes labeled as [1 2 3 4 5 6 8] and [7 9 10]. The leaf nodes from left to right in this tree are label as [1], [2], [3], [4], [5], [6], [8], [7], [9] and [10]. This tree has a

maximum depth of 6 which is the path from the parent node [1 2 3 4 5 6 7 8 9 10] to [5] or [6].

For this problem there are several steps in learning the tree. The first step is to map the input training set  $\mathbf{x}_i = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\ell] \in \mathbb{R}^n$ ,  $i = 1, \dots, \ell$ ,  $\phi(\mathbf{x}_i): X \rightarrow F$  from input space into a potential higher dimensional space  $F \in \mathbb{R}^\ell$  called kernel space. The mapping  $\phi(\mathbf{x}_i)$  is represented by a kernel function  $K(\mathbf{x}_i, \mathbf{x}_j)$  that defines an inner product in  $\mathbb{R}^\ell$ . Each sample in the training set contains one target value  $y_i \in \mathbf{C} = [C_1, C_2, \dots, C_c]$ ,  $i = 1, 2, \dots, \ell$ . which describes the class to which the sample is a member of. The parameters for calculating the kernel matrix are important when training the tree and the two-class classifiers at each node. The kernels used in this research are as follows

1. linear:  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
2. polynomial:  $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \gamma > 0$
3. radial basis function (RBF):  $K(\mathbf{x}_i, \mathbf{x}_j) = e^{\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)}, \gamma = \frac{1}{2\sigma^2} > 0$
4. sigmoid:  $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$

where,  $\gamma$ ,  $r$ , and  $d$  are kernel parameters.

The distance measure used in this section is an expansion of the KFD (Mika et al., 1999). The second step is to calculate the initial alpha vectors for a multi-class problem. The alpha vectors are defined as follows:

$$\hat{\alpha} = \frac{\bar{M}}{N_\mu} \quad (3.33)$$

where

$$\begin{aligned}\bar{M} &= \frac{1}{[c(c-1)/2]} \sum_{p=1}^{c-1} \sum_{q=2}^c (M_{C_p} - M_{C_q}) \\ M_{C_k} &= \frac{1}{\ell_{C_k}} \sum_{\substack{j=1 \\ j \in C_k}}^{\ell_{C_k}} K(\mathbf{x}_i, \mathbf{x}_j)\end{aligned}\tag{3.34}$$

and

$$\begin{aligned}N_\mu &= N + \mu I \\ N &= \sum_{\substack{j=1 \\ j \in \mathbf{C}}}^{\ell_{\mathbf{C}}} K(\mathbf{x}_i, \mathbf{x}_j) \left( I - 1/\ell_{\mathbf{C}} \right) K(\mathbf{x}_i, \mathbf{x}_j)^T\end{aligned}\tag{3.35}$$

The regularization value  $\mu$  must be large enough so that the  $(N_\mu)^{-1}$  is positive definite (Mika et al., 1999). The next step is to use the alpha vectors and the kernel matrix to project the input feature space into a one dimensional space as follows:

$$\hat{\mathbf{x}}_i = K(\mathbf{x}_i, \mathbf{x}_j) \hat{\alpha}\tag{3.36}$$

where  $\hat{\mathbf{x}}$  is an  $[\ell \times 1]$  vector. Now the individual class distance can be calculated as

$$\hat{D}_{C_k} = \frac{1}{\ell_{C_k}} \sum_{\substack{i=1 \\ i \in C_k}}^{\ell_{C_k}} \hat{\mathbf{x}}_i\tag{3.37}$$

The distance vector  $\hat{D}_{C_k}$  is of length  $c$ . Once the distance vectors are calculated, the next step is to taking the average of  $\hat{D}_{C_k}$  which provides a separation point between classes



when  $C_k > 2$ . For example the top node in Figure 3.13 contains classes [1 2 ... 10] and the classes are divided into two branches. The left branch contains classes [1 2 3 4 5 6 8] while the right branch contains classes [7 9 10]. Figure 3.14 is the corresponding figure to Figure 3.13 which contains the 10 classes totaling 1000 samples as shown on the x-axis and the sample values on the y-axis. The distance values of  $\hat{D}_{C_k}$  are shown within the figure as well. Taking the average of  $\hat{D}_{C_k}$  is -5.0313 which is the value used to separate the ten classes into two sub classes.

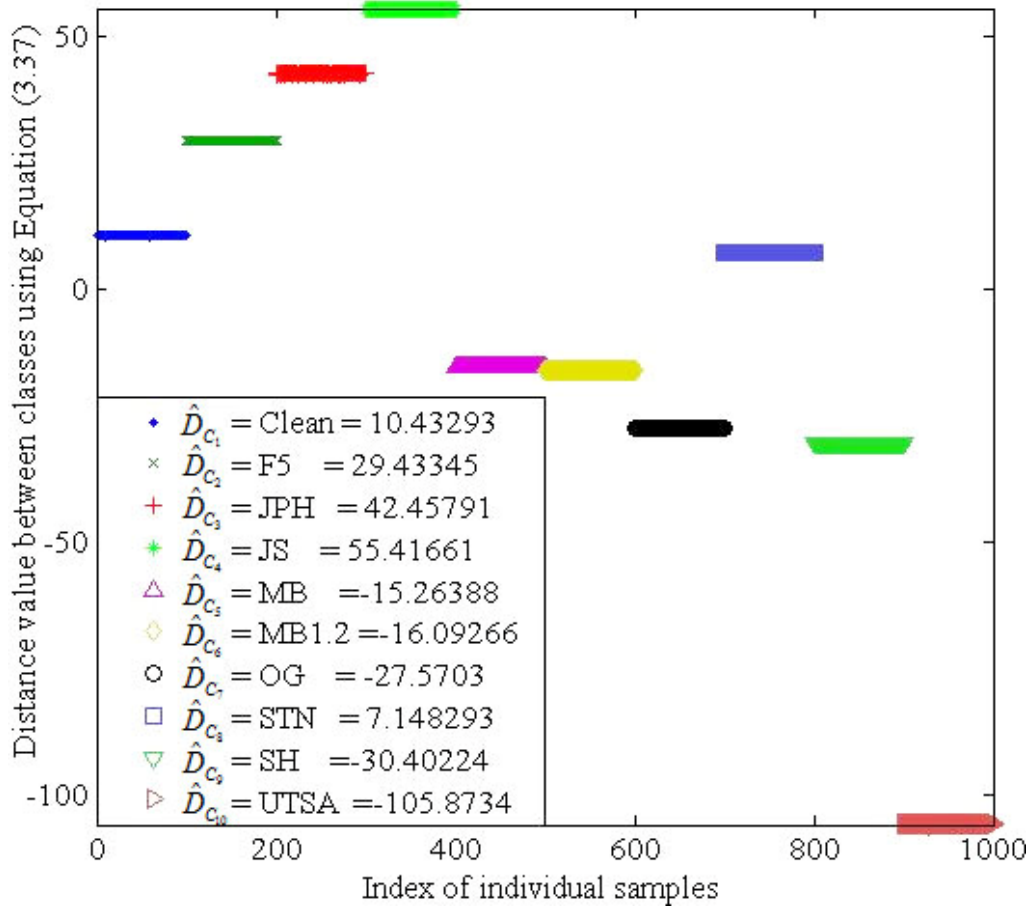


Figure 3.14. Distance values  $\hat{D}_{C_k}$  for a 10 class problem.

Once a new branch with more than two-classes is built the distance measure is calculated again. The nodes of the tree are expanded from left to right until a leaf node is reached.

Consider the node labeled as [2 3], this node will contain two leaf nodes labeled [2] and [3].

The decision tree learning algorithm is shown as follows:

1. Input training data  $\mathbf{x}_i$  with class labels, the kernel parameters and the classifier (KFD or SVM).
2. If  $\mathbf{x}_i$  is empty return.
3. Else if class labels of  $\mathbf{x}_i$  are all the same make a leaf node and return.
4. Else if  $\mathbf{x}_i$  contains two-classes make two leaf nodes, a left and right, and return.
5. Else if  $\mathbf{x}_i$  contains three-classes calculate the average distance for each class.
6. Divide the input data into two classes creating two branches, a left and right.
  - i. If the left Branch contains more than two classes step 1.
  - ii. Else make a leaf node and go to step iii.
  - iii. If the right branch contains more than two classes go to step 1.
  - iv. Else make a leaf node and return.
7. Return tree
8. Train the two-class classifiers for each node of the tree.

### 3.4 Fusion of Multi-Class Classification Systems

In this section the fusion methods of the multi-class detection systems is covered. The class labels of the 8 multi-class detection systems are fused. In this research there are 10 image classes, consisting of clean, F5 (Westfeld, 2001; 2003), JP Hide (Latham, 1999), JSteg (Upham, 1993), Model-base (Sallee, 2003; 2006), Model-based Version 1.2 (Sallee, 2008a), OutGuess (Provos, 2004), Steganos (2008), StegHide (Hetzl, 2003) and UTSA (Agaian et al., 2006). The three fusion methods, AdaBoost (Bishop, 2006, pp. 358), Bayesian Belief Networks (Murphy, 2001) and Probabilistic Neural Networks (Leap et al., 2007), used in this section were described in Chapter 2 Section 2.7.

### 3.4.1 AdaBoost Boosting

In this sub section the 7 detection systems are fused using AdaBoost (Rodriguez and Peterson, 2008b). Each classification model is defined as  $M_k$ . The input training set is  $\mathbf{x}_i = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\ell] \in \mathbb{R}^n$ , with each sample in the training set contains one target value  $\mathbf{C} = C_k = [C_1, C_2, \dots, C_c]$ ,  $k = 1, 2, \dots, c$ , (known as the class labels  $y_i \in \mathbf{C}$ ,  $i = 1, 2, \dots, \ell$ ). The method implemented in this research is from Bishop (2006, pp. 658). The method described by Bishop (2006) has three steps as follows

1. The data weighting coefficients  $\{w_i\}$  are initialized as  $w_i^{(1)} = 1/\ell$  for  $i = 1, \dots, \ell$ .
2. For  $k = 1, \dots, 7$ :
  - (a) Fit a classifier  $M_k(\mathbf{x})$  to the training data by minimizing the weighted error function

$$J_k = \sum_{i=1}^{\ell} w_i^{(k)} I(M_k(\mathbf{x}_i) \neq y_i)$$

where  $I(M_k(\mathbf{x}_i) \neq y_i)$  is the indicator function and equals 1 when  $M_k(\mathbf{x}_i) \neq y_i$  and 0 otherwise.

- (b) Evaluate the quantities

$$\mathcal{E}_k = \frac{\sum_{i=1}^{\ell} w_i^{(k)} I(M_k(\mathbf{x}_i) \neq y_i)}{\sum_{i=1}^{\ell} w_i^{(k)}}$$

and then use these to evaluate

$$\alpha_k = \ln \left\{ \frac{1 - \mathcal{E}_k}{\mathcal{E}_k} \right\}$$

- (c) Update the data weighting coefficients

$$w_i^{(k+1)} = w_i^{(k)} e^{\alpha_k I(M_k(\mathbf{x}_i) \neq y_i)}$$

3. Making a prediction using the final trained model for an input image sample

$\mathbf{x}_0 \in \{C, F5, JPH, JS, MB, MB1.2, OG, STN, SH, UTSA\}$  is given by

$$f(\mathbf{x}_0) = \sum_{k=1}^7 \alpha_k M_k(\mathbf{x}_0)$$

### 3.4.2 Bayes Network for Model Averaging

In this sub section the 7 detection systems are fused using a Bayesian network (Rodriguez et al., 2008b). Each classification model is defined as  $M_k$  as shown in Figure 3.15.

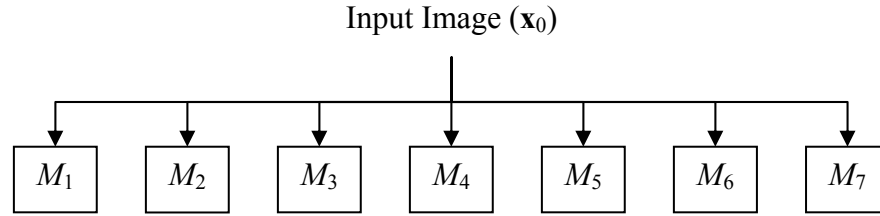


Figure 3.15. Detection structure for 8 classification models.

Table 3.2 shows the prior probabilities that a target  $T$  is a clean ( $C$ ), F5, JP Hide ( $JPH$ ), JSteg ( $JS$ ), Model-based ( $MB$ ), Model-based Version 1.2 ( $MB1.2$ ), OutGuess ( $OG$ ), Steganos ( $STN$ ), StegHide ( $SH$ ) and UTSA ( $UTSA$ ) image.

Table 3.2. Distribution of the image types.

Target( $T$ )									
$T =$ $C$	$T =$ $F5$	$T =$ $JPH$	$T =$ $JS$	$T =$ $MB$	$T =$ $MB1.2$	$T =$ $OG$	$T =$ $STN$	$T =$ $SH$	$T =$ $UTSA$
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

For example an input image sample  $\mathbf{x}_0 \in \{C, F5, JPH, JS, MB, MB1.2, OG, STN, SH, UTSA\}$  as shown in Figure 3.15 fed into each of the trained classification detection systems will have a class label assigned from

each of the systems. So, to determine the probability that the class label is  $C$  when each of the models returns a class label as  $C$  the model averaging topology dictates a joint pdf as

$$P(T, M_1, M_2, M_3, M_4, M_5, M_6, M_7) = \\ P(M_1|T)P(M_2|T)P(M_3|T)P(M_4|T)P(M_5|T)P(M_6|T)P(M_7|T)P(T)$$

The method used to facilitate the computations in the model averaging is Murphy's (2001) Bayes Net Toolbox (BNT) for Matlab resulting in the following calculations.

$$P(T = C | M_1 = "C", M_2 = "C", M_3 = "C", M_4 = "C", M_5 = "C", M_6 = "C", M_7 = "C") \\ = \frac{P(T = C, M_1 = "C", M_2 = "C", M_3 = "C", M_4 = "C", M_5 = "C", M_6 = "C", M_7 = "C")}{P(M_1 = "C", M_2 = "C", M_3 = "C", M_4 = "C", M_5 = "C", M_6 = "C", M_7 = "C")} \\ = \frac{P(T = C, M_1 = "C", M_2 = "C", M_3 = "C", M_4 = "C", M_5 = "C", M_6 = "C", M_7 = "C")}{\sum_{\mathbf{x}_0} P(T = \mathbf{x}_0, M_1 = "C", M_2 = "C", M_3 = "C", M_4 = "C", M_5 = "C", M_6 = "C", M_7 = "C")}$$

Using Bayes' Rule the numerator can be represented as

$$P(M_1|C)P(M_2|C)P(M_3|C)P(M_4|C)P(M_5|C)P(M_6|C)P(M_7|C)P(T) \\ = (P(M_1 = "C" | T = C)P(M_2 = "C" | T = C)P(M_3 = "C" | T = C)P(M_4 = "C" | T = C) \\ P(M_5 = "C" | T = C)P(M_6 = "C" | T = C)P(M_7 = "C" | T = C)P(T = C))$$

and the denominator as

$$\sum_{\mathbf{x}_0} P(T = \mathbf{x}_0, M_1 = "C", M_2 = "C", M_3 = "C", M_4 = "C", M_5 = "C", M_6 = "C", M_7 = "C") \\ = \sum_{\mathbf{x}_0} (P(M_1 = "C" | T = \mathbf{x}_0)P(M_2 = "C" | T = \mathbf{x}_0)P(M_3 = "C" | T = \mathbf{x}_0)P(M_4 = "C" | T = \mathbf{x}_0) \\ P(M_5 = "C" | T = \mathbf{x}_0)P(M_6 = "C" | T = \mathbf{x}_0)P(M_7 = "C" | T = \mathbf{x}_0)P(T = \mathbf{x}_0))$$

### 3.4.3 Probabilistic Neural Network (PNN) Fusion

In this method the outputs of individual classification models are treated as input features to train the PNN fusion system. The key is to use the class labels from each of the systems as posterior probability estimates and employing them as features in the neural network. It should be noted that one of the posterior probabilities from the input classifier should be removed. For the seven individual ten-class classifiers used in this research each of the classification models,  $M_k$ , will contribute seven inputs for training the PNN.

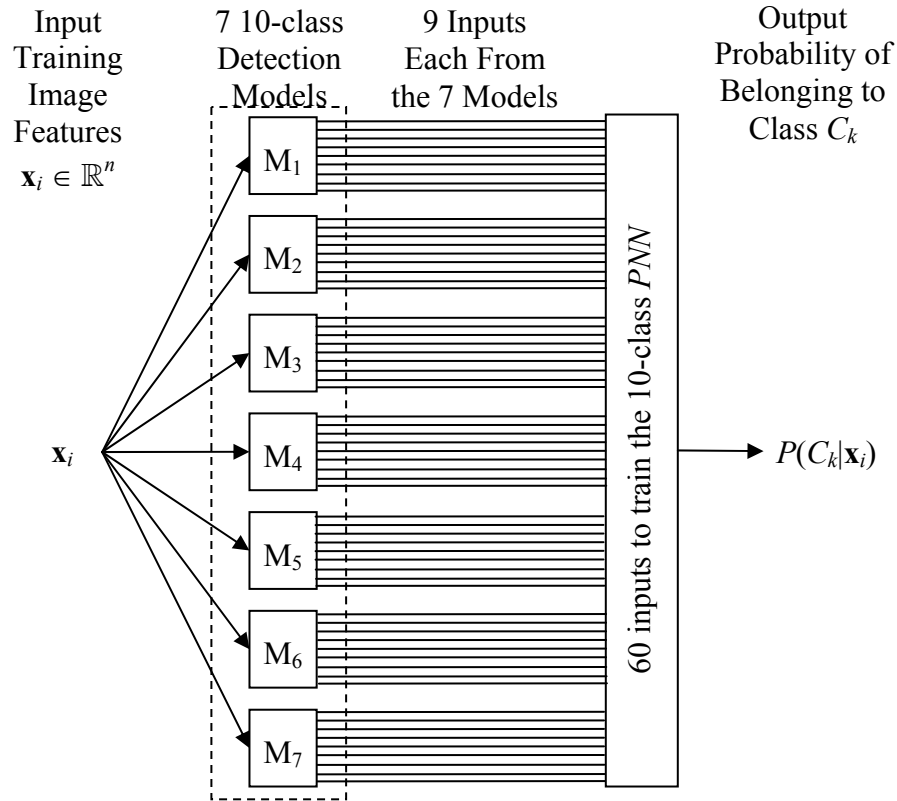


Figure 3.16. Probabilistic Neural Network Classification Structure.

### 3.5 Summary

This chapter presented three new methods for improving multi-class detection systems for the kernel Fisher's discriminant and support vector machines. The first method used in the system is the generation of features using the DCT for JPEG images. The major components of the new feature generation method are the decomposition of the DCT coefficients and the use four different predictors. The second new method consists of a new feature ranking method which uses the individual classifiers to rank the order of the features on class separability in the kernel space. The final method consists of a multi-class tree which is expanded with the use of a distance measure between classes in the kernel space. In addition to the three new methods used in the development of multi-class classification for KFD and SVM is the fusion of multiple steganalysis systems. The fusion techniques used are based on modified implementation from AdaBoost (Bishop, 2006), Bayesian networks (Murphy, 2001) and probabilistic neural networks (Leap et al., 2007).

Chapter 4 demonstrates results with an increase in classifier performance. The results shown compare an existing multi-class SVM classifier with the new methods shown in this chapter, feature selection, multi-class classifier and a modified simple fusion method.

## IV. Analysis and Results

The goal of the steganalysis classification system is to identify an input JPEG image as a clean image or identify the embedding algorithm used. The nine embedding algorithms tested over include F5 (Westfeld, 2001; 2003), JP Hide (Latham, 1999), JSteg (Upham, 1993), Model-base (Sallee, 2003; 2006), Model-based Version 1.2 (Sallee, 2008a), OutGuess (Provos, 2004), Steganos (2008), StegHide (Hetzl, 2003) or UTSA (Agaian et al., 2006). This chapter compares the performance of the KFD and SVM multi-class system developed against four (i.e., EM,  $k$ -NN, Parzen window and PNN) multi-class and three fusion (i.e., AdaBoost, Bayes and PNN fusion) classification techniques. In order to statistically compare the systems,  $k$ -fold cross validation is used for both training and testing the system within a clean JPEG image dataset and nine stego image datasets. The statistical tool applied for analysis is the two tailed student  $t$ -test.

The clean JPEG image dataset used as a cover image set for analyzing the system includes 1000 RGB images of size 512×512 with a quality factor of 75%. Nine stego image datasets are generated from the clean dataset with a stego message from the aforementioned nine embedding tools of 4000 characters which is equivalent to one page of text. The number of DCT coefficients altered within a color layer of a JPEG image is known as the *embedding rate* (Kharrazi et al., 2005). The average embedding rate of the coefficients altered for each stego image dataset are as follows:

- F5 has an average embedding rate 6.25%.
- JP Hide (JPH) has an average embedding rate 3.76%
- JSteg (JS) has an average embedding rate 7.53%
- Model-based (MB) has an average embedding rate 5.36%
- Model-based Version 1.2 (MB1.2) has an average embedding rate 5.68%
- OutGuess (OG) has an average embedding rate 3.24%
- Steganos (STN) has an average embedding rate 0.75%
- StegHide (SH) has an average embedding rate 2.30%
- UTSA has an average embedding rate 5.38%



Note that in testing and training, 100 images are chosen from each clean and stego image dataset. The clean images used within the clean image dataset do not appear as stego images used within the stego image datasets, nor does any stego image reappear embedded with another steganography algorithm. For example, none of the F5 images were the same as the JSteg images.

This chapter demonstrates the performance of the steganalysis classification system developed in this research. Section 4.1 describes the statistical methods of measure used for testing and validation in the experiment. The results include a comparison of the feature generation methods: wavelet feature generation, DCT feature generation and DCT directional and frequency decomposition feature generation. In Section 4.3, results on the steganalysis dataset for eight multi-class classification methods including expectation maximization with mixture models (EM),  $k$ -nearest neighbors ( $k$ -NN), kernel Fisher's discriminant (KFD), Parzen window, probabilistic neural networks (PNN), support vector machines (SVM) and StegoWatch are discussed, respectively. Section 4.4 demonstrates a performance improvement when utilizing and fusing several classification algorithms together. Experimental results of three fusion techniques using AdaBoost, Bayesian neural network, and probabilistic neural network, are shown. Finally, a summary of all the results is presented in Section 4.5.

#### **4.1 Confirming and Validating the Analysis**

In statistics a result is *statistically significant* if it is unlikely to have occurred by chance. A statistically significant difference between two sets of results simply implies that there is statistical evidence that there is a difference. This however, does not indicate that the difference is necessarily large. In this research the results are generated using  $k$ -fold cross validation to determine the classification accuracy of the classification models. A  $t$ -test between paired samples about the means with a confidence level of 95% is used to determine the statistical significance of the results.

In  $k$ -fold cross-validation, the original sample is partitioned into  $k$  subsamples. Of the  $k$  subsamples, a single subsample is retained as the test data for testing the model, and the remaining  $k-1$  subsamples are used as training data. The cross-validation process is then repeated  $k$  times (the *folds*), with each of the  $k$  subsamples used exactly once as the validation data. The  $k$  results from the folds are averaged to produce a single estimation (Kohavi, 1995; Mitchell, 1997; Russell and Norvig, 2003).

In this chapter the data is partitioned into five groups of equal size as shown in Figure 4.1. For each run four of the groups are used for training the classification model and the remaining group is used for testing the model. This procedure is repeated for five runs where the runs are for all five possible choices of the held out test group.

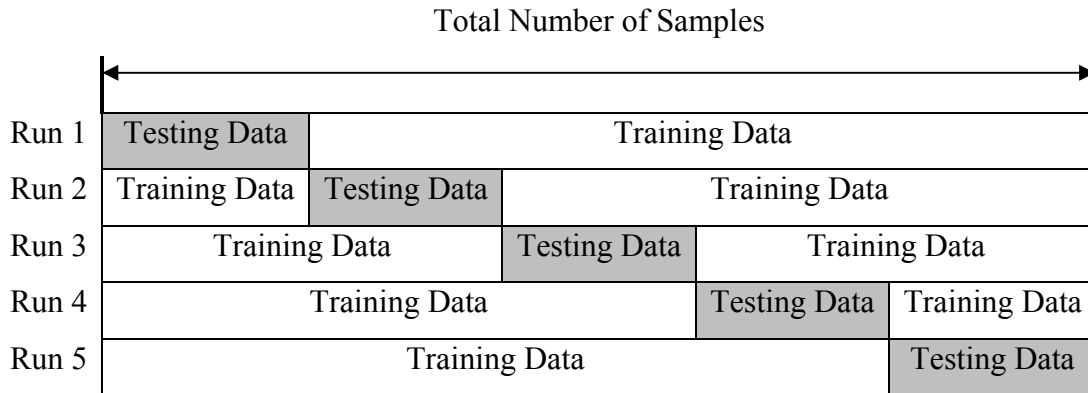


Figure 4.1. 5-fold cross-validation with 5 runs consisting of 80% of the data for training the classification model and 20% for testing the training model.

To ensure that the test of significance is calculated properly the Lilliefors test for normality is used to determine if the results being analyzed are normally distributed (Lilliefors, 1967; Abdi and Molin, 2007). If the result is determined that the results are normally distributed the  $t$ -test is used to test for statistical significance (Hogg, and Tanis, 1993; Kohavi, 1995; Rice, 1995; Wackerly et al., 1996). On the other hand, if the test for normality fails then the Wilcoxon test is used to determine if the results are significant.

In the next section the results are shown in tables using the 5-fold cross validation. The tables are accompanied by analysis to determine if the reported results are statistically significant.

## 4.2 Feature Generation Method Comparison

The results in this section show a comparison between the three feature generation methods of wavelet features, DCT features, and DCT directional and frequency decomposition features, and results that use all three feature generation methods combined. Prior to classification the data is prepared using the data standardization described in subsection 2.3.1 (Dillon and Goldstein, 1984, pp. 12-13). Feature discrimination capability results from executing a SVM two-class classifier without and with the SVM-kernel feature ranking described in subsection 3.2.1. The SVM method used is *SVMlight* (Joachims, 1998, 2007). The feature ranking method used is the SVM-kernel feature ranking method presented in Section 3.2. The kernel function  $K(\mathbf{x}_i, \mathbf{x}_j)$  used is the radial basis function  $e^{\left(-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2\right)}$  with the parameter  $\gamma=1/\left(2(12)^2\right)$  and the upper bound  $\hat{C} = 12$ . The results of the analysis include the percentage of true positive and true negatives shown on a class-by-class basis where the clean image sets are compared against each steganography embedding image set. The true negative indicates the percentage of clean images correctly classified as clean images, while the true positive indicates the percentage of stego images correctly classified as stego images. The average of true negative and true positive is the classification accuracy (CA).

#### 4.2.1 Wavelet Feature Generation (Lyu and Farid, 2004)

The results for the wavelet feature generation, which generates 72 features, are shown without and with feature ranking in Tables 4.1 and 4.2, respectively.

Table 4.1. Classification accuracy for wavelet feature generation.

	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
True Negative	64.8± 5.0	94.3± 9.8	98.1± 2.6	59.4± 13.6	59.5± 10.0	71.3± 8.7	74.8± 11.9	50.7± 6.1	80.7± 8.5
True Positive	66.6± .34	81.6± 6.3	98.1± 2.6	56.3± 9.7	56.9± 7.5	70.7± 9.2	68.5± 6.7	50.2± 7.0	78.4± 7.6
Classification Accuracy	65.7± 3.9	87.9± 5.1	98.1± 1.0	57.8± 11.6	58.2± 8.7	71.0± 8.9	71.6± 8.0	50.4± 6.5	79.5± 6.7

Table 4.2. Classification accuracy for wavelet feature generation using feature ranking.

	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
No. of Features	25	25	15	19	20	22	16	12	39
True Negative	74.9± 4.4	99.0± 2.1	99.1± 2.1	71.6± 6.0	73.1± 6.9	74.1± 7.2	83.8± 2.8	64.6± 4.5	86.4± 4.0
True Positive	78.0± 4.4	91.6± 8.6	98.1± 2.6	66.4± 3.4	69.6± 5.1	74.0± 6.5	72.6± 3.4	61.4± 2.6	82.9± 3.5
Classification Accuracy	76.4± 2.7	95.3± 3.8	98.5± 1.3	69.0± 4.1	71.3± 5.5	74.0± 6.7	78.2± 2.4	63.0± 2.6	84.6± 3.2

The results shown in Table 4.2 indicate an improvement of detection accuracy by proper selection of features during training. The second row shows the number of features among 72 identified by the SVM-kernel feature ranking method. The statistical significance of selecting features with the proposed feature saliency metric is depicted in Table 4.3. As can be seen in the significance testing for classification accuracy, the Clean vs. F5 image classes, Clean vs. MB1.2, and Clean vs. SH comparisons show statistically

significant difference in the mean, while the difference in the mean for the other embedding methods are not statistically significant.

Table 4.3. *t*-test; paired two samples for means between Tables 4.1 and 4.2.

<i>t</i> -Critical Two-Tail; $t_{4, 0.975} = 2.776$ , $n_1 = n_2 = 5$ (corresponding to 5-fold), $\alpha = 0.05$									
Image Classes	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
<i>t</i> -Stat	5.97	2.39	0.186	2.42	5.03	0.86	1.68	4.09	1.86
Statistically Significant	Yes	No	No	No	Yes	No	No	Yes	No

#### 4.2.2 DCT Feature Generation (Pevny and Fridrich, 2006)

The results for the DCT feature generation (Pevny and Fridrich, 2006) are shown without and with feature selection in Tables 4.4 and 4.5. The 274 features generated (Pevny and Fridrich, 2006) are an extension of the original features described in Section 2.2.2 developed by Fridrich (2004).

Table 4.4. Classification accuracy for DCT feature generation.

	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
True Negative	100± 0.0	100± 0.0	100± 0.0	99.0± 2.1	100± 0.0	100± 0.0	86.5± 6.9	100± 0.0	100± 0.0
True Positive	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	87.8± 6.0	100± 0.0	100± 0.0
Classification Accuracy	100± 0.0	100± 0.0	100± 0.0	99.5± 1.1	100± 0.0	100± 0.0	87.1± 6.0	100± 0.0	100± 0.0

Table 4.5. Classification accuracy for DCT feature generation using feature ranking.

	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
No. of Features	12	24	5	7	7	5	23	5	5
True Negative	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	89.1± 3.6	100± 0.0	100± 0.0
True Positive	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	88.5± 5.7	100± 0.0	100± 0.0
Classification Accuracy	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	88.7± 2.8	100± 0.0	100± 0.0

The results shown in Table 4.5 indicate after the SVM-kernel feature ranking, only a few of the 274 features are necessary for a perfect classification accuracy in most of the cases except the Clean vs. STN image classes. The statistical significance of selecting features with the proposed feature ranking is depicted in Table 4.6. As can be seen in the significance testing for classification accuracy, only the Clean vs. STN image classes show significant difference in the mean, while the difference in the mean for the other stego embedding methods are not statistically significant. Although there are no improvement (quite difficult to improve from a perfect classification) in the classification accuracy even with the inclusion of a feature ranking method, the utility is apparent in the reduced number of features necessary to still achieve perfect classification.

Table 4.6. *t*-test; paired two samples for means between Tables 4.4 and 4.5.

<i>t</i> -Critical Two-Tail; $t_{4,0.975} = 2.776$ , $n_1 = n_2 = 5$ (corresponding to 5-fold), $\alpha = 0.05$									
Image Classes	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
<i>t</i> -Stat	0.0	0.0	0.0	1.00	0.0	0.0	0.43	0.0	0.0
Statistically Significant	No	No	No	No	No	No	Yes	No	No

### 4.2.3 DCT Directional and Frequency Decomposition

The results for the DCT directional and frequency decomposition feature generation described in Section 3.1 are shown without feature selection in Table 4.7 and with feature selection in Table 4.8. This feature generation method results in 180 features.

Table 4.7. Classification accuracy for DCT directional and frequency feature generation.

	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
True Negative	95.4± 5.3	99.0± 2.1	99.0± 2.1	99.0± 2.1	96.4± 5.7	94.5± 5.8	96.2± 6.0	98.0± 2.8	100± 0.0
True Positive	95.4± 5.3	100± 0.0	98.2± 4.1	93.7± 4.9	93.8± 6.6	97.0± 2.7	89.2± 4.2	92.9± 6.3	100± 0.0
Classification Accuracy	95.4± 2.1	99.5± 1.1	98.6± 2.0	96.3± 1.8	95.1± 2.3	95.7± 2.5	92.7± 1.9	95.4± 3.0	±100± 0.0

Table 4.8. Classification accuracy for DCT directional and frequency feature generation using feature ranking.

	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
No. of Features	21	35	22	26	27	24	23	25	22
True Negative	98.2± 4.1	100± 0.0	100± 0.0	98.2± 4.1	98.2± 4.1	98.1± 2.6	100.0± 0.0	100± 0.0	100± 0.0
True Positive	100± 0.0	100± 0.0	100± 0.0	98.1± 2.6	98.1± 2.6	98.1± 2.6	97.1± 2.6	96.3± 3.8	100± 0.0
Classification Accuracy	99.1± 2.0	100± 0.0	100± 0.0	98.1± 1.9	98.1± 1.9	98.1± 1.1	98.5± 1.3	98.1± 1.9	100± 0.0

The results shown in Table 4.8 indicate an improvement of detection accuracy by proper ranking of features during training. The second row shows the number of features among 180 identified by the presented feature saliency metric, i.e., the SVM-kernel feature ranking method. The statistical significance of selecting features with the proposed feature ranking is depicted in Table 4.9. As can be seen for the classification accuracy and significance testing, the Clean vs. F5, Clean vs. MB1.2, Clean vs. STN, Clean vs. SH

embedding methods show significant difference in the mean, while the difference in the mean for the other embedding methods are not statistically significant.

Table 4.9. *t*-test; paired two samples for means between Tables 4.7 and 4.8.

<i>t</i> -Critical Two-Tail; $t_{4, 0.975} = 2.776$ , $n_1 = n_2 = 5$ (corresponding to 5-fold), $\alpha = 0.05$									
Image Classes	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
<i>t</i> -Stat	4.06	1.00	1.51	1.69	6.37	2.23	5.94	3.29	0.0
Statistically Significant	Yes	No	No	No	Yes	No	Yes	Yes	No

#### 4.2.4 Combined Features

The wavelet features (Lyu and Farid, 2004), DCT features (Pevny and Fridrich, 2006) and DCT directional and frequency decomposition features are combined to increase the classification accuracy for each of the targeted embedding methods. The results for the combined features are shown with feature selection in Table 4.10. The total number of features in the combination of the three methods is 526.

Table 4.10. Classification accuracy for combined feature generation using feature ranking.

	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
No. of Features	11	18	5	6	10	5	15	7	5
True Negative	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0
True Positive	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0
Classification Accuracy	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0	100±0.0

The results shown in Table 4.10 indicate that perfect detection accuracies are obtained for each image class by combining the three feature generation methods and performing a proper ranking of the 526 features. Statistical significance comparisons are performed for



the combined features versus the first three compared methods in this chapter. The statistical significance shown in Table 4.11 is the classification accuracy comparison between the combined features from Table 4.10 and the wavelet feature generation results of Table 4.2.

Table 4.11. *t*-test: paired two samples for means of wavelet features with feature ranking vs. combined features with feature ranking.

<i>t</i> -Critical Two-Tail; $t_{4, 0.975} = 2.776$ , $n_1 = n_2 = 5$ (corresponding to 5-fold), $\alpha = 0.05$									
Image Classes	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
<i>t</i> -Stat	19.4	2.73	2.44	16.5	11.4	8.55	20.0	30.8	10.6
Statistically Significant	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes

The statistical significance shown in Table 4.12 is the classification accuracy comparison between the combined features from Table 4.10 and the DCT feature generation results of Table 4.5.

Table 4.12. *t*-test: paired two samples for means of DCT features with feature ranking vs. combined features with feature ranking.

<i>t</i> -Critical Two-Tail; $t_{4, 0.975} = 2.776$ , $n_1 = n_2 = 5$ (corresponding to 5-fold), $\alpha = 0.05$									
Image Classes	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
<i>t</i> -Stat	0.0	0.0	0.0	0.0	0.0	0.0	8.70	0.0	0.0
Statistically Significant	No	No	No	No	No	No	Yes	No	No

The statistical significance shown in Table 4.13 is the classification accuracy comparison between the combined features from Table 4.10 and the DCT directional and frequency feature generation results of Table 4.8.

Table 4.13. *t*-test: paired two samples for means of DCT directional and frequency features with feature ranking vs. combined features with feature ranking.

<i>t</i> -Critical Two-Tail; $t_{4, 0.975} = 2.776, n_1 = n_2 = 5$ (corresponding to 5-fold), $\alpha = 0.05$									
Image Classes	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
<i>t</i> -Stat	1.0	0.0	0.0	2.17	2.17	4.00	2.17	0.0	0.0
Statistically Significant	No	No	No	No	No	Yes	No	No	No

The results shown in Table 4.11 indicate a significant improvement in classification accuracy when comparing the *wavelet features with feature ranking* versus the combined features with feature ranking method for all embedding methods, except JPHide and JSteg. Table 4.12 only shows classification accuracy improvement for STN when comparing *DCT features with feature ranking* (using 23 features) versus the combined features with feature ranking (using 15 features). Similarly, in Table 4.13 classification accuracy improvement is achieved for the detection of OG when comparing *DCT decomposition features with feature ranking* (using 24 features) versus combined feature with feature ranking (using 5 features). This analysis further highlights the strengths and weaknesses of each of the feature generation methods and its capability of detecting certain embedding methods. By combining the features from the three feature generation methods and applying the SVM-kernel feature ranking method the classification accuracy is improved in identifying stego images from clean images.

#### 4.2.5 Summary of Feature Generation Methods

From subsection 4.2.1 to 4.2.4, the results from each individual feature generation method and the combined features are demonstrated. A summary table on classification accuracies is shown in Table 4.14. It is apparent that the combined features integrate the capability of the three methods and achieves perfect classification accuracy.

Table 4.14. Classification accuracy summary for the individual feature generation and combined features when feature ranking is used.

	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
Wavelets	76.4± 2.7	95.3± 3.8	98.5± 1.3	69.0± 4.1	71.3± 5.5	74.0± 6.7	78.2± 2.4	63.0± 2.6	84.6± 3.2
DCT	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	88.7± 2.8	100± 0.0	100± 0.0
DCT Decomp	99.1± 2.0	100± 0.0	100± 0.0	98.1± 1.9	98.1± 1.9	98.1± 1.1	98.5± 1.3	98.1± 1.9	100± 0.0
Combined	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0	100± 0.0

The SVM-kernel feature ranking method has shown that the best subset of features can be identified to improve classification accuracy of the two-class classifier. Table 4.15 shows how each of the feature generation method contributes to the various stego methods in the number of features to obtain the combined features for clean versus each stego image class as in Table 4.10. For a list of specific features associated with the methods in Table 4.15 the reader is referred to Appendix A.

Table 4.15. Number of features used from each of the feature generation method in feature combination.

	Clean vs. F5	Clean vs. JPH	Clean vs. JS	Clean vs. MB	Clean vs. MB1.2	Clean vs. OG	Clean vs. STN	Clean vs. SH	Clean vs. UTSA
No. of Features	11	18	5	6	10	5	15	7	5
Wavelets	1	3	0	0	0	0	2	0	0
DCT	5	12	5	5	5	5	7	5	5
DCT Decomp	5	3	0	1	5	0	6	2	0

### 4.3 Results for Individual Multi-class Detection Systems

This section provides results for the seven multi-class classification systems designed to solve the steganalysis problem of identifying the embedding methods. For each multi-class detection system the process of performing feature preprocessing, feature extraction, feature ranking, classification and multi-class classification is followed. In this section the six classification methods described in Section 2.6 and a commercial tool are used as part of seven individual multi-class detection systems: expectation maximization,  $k$ -nearest neighbors, Parzen window, probabilistic neural networks, kernel Fisher's discriminant, support vector machines, and StegoWatch, which is a commercial detection tool. The features used for classification are the combination of wavelet features, DCT features and the presented DCT directional and frequency decomposition features. The feature improvement includes data standardization, feature extraction and feature ranking methods which are used in conjunction with the multi-class systems. All normalization, feature ranking/selection, and settings were tested where only the best performing combination is presented. For example, in the EM method in Section 4.3.1, the Bhattacharyya distance is used instead of the other four feature ranking/selection discussed in Section 2.5 since the Bhattacharyya distance provided the highest classification accuracy combined with the other parameter combinations.

#### 4.3.1 Expectation Maximization

Table 4.16 shows the classification accuracy from a 5-fold cross validation when performing multi-class classification using expectation maximization (EM). The feature improvement methods and classification parameters used in expectation maximization are listed in the following, in which the combination of parameters provides the highest classification accuracy.

- The data for this model is not normalized;
- Bhattacharyya distance is used for feature ranking with the top 34 out of 526 features selected;

- PCA is performed on the subset of un-normalized 34 features resulting in 12 principal components with eigenvalues greater than 1;
- The number of clusters are determined by using a clustering algorithm on each of the training classes (Sanguinetti et al., 2005) prior to training the EM algorithm where two-clusters are used for each class with the exception of the Steganos class which requires three clusters, and each class is trained individually where the 10 individual models return the mean and covariance's used with the Bayes classifier.

Table 4.16. Classification accuracy for 10-class expectation maximization classifier.

		Actual									
Predicted		Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
	Clean	83± 5.7	0± 0.0	0± 0.0	0± 0.0	1± 2.2	0± 0.0	0± 0.0	14± 4.4	0± 0.0	0± 0.0
	F5	0± 0.0	88± 9.0	2± 2.7	0± 0.0	1± 2.2	0± 0.0	0± 0.0	2± 4.4	0± 0.0	4± 4.1
	JPH	0± 0.0	2± 2.7	90± 7.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	5± 5.0	0± 0.0	0± 0.0
	JS	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	MB	0± 0.0	0± 0.0	0± 0.0	0± 0.0	51± 11.9	49± 11.9	0± 0.0	0± 0.0	5± 7.0	0± 0.0
	MB12	0± 0.0	0± 0.0	0± 0.0	0± 0.0	38± 7.5	42± 9.0	0± 0.0	0± 0.0	6± 10.8	0± 0.0
	OG	1± 2.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	99± 2.2	0± 0.0	0± 0.0	0± 0.0
	STN	16± 6.5	0± 0.0	6± 8.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0	79± 18.5	1± 2.2	0± 0.0
	SH	0± 0.0	4± 4.1	2± 4.4	0± 0.0	9± 6.5	9± 6.5	1± 2.2	0± 0.0	86± 6.5	0± 0.0
	UTSA	0± 0.0	6± 6.5	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	2± 2.7	96± 4.1

In Table 4.16, the results show that the MB and MB12 image classes cannot be separated by the EM multi-class system since their classification accuracies are of mean values 51% and 42%, respectively. The results show that a MB stego image for testing has a 38% and 9% probability of being misclassified as MB12 and SH, respectively. On the other hand, a MB12 stego image for testing has a 49% and 9% probability of being misclassified as MB and SH, respectively. EM performs best in identifying JPH, JS, OG

and UTSA image classes with classification accuracies  $\geq 90\%$ . EM performs fairly well in identifying Clean, F5, STN and SH image classes with classification accuracies between 75% to 89%.

#### **4.3.2 $k$ -Nearest Neighbors ( $k$ -NN)**

Table 4.17 shows the classification accuracy from 5-fold cross validation when performing multi-class classification using  $k$ -nearest neighbors. The feature improvement methods and classification parameters used in  $k$ -nearest neighbors are listed in the following, in which the combination of parameters provides the highest classification accuracy.

- The data is normalized using min-max normalization;
- Fisher's linear discriminant is used for ranking the features with the top 34 out of 526 features selected;
- The number of nearest neighbors are determined experimentally based on classification accuracy with  $k = 5$ .

Table 4.17. Classification accuracy for 10-class  $k$ -NN classifier.

		Actual									
Predicted		Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
	Clean	78± 7.5	0± 0.0	2± 2.7	0± 0.0	1± 2.2	0± 0.0	1± 2.2	30± 15.4	1± 2.2	0± 0.0
	F5	1± 2.2	95± 6.1	1± 2.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2	6± 6.5
	JPH	0± 0.0	1± 2.2	92± 2.7	0± 0.0	0± 0.0	0± 0.0	0± 0.0	5± 8.6	0± 0.0	0± 0.0
	JS	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	MB	0± 0.0	0± 0.0	0± 0.0	0± 0.0	54± 6.5	52± 9.0	0± 0.0	0± 0.0	7± 10.9	0± 0.0
	MB12	0± 0.0	0± 0.0	0± 0.0	0± 0.0	38± 4.4	47± 9.7	0± 0.0	0± 0.0	3± 4.4	0± 0.0
	OG	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	99± 2.2	0± 0.0	1± 2.2	0± 0.0
	STN	21± 8.2	1± 2.2	5± 3.5	0± 0.0	0± 0.0	0± 0.0	0± 0.0	65± 11.1	1± 2.2	0± 0.0
	SH	0± 0.0	0± 0.0	0± 0.0	0± 0.0	7± 5.7	1± 2.2	0± 0.0	0± 0.0	86± 10.8	0± 0.0
	UTSA	0± 0.0	3± 2.7	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	94± 6.5

In Table 4.17, the results show that the MB and MB12 image classes cannot be separated by the  $k$ -NN multi-class system since their classification accuracies are of mean values 54% and 47%, respectively. This indicates that a MB stego image for testing has a 38% and 7% probability of being misclassified as MB12 and SH, respectively. On the other hand, a MB12 stego image for testing has a 52% probability of being misclassified as MB. In addition,  $k$ -NN barely does better than a coin toss in classifying STN with a classification accuracy of 65% with a 30% probability of misclassifying STN as Clean.  $k$ -NN performs best in identifying F5, JPH, JS, OG and UTSA image classes with classification accuracies  $\geq 90\%$ .  $k$ -NN performs fairly well in identifying Clean and SH image classes with classification accuracies between 75% to 89%. When comparing Table 4.17 with Table 4.16, both methods appear to misclassify MB and MB12. This is in large part due to the features being used while two different feature ranking methods are used, i.e., expectation maximization uses Bhattacharyya feature ranking with 34 features and  $k$ -NN uses Fisher's linear discriminant with 34 features, 30 of the 34 feature are the same in both.

### 4.3.3 Probabilistic Neural Networks (PNN)

Table 4.18 shows the classification accuracy from 5-fold cross validation when performing multi-class classification using PNN. The feature improvement methods and classification parameters used in probabilistic neural networks are listed in the following, in which the combination of parameters provides the highest classification accuracy.

- The data is normalized using Z-score normalization;
- The feature ranking is conducted using signal-to-noise ratio with the top 58 out of 526 features selected;
- Spread parameter  $\sigma = 0.24$ .

Table 4.18. Classification accuracy for 10-class PNN classifier.

		Actual									
Predicted		Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
	Clean	84± 5.4	0± 0.0	0± 0.0	0± 0.0	1± 2.2	0± 0.0	1± 2.2	53± 18.2	2± 2.7	0± 0.0
	F5	0± 0.0	99± 2.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2	0± 0.0
	JPH	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2	0± 0.0	0± 0.0
	JS	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	MB	0± 0.0	0± 0.0	0± 0.0	0± 0.0	57± 9.0	42± 6.7	0± 0.0	0± 0.0	5± 7.0	0± 0.0
	MB12	0± 0.0	0± 0.0	0± 0.0	0± 0.0	37± 7.5	58± 6.7	0± 0.0	0± 0.0	1± 2.2	0± 0.0
	OG	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	98± 2.7	0± 0.0	0± 0.0	0± 0.0
	STN	16± 5.4	0± 0.0	0± 0.0	0± 0.0	1± 2.2	0± 0.0	0± 0.0	45± 15.8	0± 0.0	0± 0.0
	SH	0± 0.0	1± 2.2	0± 0.0	0± 0.0	4± 6.5	0± 0.0	1± 2.2	1± 2.2	91± 7.4	0± 0.0
	UTSA	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	100± 0.0

In Table 4.18, the results show that the MB, MB12 and STN image classes cannot be separated by the PNN multi-class system since their classification accuracies are of mean values 57%, 58% and 45%, respectively. Other than EM and  $k$ -NN in Table 4.16 and 4.17,



PNN classifies five stego methods, F5, JPH, JS, OG and UTSA, with a 98% classification accuracy or better; however, it fails to separate STN from Clean and MB from MB12.

#### 4.3.4 Parzen window

Table 4.19 shows the classification accuracy from 5-fold cross validation when performing multi-class classification using Parzen window. The feature improvement methods and classification parameters used in Parzen window are listed in the following, in which the combination of parameters provides higher classification accuracy.

- The data is normalized using Z-score normalization;
- Fisher's linear discriminant is used for ranking the features with the top 36 out of 526 features selected;
- Window width  $\sigma = 0.85$ .

Table 4.19. Classification accuracy for 10-class Parzen window classifier.

		Actual									
Predicted		Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
	Clean	82± 9.0	0± 0.0	4± 4.1	0± 0.0	0± 0.0	0± 0.0	1± 2.2	30± 28.9	0± 0.0	0± 0.0
	F5	0± 0.0	99± 2.2	1± 2.2	0± 0.0	1± 2.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0	4± 4.1
	JPH	0± 0.0	0± 0.0	90± 6.1	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	JS	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	MB	0± 0.0	0± 0.0	0± 0.0	0± 0.0	57± 14.5	53± 15.2	0± 0.0	0± 0.0	1± 2.2	0± 0.0
	MB12	0± 0.0	0± 0.0	0± 0.0	0± 0.0	33± 9.7	42± 10.3	0± 0.0	0± 0.0	1± 2.2	0± 0.0
	OG	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	99± 2.2	0± 0.0	0± 0.0	0± 0.0
	STN	18± 9.0	0± 0.0	5± 6.1	0± 0.0	1± 2.2	0± 0.0	0± 0.0	70± 28.9	2± 2.7	0± 0.0
	SH	0± 0.0	0± 0.0	0± 0.0	0± 0.0	8± 10.3	5± 7.0	0± 0.0	0± 0.0	96± 4.1	0± 0.0
	UTSA	0± 0.0	1± 2.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	96± 4.1

In Table 4.19, the results show that Parzen window method is able to classify F5, JPH, JS, OG, SH, and UTSA with a 90% classification accuracy or better. Although it fails to separate STN from Clean, the classification accuracy using Parzen window instead of  $k$ -NN and PNN improves to 70%. As compared to Table 4.16, the Parzen window method performs better on SH with a 96% classification accuracy versus 86% in EM.

#### **4.3.5 Kernel Fisher's Discriminant (KFD) with Multi-class Tree**

Table 4.20 shows the classification accuracy from 5-fold cross validation when performing multi-class classification using KFD. The feature improvement methods and classification parameters used in kernel Fisher's discriminant are listed in the following, in which the combination of parameters provides the highest classification accuracy.

- The data is normalized using Z-score normalization;
- The feature ranking at each of the nodes is conducted using kernel feature ranking;
- The nodes correspond to Figure 4.2 where the top 50 features are used for classification in node A (i.e., classes 1 2 3 4 5 6 7 8 9 and 10), the top 46 features selected for node B, the top 34 features for node D, the top 24 features for node G, the top 36 features for node E, the top 32 features selected for node H, the top 26 features selected for node I, the top 31 features selected for node C, the top 25 features selected for node F;

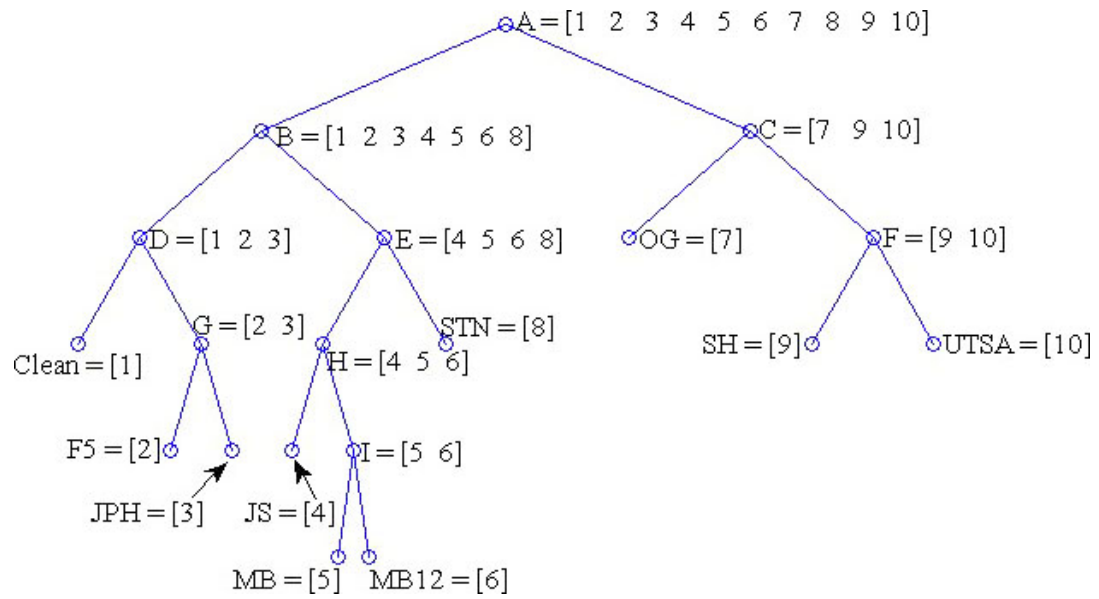


Figure 4.2. Decision tree for a 10-class classification problem with 10 leaf nodes, 9 parent nodes and a maximum depth of 6.

- The kernel used is the radial basis function with the normalizing constant  $\hat{C} = 12$  and  $\sigma = 3$ .

Table 4.20. Classification accuracy for 10-class KFD classifier.

		Actual									
Predicted		Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
	Clean	78± 5.7	0± 0.0	1± 2.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0	20± 12.7	0± 0.0	0± 0.0
	F5	0± 0.0	94± 6.5	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2	2± 2.7
	JPH	2± 2.7	0± 0.0	92± 4.4	0± 0.0	0± 0.0	0± 0.0	0± 0.0	2± 2.7	0± 0.0	0± 0.0
	JS	0± 0.0	0± 0.0	0± 0.0	94± 6.5	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	MB	0± 0.0	1± 2.2	0± 0.0	0± 0.0	54± 2.2	40± 10.6	2± 2.7	0± 0.0	8± 7.5	1± 2.2
	MB12	0± 0.0	0± 0.0	0± 0.0	0± 0.0	40± 3.5	59± 10.8	0± 0.0	0± 0.0	1± 2.2	0± 0.0
	OG	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	98± 2.7	0± 0.0	0± 0.0	0± 0.0
	STN	19± 5.4	1± 2.2	0± 0.0	5± 7.0	1± 2.2	0± 0.0	0± 0.0	78± 14.4	0± 0.0	0± 0.0
	SH	1± 2.2	0± 0.0	7± 4.4	1± 2.2	5± 5.0	1± 2.2 0.0	0± 0.0	0± 0.0	90± 7.9	0± 0.0
	UTSA	0± 0.0	4± 4.1	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	97± 2.7

In Table 4.20, the results show that using KFD is able to classify F5, JPH, JS, OG, SH and UTSA with a 90% classification accuracy or better. Comparing Table 4.16 to Table 4.19, KFD might not have perfect classification accuracies on certain methods, however, it performs better on average for all the image classes.

#### 4.3.6 Support Vector Machines (SVM) with Multi-class Tree

Table 4.21 shows the classification accuracy from 5-fold cross validation when performing multi-class classification using SVM.

The feature improvement methods and classification parameters used in support vector machines with multi-class tree are listed in the following, in which the combination of parameters provides the highest classification accuracy.

- The data is normalized using Z-score normalization;

- The feature ranking at each of the nodes was conducted using kernel feature ranking;
- The nodes correspond to Figure 4.3 with the top 90 features selected for node A, the top 44 features selected for node B, the top 46 features for node D, the top 21 features for node G, the top 63 features for node E, the top 48 features selected for node H, the top 19 features selected for node I, the top 46 features selected for node C, the top 22 features selected for node F;

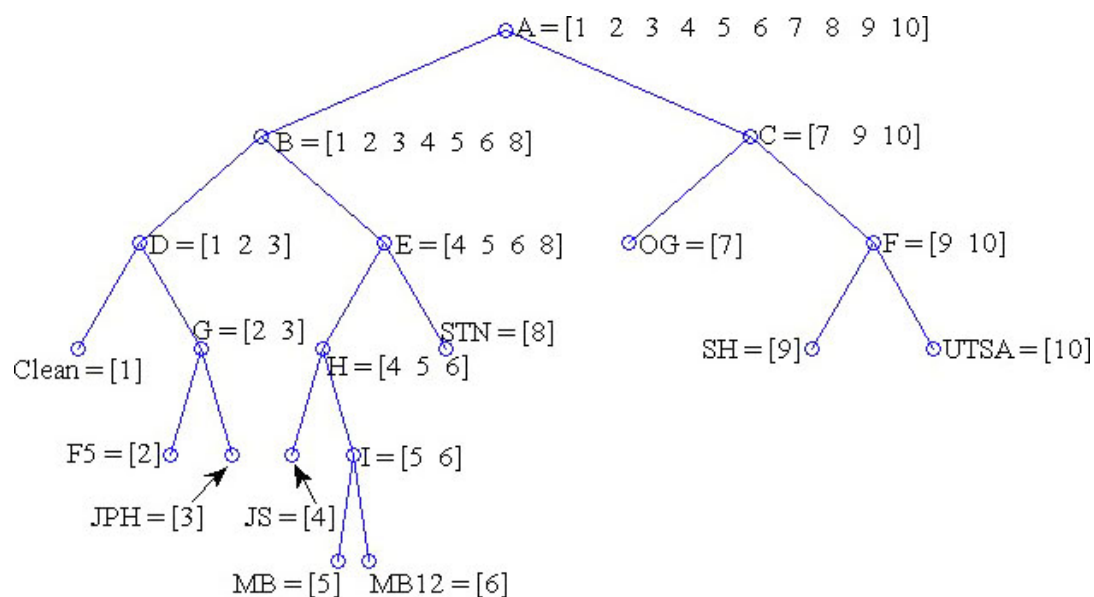


Figure 4.3. Decision tree for a 10-class classification problem with 10 leaf nodes, 9 parent nodes and a maximum depth of 6.

- The kernel used was the radial basis function, the normalizing constant  $\hat{C} = 6$ , and  $\sigma = 3$ .

Table 4.21. Classification accuracy for 10-class SVM classifier.

		Actual									
Predicted		Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
	Clean	86± 4.1	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2	17± 16.0	1± 2.2	0± 0.0
	F5	1± 2.2	95± 3.5	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2
	JPH	0± 0.0	1± 2.2	92± 4.4	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	JS	1± 2.2	4± 2.2	6± 6.5	100± 0.0	0± 0.0	0± 0.0	2± 2.7	0± 0.0	2± 4.4	1± 2.2
	MB	0± 0.0	0± 0.0	0± 0.0	0± 0.0	53± 7.5	43± 12.0	0± 0.0	0± 0.0	1± 2.2	0± 0.0
	MB12	0± 0.0	0± 0.0	0± 0.0	0± 0.0	46± 8.2	56± 13.8	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	OG	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	97± 2.7	0± 0.0	0± 0.0	0± 0.0
	STN	11± 2.2	0± 0.0	2± 2.7	0± 0.0	0± 0.0	0± 0.0	0± 0.0	82± 16.0	0± 0.0	0± 0.0
	SH	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2	1± 2.2	0± 0.0	1± 2.2	96± 4.1	0± 0.0
	UTSA	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	98± 2.7

In Table 4.21, the results show that the MB and MB12 image classes cannot be separated by the SVM multi-class system since their classification accuracies are of mean values 53% and 56%, respectively. However, Table 4.21 shows that SVM with multi-class tree performs better in general on other image classes when comparing to other classifiers from Table 4.16 to Table 4.20. For instance, Clean has a classification accuracy of 86% and STN has a classification accuracy of 82% which are both larger than the other five multi-class classifiers.

#### 4.3.7 StegoWatch

Table 4.22 shows the classification accuracy from 5-fold cross validation when performing multi-class classification using StegoWatch. Observe from Table 4.22 that StegoWatch clearly targets the identification of F5 embedding method above all others. For this tool the results are returned as either *H*, *M* or *L* for a high, medium or low stego detection level. If the image is clean an *OK* is returned indicating that the image is clean. For the image data set being analyzed in this research StegoWatch also returns a

comment indicating that *F5* has been identified. For this tool three classes are assigned. In the event an *H* or *M* is returned the image is considered as being stego, if an *L* or *OK* is returned the image is labeled as clean and if the comment indicates that *F5* was identified then *F5* is the class label.

Table 4.22. Classification accuracy for StegoWatch detection system.

	Actual			
		Clean	F5	Stego
Predicted	Clean	51± 6.9	0± 0.0	48± 12.4
	F5	0± 0.0	100± 0.0	0± 0.0
	Stego	49± 6.9	0± 0.0	52± 12.4

In Table 4.22, the results show the classification accuracies on Clean, F5 and all of the other (Stego) image classes. Except F5, the other image classes cannot be separated by the multi-class system since their classification accuracies are around 50%.

#### 4.3.8 Summary of Steganalysis Multi-Class results

Table 4.23 summarizes the classification accuracies of the seven multi-class classifiers that were examined in this chapter. Since StegoWatch is clearly specialized in identifying F5, it will not be included in the comparison performed in the proceeding analysis of identifying the multi-class classifier that targets specific embedding methods. However, for completeness the StegoWatch classification accuracy is still depicted in Table 4.23. Statistical significance comparing the best of the true multi-class classifiers (i.e., EM, *k*-NN, Parzen and PNN) with the best of the tree structure multi-class classifiers (i.e., KFD and SVM) is conducted using a *t*-test and shown in Table 4.24. The best classifiers according to the defined grouping are indicated in bold in Table 4.23, which are then used in the statistical comparison in Table 4.24. Based on overall classification accuracy in Table 4.23, the best individual system appears to be SVM with used in a multi-class tree structure.

Table 4.23. Classification accuracy for multi-class detection system.

	Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA	CA
EM	83± 5.7	88± 9.0	90± 7.0	100± 0.0	51± 11.9	42± 9.0	<b>99±</b> <b>2.2</b>	<b>79±</b> <b>18.5</b>	86± 6.5	96± 4.1	81.4± 20.5
<i>k</i> -NN	78± 7.5	95± 6.1	92± 2.7	100± 0.0	54± 6.5	47± 9.7	99± 2.2	65± 11.1	86± 10.8	94± 6.5	81± 19.6
Parzen	82± 9.0	99± 2.2	90± 6.1	100± 0.0	57± 14.4	42± 10.3	99± 2.2	70± 28.9	<b>96±</b> <b>4.1</b>	96± 4.1	83.1± 22.0
PNN	<b>84±</b> <b>5.4</b>	<b>99±</b> <b>2.2</b>	<b>100±</b> <b>0.0</b>	<b>100±</b> <b>0.0</b>	<b>57±</b> <b>9.0</b>	<b>58±</b> <b>6.7</b>	98± 2.7	45± 15.8	91± 7.4	<b>100±</b> <b>0.0</b>	<b>83.2±</b> <b>21.5</b>
KFD	78± 5.7	94± 6.5	92± 4.4	94± 6.5	<b>54±</b> <b>2.2</b>	<b>59±</b> <b>10.8</b>	<b>98±</b> <b>2.7</b>	78± 14.4	90± 7.9	97± 2.7	83.4± 16.5
SVM	<b>86±</b> <b>4.1</b>	<b>95±</b> <b>3.5</b>	<b>92±</b> <b>4.4</b>	<b>100±</b> <b>0.0</b>	53± 7.5	56± 13.8	97± 2.7	<b>82±</b> <b>16.0</b>	<b>96±</b> <b>4.1</b>	<b>98±</b> <b>2.7</b>	<b>85.5±</b> <b>17.9</b>
Stego Watch	51± 6.9	100± 0.0	52± 12.4								67.7± 10.6

Table 4.24. t-test: paired two samples for means.

<i>t</i> -Critical Two-Tail; $t_{4, 0.975} = 2.776$ , $n1 = n2 = 5$ (corresponding to 5-fold), $\alpha = 0.05$										
Image Class	Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
Classifier Comparison	PNN vs. SVM	PNN vs. SVM	PNN vs. SVM	PNN vs. SVM	PNN vs. KFD	PNN vs. KFD	EM vs. KFD	EM vs. SVM	Parzen vs. SVM	PNN vs. SVM
<i>t</i> -Stat	0.49	2.13	4.0	0.0	0.6	0.27	1.0	0.8	0.0	1.63
Statistically Significant	No	No	Yes	No	No	No	No	No	No	No

As can be seen in Table 4.24, only the JPH image class shows significant difference in the mean between the best result from a multi-class classifier and the best result from the multi-class tree results. The difference in the mean for the rest of the image classes are not statistically significant. In addition, the results from the individual tables show that the various classifiers each have individual strengths when identifying the various embedding methods. To take advantage of the individual classifiers the next section uses fusion to combine the seven detection systems.



## 4.4 Fusion

From Section 4.3, no advantage of single multi-class classifiers has been shown; instead each of the multi-class classifier has individual strength. To make use of the individual strengths of the classifiers, the three fusion techniques presented in Section 3.4 are used and the results shown in this section. For AdaBoost and Bayesian network fusion the class labels are fused as discrete values. For the commercial tool, StegoWatch, the results are returned as either *L*, *OK* indicating a clean class label, or *F5*. The PNN fusion however, requires that the results feed into the fusion system be posterior probabilities. To solve this problem for the commercial tool two inputs are used, clean or *F5*. If the result returned is clean, *L* or *OK*, a posterior probability of 0.9 is assigned and the *F5* input is assigned a 0.01. If the result returned is *F5* a posterior probability of 0.9 is assigned and the *clean* input is assigned a 0.01. For the 10-class classifiers probabilities are assigned to each of the 10 classes but only 9 of the 10 labels from each of the classifiers is used to train the fusion system, allowing proper training of the weights.

### 4.4.1 AdaBoost

The results for AdaBoost fusion are shown in Table 4.25. Fusing the seven multi-class systems results in detecting the Clean and Steganos (STN) classes as well as the Model-based and Model-based version 1.2 are improved.

Table 4.25. Classification accuracy for AdaBoost fusion.

		Actual									
Predicted		Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
	Clean	86± 4.1	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	16± 19.8	0± 0.0	0± 0.0
	F5	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	JPH	1± 2.2	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	JS	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	MB	0± 0.0	0± 0.0	0± 0.0	0± 0.0	63± 7.5	37± 10.3	0± 0.0	0± 0.0	2± 4.4	0± 0.0
	MB12	0± 0.0	0± 0.0	0± 0.0	0± 0.0	34± 6.5	61± 8.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	OG	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0
	STN	13± 4.4	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	84± 19.8	1± 2.2	0± 0.0
	SH	0± 0.0	0± 0.0	0± 0.0	0± 0.0	3± 2.7	2± 2.7	0± 0.0	0± 0.0	97± 4.1	0± 0.0
	UTSA	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	100± 0.0

With AdaBoost, the classification accuracies of MB and MB12 are 63% and 61%. Comparing to the best individual classifier as shown in Table 4.23, AdaBoost actually improves the classification capability of these two image classes.

#### 4.4.2 Bayes Fusion

The results for Bayes fusion are shown in Table 4.26. Similar to AdaBoost, this fusion method also improved the classification accuracy between the Clean and Steganos (STN) classes as well as the Model-based and Model-based version 1.2.

Table 4.26. Classification accuracy for Bayes fusion.

		Actual									
Predicted		Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
	Clean	89± 2.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	13± 16.4	1± 2.2	0± 0.0
	F5	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2	0± 0.0
	JPH	1± 2.2	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	JS	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2	0± 0.0
	MB	0± 0.0	0± 0.0	0± 0.0	0± 0.0	63± 8.3	37± 8.3	0± 0.0	0± 0.0	1± 2.2	0± 0.0
	MB12	0± 0.0	0± 0.0	0± 0.0	0± 0.0	34± 10.8	63± 8.3	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	OG	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0
	STN	10± 3.5	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	86± 15.5	0± 0.0	0± 0.0
	SH	0± 0.0	0± 0.0	0± 0.0	0± 0.0	3± 2.7	0± 0.0	0± 0.0	1± 2.2	96± 4.1	0± 0.0
	UTSA	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	100± 0.0

As compared to the best individual classifier shown in Table 4.23, this fusion technique actually at the very least maintains or improves every classification accuracy for all the image classes.

#### 4.4.3 PNN Fusion

The results for PNN fusion are shown in Table 4.27. Similar to the previous two fusion systems, the classification accuracy between the Clean and STN classes as well as the MB and MB12 are also improved.

Table 4.27. Classification accuracy for PNN fusion.

		Actual									
Predicted		Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
	Clean	88± 2.7	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	15± 17.6	1± 2.2	0± 0.0
	F5	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	JPH	1± 2.2	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	1± 2.2	0± 0.0	0± 0.0
	JS	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	3± 4.4	0± 0.0
	MB	0± 0.0	0± 0.0	0± 0.0	0± 0.0	56± 13.8	34± 10.8	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	MB12	0± 0.0	0± 0.0	0± 0.0	0± 0.0	40± 14.5	63± 8.3	0± 0.0	0± 0.0	0± 0.0	0± 0.0
	OG	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	100± 0.0	0± 0.0	0± 0.0	0± 0.0
	STN	11± 2.2	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	84± 19.8	0± 0.0	0± 0.0
	SH	0± 0.0	0± 0.0	0± 0.0	0± 0.0	4± 2.2	3± 2.7	0± 0.0	0± 0.0	96± 4.1	0± 0.0
	UTSA	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	0± 0.0	100± 0.0

With PNN fusion, the classification accuracies of Clean, F5, MB, MB12 and STN image classes are improved as compared to the best individual classifier shown in Table 4.23.

#### 4.4.4 Summary of Multi-class Steganalysis Fusion Techniques

Table 4.28 shows the classification accuracy of the fusion methods and the best individual classifier, i.e., SVM with multi-class tree. It is chosen as the best individual classifier purely based on the overall classification accuracy of 85.5% (Table 4.23). Table 4.28 shows that by using any fusion technique classification accuracy improves over the best individual classifier. It shows that each of the fusion methods has an equal or higher classification accuracy over any of the best individual classifiers results.

Table 4.28. Classification accuracy comparisons between the best individual results and the three fusion methods.

	Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA	CA
SVM	86± 4.1	95± 3.5	92± 4.4	100± 0.0	53± 7.5	56± 13.8	97± 2.7	82± 16.0	96± 4.1	98± 2.7	85.5± 17.3
AdaBoost Fusion	86± 4.1	100± 0.0	100± 0.0	100± 0.0	63± 7.5	61± 8.2	100± 0.0	84± 19.8	96± 4.1	100± 0.0	89± 15.3
Bayes Fusion	89± 2.2	100± 0.0	100± 0.0	100± 0.0	63± 7.5	63± 8.3	100± 0.0	86± 15.5	96± 4.1	100± 0.0	89.7± 15.3
PNN Fusion	89± 2.2	100± 0.0	100± 0.0	100± 0.0	65± 3.5	63± 8.3	100± 0.0	86± 15.5	96± 4.1	100± 0.0	89.9± 14.9

The three fusion techniques are equally valid choices for combining the individual multi-class classifier from Section 4.3. In Table 4.29 the  $t$ -test is performed between the PNN fusion (highest overall CA in the fusion methods examined) and the SVM multi-class classifier (highest overall CA of the individual multi-class classifier) to determine whether the difference in the means between these two methods is statistically significant. As noted in Table 4.29 the two methods show statistical differences for the F5, JPH, MB, and STN image classes.

Table 4.29.  $t$ -test: paired two samples for means of classification accuracy between PNN fusion and SVM.

$t$ -Critical Two-Tail; $t_{4, 0.975} = 2.776$ , $n1 = n2 = 5$ (corresponding to 5-fold), $\alpha = 0.05$										
Image Class	Clean	F5	JPH	JS	MB	MB12	OG	STN	SH	UTSA
$t$ -Stat	1.5	3.16	4	0.0	2.9	1.20	2.44	4	0.0	1.63
Statistically Significant	No	Yes	Yes	No	Yes	No	No	Yes	No	No

In this subsection it was shown that the fusion techniques have equal or greater classification accuracy over any of the individual classifiers. In addition to the statistical significance for certain image classes shown in Table 4.29, the fusion methods also show an increase in classification accuracy over any of the individual detection systems.

## 4.5 Summary

In this chapter, the DCT decomposition feature generation, kernel feature ranking, decision tree for multi-class classification and the fusion techniques have shown improvements in classification accuracy when determining the stego algorithm used to create a stego image. Results comparing the feature generation methods described in Section 3.2 show that the new features are able to distinguish between Steganos and the other classes while the wavelet feature generation method (Lyu and Farid, 2002) and DCT feature generation (Pevny and Fridrich, 2006) have shown difficulty in identifying Steganos. It has also been shown that by combining all of the feature generation methods, detection improves. Additionally, by performing feature ranking, detection results for the SVM classifier are improved. The third area of improvement is the development of a multi-class tree that is used with two-class KFD and SVM classifiers. The tree in this case is expanded by using a distance measure in the kernel space. While the classification tree shows promise, the results can additionally be improved through the use of a fusion technique. The fusion techniques use the strengths of each individual multi-class detection systems to better predict the embedding method. The *t*-test was used in this chapter to determine if the methods used to improve the classification of individual steganography methods are statistically significant. While no individual system showed to be statistically significant over any of the others, it is important to note that the real utility of the methods in this research lies in using each and every available detection system to improve the identification of steganography methods.

## **V. Conclusion and Recommendations**

This research demonstrated a steganalysis classification system that identifies the steganalysis embedding method in a given JPEG. The system includes feature preprocessing, feature extraction, feature ranking, classification and multi-class classification. The methodology, analyses and experimental results with system validation have been described and demonstrated in Chapters 3 and 4. The results show the statistical difference of the proposed classification system which is essential for such a system. This chapter summarizes the research conducted and also provides the advantage and disadvantage of this steganalysis classification system. Further research can be applied not only based on the constraints and limitations when developing the system but by its application to other areas.

### **5.1 Application of Results**

This research proposes a novel multi-class detection system applied to the problem of steganalysis. The complete system is shown in Figure 5.1. With the input including the clean and stego image sets using the embedding methods either F5, JP Hide, JSteg, Model-base, Model-based Version 1.2, OutGuess, Steganos, StegHide or UTSA, features are generated from each image and each feature set is assigned a class label identifying the embedding method used. Three components, Multi-class Detection for EM/k-NN/Parzen/PNN, Multi-class Detection for KFD/SVM, and Commercial Detection Systems are integrated as an 8 multi-class system. The components analyze the raw features and their results are fused in order to assign a final class label.

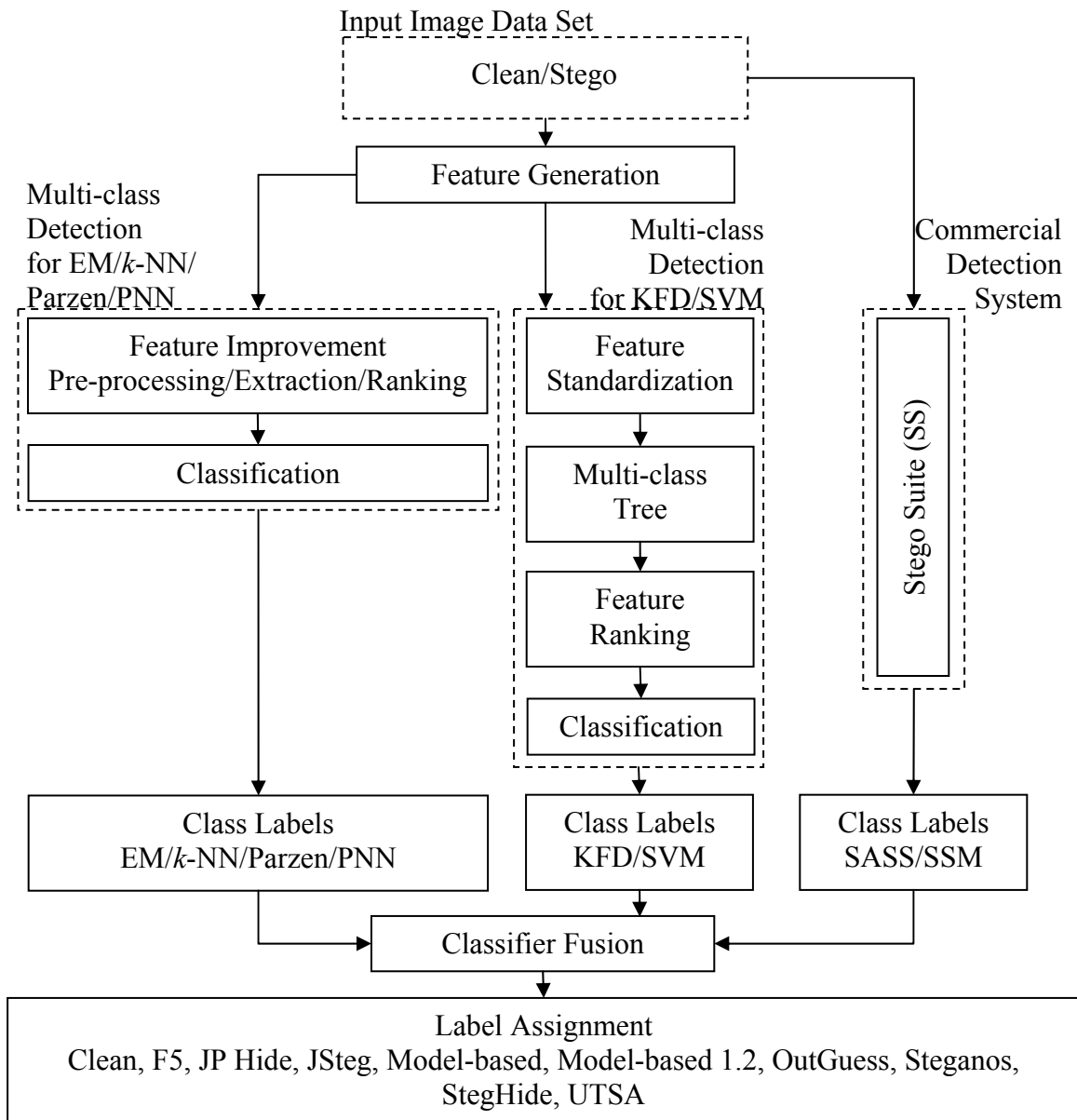


Figure 5.17. Detection system.

The multiclass fusion system developed in this dissertation provides the steganalyst the ability to use all available tools from both the research community and the commercial industry to be combined in one detection system. For certain law enforcement agencies that use detection methods not available to outside agencies (i.e., ILook Investigator, Detica's Inforenz Forager, SecureStego (AFRL) and WetStone's Stego Suite with added



applications) the fusion method provides a means to incorporate the class labels of any tools necessary for identifying stego methods.

As shown in Section 4.2, the classification accuracy for each feature generation method, Wavelet features, DCT features, and DCT decomposition features, with feature ranking is increased. Using SVM-kernel feature ranking, the wavelet feature generation method has an average increase of 7% in classification accuracy when feature ranking is not used. The DCT features show an increase in classification accuracy on Clean vs. MB and Clean vs. STN with 0.5% and 1.6% respectively when using SVM-kernel feature ranking. The DCT decomposition features have an average increase of 2.5% classification accuracy in comparison to not using feature ranking. Between the three feature generation methods shown in Table 4.14, while the DCT features are able to classify most of the stego methods accurately, the proposed DCT decomposition features has an increase in classification accuracy of 10% on Steganos (STN) over the DCT features. This allows the combination of features with feature ranking to separate the Clean vs. all the Stego image classes as shown in Table 4.14 with perfect classification accuracy. By creating a multi-class classifier using the decision tree in Section 3.3 the proposed SVM with tree structure has an increase of classification accuracy of 2.3% over PNN as shown in Table 4.23. Furthermore, with the use of fusion techniques, the overall classification accuracy of the best individual classifier increases from 85.5% to 89% (see Table 4.28). AdaBoost, Bayes, and PNN fusion obtain the classification accuracy of 89%, 89.7% and 89.9%, respectively.

## **5.2 Recommendations for Future Work**

The need to extract the hidden information is necessary for law enforcement to build a criminal case if it is to hold up in court. This problem of extraction leads to an intermediate step of identifying the embedding method used to create the stego file. Another problem exist for the steganalyst in which several tools are available to detect whether an image is clean or stego. The multi-class classification system developed needs

to be expanded to identify more steganography algorithms. This expansion includes the ability to identify embedding techniques other than DCT coefficients, such as header analysis, and spatial embedding methods. Additionally, the techniques should be extended to classify JPEG images with different image sizes, quality factors, and camera types. Also, the difficulties are currently not well understood when it comes to images taken from entirely different scenes or computer generated.

In header analysis, stego images created by methods such as F5 (Westfeld, 2001; 2003) and Invisible Secrets (2008) manipulate the header of an image in different ways. By analyzing the header of an image various embedding methods used to manipulate in headers can be identified. In both StegAlyzerSS and StegoWatch the default header for F5 was identified, however, for Invisible Secretes neither of these detectors is capable of identifying this method. The work by Pevny and Fridrich (2006) analyze various image sizes, quality factors and camera types and are supported by the Air Force Research Laboratory. Their research of these categories can be used in conjunction with the presented detection system to improve the identification of the embedding methods used.

Another area of improving stego method identification is to separate images into different scenes, e.g., images of an aircraft with blue sky should not be in the same data set as images of an individual smiling. By separating images into the various categories the problems encountered in Section 4.4 with outliers can be avoided. The number of varying scenes is a research topic that has been extensively studied can be incorporated into the work provided in this document.

### **5.3 Conclusion**

This dissertation proposes a novel multi-class classification system on steganalysis. This research of developing the steganalysis classification system has contribution in four advancements: feature generation, feature ranking, multi-class for kernel Fisher's discriminant as well as support vector machines and fusion of detection systems. First,

the new features are generated from the frequency bands and directions of the Discrete Cosine Transform (DCT) coefficients of JPEG images. The second improvement is a new feature ranking method. From the original input feature set, it selects a subset of features specifically designed for the kernel Fisher's discriminant (KFD) and the support vector machines (SVM). The third improvement is a multi-class classification tree designed for the KFD and SVM classifiers. The final contribution of this steganalysis classification system is a multi-class classifier fusion with classifier selection and fusion. The complete system performance shows an increase in classification accuracy of 10% as well as being statistically different from existing detection techniques. In addition, this system provides a solution for identifying steganographic fingerprints as well as the ability to include future multi-class classification tools.

## Appendix A

Table A1. Number of features used from each of the feature generation method in feature combination (Table 4.15 Detail for Clean vs. F5, Clean vs. JPH and Clean vs. JS).

Method ↓	Total No. of Features →	Clean vs. F5	Clean vs. JPH	Clean vs. JS
Wavelets	No. of Features	11	18	5
	(Number of Features) Description of Feature: Statistic Calculated, Orientation, Subband Scale (level), either Wavelet Coefficients or Log Error	(1) Variance, Horizontal, 1, Log Error	(1) Mean, Diagonal, 1, Log Error (1) Variance, Horizontal, 1, Log Error (1) Kurtosis, Vertical, 1, Log Error	(0)
DCT	No. of Features	5	12	4
	(Number of Features) Description of Feature: either Global Histogram AC Histogram Dual Histogram Variation Blockiness Co-occurrence Markov	(1) AC Histogram (3) Dual Histogram (1) Markov	(3) AC Histogram (3) Dual Histogram (5) Co-occurrence (1) Markov	(3) Co-occurrence (1) Markov
DCT Decomp	No. of Features	5	3	1
	(Number of Features) Description of Feature: Statistic Calculated, Orientation, Frequency, either Regression Mean Difference Avg. Neighboring Shifted Right Shifted Down Shifted Diagonal	(1) Variance, Diagonal, Low, Regression (1) Variance, Horizontal, Low, Avg. Neighboring (1) Variance, Horizontal, Low, Shifted Diagonal (1) Variance, Vertical, Low, Shifted Diagonal (1) Entropy, Vertical, Low, Shifted Diagonal	(1) Variance, Diagonal, Low, Regression (1) Entropy, Horizontal, Low, Avg. Neighboring (1) Variance, Vertical, Low, Shifted Diagonal	(1) Variance, Diagonal, Medium, Shifted Diagonal

Table A2. Number of features used from each of the feature generation method in feature combination (Table 4.15 Detail for Clean vs. MB, Clean vs. MB1.2 and Clean vs. OG).

		Clean vs. MB	Clean vs. MB1.2	Clean vs. OG
Method ↓	Total No. of Features →	6	10	5
Wavelets	No. of Features	0	0	0
	(Number of Features) Description of Feature: Statistic Calculated, Orientation, Subband Scale (level), either Wavelet Coefficients or Log Error	(0)	(0)	(0)
DCT	No. of Features	5	5	4
	(Number of Features) Description of Feature: either Global Histogram AC Histogram Dual Histogram Variation Blockiness Co-occurrence Markov	(4) Co- occurrence (1) Markov	(2) Co- occurrence (3) Markov	(1) AC Histogram (3) Markov
DCT Decomp	No. of Features	1	5	1
	(Number of Features) Description of Feature: Statistic Calculated, Orientation, Frequency, either Regression Mean Difference Avg. Neighboring Shifted Right Shifted Down Shifted Diagonal	(1) Variance, Diagonal, Low, Regression	(1) Variance, Diagonal, Low, Regression (1) Variance, Horizontal, Low, Avg. Neighboring (1) Variance, Diagonal, Low, Mean Difference (1) Variance, Vertical, Low, Shifted Diagonal (1) Entropy, Vertical, Low, Shifted Diagonal	(1) Mean, Vertical, Medium, Regression

Table A3. Number of features used from each of the feature generation method in feature combination (Table 4.15 Detail for Clean vs. STN, Clean vs. SH and Clean vs. UTSA).

		Clean vs. STN	Clean vs. SH	Clean vs. UTSA
Method ↓	Total No. of Features →	15	7	5
Wavelets	No. of Features	2	0	0
	(Number of Features) Description of Feature: Statistic Calculated, Orientation, Subband Scale (level), either Wavelet Coefficients or Log Error	(1) Variance, horizontal subband at scale 1, log error (1) Mean, diagonal subband at scale 1	(0)	(0)
DCT	No. of Features	7	5	5
	(Number of Features) Description of Feature: either Global Histogram AC Histogram Dual Histogram Variation Blockiness Co-occurrence Markov	(1) Global histogram (1) AC histogram (3) Dual histogram (2) Co- occurrence	(4) Co-occurrence (1) Markov	(4) Co- occurrence (1) Markov
DCT Decomp	No. of Features	6	2	0
	(Number of Features) Description of Feature: Statistic Calculated, Orientation, Frequency, either Regression Mean Difference Avg. Neighboring Shifted Right Shifted Down Shifted Diagonal	(1) Variance, Diagonal, Low, Regression, (1) Variance, Horizontal, Low, Average Neighboring (1) Variance, Horizontal, Low, Shifted Diagonal (1) Variance, Vertical, Low, Shifted Diagonal (1) Variance, Diagonal, Low, Mean Difference (1) Entropy, Vertical, Low, Shifted Diagonal	(1) Variance, Horizontal, Low, Avg. Neighboring (1) Variance, Vertical, Low, Shifted Diagonal	(0)

## Bibliography

- Abdi, H. and Molin, P. (2007). Lilliefors test of normality. *Encyclopedia of Measurement and Statistics*, N.J. Salkind (Ed.), pp. 540-544, Thousand Oaks, CA.
- Addison, P. S. (2002). *The Illustrated Wavelet Transform Handbook, Introductory Theory and Applications in Science, Engineering, Medicine and Finance*. Bristol BS1 6Be, UK: IOP Publishing Ltd.
- Agaian, S., Schneider, E. and Cherukuri, R. (2006). Transform-domain Steganographic Algorithms Exploring Pixel Intensity Variations. *IEEE Signal Processing Society, 2006 International Conference on Multimedia & Expo*, Toronto, Canada, pp. 1-4.
- Astrowsky, B.H. (2000). STEGANOGRAPHY: Hidden Images, A New Challenge in the Fight Against Child Porn. *UPDATE*, Volume 13, Number 2, pp. 1-4, Retrieved June 3, 2008, <http://www.antichildporn.org/steganog.html>.
- Backbone Security. (2008). Steganography Analysis and Research Center. Retrieved June 3, 2008, <http://www.sarc-wv.com/safdb.aspx>.
- Barnett, V. and Lewis, T. (1994). *Outliers in Statistical Data 3rd Edition*. Chichester, England: John Wiley & Sons.
- Bauer K. W., Alsing, S. G., and Greene, K. A. (2000). Feature screening using signal-to-noise ratios. In *Neurocomputing*, Volume 31, pp. 29-44.
- Belue, L. M. (1992). *Multilayer Perceptrons for Classification*. University published master's thesis, AFIT/GOR/ENS/92M-02, Air Force Institute of Technology, Wright-Patterson AFB, OH.
- Belue, L. M. and Bauer, K. W. Jr., (1995). Determining input features for multilayered perceptrons. *Neurocomputing*, Volume 7, pp. 111-121.
- Bhattacharyya, A., (1943). On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, Number 35, pp. 99-110.
- Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York, NY: Oxford University Press Inc.

- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York, NY: Springer.
- Bonferroni, C. E. (1935). Il calcolo delle assicurazioni su gruppi di teste. *Studi in Onore del Professore Salvatore Ortu Carboni*, Rome, pp. 13-60.
- Bonferroni, C. E. (1936). Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, Volume 8, pp. 3-62.
- Brown, W. and Shepherd, B. J. (1995). *Graphics File Formats: Reference and Guide*. Greenwich, CT: Manning Publications.
- Burgers, C., (1998). A tutorial on Support Vector Machines for Pattern Recognition, *Data Mining and Knowledge Discovery*. Volume 2, Number 2, pp. 121-167.
- Burns, J. (2006). Image Formats. Retrieved March 25, 2006, [http://www.htmlgoodies.com/tutorials/web\\_graphics/article.php/3479931](http://www.htmlgoodies.com/tutorials/web_graphics/article.php/3479931).
- Celeux, G. and G. Govaert, (1995). Gaussian Parsimonious Clustering Models. *Pattern Recognition*, Volume 28, pp. 781-793.
- Chandramouli, R. (2002). Mathematical approach to steganalysis. *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, Volume 4675. International Society for Optical Engineering, San Jose, California, pp. 14-25.
- Chang, C.-C., and Lin, C.-J. (2001). LIBSVM: a library for support vector machines. Retrieved February 2006, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Cover, T. and Hart, P. E. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*. Volume 13, pp. 21-27.
- Cristianini, N., and Shawe-Taylor J., (2000). *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge CB2 2RU, UK: Cambridge University Press.
- Cristianini, N., and Shawe-Taylor J., (2004). *Kernel Methods for Pattern Analysis*. Cambridge CB2 2RU, UK: Cambridge University Press.
- Crammer, K. and Singer, Y. (2000). On the Learnability and Design of Output Codes for Multiclass Problems. *Computational Learning Theory*, pp. 35-46.



- Crammer, K. and Singer, Y. (2001). Ultraconservative Online Algorithms for Multiclass Problems. *14th Annual Conference on Computational Learning Theory, {COLT} 2001 and 5th European Conference on Computational Learning Theory*, Amsterdam, The Netherlands, Volume 2111, pp. 99-115.
- Davis, C.H. (1857). *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections, A Translation of Gauss's "Theoria Motus." With an Appendix*. Boston, MA: Little, Brown and Company. (Original work published 1809).
- Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, Volume 39, Number 1, pp.1-22.
- Dillon, W. R. and Goldstein, M. (1984). *Multivariate Analysis Method and Applications*. New York, NY: John Wiley & Sons, Inc.
- Draper, N. R. and Smith, H. (1998). *Applied Regression Analysis Third Edition*. New York, NY: Wiley Series in Probability and Statistics, John Wiley & Sons, Inc.
- Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. New York, NY: Wiley-Interscience Publications.
- Duda, R.O., Hart, P.E. and Stork, D.G. (2001). *Pattern Classification* (2nd. Ed.). New York, NY: John Wiley & Sons.
- Duin, R. P. W. (2002). The Combining Classifier: to Train or Not to Train. Editors R. Kasturi, D. Laurendeau, C. Suen (eds.), *ICPR16, Proceedings 16th International Conference on Pattern Recognition*, Volume 2, IEEE Computer Society Press, Los Alamitos, pp. 765-770.
- Duin, R. P. W. and Tax, D. M. J. (2000). Experiments with Classifier Combining Rules. in: J. Kittler, F. Roli (eds.), *Lecture Notes in Computer Science Multiple Classifier Systems (Proc. First International Workshop, MCS 2000, Cagliari, Italy, Volume 1857*, Springer, Berlin, pp. 16-29.
- Eibl, G. and Pfeiffer, K-P. (2005). Multiclass Boosting for Weak Classifiers. *Journal of Machine Learning Research* Volume 6, pp. 189-210.
- Elysium Ltd. (2004). Home of the JPEG committee. Retrieved February 20, 2005, <http://www.JPEG.org/>.

- Farid, H. (2002). Detecting Hidden Messages Using Higher-Order Statistical Models. *IEEE International Conference on Images Processing*, Volume 2, Rochester, NY, pp. 905-908.
- Fisher, R. A., (1936). The use of Multiple Measurements in Taxonomic Problems. *Proceedings of Annals of Eugenics*, Number 7, pp. 179-188.
- Fisher, R. A. (1943) A Theoretical Distribution for the Apparent Abundance of Different Species. *Journal of Animal Ecology*, Volume 12, pp. 54-58.
- Franc, V. and Hlavac, V. (2007). Statistical Pattern Recognition Toolbox. Retrieved October 30, 2007, <http://cmp.felk.cvut.cz/cmp/software/stprtool/index.html>.
- Freund, Y. and Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, Number 55, pp. 23-37.
- Freund, Y. and Schapire, R. E. (1999). A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, Volume 14, Issue 5, pp 771-780.
- Friedberg, S. H., Insel, A. J. and Spence, L. E. (1989). *Linear Algebra Second Edition*. Englewood Cliffs, NJ: Prentice Hall.
- Fridrich, J. (2004). Feature-Based Steganalysis for JPEG Images and its Implications for Future Design of Steganographic Schemes. *LNCS 6th Information Hiding Workshop*, Editor Fridrich, J., Volume 3200, Springer-Verlag, pp. 67-81.
- Friess, T-T., Cristianini, N. and Campbell, C., (1998). The Kernel-Adatron algorithm: a fast and simple learning procedure for support vector machines. *Proceedings On Machine Learning 15th International Conference*, pp. 188-196.
- Fu, D., Shi, Y. Q., Zou, D. and Xuan, G., (2006). JPEG Steganalysis Using Empirical Transition Matrix in Block DCT Domain. *2006 IEEE 8th Workshop on Multimedia Signal Processing*, Victoria, BC, Canada, pp.310-313.
- Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition Second Edition*. San Diego, CA: Academic Press, Inc.
- Gauss, C. F. (1809). *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientum*. Perthes and Besser, Hamburg.

- Goebel, K. and Yan, W. (2004). Choosing Classifiers for Decision Fusion. *The 7th International Conference on Information Fusion*, Stockholm, Sweden.
- Gonzalez, R. C. and Woods R. (1992). *Digital Image Processing*. Reading, MA: Addison-Wesley.
- Gonzalez, R. C. and Woods R. (2002). *Digital Image Processing, 2nd Edition*. Upper Saddle River, NJ: Prentice Hall.
- Gonzalez, R. C. and Woods R. (2007). *Digital Image Processing, 3rd Edition*. Upper Saddle River, NJ: Prentice Hall.
- Gonzalez, R. C., Woods, R. E. & Eddins, S. L. (2004). *Digital Image Processing Using Matlab*. Upper Saddle River, NJ: Prentice Hall.
- Guyon, I., (2007). Feature Selection with CLOP. Retrieved October 28, 2007 [http://clonet.com/isabelle/Projects/ETH/Feature\\_Selection\\_w\\_CLOP.html](http://clonet.com/isabelle/Projects/ETH/Feature_Selection_w_CLOP.html).
- Guyon, I., Gunn, S., Nikravesh, M. and Zadeh, L. A. (Eds.). (2006). *Feature Extraction, Foundations and Applications*. New York, NY: Springer.
- Guyon, I., Weston, J., Barnhill, S. and Vapnik, V., (2002). Gene Selection for Cancer Classification Using Support Vector Machines. *Machine Learning*, Volume 46, pp. 389-422.
- Hetzl, S., (2003). StegHide. Retrieved July 20, 2005 <http://steghide.sourceforge.net/>.
- Higgins, K. J. (2007). Research Shows Image-Based Treat on the Rise. *Dark Reading*, Retrieved October 25, 2007, [http://www.darkreading.com/document.asp?doc\\_id=136702&f\\_src=darkreading](http://www.darkreading.com/document.asp?doc_id=136702&f_src=darkreading).
- Hoeting, J., Madigan, D., Raftery, A. and Volinsky, C. (1999). *Statistical Science*, Volume 14, Number 4, pp. 382-401.
- Hogg, R. V. and Tanis, E. A. (1993). *Probability and Statistical Inference*. New York, NY: MacMillan Publishing Company.
- Hotelling, H., (1933). Analysis of a complex of statistical variables into principal components, *Journal of Educational Psychology*, Number 24, pp. 417-441.

- Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2006). A practical guide to support vector classification. Department of Computer Science and Information Engineering National Taiwan, Retrieved March 15, 2006  
<http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>.
- Hsu, C.-W., Chang, C.-C. and Lin, C.-J. (2002). A comparison on methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, Volume 13, pp. 415-425.
- Hsu, C.-W. and Lin, C.-J. (2002). A Simple Decomposition Method for Support Vector Machines. *Machine Learning*, Volume, 46 pp. 291-314.
- Independent JPEG Group (1998). Independent JPEG Group. Retrieved February 20, 2005, <http://www.ijg.org/>.
- Jaakola, T. S. and Haussler, D. (1998). Exploring Generative Models in Discriminative Classifiers. *Advances in Neural Information Processing Systems*. Kearns, M.S., Soll, S. A. and Cohn, D. A. (Eds.), Volume 11, Cambridge, MA: MIT Press.
- Jackson, J. T. (2003). Targeting Covert Messages: A Unique Approach for Detecting Novel Steganography. University published master's thesis, AFIT/GCE/ENG/03-02, Air Force Institute of Technology, Wright-Patterson AFB OH.
- Jaeger, S. (2004). Informational Classifier Fusion. *Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004*, Volume 1, pp. 216-219.
- Jain, A. K., Duin, R. P. W. and Mao, J., (2000). Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 22, Number 1, pp. 4-37.
- Joachims, T. (1998). Making Large-Scale Support Vector Machine Learning Practical. *Advances in Kernel Methods: Support Vector Machines*, Scholkopf, B, Burges, C. and Smola, A. (Eds.), Cambridge, MA: MIT Press.
- Joachims, T. (2007). SVMlight Support Vector Machines. Retrieved October 30, 2007  
<http://svmlight.joachims.org/>.
- Johnson, N. F. and Jajodia, S. (1998A). Exploring steganography: Seeing the unseen. *IEEE Computer*, Volume 31, Number 2, pp. 26-34.

- Johnson, N. F. and Jajodia, S. (1998B). Steganalysis of images created using current steganography software. *Proceedings of the Second International Workshop on Information Hiding (IH '98), Lecture Notes in Computer Science*, Volume 1525, pp. 273-289, Berlin, Germany: Springer-Verlag.
- JPEG (1994). Information technology - Digital compression and coding of continuous-tone still images: Requirements and guidelines, ISE/IEC IS 10918-1, American National Standards Institute, Retrieved June 15, 2008  
<http://www.w3.org/Graphics/JPEG/itu-t81.pdf>.
- Kahn, D., (1996). The History of Steganography. *Lecture Notes in Computer Science, First International Workshop in Information Hiding Proceedings*, pp. 1-5, Cambridge, U.K: Springer Press.
- Kaiser, H. F. (1960). The application of electronic computers to factor analysis. *Educational and Psychological Measurement*, Volume 20, Number 1, pp. 141-151.
- Katzenbeisser, S. and Petitcolas, F. A. P. (Eds.). (2000). *Information Hiding Techniques for Steganography and Digital Watermarking*. Norwood, MA: Artech House, Inc.
- Kelley, J., (2001). Terror Groups Hide Behind Web Encryption, *USA Today*, February 5, 2001.
- Kharrazi, M., Sencar, H. T. and Memon, N. (2005). Benchmarking steganographic and steganalysis techniques. *Security, Steganography, and Watermarking of Multimedia Contents VII*, Delp, E. J., III and Wong, P. W. (Eds.), Proceedings of the SPIE, Volume 5681, pp. 252-263.
- Kharrazi, M., Sencar, H. T. and Memon, N. (2006). Performance study of common image steganography and steganalysis techniques. *Journal of Electronic Imaging*, Volume 15, Issue 4, pp. 041104 1-16.
- Kessler, G. C. (2004). An Overview of Steganography for the Computer Forensics Examiner. *Forensic Science Communications*, Volume 6, Number 3, pp. 1-29.
- Kittler, J., (2002). A Framework for Classifier Fusion: Is It Still Needed?. *Lecture Notes In Computer Science*, Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition, Volume 1876, pp. 45 – 56, London, UK: Springer-Verlag.

- Kittler, J., Hatef, M., Duin, R. P. W. and Matas, J., (1998). On Combining Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 20, Number 3.
- Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Volume 2, Number 12, pp. 1137–1143.
- Kohavi, R. and Provost F. (1998). Glossary of Terms. *Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process*, Volume 30, Number 2/3.
- Kuncheva, L. I. (2004). *Combining Pattern Classifiers Methods and Algorithms*. New York, NY: John Wiley & Sons, Inc.
- Kuncheva, L. I., Bezdek J. C., and Duin R. P.W. (2001). Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, Volume 34, pp. 299-314.
- Latham, A. (1999). Steganography. Retrieved July 20, 2005  
<http://linux01.gwdg.de/~alatham/stego.html>.
- Leap, N. J., Clemans, P. P., Bauer, K. B. Jr. and Oxley, M. E. (2004). An Investigation of the Effects of Correlation and Autocorrelation on Classifier Fusion and Optimal Classifier Ensembles. *Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 2004)*, Dagli, C., Enke, D., Buczak, A., Embrechtz, M. and Ersoy, O. (Eds.), pp. 149-154.
- Legendre, A. M. (1805). Nouvelles méthodes pour la détermination des orbites des comètes, A Paris, Chez Firmin DIDOT, Libraire pour les Mathématiques, la Marine, l'Architecture, et les Éditions stéréotypes, rue de Thionville, no 116.
- Lie, W.-N and Lin, G.-S. (2005). A feature-based classification technique for blind image steganalysis. *IEEE Transactions on Multimedia*, Volume 7, Number 6, pp. 1007-1020.
- Lilliefors, H. (1967). On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, Volume 62, pp. 399-402.

- Lin, H.-T., Lin, C.-J. and Weng, R. C. (2003). A note on Platt's probabilistic outputs for support vector machines. *Technical report, Department of Computer Science, National Taiwan University*, Retrieved March 2006, <http://www.csie.ntu.edu.tw/~cjlin/papers/plattprob.ps>.
- Littlebury, I. (1737a). *The History of Herodotus, Translated from Greek*, Volume I, The Third Edition, London.
- Littlebury, I. (1737b). *The History of Herodotus, Translated from Greek*, Volume II, The Third Edition, London.
- Liu, Y. and Zheng, Y. F. (2005). One-Against-All Multi-Class SVM Classification Using Reliable Measures. Retrieved November 2006, <http://www.stat.umn.edu/~xshen/paper/One-Against-All-Zheng.pdf>.
- Looney, C. G. (1997). *Pattern Recognition Using Neural Networks*. New York, NY: Oxford University Press.
- Louw, N. and Steel, S. J. (2006). Variable Selection in Kernel Fisher Discriminant Analysis by Means of Recursive Feature Elimination, *Computational Statistics and Data Analysis*, Volume 51, pp. 2043-2055.
- Lyu, S. and Farid, H. (2002). Detecting Hidden Messages Using Higher-Order Statistical Models. *International Conference on Image Processing (ICIP)*, Rochester, NY.
- Lyu, S. and Farid, H. (2004). Steganalysis Using Color Wavelet Statistics and One-Class Support Vector Machines. *SPIE Symposium on Electronic Imaging*, San Jose, CA.
- Mak, G. (2000). The Implementation of Support Vector Machines Using the Sequential Minimal Optimization Algorithm, University published master's thesis, Retrieved October 28, 2007, [http://moncs.cs.mcgill.ca/MSDL/10\\_people.html](http://moncs.cs.mcgill.ca/MSDL/10_people.html).
- Martinez W. L. and Martinez, A. R. (2002). *Computational Statistics Handbook with MATLAB*, Boca Raton, FL: Chapman & Hall/CRC.
- MATLAB (2004). *Image Processing Toolbox User's Guide*, Version 4. Natick, MA: The Mathworks, Inc.
- MATLAB (2007). *Optimization Toolbox, Users Guide*, Version, 3.1.1. Natick, MA: The Mathworks, Inc.

- Middelmann, W., Ebert, A. and Thoennessen, U. (2006). Assessment of a Novel Decision and Reject Method for Multi-Class Problems in a Target Classification Framework for SAR Scenarios. *Algorithms for Synthetic Aperture Radar Imagery XIII*, Zelnio, E. G. and Garber, F. D. (Eds.), Proceedings of the SPIE, Volume 6237, Number 0P, pp. 1-9.
- Mika, S., G. Ratsch, G., Weston, J., Scholkopf, B. and Muller, K. (1999). Fisher Discriminant Analysis with Kernels. *Neural Networks for Signal Processing IX*, Proceedings of the 1999 IEEE Signal Processing Society Workshop, pp. 41-48.
- Mitchell, T. M. (1997). *Machine Learning*. Boston, MA: McGraw-Hill.
- Murphy, K., (2001). The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, Volume 33, pp. 1-20.
- Murry, J. D. and vanRyper, W. (1994). *Encyclopedia of Graphics File Formats*. Sebastopol, CA: O'Reilly & Associates, Inc.
- Neter, J., Kutner, M. H., Wasserman, W. and Nachtsheim, C. J. (1996). *Applied Linear Statistical Models*, Fourth Edition. Boston, MA: The McGraw-Hill Companies, Inc.
- The Oxford English dictionary: being a corrected re-issue*. (1933). Oxford: Clarendon Press.
- Parzen, E. (1962). On the Estimation of a Probability Density Function and Mode. *Annals of Mathematical Statistics*. Volume 33 pp. 1065-1076.
- Petitcolas, F., Anderson, R. and Kuhn, M. (1999). Information hiding—A survey. *Proceedings IEEE*, Volume 87, pp. 1062–1078.
- Pevny, T. and Fridrich, J. (2006). Determining the Stego Algorithm for JPEG Images. *Special Issue of IEE Proceedings - Information Security*, Volume 153, Number 3, pp. 75-139.
- Pevny, T. and Fridrich, J., (2007). Merging Markov and DCT Features for Multi-Class JPEG Steganalysis. *Security, Steganography, and Watermarking of Multimedia Contents IX*, Delp, E. J., III; Wong, P. W. (Eds.), Proceedings of the SPIE, Volume 6505, pp. 650503 1-13.
- Platt, J. (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods, *Advances in Large Margin Classifiers*, Smola, A., Bartlett, P., Scholkopf, B. and Schuurmans, D. (Eds.), Cambridge, MA: MIT Press.



- Platt, J. C., Cristianini, N. and Shawe-Taylor, J. (2000). Large Margin DAGs for Multiclass Classification. *Advances in Neural Information Processing Systems 12*, Solla, S.A., Leen, T.K. and Mueller, K.-R. (Eds.), pp. 547-553.
- Provos, N. and Honeyman, P. (2003) Hide and Seek: An Introduction to Steganography. *IEEE Security & Privacy Magazine*, May/June, pp. 32-44.
- Provos, N. (2001). Defending Against Statistical Steganalysis. *10th USENIX Security Symposium*, Volume 10, pp. 323-336.
- Provos, N. (2004) OutGuess. Retrieved July 26, 2006, <http://www.outguess.org/>.
- Radcliff, D. (2002). Steganography: Hidden Data. *Computerworld, Inc.*, June 10, 2002, Retrieved March 6, 2008, <http://www.computerworld.com/securitytopics/security/story/0,10801,71726,00.html>.
- Rakotomamonjy, A. (2003). Variable Selection Using SVM based Criteria, *The Journal of Machine Learning Research*, Volume 3, pp. 1357-1370.
- Rao, K. P. and Yip, P. (1990). *Discrete Cosine Transform Algorithms, Advantages, Applications*. San Diego, CA: Academic Press, Inc.
- Rawlinson, G. (1889). *The History of Herodotus*, Volume I, New York, NY: D. Appleton and Company.
- Rawlinson, G. (1875). *The History of Herodotus*, Volume II, Albemarle Street, London: John Murray.
- Rawlinson, G. (1862). *The History of Herodotus*, Volume III, Albemarle Street, London: John Murray.
- Rawlinson, G. (1880). *The History of Herodotus*, Volume IV, Albemarle Street, London: John Murray.
- Rencher, A. C. (2002). *Methods of Multivariate Analysis Second Edition*. New York, NY: John Wiley & Sons, Inc.
- Rice, J. A. (1995). *Mathematical Statistics and Data Analysis Second Edition*. Belmont, CA: Wadsworth Publishing Company.

- Rifkin, R. and Klautau, A. (2004). In Defense of One-Vs.-All Classification. *The Journal of Machine Learning Research*, Volume 5, pp. 101-141.
- Rodriguez, B. M. and Peterson, G. L. (2007). Steganography Detection Using Multi Class Classification. In Craiger, P., and Shenoi, S. (Eds.), *Advances in Digital Forensics III*, (pp. 193-204). Boston, MA: Springer Science+Business Media.
- Rodriguez, B. M. and Peterson, G. L. (2008a). Classifier Dependent Feature Preprocessing Methods, *Mobile Multimedia/Image Processing, Security, and Applications 2008*, Agaian, S. S. and Jassim, S. A. (Eds.), Proceedings of the SPIE, Volume 6982, pp. 69820S 1-12.
- Rodriguez, B. M. and Peterson, G. L. (2008b). Multi-Class Classification Fusion using Boosting for Identifying Steganography Methods. *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2008*, Dasarathy, B. V. (Ed.), Proceedings of the SPIE, Volume 6974, pp. 697407 1-10.
- Rodriguez, B. M., Peterson, G. L. and Bauer, K. W. (2008a). Feature Ranking for SVM and Kernel Fishers Discriminant Classifiers, In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Submitted, June 2008.
- Rodriguez, B. M., Peterson, G. L. and Bauer, K. W. (2008b). Fusion of Steganalysis Systems using Bayesian Model Averaging. Manuscript Submitted for publication.
- Ruck, D. W. (1990). Characterization of Multilayer Perceptrons and Their Application to Multisensor Automatic Target Detection. University published dissertation, AFIT/DS/ENG/90-2, Air Force Institute of Technology, Wright-Patterson AFB, OH.
- Ruck, D. W., Rogers, S. K. and Kabrisky, M. (1990). Feature Selection Using a Multilayer Perceptron. *Journal of Neural Network Computing*, Volume 2, Number 2, pp 40-48.
- Russell, S. and Norvig, P. (2003). *Artificial Intelligence A Modern Approach Second Edition*, Upper Saddle River, NJ: Pearson Education, Inc.
- Ruta, D. and Gabrys, B. (2000). An Overview of Classifier Fusion Methods, *Computing and Information Systems*, Volume 7, pp. 1-10.
- Ruta, D. and Gabrys, B. (2001). Application of the Evolutionary Algorithms for Classifier Selection in Multiple Classifier Systems with Majority Voting. *Lecture Notes in Computer Science*, Volume 2096, pp 399-408.

- Ruta, D. and Gabrys, B. (2005). Classifier Selection for Majority Voting. *Information Fusion*. Volume 6, Issue 1, pp. 63-81.
- Sallee, P. (2003). Model-Based Steganography. *Lecture Notes in Computer Science*, Volume 2939, pp. 154-167, Berlin Heidelberg: Springer-Verlag.
- Sallee, P. (2006) Model-Based Steganography. Retrieved July 20, 2006 <http://redwood.ucdavis.edu/phil>.
- Sallee, P. (2008a) Matlab JPEG Toolbox. Retrieved June 15, 2008 <http://www.philsallee.com/jpegtbx/index.html>.
- Sallee, P. (2008b) Model-Based Steganography Version 1.2. Retrieved June 15, 2008 <http://www.philsallee.com/mbsteg/index.html>.
- Sanguinetti, G., Laidler, J., Lawrence, N. D., (2005). Automatic Determination of the Number of Clusters Using Spectral Algorithms. *Machine Learning for Signal Processing*, 2005 IEEE Workshop on, pp.55-60.
- Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Müller, K.-R., Rätsch, G. and Smola, A. (1999). Input Space vs. Feature Space in Kernel-Based Methods. *IEEE Transactions on Neural Networks*, Volume 10, Number 5, pp. 1000–1017.
- Scholkopf, B.; Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT Press.
- Scholkopf, B., Smola, A. and Muller, K.R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, Volume 10, pp. 1299–1319.
- Schwenker, F. (2000). Hierarchical Support Vector Machines for Multi-Class Pattern Recognition. *Proceedings, Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, Volume 2, pp. 561-565.
- Shi, Y.Q., Xuan, G., Zou, D., Gao, J., Yang, C., Zhang, Z., Chai, P., Chen, W. and Chen, C. (2005). Image steganalysis based on moments of characteristic functions using wavelet decomposition, prediction-error image, and neural network. *IEEE International Conference on Multimedia and Expo*, pp. 1-4.
- Shipp, C. A. and Kuncheva, L. I. (2002). Relationships between combination methods and measures of diversity in combining classifiers, *Information Fusion*, Volume 3, Number 2, June 2002 , pp. 135-148.

- Silman, J. (2001). Steganography and Steganalysis; An Overview. SANS Institute, Retrieved June 3, 2008, [www.sans.org/reading\\_room/whitepapers/steganography/553.php](http://www.sans.org/reading_room/whitepapers/steganography/553.php).
- Simmons, G.J. (1984). The Prisoners' Problem and the Subliminal Channel. *Proceedings of CRYPTO'83 Advances in Cryptology*, pp. 51-67.
- Simoncelli, E. P. and Adelson, E. H. (1990). Subband Transforms. *Subband Image Coding*, Woods, J. (Ed.), Norwell, MA: Kluwer Academic Press.
- Specht, D. F. (1998). Probabilistic Neural Networks for Classification, Mapping , or Associative Memory. *IEEE International Conference on Neural Networks*, pp. 525-532.
- Specht, D. F. (1990). Probabilistic Neural Networks. In *Neural Networks*, Volume 3, pp. 109-118.
- Steganos, (2008). Steganos Privacy Suite 2008, <http://www.steganos.com/us/>.
- Stone, M. (1974). Cross-validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society, Series B (Methodological)*, Volume 36, Number 2, pp. 111-147.
- Tax, D.M.J. Duin, R.P.W. (2002). Using two-class classifiers for multiclass classification. *Proceedings 16th International Conference on Pattern Recognition*, Volume 2, pp. 124 – 127.
- Theodoridis, S. and Koutroumbas, K. (2006). *Pattern Recognition Third Edition*. San Diego, CA: Academic Press.
- Tomasi, C. (2006). Estimating Gaussian Mixture Densities with EM – A Tutorial. *Duke University Course Notes*, Retrieved Sept 15, 2006 <http://www.cs.duke.edu/courses/spring04/cps196.1/handouts/EM/tomasiEM.pdf>.
- Trujillo-Ortiz, A., Hernandez-Walls, R., Castro-Perez, A. and Barba-Rojo, K. (2008). MOUTLIER1: Detection of Outlier in Multivariate Samples Test. A MATLAB file, Retrieved March 15, 2008 <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=12252>.
- Upham, D (1993) JPEG - JSteg. Retrieved July 20, 2005 <ftp://ftp.funet.fi/pub/crypt/steganography/>.

- van der Heijden, F., Duin, R.P.W., de Ridder, D. And Tax, D.M.J. (2004). *Classification, Parameter Estimation and State Estimation An Engineering Approach using Matlab*. England: John Wiley & Sons, Inc.
- Vanderbei, R.J. and Shanno, D.F. (1999). An Interior Point Algorithm for Nonconvex Nonlinear Programming, *Proceedings of the journal of Computational Optimization and Applications*, Volume 13, pp. 231-252.
- Vapnik, V., (1998). *Statistical Learning Theory*. New York, NY: John Wiley & Sons, Inc.
- Wackerly, D. D., Mendenhall, W. and Scheaffer, R. L. (1996). *Mathematical Statistics with Applications*. Belmont, CA: Wadsworth Publishing Company.
- Wand, M. and Jones, M. (1995). *Kernel Smoothing*. London: Chapman & Hall.
- Wang, Y. and Moulin, P. (2007). Optimized Feature Extraction for Learning-Based Image Steganalysis. *IEEE Transactions on Information Forensics and Security*, Volume 2, Number 1, pp. 31-45.
- Wang, Y-C. and Casasent, D. (2005). New Hierarchical SVM Classifier for Multi-class Target Recognition. *Intelligent Robots and Computer Vision XXIII: Algorithms, Techniques, and Active Vision*, Casasent, D. P., Hall, E. L. and Juha, R. (Eds.), Proceedings of the SPIE, Volume 6006, pp. 359-370.
- Westfeld, A. (2001). F5—A Steganographic Algorithm High Capacity Despite Better Steganalysis. *Lecture Notes in Computer Science*, Volume 2137, pp. 289-302, Berlin Heidelberg: Springer-Verlag.
- Westfeld, A. (2003). Steganography Software F5. Retrieved July 26, 2006 <http://www.rn.inf.tu-dresden.de/~westfeld/f5.html>.
- Westfeld, A. and Pfitzmann, A. (1999). Attacks on Steganographic Systems. *Lecture Notes in Computer Science*, Volume 1768, pp. 61-76, London, UK: Springer-Verlag.
- Weston, J., Elisseeff, A., BakIr, G. and Sinz, F. (2006). The Spider. Retrieved October 19, 2007, <http://www.kyb.tuebingen.mpg.de/bs/people/spider/main.html>.
- Weston, J., Elisseeff, A., Scholkopf, B. and Tipping, M., (2003) Use of the Zero-Norm with Linear Models and Kernel Methods, *The Journal of Machine Learning Research*, Volume 3, pp. 1439-1461.

- Wilks, S. S. (1963). Multivariate Statistical Outliers. *The Indian Journal of Statistics*, Volume 25, Issue A, pp. 407-426.
- Woods, K., Kegelmeyer, W. P. and Bowyer, K. (1997). Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 19 Number 4 pp. 405–410.
- Xuan, G., Shi, Y. Q., Gao, J., Zou, D., Yang, C., Zhang, Z., Chai, P., Chen, C. and Chen, W. (2005). Steganalysis Based on multiple features formed by statistical moments of wavelet characteristic functions. *Lecture Note in Computer Science*, Barni, M.; Herrera Joancomartí, J.; Katzenbeisser, S.; Pérez-González, F. (Eds.), Volume 3727, pp. 262-277, Berlin Heidelberg: Springer-Verlag.
- Yang, J., Yang, X. and Zhang, J. (2006). A Parallel Multi-Class Classification Support Vector Machine Based on Sequential Minimal Optimization. *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences*, Volume 1, pp. 443-446.
- Yip, P. and Rao, K. (1987). On the shift properties of DCT's and DST's. *IEEE Transactions on Acoustics, Speech and Signal Processing* Volume 35, Issue 3 pp. 404-406.

## **Vita**

Benjamin M. Rodriguez II received his Bachelor's degree in Electrical Engineering from the University of Texas at San Antonio in 2003. He also received a Master of Science degree in Electrical Engineering from the University of Texas at San Antonio in August 2005. Upon completion of his Ph.D. degree in Electrical and Computer Engineering from the Air Force Institute of Technology, Wright Patterson Air Force Base, Ohio in the Fall of 2008 Mr. Rodriguez will start work in the Washington D.C. area.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
1. REPORT DATE (DD-MM-YYYY) 11 September 2008		2. REPORT TYPE Ph. D. Dissertation		3. DATES COVERED (From – To) September 2005 – September 2008	
4. TITLE AND SUBTITLE  MULTI-CLASS CLASSIFICATION FOR IDENTIFYING JPEG STEGANOGRAPHY EMBEDDING METHODS				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)  Rodriguez II, Benjamin, M., Civ.				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Street, Building 642 WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/DEE/ENG/08-20	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Steganographic Intelligence Technologies Program Manager Air Force Research Laboratory Information Directorate Multi-Sensor Exploitation Branch (AFRL/RIEC) Attn: Chad D. Heitzenrater, CSDP 525 Brooks Road, Rome NY 13441 COM: 315-330-2575 (DSN: 587-2575) Email: chad.heitzenrater@rl.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT  APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT <p>Over 725 steganography tools are available over the Internet, each providing a method for covert transmission of secret messages. This research presents four steganalysis advancements that result in an algorithm that identifies the steganalysis tool used to embed a secret message in a JPEG image file. The algorithm includes feature generation, feature preprocessing, multi-class classification and classifier fusion. The first contribution is a new feature generation method which is based on the decomposition of discrete cosine transform (DCT) coefficients used in the JPEG image encoder. The generated features are better suited to identifying discrepancies in each area of the decomposed DCT coefficients. Second, the classification accuracy is further improved with the development of a feature ranking technique in the preprocessing stage for the kernel Fisher's discriminant (KFD) and support vector machines (SVM) classifiers in the kernel space during the training process. Third, for the KFD and SVM two-class classifiers a classification tree is designed from the kernel space to provide a multi-class classification solution for both methods. Fourth, by analyzing a set of classifiers, signature detectors, and multi-class classification methods a classifier fusion system is developed to increase the detection accuracy of identifying the embedding method used in generating the steganography images. Based on classifying stego images created from research and commercial JPEG steganography techniques, F5, JP Hide, JSteg, Model-based, Model-based Version 1.2, OutGuess, Steganos, StegHide and UTSA embedding methods the performance of the system shows a statistically significant increase in classification accuracy of 10%. In addition, this system provides a solution for identifying steganographic fingerprints as well as the ability to include future multi-class classification tools.</p>					
15. SUBJECT TERMS Steganography; Steganalysis; Multi-Class Classification; Classifier Fusion					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Gilbert L. Peterson (ENG) gilbert.peterson@afit.edu
U	U	U	UU	191	19b. TELEPHONE NUMBER (Include area code) (937) 255-6565, ext 4281