Towards Robust Steganalysis:

binary classifiers and large, heterogeneous data

Ivans Lubenko
Wadham College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Michaelmas 2013

Towards Robust Steganalysis:

binary classifiers and large, heterogeneous data

Ivans Lubenko
Wadham College, Oxford

A thesis submitted for the degree of Doctor of Philosophy
Michaelmas Term 2013

Abstract

The security of a steganography system is defined by our ability to detect it. It is of no surprise then that steganography and steganalysis both depend heavily on the accuracy and robustness of our detectors. This is especially true when real-world data is considered, due to its heterogeneity. The difficulty of such data manifests itself in a penalty that has periodically been reported to affect the performance of detectors built on binary classifiers; this is known as cover source mismatch.

It remains unclear how the performance drop that is associated with cover source mismatch is mitigated or even measured. In this thesis we aim to show a robust methodology to empirically measure its effects on the detection accuracy of steganalysis classifiers. Some basic machine-learning based methods, which take their origin in domain adaptation, are proposed to counter it.

Specifically, we test two hypotheses through an empirical investigation. First, that linear classifiers are more robust than non-linear classifiers to cover source mismatch in real-world data and, second, that linear classifiers are so robust that given sufficiently large mismatched training data they can equal the performance of any classifier trained on small matched data.

With the help of theory we draw several nontrivial conclusions based on our results. The penalty from cover source mismatch may, in fact, be a combination of two types of error; estimation error and adaptation error. We show that relatedness between training and test data, as well as the choice of classifier, both have an impact on adaptation error, which, as we argue, ultimately defines a detector's robustness. This provides a novel framework for reasoning about what is required to improve the robustness of steganalysis detectors. Whilst our empirical results may be viewed as the first step towards this goal, we show that our approach provides clear advantages over earlier methods.

To our knowledge this is the first study of this scale and structure.

Acknowledgements

Andrew has provided me with advice, support and supervision far and beyond what I could have ever expected. I thoroughly enjoyed my time as a DPhil student and it is without doubt largely because of his excellent act in the role of my supervisor. Working with him contributed more than anything towards my understanding of steganography, steganalysis and science in general. It has been a pleasure - a wholehearted thank you.

Many people helped me with my work as a DPhil student, they are not all named but I am grateful to all. I would like to thank Professor Dominic Welsh for his time and patience tutoring me in Information Theory in my first year. I would like to also say thank you to Tomas Pevny and Jan Kodovsky and everybody else who took their interest in my work and several helpful discussions that we had along the way. I would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out much of the empirical work presented here.

I am grateful to the Engineering and Physical Sciences Research Council, the Department of Computer Science and to Wadham College who provided me with financial support over the years.

Special thanks go to my girlfriend's family who looked after me on the run up to the deadline. I felt genuinely welcome and rather spoilt with all the food. Finally I would like to thank my Mom and Dad - none of this would have been possible without their support. And to Helen, thank you.

Contents

1 Introduction **3**

1.1 Formalising the problem 5

1.2 Thesis scope 8

1.3 Thesis statement 8

1.4 Dissertation plan 9

2 Steganography in JPEG files **12**

2.1 Structure of JPEG files 12

2.2 Usable cover size, embedding rate, efficiency and capacity 17

2.3 Components of a steganography algorithm 17

2.3.1 Coding procedure 18

2.3.2 Distortion function 19

2.3.3 Embedding operation 19

2.4 Aims of JPEG steganography design 20

2.5 Literature review of steganography algorithms 21

2.6 Steganography techniques for this study 23

2.6.1 Non-shrinkage F5 24

2.6.2 Perturbed Quantization 26

2.6.3 Usage in experiments 31

3 Features **33**

3.1 Introduction 33

3.2 Main concepts 34

3.2.1 Filtering 35

3.2.2 Cartesian calibration 35

3.2.3 Histograms, co-occurrences and estimates of conditional probability mass functions . . . 37
3.3 Challenges in steganalysis data . . . 38
3.3.1 General . . . 38
3.3.2 From cover source mismatch . . . 39
3.4 Literature review: summary of features sets . . . 41
3.5 Example feature set: C300 features . . . 43

4 Classifiers **46**
4.1 Binary classification . . . 46
4.2 Linear classifiers . . . 51
4.2.1 Average Perceptron . . . 51
4.2.2 Linear Support Vector Machine . . . 53
4.2.3 Fisher Linear Discriminant . . . 54
4.3 Non-linear classifiers . . . 55
4.3.1 Kernel Support Vector Machine . . . 55
4.3.2 Ensemble classifiers . . . 58
4.3.2.1 Ensemble FLD . . . 59
4.3.2.2 Ensemble Average Perceptron . . . 60
4.3.3 Other non-linear methods . . . 60
4.3.4 Analysis . . . 60
4.4 Conclusion . . . 61

5 Designing experiments **62**
5.1 Data . . . 62
5.2 Experimental modelling of classification-based steganalysis scenarios 65
5.3 Methodology . . . 69
5.3.1 Performance measures . . . 69
5.3.2 Model error . . . 71
5.3.3 Calibration error . . . 72
5.3.3.1 Data-related issues . . . 72
5.3.3.2 Classifier-related issues . . . 74
5.3.4 Computational constraints . . . 83

5.3.4.1 One-pass or iterations until convergence? 88
5.3.5 Uncertainty quantification (benchmarks) 92
5.4 Some implementation details 92

6 Experimental results **94**
6.1 Background and motivation 94
6.2 A review of solutions 97
6.3 Hypothesis 100
6.4 Experiment 1: Steganalysis in "laboratory" conditions 101
6.4.1 Procedure for Experiment 1 102
6.4.2 Results for Experiment 1 102
6.5 Experiment 2: Cover source mismatch in "laboratory" conditions . . 104
6.5.1 Results for Experiment 2 105
6.6 Experiment 3: Real-world data: linear and non-linear classification in mismatched data sources 108
6.6.1 Procedure for Experiment 3 108
6.6.2 Results for Experiment 3 108
6.7 Reconciling results across the three experiments 112
6.8 Conclusions and critical assessment 118

7 Analysis **120**
7.1 Statistical learning theory for steganalysis 121
7.1.1 Introduction 121
7.1.2 Formalising concepts 122
7.1.3 Link to steganalysis 126
7.2 Domain adaptation bounds for classification 127
7.2.1 Analysis of experimental results 129
7.3 Can we do any better? 133
7.3.1 Agnostic learning in target domain > domain adaptation . . 134
7.3.2 Further works on domain adaptation 135
7.4 Consequences for steganalysis 135

8 Conclusions **137**

8.1 Open questions 138

8.2 Limitations 140

8.3 Further work 142

A **143**

A.1 KSVM grid search on homogeneous and heterogeneous data 144

B **146**

C **148**

C.1 Results from the final models when the classifiers are trained online under the normal computational constraints 149

D **151**

D.1 Combined tables for homogeneous-matched (major diagonal, highlighted in bold) and homogeneous-mismatched (off-diagonal entries). 151

Bibliography **158**

Preface

As a steganalysis researcher, I have the great fortune of working in a young research field with a vast range of fascinating, yet unexplored, problems. If you have never heard of steganalysis and steganography before, here is the elevator description.

Steganography is a science which studies hidden communication. Its general goal is to enable perfectly undetectable communication over public communication channels. It attempts to achieve this by embedding a message into the subliminal channel of an innocuous digital object, which may then be transmitted publicly. The subliminal channel can be any source of non-determinism present in such digital objects which are plausible to appear on some public channel (even when the source that generates them is largely known). Multimedia files of various formats are well suited to serve as so-called cover objects for embedding steganographic messages.

Steganalysis is the complementary subject to steganography and is primarily concerned with breaking it. A steganographic system is formally considered to be broken if there exists a steganalysis detector which is better than random at identifying the presence of hidden communication. This definition takes its roots from cryptography where the problem of decryption (in our case detection) is modelled as the Prisoners' Problem and adheres to Kerckhoffs' principle, which states that the enemy knows the system. It is straightforward to see why this may not be a satisfactory definition for any practical application of steganalysis - the detectors need to provide robustness guarantees to stand a chance of being deployed in the real world.

Robustness has been considered in the literature and the problem of cover source mismatch has been identified as a major obstacle to achieving successful real-world steganalysis. Cover source mismatch is a term given to the phenomenon which describes the difference in detection performance of a steganalysis detector on data generated by different sources. It occurs when a detector is tuned to one data set and tested on a different data set, which is not the "correct" source of covers. This represents an obvious challenge to real-world applications, where robustness may be critical to the success of deployment of a steganalysis system.

Chapter 1

Introduction

This thesis is an empirical study of steganalysis in heterogeneous covers. The main focus is on binary classification in digital images. This rather simplified view of steganalysis has established itself as *de facto* in modern steganography and steganalysis literature, where advancements are made from both sides in competition and the definition of security is strictly empirical - a steganography method is as secure as its best detector. The 'best detector' is a loose term because our ability to detect steganalysis relies heavily on our knowledge of a great number of variables defining various practicalities of the problem. Unlike cryptography where Kerckhoffs' principle is sufficient to encompass all of the relevant knowledge about the problem, in steganography and steganalysis similar conditions only work for covers reduced to purely theoretical constructs. In these conditions theoretical guarantees about security have been found [52, 111, 117]. For empirical covers the problem appears to be harder if not impossible - arguments have been made that a perfect detector does not exist [15].

This has not, however, stopped the field from trying to improve the practical tools. Much of this improvement has come from the side of steganography and the consensus is that the practical methods here are more advanced than those in steganalysis. In steganalysis, research efforts have largely been focused on improvements in feature representations. The first representations were based on structural information, but modern representations encompass more general statistics and are used as features in conjunction with machine learning classifiers. There is fierce competition

over the development of new, better features, which has seen gradual improvement of detection rates. New feature sets have been proposed at an approximate rate of one every six months claiming to better the state of the art. The most modern examples offer to automate many processes associated with feature engineering, such as dimensionality control [51]. One can envisage that such automation might soon be taken one step further and feature engineering will be replaced with Deep Learning methods, which are designed to automatically construct the best feature representation.

By definition then, machine learning has become an essential tool in steganalysis. Most classifiers used to date are unsophisticated, based on the standard machine learning tools such as the Fisher Linear Discriminant, the Support Vector Machine with Gaussian kernel and ensemble methods based on buckets of models. Several atypical proposals, notably the Extreme Learning Machine [88] and Multitask Logistic Regression [86], have been shown to positively contribute to binary steganalysis. Progress in this area has been slow and there exist many specialised machine learning methods yet to be explored. On the other hand it may be argued that some improvements in detection may only be possible through the use of instruments from machine learning designed specifically to tackle specific learning tasks.

One example of such a task keeps reoccurring despite all the advances in the area of feature engineering. Cover source mismatch has been reported to affect the detection performance in several publications [4, 20, 40, 59, 94, 44, and more]. It is known that the differences in digital images that naturally occur in the real world affect the accuracy of steganalysis. It has been reported that camera model, image processing and even image scene all have some effect [35]. Individual studies have been conducted to pinpoint exact effects on features. For example, Ker studied the effects of cover size on capacity [55]; Böhme [15] studied saturation and local variance from the point of view of their influence on detection accuracy, Kodovsky [72] found resizing had negative effect on steganalysis; several other studies exist [92, 15, 20, 36, amongst others]. Such effects may or may not be specific to a particular type of covers (e.g. spatial-domain images or JPEG images) and future formats

are not guaranteed to have the same effects.

It remains unclear how the performance drop that is associated with the problem of cover source mismatch is mitigated or even measured [58]. In this thesis we aim to show a robust methodology to empirically measure the effects of cover source mismatch on the detection accuracy. Some basic machine-learning based methods, which take their origin in domain adaptation, are proposed to counter this problem. We choose a real-world source of images. To our knowledge this is the first study of this scale and structure.

1.1 Formalising the problem

The idea of viewing steganalysis as a binary classification problem is due to Simmons [109] who formalised it as the Prisoners' Problem.

Let us consider Simmons' problem as it is viewed in contemporary steganalysis literature. In this model, Alice is trying to communicate secret messages to Bob. All communications are monitored by the Warden and the two prisoners' aim is to communicate effectively without raising suspicion. They can achieve this by using steganography to hide their messages using subliminal channels of otherwise innocuous objects, called cover objects. We assume digital images as cover objects, but in practice this can be one of many different types of multimedia or other high-capacity digital objects. To enable steganographic communication Alice and Bob agree on a protocol. Let $\mathcal{C}$ be the set of all objects of a fixed type, e.g. JPEG-format images of fixed quality. Let $\mathcal{K}$ be the set of all possible secret keys and $\mathcal{M}$ the set of all possible messages.

The protocol consists of an embedding function Emb, which takes three inputs: a *predetermined*[1] secret key $k \in \mathcal{K}$, one of *plausible*[1] cover objects $c \in \mathcal{C}$ and the

[1]The problem of establishing a shared key is important in practice however we omit the technical details in this thesis for brevity. The concept of plausible cover objects is closely related to the idea of a cover source which we will be revisited throughout this thesis.

desired secret message $m \in \mathcal{M}$:

$$\mathsf{Emb} \;:\; \mathcal{C} \times \mathcal{K} \times \mathcal{M} \to \mathcal{C} \;.$$

Emb applies embedding changes to cover object c in locations chosen by the secret key k to embed the message m and produce a stego object c'. Extraction function Ext that recovers embedded messages from stego objects using the key is written as:

$$\mathsf{Ext} \;:\; \mathcal{C} \times \mathcal{K} \to \mathcal{M} \;.$$

The goals of this protocol are three-fold: first, to preserve the *correctness* of message delivery:

$$\mathsf{Ext}(\mathsf{Emb}(c, k, m), k) = m \;,$$

second, to achieve *undetectability*, i.e. to avoid perceptible or statistical detection and, third, to achieve relatively *high bandwidth*. A skeleton of such protocol is illustrated in Figure 1.1.

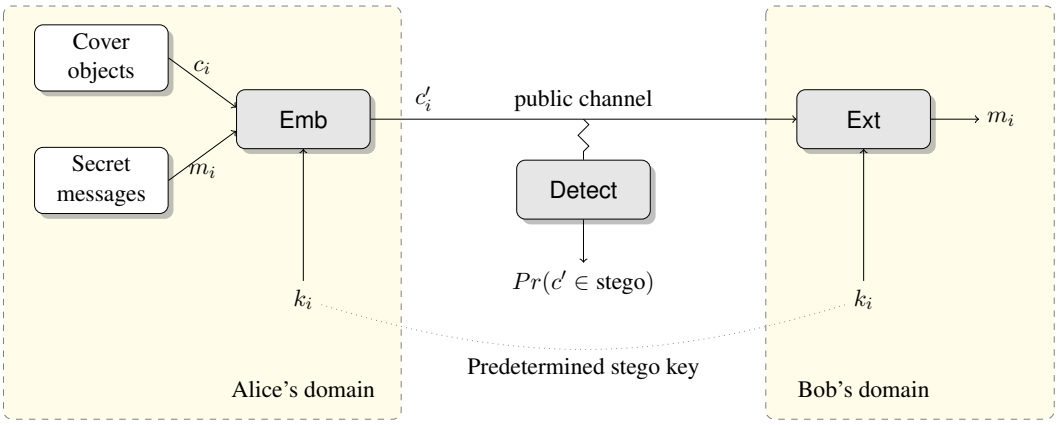


Figure 1.1: High level diagram of the Prisoners' Problem.

A minimum of three parts of this protocol must be known only to the participants: the original cover objects (Alice only), the hidden message (Alice and Bob) and the shared key (Alice and Bob). All other variables are assumed to be known also to the Warden under Kerckhoffs' principle. There are, however, different levels of knowledge that may represent the real-world steganalysis scenario better. For this reason Ker categorises the relevant knowledge into the following scenarios (taken verbatim from [56]):

(A*) The Warden knows the content of the hidden payload and the embedding algorithm used.

(A) The Warden knows the length of the hidden payload, but not its content, and knows the embedding algorithm used.

(B) The Warden knows the embedding algorithm used, but nothing about the payload.

(C) The Warden does not know the embedding algorithm used.

which describe what the Warden knows about the embedding. These go in conjunction with the Warden's knowledge about the covers (also verbatim from [56]):

(1) The Warden knows the exact characteristics of Alice's cover source[2].

(2) The Warden does not know the exact characteristics of Alice's cover source, but can learn about it by seeing examples.

(3) The Warden does not have information about Alice's cover source, but can learn about a similar one by seeing examples.

(4) The Warden knows nothing about Alice's cover source.

We focus our attention on breaking the second goal of a steganographic system: undetectability. It is reached when steganographic objects cannot be distinguished from all other 'plausible' objects. The above cases (1)-(4) characterise different levels of our knowledge about what is plausible in the system. Whilst obvious game-theoretic considerations spring to mind which are not covered by these, it is still a reasonable categorisation and encompasses a wider perspective than is generally accepted in the literature. It has been argued [15] that covers are the least controllable

part in steganalysis and steganography due to them being, in part, manifestations of reality, be it a (part of a) visual scene (in digital images) or compositions of audible sounds (in music recordings), which explains the often-made simplification to case (2), where examples from target source are made available to the steganalyst. Hence, for case (2), the notion of plausible objects is somewhat pre-defined. Under the same argument, case (1) is thought to be only realisable in theoretical constructs. This thesis studies cases (3) and (4).

1.2 Thesis scope

We will assume the job of a passive warden whose sole aim is to test a binary hypothesis:

H_0 : $\boldsymbol{x}$ is a cover image

H_1 : $\boldsymbol{x}$ carries a payload of known length and embedding algorithm

Our task is to learn a decision function $h(\boldsymbol{x})$ that determines the answer for the above hypothesis test. Imagine the decision function being a binary (machine learning) classifier and consider the difference between training $h(\boldsymbol{x})$ for cases (A)(2) and (A)(3)/(A)(4). The former leads to a straightforward problem of training and testing a classifier on data from one (Alice's) source. The latter however calls for non-trivial training. It has been shown that a great number of attributes that we may or may not know about Alice's source affect the accuracy of the decision function $h(\boldsymbol{x})$. Each source has characteristics that may or may not be unique to it therefore the risk of training a detector on a mismatched source needs to be taken into account. This thesis is largely about studying the risk associated with this mismatch and several simple ways of correcting it.

1.3 Thesis statement

This thesis is about steganalysis in real-world data. Specifically, it aims to explore the benefits, as well as challenges, that arise when we switch from using laboratory data, where many variables can be controlled for, to using a large collection of mismatched real-world sources, where the same levels of control may not be possible

due to a steganalyst's limited knowledge of the source in question. Unsurprisingly we encounter cover source mismatch - it therefore takes centre stage in several discussions. Our main approach is simple and takes its motivation from the theory of statistical learning and can be summarised through the following quote from Ng et al. [89]: "Assuming that we have a sufficiently powerful learning algorithm, one of the most reliable ways to get better performance is to give the algorithm more data." We make a claim that under the assumptions that steganalysis training data from the target source may be scarce, if not completely unavailable, it may be beneficial to use large amounts of data from mismatched sources to train a detector for the said target - it all depends on the relatedness between the target source and the available mismatched training data. A steganalyst's choices with regards to the classifier, training data and, to a lesser extent, features are considered. This is investigated through the help of three carefully designed experiments which use real-world data and modern steganalysis classifiers and features. We seek help from statistical learning theory and its recent extensions to domain adaptation in order to analyse the results. A number of questions are raised, some of which are answered with the help of the theory and others are presented as open questions for future work. To the best of our knowledge this is the first work that puts steganalysis in the framework of domain adaptation, its theory and its simplest algorithmic solutions.

1.4 Dissertation plan

The remainder of this thesis is organised as follows. Chapters 2 to 4 are designed to provide a detailed description, as well as any necessary background including literature reviews, for the tools used in our experiments: steganographic embedding algorithms, steganalysis features and binary classifiers. Chapters 5, 6 and 7 in turn introduce our experimental methodology, our experiments and their results and the detailed analysis using theory. In more detail they are:

Chapter 2

Chapter 2 is about JPEG steganography. It starts with an introduction to the JPEG image format. One section is devoted to the necessary details of the components of a typical JPEG steganography scheme. A brief literature

review of different algorithms follows. Finally, we provide details of the nsF5 and PQ steganography schemes which are used in our experiments.

Chapter 3

This chapter provides the background for steganalysis features. We first give an overview of the common characteristics of steganalysis feature sets and the challenges that exist in their design. A brief literature review follows. We conclude with a detailed description of the CC-C300 feature set which is used in our experiments.

Chapter 4

A description of each of the classifiers used in our experiments is given in this chapter. It proposes to test classifiers that have not previously been considered for steganalysis, in particular the Average Perceptron, linear Support Vector Machine and an online ensemble classifier. Chapter 4 also discusses their potential advantages.

Chapter 5

Experimental design is the main subject of this chapter. It includes the details of the data and how it allows us to create multiple distinct models of the steganalysis problem, including the laboratory conditions, in which we measure the performance of classifiers trained on matched as well as mismatched data, and the more realistic conditions, in which a large amount of mismatched data is made available for training. It also provides an in-depth look at our experimental methodology including the metrics used to assess the classifiers' performance.

Chapter 6

Our experiments and their results are presented in this chapter. Experiment 1 replicates the laboratory conditions studied in most steganalysis work to date. The trained classifiers are evaluated on mismatched testing data in Experiment 2. Experiment 3 proposes a new approach to training a steganalysis detector: the classifiers are trained from as large and varied selection of images as possible. We discuss the clear advantages of approach in Experiment 3 compared

to the approach in Experiment 2 and point out its deficiencies compared to the idealised conditions of Experiment 1.

Chapter 7

This chapter puts steganalysis in the framework of theoretical machine learning, specifically comparing the regular training to domain adaptation based settings. With the help of recent advancements in statistical learning theory, in particular those that extend it to domain adaptation, we provide an analysis of our experimental results. Several key findings are made here and their consequences to steganalysis explained. This chapter raises many open questions.

Chapter 8

The final chapter summarises the key findings and contributions of this thesis to the field, as well as providing the necessary critique and limitations. Several questions open for future work are discussed.

Chapter 2

Steganography in JPEG files

Two steganography algorithms were chosen for this study, namely no-shrinkage F5 (nsF5) and Perturbed Quantisation (PQ). As our aim is to evaluate detectors performance rather than intrinsic properties of the chosen algorithms, such as embedding efficiency or optimality of distortion function, we are able to, without the loss of generality, simplify our experiments by implementing fast simulations of the two algorithms. We start this chapter with the necessary introduction to the JPEG image format, including the compression algorithm, as this will be required for our simulation of PQ. Section 2.3 provides the necessary details of the components of a typical JPEG steganography scheme. A brief review of image-based steganography is given is Section 2.5. This is followed by Sections 2.6.1 and 2.6.2 which provide details of nsF5 and PQ steganography schemes which we use in our experiments. The chapter concludes with a brief analysis of the properties of the two algorithms.

2.1 Structure of JPEG files

JPEG [116] is a popular lossy image-compression standard and is by far the most common format for storing digital images. The lossy compression, which leads to smaller file sizes, is achieved by approximating images based on the human perception of visual information. The idea is to preserve only perceptually important content. In practice this means that, for example, brightness is taken as being more important than colour and low spatial frequency as more important than high spatial frequency. The less important content is approximated or lost through the lossy

steps of downsampling and quantisation. The amount of compression, and as such the quality of approximation, is specified by the user through the quality factor parameter f.

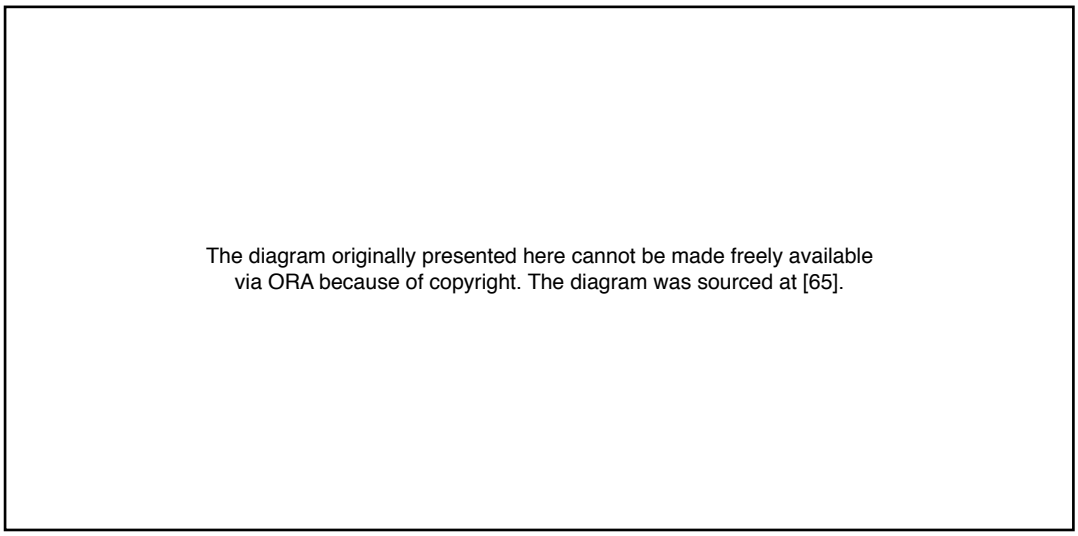


Figure 2.1: JPEG compression algorithm illustrated. Copied from [65].

JPEG images consist of a header, which carries the metadata about the image (e.g. its size and quantisation table), and a losslessly compressed stream of quantised discrete cosine transform (DCT) coefficients. In outline, the JPEG compression algorithm performs the following sequence of steps to arrive at the final stream of data as shown in Figure 2.1:

(1) Colour space transformation linearly transforms each pixel of the original uncompressed image from the RGB (or grayscale) colour model into the YC_bC_r colour model (if needed)[1]. This is a linear transform. Y is the luminance component and C_b and C_r are the blue and red chrominance components. The three components are then processed separately.

(2) Spatial downsampling: the C_b and C_r components are taken at lower resolution (typically 2 to 1 in both horizontal and vertical directions).

[1]There are different ways of encoding information about colour: RGB and YC_bC_r are two well-known models.

(3) Discrete cosine transform (DCT) is applied to individual 8-by-8 blocks of pixels with padding if necessary.

(4) Each block undergoes quantisation using an 8-by-8 table of quantisation coefficients with rounding to integers.

(5) Lossless compression using the Huffman coding.

In steganography we are particularly interested in step (4). Quantisation is a lossy operation and therefore the steganographic embedding changes are naturally made to the transformed data, i.e. the quantised DCT coefficients[2]. We will be using JPEGs for simulating steganographic embedding using different algorithms and thus a short description of DCT transform and quantisation is given below.

The Discrete Cosine Transform (DCT) is a major step in the JPEG compression algorithm and is used to transform blocks of an image from the spatial domain (pixel colour values) to the frequency domain (DCT coefficients). We can visualise this procedure by looking at the inverse of this operation (IDCT). For a given 8-by-8 block of DCT coefficients C we can retrieve its corresponding block of pixel values B using the IDCT transform, which takes the form of a linear combination of DCT basis vectors A^{uv} multiplied by the respective DCT coefficients $c_{uv} \in C$ (adapted from [56]):

$$B = \sum_{u=1}^{8} \sum_{v=1}^{8} c_{uv} A^{uv} \ .$$

The elements a_{ij}^{uv} for $i, j \in [1..8]$ of each basis block A^{uv} for $u, v \in [1..8]$ are given by (adapted from [56]):

$$a_{ij}^{uv} = \frac{c_u c_v}{8} \cos\left(\frac{\pi}{8}\left(i + \frac{1}{2}\right)u\right) \cos\left(\frac{\pi}{8}\left(j + \frac{1}{2}\right)v\right) \ ,$$

where $c_0 = \sqrt{2}$ and $c_n = 1$ for all $n \neq 0$. Selected basis blocks A^{uv} are shown in Figure 2.2. The coefficients which are closer to the top left corner of a block are the low frequency coefficients, and the bottom right corner are the high frequency

[2]Otherwise a level of redundancy must be introduced to account for errors (e.g. see Yet Another Steganography Scheme [110]), which goes in tension with one of the main principles of steganography - minimum distortion.

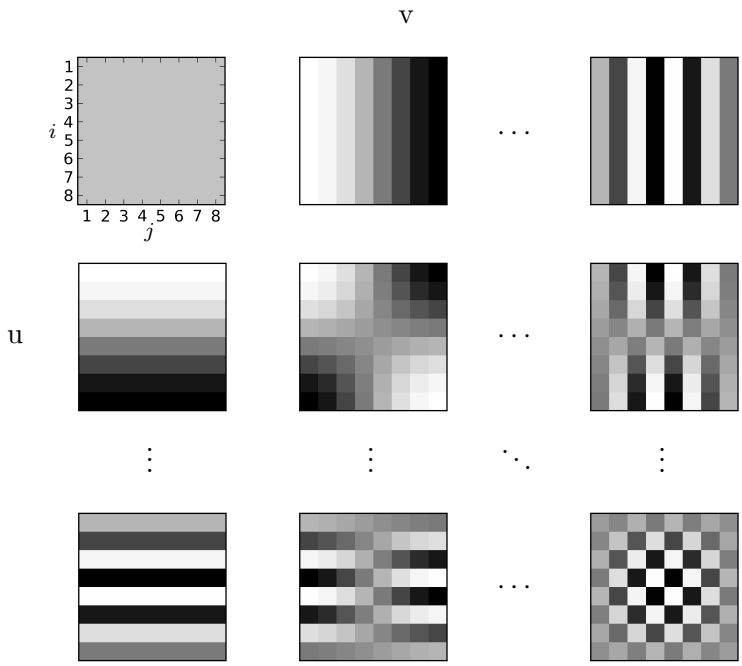


Figure 2.2: Selected DCT basis blocks for inverse DCT transform.

coefficients, likewise basis blocks with low indices correspond to low frequency and block with high indices to high frequency.

Quantisation is the process of rounding often real-valued DCT coefficients c_{ij} to the nearest integer q_{ij} where q_{ij} is called the quantisation step and can be different for different coefficients based on their position (indexed by (i, j), sometimes called mode) in the DCT-block. Larger q_{ij} values result in harsher quantisation and hence larger information loss. An 8-by-8 table of qs is called the quantisation table Q_f for some fixed quality factor $f \in [1..100]$ and is unique to each of the three components (YC_bC_r) of an image. The direction of rounding may vary between different JPEG compression libraries.

Two example luminance (Y component) quantisation tables Q_f for quality factors $f = 85$ and $f = 70$ are shown in Figure 2.3. The standard quantisation tables like these are calculated by a formula from the base table for quality factor 50 and are

$$\begin{pmatrix} 5 & 3 & 3 & 5 & 7 & 12 & 15 & 18 \\ 4 & 4 & 4 & 6 & 8 & 17 & 18 & 17 \\ 4 & 4 & 5 & 7 & 12 & 17 & 21 & 17 \\ 4 & 5 & 7 & 9 & 15 & 26 & 24 & 19 \\ 5 & 7 & 11 & 17 & 20 & 33 & 31 & 23 \\ 7 & 11 & 17 & 19 & 24 & 31 & 34 & 28 \\ 15 & 19 & 23 & 26 & 31 & 36 & 36 & 30 \\ 22 & 28 & 29 & 29 & 34 & 30 & 31 & 30 \end{pmatrix} \qquad \begin{pmatrix} 10 & 7 & 6 & 10 & 14 & 24 & 31 & 37 \\ 7 & 7 & 8 & 11 & 16 & 35 & 36 & 33 \\ 8 & 8 & 10 & 14 & 24 & 34 & 41 & 34 \\ 8 & 10 & 13 & 17 & 31 & 52 & 48 & 37 \\ 11 & 13 & 22 & 34 & 41 & 65 & 62 & 46 \\ 14 & 21 & 33 & 38 & 49 & 62 & 68 & 55 \\ 29 & 38 & 47 & 52 & 62 & 73 & 72 & 61 \\ 43 & 55 & 57 & 59 & 67 & 60 & 62 & 59 \end{pmatrix}$$

(a) $f = 85$ (b) $f = 70$

Figure 2.3: Quantisation tables for quality factors 85 (left) and 70 (right)

obtained using the following approximating equation[3] [35]:

$$Q_f = \left[Q_{50} \frac{100 - f}{50} \right], \text{ where } Q_{50} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix} \tag{2.1}$$

This equation reveals an important relationship between quantisation tables of different quality factors, which is exploited by some steganography algorithms such as Perturbed Quantisation which we study later. In particular, we can observe that in Figure 2.3 many of the values of quantisation factors in (b) are a factor (of 2) away from the values in (a). This property is true for many pairs of quantisation tables, including all pairs whose quality factors satisfy:

$$f' = 2(f - 50), \text{ s.t. } f \text{ and } f' \in \mathbb{Z}^+. \tag{2.2}$$

It is possible to define custom quantisation tables for JPEG compression. We will see that calculating them using Equation (2.1), which gives similar but not exactly the standard JPEG quantisation tables for a fixed quality factor f, will prove to be particularly useful for steganography. The custom and the standard quantisation tables would produce very similar images. We use this idea in Section 2.6.2.

[3]The brackets [ ] in this equation mean integer rounding.

For simplicity we assume working in luminance (Y) component only, however all arguments can be extended to other components in a straightforward manner.

2.2 Usable cover size, embedding rate, efficiency and capacity

Before we discuss JPEG steganography in detail we first need to formalise some concepts of embedding measures that are common to spatial- and JPEG-domain algorithms. We start with the notion of cover size, which is equal to the total number of pixels in a spatial-domain cover or the total number of non-zero DCT coefficients in a JPEG cover. We define usable cover size as the total number of elements (pixels or DCT coefficients) in an image which are eligible for embedding under the embedding operation of a given steganography algorithm. Embedding efficiency is defined as the average number of message bits embedded per one change to the cover. Capacity is the maximum payload length that can be embedded using a given steganography algorithm into a given cover expressed as a proportion of the cover size. Embedding rate is the proportion of capacity used for embedding:

$$\text{true embedding rate} = \text{efficiency} \times \text{change rate} \times \frac{\text{usable cover size}}{\text{cover size}} \,. \tag{2.3}$$

Throughout this thesis we will use the term "embedding rate" to refer to the quantity:

$$\text{"embedding rate"} = \text{efficiency} \times \text{change rate} \,. \tag{2.4}$$

This allows us to reason about what proportion of the usable cover size was used for embedding in each case.

2.3 Components of a steganography algorithm

One can differentiate several components of an embedding algorithm: the coding procedure, the distortion function and the embedding operation. A high-level diagram is shown in Figure 2.4. In this thesis we study how steganography changes the quantised coefficients. We therefore abstract away everything related to the message - in our simulation *the three components reduce to the embedding operation only.* We note however that this level of simplification can only be achieved

for non-adaptive embedding schemes or weakly-adaptive schemes (see Section 2.5) where the distortion function is part of the embedding operation.

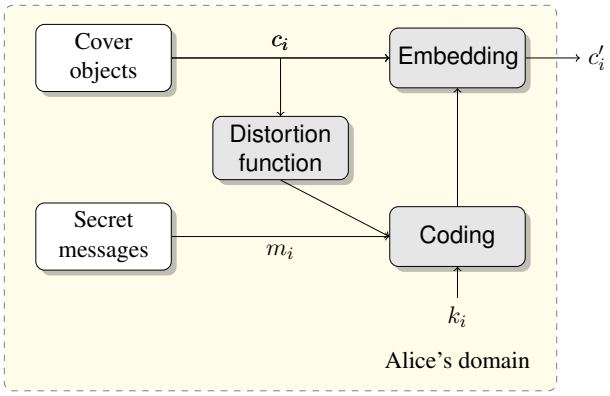


Figure 2.4: Schematic diagram of the three components of a steganography algorithm: coding procedure, distortion function and embedding operation.

2.3.1 Coding procedure

A coding procedure is usually used for one or more of the following purposes:

a) As a workaround for the non-shared selection channel problem.[4]

b) To increase the embedding efficiency, which means that fewer embedding changes of the same magnitude[5] are required to embed a given payload.

c) More generally to minimise some expected distortion.

It takes into account the arrangement of LSBs in some predefined locality in conjunction with the values of several message symbols at a time in order to find the minimum number of embedding changes required to encode those symbols into that locality. Examples of a) include Wet Paper Codes [32], examples of b) include Hamming codes [113] and examples of c) include Syndrome Trellis Codes [30].

[4]This occurs in some embedding operations such as the F5, when the receiver has only partial knowledge of the location of the payload.

[5]By magnitude here we mean the average number of bits changed per one embedded symbol.

2.3.2 Distortion function

Distortion function measures the cost of embedding and is used by adaptive embedding algorithms as side-information to position the embedding changes into the low-cost high-complexity areas of the image[6]. Different distortion functions may choose different embedding paths. Early examples simply counted the number of embedding changes (see Modified Matrix Encoding (MMx) [64]). All current distortion functions, such as Highly Undetectable Steganography (HUGO) [95], Uniform Embedding Distortion (UED) [45] and Universal Wavelet Relative Distortion (UNIWARD) [50] have different costs associated with different possible changes. These functions are additive which means they can be effectively minimised. Studying distortion functions is outside of the scope of this thesis and we refer the reader to the relevant references for more information.

2.3.3 Embedding operation

The embedding operation is the most low level part of the algorithm that deals with how the cover elements are changed. It is used to perform embedding changes at locations predetermined by the shared selection channel from the collection of locations allowed by the distortion function. In JPEG images it has a secondary function of preserving the undetectability property of the stego system, in particular to avoid visual detection. Many JPEG embedding operations include heuristics which are based on our knowledge of the JPEG format. These heuristics are designed to ensure that obvious visual artefacts are not introduced into the image. The most common heuristics specific to the JPEG stego system are:

a) The absolute value of a DCT coefficient should never be increased [68]. The need for this stems from the fact that changing one JPEG coefficient affects its block of 8-by-8 pixels. Furthermore some coefficients (higher frequency) are quantised harsher than others, therefore changing low frequency coefficients. In practice this means that a one bit increase in the value of a high-frequency coefficient caused by an embedding operation such as LSB-replacement will be multiplied by a large relative value of the quantisation factor and as such have

[6]The assumption here is that such areas are harder to model.

a disproportionate contribution to the $YCbCr$ values of the corresponding block after the inverse DCT transform.

b) Avoid recompression as it may introduce visual and statistical artefacts, which might not be present in the original domain of the sender [67]. Visual artefacts common to JPEG compression include blockiness, colour distortion, ringing and blurring.

c) If compression is performed during embedding, the algorithm must match that of the covers [42].

These are just a few examples that may lead to a violation of statistical and visible structures found in JPEG images.

State-of-the-art steganography schemes encompass all three of the above components however the distortion function requires side information. Side information usually means the knowledge of the original uncompressed image (often called pre-cover) which may or may not be a feasible assumption in the real-world scenario. In this dissertation we will focus on those JPEG steganography schemes that do not require the knowledge of the uncompressed image. Because JPEG is a lossy compression algorithm, the steganographic embedding changes must be applied after all the stages of the algorithm where the information is lost - the embedding changes will not persist otherwise. The next section introduces all steps of the JPEG compression algorithm and their outputs, paying special attention to those steps where the embedding can be made.

2.4 Aims of JPEG steganography design

Just like with steganography in spatial-domain images most JPEG-domain algorithms aim for increased security based on some collection of heuristics. The heuristics can include one of the following (alongside examples of their associated algorithms; for more details of these algorithms see Section 2.5):

- Resist existing steganalysis attacks (e.g. LSB-matching (avoids the parity structure in the histogram), OutGuess (preserves first-order features))

- Minimise the number of changes (F5, nsF5)
- Minimise a more general distortion function[7] (any adaptive embedding technique which requires original never-compressed image, e.g. HUGO)
- Emulate natural randomness of the operations underlying the JPEG compression algorithm (PQ)

2.5 Literature review of steganography algorithms

We continue with a brief review of literature on image-based steganography. Looking back at the development of algorithms for steganography one might notice that only a few of them introduced new embedding operations. Other algorithms utilise those embedding operations and extend them to improve efficiency or security or both through the use of coding and distortion functions.

Whilst it is unclear when or how the Least Significant Bit replacement (LSB-r) came in to existence, the consensus is that it is probably the first steganography method for digital images. It is a simple spatial-domain algorithm that can be summarised by its embedding operation (or an 80-character-long Perl code [53]), which works by overwriting the least significant bits of image pixels with the message encoded in binary. Several variants of this algorithm exist that improve its efficiency by coding the message using ternary or any more general q-ary symbols and embedding each symbol in two or more least significant bits. A related algorithm is LSB-matching (LSB-m) [107] which improves the security of LSB-r by defining a new embedding operation which, if change is required, increases or decreases pixels at random instead of simply overwriting their LSBs.

To make these types of algorithms compatible with the JPEG image format a modification was proposed, called JSteg [112]. Due to the nature of what JPEG coefficients represent compared to the normal image pixels, an algorithm that blindly (regardless of the complexity of the local content) overwrites LSBs with message bits inevitably

[7] Minimising number of changes can also be viewed as a distortion function. More general distortion functions minimise the total cost, where each change has non-trivial cost.

increases the absolute values of low-frequency coefficients which results in visible artefacts (appearing as an 8-by-8 checkerboard pattern) in flat low-complexity areas of an image such as the sky. The algorithm for JSteg addresses the problem by simply avoiding embedding into the coefficients with values -1, 0 or 1.

Several other algorithms used LSB-replacement operations. For example, OutGuess [98] was designed to resist early steganalysis attacks which used first-order statistics. It keeps track of all unmodified coefficients and uses them to restore the histogram of the original image after the message was embedded. It was shown to be detectable by steganalysis features based on second- or higher-order statistics.

Other embedding operations were also proposed. The F5's [118] embedding operation, which was later shown to be empirically optimal for JPEG images [68], only allows changes that decrease the absolute values of image coefficients. Non-shrinkage F5 (nsF5) [41], an improved version of the F5 algorithm which uses the same embedding operation established itself as one of the most widely studied in JPEG steganalysis research and we use it in this thesis and discuss it in more detail in Section 2.6.1 of this chapter.

Other algorithms proposing alternative embedding operations were YASS [110] and Steghide [48]. YASS was designed to make embedding changes in spatial or wavelet domains in such a manner that they are preserved after subsequent compression into, say, JPEG. It was shown that the necessary robustness was a major drawback leading to easy detection given the right features [74]. StegHide introduced a graph-theoretic algorithm for exchanging the values of two or more "adjacent" pixels. It also proved to be easily detectable.

It has been shown that the ranking of non-adaptive JPEG algorithms is [61]:

$$\text{nsF5} > \text{F5} > \text{StegHide} > \text{OutGuess} > \text{Jsteg}$$

So far all examples we considered assumed uniform cost of embedding changes (apart from the restricted elements like -1, 0 and 1 in StegHide which had infinite cost). A

major advancement came with the introduction of side-informed embedding using the notion of distortion functions. Most modern algorithms such as Highly Undetectable Steganography (HUGO) [95], Wavelet Obtained Weights (WOW) [49], Universal Wavelet Relative Distortion (UNIWARD) [50], Perturbed Quantisation (PQ) [33] and others were designed to exploit properties of their respective embedding domain through the use of side-information. They are capable of choosing where the embedding changes are made based on the content of the image which is found from a pre-cover. In spatial-domain the pre-cover can be the original uncompressed image itself, whilst in JPEG-domain they can be either a never-compressed image or a higher-quality JPEG that was used to produce the cover JPEG (in the latter case resulting in a lower quality stego image also in JPEG format). Adaptive embedding is considered to be more secure, which has so far been supported by empirical evidence.

An argument is yet to be made about the practicality of adaptive schemes which use never-compressed images as pre-covers and only select adaptive schemes are capable of embedding using JPEGs as pre-covers. PQ is one of them and is used in our experiments. Other examples that are also capable of side-informed embedding into JPEGs are BCHOpt [104] and SI-UNIWARD [50].

2.6 Steganography techniques for this study

It has been shown that non-adaptive embedding algorithms are not very secure given the current steganalysis features. On the other hand it can be argued that adaptive algorithms, especially those that use never-compressed images as pre-covers, are not very realistic. Furthermore as a major part of this thesis is based around a large database of JPEG images we are not able to use spatial domain techniques for two reasons. First, decompressed JPEGs are only approximations of natural images - using them as pre-covers in place of never-compressed natural images may lead to rather biased experimental results[8]. Second, obtaining circa 2 million never-

[8]This depends on the quality factor of the given JPEG in the first place. Images in our database have quality factor of 85 which means they are likely to exhibit statistical artefacts in the spatial-domain and lead to the aforementioned problem.

compressed spatial-domain images is infeasible in most circumstances.

In this thesis we chose to work with two steganography algorithms, one from each group. nsF5 and PQ are two well-established examples from the research literature. They are perhaps the simplest algorithms in their respective groups (non-adaptive and adaptive schemes) that are not obviously flawed unlike, for example, StegHide or YASS. At the time of writing there exist more modern examples which are claimed to be more secure, but they were not introduced until our experiments for this thesis were well under way.

Our aim is to study steganography from the point of view of its effects on steganalysis features. We can therefore focus our attention on the embedding operation and simulate the steganography algorithm by implementing only the functionality necessary to its embedding operation. We abstract away everything that is related to the message which means that any intrinsic properties of the steganography algorithm such as embedding efficiency or optimality of the distortion function are deliberately not taken into account, allowing us to achieve computational efficiency. We use pseudorandom streams of bits as simulated messages.

This section provides a detailed description of nsF5 and PQ and our simulated implementations of their embedding operations.

2.6.1 Non-shrinkage F5

Non-shrinkage F5 (nsF5) is a more secure version of the F5 steganography algorithm and uses F5's embedding operation. The F5 algorithm was first introduced by Westfeld in [118]. Its embedding operation is straightforward: each message bit is embedded into one of any non-zero DCT coefficients of the cover image which have not yet been visited by either reducing its absolute value by one or leaving it unchanged if they match. The locations for the payload are chosen according to a shared selection channel (e.g. they are defined simply by a key used as a seed to pseudorandom number generator) and all zero-valued coefficients are skipped. There is a slight complication to the F5 algorithm, which is called the "shrinkage":

in the event when the DCT coefficient has the value -1 or 1 it will be changed to zero and must be re-embedded. This "shrinkage" can be easily detected by modern steganalysis schemes. It was later resolved in the non-shrinkage F5 (nsF5) algorithm, which avoids re-embedding by the means of Wet Paper Codes [32]. The nsF5 is one of the most studied algorithms in steganalysis research due to its superior security compared to other schemes and empirically optimal embedding operation [68].

Algorithm 1 Simulated non-shrinkage F5

1: **procedure** (SIMULATED) NSF5
2: $S \leftarrow$ all non-zero coefficients c_i of image $\mathcal{I}$
3: **repeat**
4: choose c_i from S at random without replacement
5: **if** change **then** ▷ with probability of 0.5.
6: $c_i = (|c_i| - 1) \times \text{sgn}(c_i)$
7: **end if**
8: **until** reached the length of payload
9: **end procedure**

Our simulation

Our simulation of the nsF5 algorithms follows from the above description and is shown in Algorithm 1. We assume that a typical message consists of independent and identically distributed random bits. All non-zero valued coefficients are contributing coefficients. From those we choose $p \times 100\%$ of them at random to match the desired embedding rate p. These are then used to embed a random payload according to the F5's embedding operation.

Embedding rate

In this simple version of the nsF5 algorithm the embedding rate (as per Equation 2.4) is equal to the true embedding rate because all non-zero coefficients can be used for embedding and the usable cover size and the cover size are the equal. We will therefore measure nsF5's embedding rate in bits per non-zero DCT coefficient. In sPQ the true embedding rate will be different as the usable cover size is almost always smaller than the cover size. We measure the embedding rate for sPQ in bits per *usable* DCT coefficient (bpuc).

2.6.2 Perturbed Quantization

In nsF5 the minimum impact of a change is 1 as we decrease the absolute value of DCT coefficient by 1. PQ goes one step further with expected minimum impact of change being 0.5. To achieve this PQ requires side information and is therefore an instance of a content adaptive steganography algorithm. Its aim is to minimise the intrusiveness of embedding changes by perturbing the normal process of rounding which is a customary part of the JPEG's quantisation step (step (4), Section 2.1). Two working modes can be differentiated. In the first mode PQ requires an original uncompressed (or lossless-format) image to calculate the side information. In the second, it uses an existing JPEG image for side-information and re-compresses it with a lower quality factor. Both variants produce a JPEG as an output stego image. Here we discuss the second but both modes follow a very similar algorithm.

For any two given quantisation tables Q and Q' one can find all DCT modes[9] which satisfy the following equation [33]:

$$kq_i = lq'_i + \frac{q'_i}{2}; \quad \text{s.t.} \quad k, l \in \mathbb{Z} \quad \text{and} \quad \frac{q'_i}{g} \text{ is even}\,, \tag{2.5}$$

where q'_i and q_i are quantisation steps at position i of tables Q and Q' respectively and g is the greatest common divisor of q'_i and q_i. For any image $\mathcal{I}$ with quantisation table Q and its re-compressed version $\mathcal{I}'$ with quantisation table Q' all pairs of DCT coefficients modes q_i, q'_i satisfying the above equation are called contributing pairs.

During re-compression every odd coefficient which appears in a contributing mode will be rounded[10] at quantisation with the rounding error of exactly 0.5. Different JPEG compression algorithms may perform rounding in different "directions". To embed into this position we simply choose the rounding direction such that the value of the least significant bit of the resulting (re-)quantised coefficient matches the required message bit. If the compression algorithm's rounding process was not deterministic, the expected impact of a change would be 0.5.

[9] We refer to each position in an 8-by-8 block as *mode*.

[10] Quantised coefficients are rounded to integers as discussed in step (4) of Section 2.1.

If we restrict our selection to only coefficients in the contributing modes we enforce a binary distortion function, in which a fixed cost is assigned to embedding into these coefficients and an infinite cost is assigned to any other change. A more general cost function would assign cost inversely proportional to the rounding error $e \in [0, 0.5]$ [50].

Because it is coefficients c_i' of the re-compressed image $\mathcal{I}'$ that get changed, we treat $\mathcal{I}'$ as the true cover object. The capacity of $\mathcal{I}'$ grows with the proportion of contributing modes to non-contributing modes. We can maximise the number of contributing modes to 64 (i.e. all of them) by enforcing a custom quantisation table during re-compression. This can be achieved by setting[11]:

$$Q' = 2 \times Q \tag{2.6}$$

Under these conditions, the rounding step of the JPEG compression algorithm by design has the desired effect on the coefficients c_i' as given in the Equation (2.5). When comparing the coefficients of the cover image $\mathcal{I}'$ to the coefficient of the original image $\mathcal{I}$, we can observe the following relationship *in most cases*:

$$c_i' = \begin{cases} \dfrac{c_i}{2} & \text{if } c_i \text{ is even} \\ \\ \dfrac{c_i}{2} \pm 0.5 & \text{if } c_i \text{ is odd} \end{cases} \tag{2.7}$$

It simply follows that we can embed into any *odd* coefficient c_i unless its corresponding coefficient c_i' in the cover image is different from $\dfrac{c_i}{2}$ by over 0.5, which happens on rare occasions due to another effect that we describe below.

In practice not all (c_i, c_i') pairs have this relationship; we observed some c_i' which have a larger absolute value than original c_i. By checking the difference between the expected values and the true values of coefficients in the re-compressed image using: $d = |(c_i - 2 \times c_i')|$ we find that d can be as large as 4 instead of the obvious maximum of 1. Approximately 0.03% of coefficients (or 0.075 of all non-zero coefficients) do

[11] One can also set Q' = 2 x k x Q for any integer k but this will decrease the number of non-zero-valued coefficients.

not result in the expected value. It means that even when sequentially performing the decompression and re-compression procedures using standard JPEG algorithms with exactly the same quantisation tables it may not result in exactly the stream of coefficients in the output image.

This is due to re-compression, which involves two operations: decompression to spatial-domain followed by a JPEG compression using the new quality factor. Even if the same quality factor is used for the original JPEG and for the new re-compressed image this turns out not to be an identity function. It can be explained by taking a closer look at some of the individual operations which take place during this procedure. In particular, truncation of the values to the $[0, 255]$ range after the inverse DCT, rounding to integers and, in the case of colour images, the downsampling of chrominance components may all be partially responsible for this effect. We leave this issue to further investigation as it does not directly affect our simulation of this steganography scheme. Because it is only a small number of coefficients that are affected by this, we assign infinite cost to them which means they are dismissed from being part of the contributing coefficients list.

Implementation

There exists an implementation of this algorithm due to Kodovsky [66], but we propose a simulation of it which focuses only on PQ's embedding changes and leaves the coding strategy out for the reasons of computational efficiency, as it is essential to our experimental strategy.

Our simulation

Our algorithm, the simulated PQ, will be based on the idea introduced in Equation (2.6). Let f be the quality factor of the original JPEG images. Then we are looking for another quality factor f' such that the set of coefficients that follow the relationship in equation (2.7) is maximised. For example, the JPEG images used for our experiments have quality factor $f = 85$. Following the argument from the previous section, we know that $f' \sim 70$ would yield the maximum number of contributing coefficient modes. However in practice using the standard JPEG quantisation table

for quality factor 70 will yield only 44 out of 64 modes as contributing when images are re-compressed from the quality factor 85. According to Equation (2.7) not every DCT coefficient which is located in a contributing mode has the potential to carry a payload, only those coefficients with odd values in $\mathcal{I}$.

This is why a self-defined quantisation table, such as $Q' = 2 \times Q$ will be a better option - the embedding changes will not be focused to a specific portion of modes and as such our simulation in this case will be more conservative. With this quantisation table 64 out of 64 modes are contributing modes and every odd coefficient in such mode is eligible for embedding[12]. As such all images in our data sets can be embedded with a non-negligible payload at a given embedding rate. Then, we find all possible embedding locations in an image and, as before, use a random selection of $p \times 100\%$ of them for embedding a random payload.

Algorithm 2 summarises the simulation of PQ's embedding changes. Let $\mathcal{I}$ be the original JPEG image compressed with quantisation table Q and $\mathcal{I}'$ be its PQ cover image with quantisation table Q'. Let M be the set of all modes which satisfy Equation (2.7). If $Q' = 2 \times Q$ then $|M| = 64$. Put all odd coefficients c_i of $\mathcal{I}$ in modes M into set S (contributing coefficients). Randomly select $p \times |S|$ of those coefficients for embedding. For each coefficient in the selection: if c_i and c_i' satisfy the Equation (2.7) then set $c_i' = c_i + 0.5$ or $c_i' = c_i - 0.5$ at random with probability 0.5.

Embedding rate

Unlike our implementation of nsF5, in this simulated PQ algorithm the true embedding rate will be different to our definition of embedding rate from Equation 2.4 as usable cover size here is almost always smaller than the cover size. sPQ's embedding rate is therefore measured in bits per usable DCT coefficient (bpuc).

[12] According to Equation (2.7) if $q_i' \neq 2 \times q$ and the equality (2.5) still holds then not all odd coefficient values are contributing but only the ones that can be found by $c' = (2c+1)\frac{q_i'}{2g}$ for all $c_i \in \mathbb{Z}$.

Algorithm 2 Simulated Perturbed Quantisation

Require: $\mathcal{I}$ - the original JPEG image with quantisation table Q

procedure (SIMULATED) PQ
  Set $Q' = 2 \times Q$
  $\mathcal{I}' \leftarrow$ image $\mathcal{I}$ recompressed with Q'
  $S \leftarrow$ indices of all odd coefficients c of image $\mathcal{I}$
  repeat
    Choose i from S at random without replacement
    if $c_i \in \mathcal{I}$ and $c'_i \in \mathcal{I}'$ satisfy equation (2.7) **then**
      Set $c'_i = c_i + 0.5$ or $c'_i = c_i - 0.5$ ▷ with probability of 0.5.
    end if
  until reached the length of payload
end procedure

In practice, images such as shown in Figure 2.5 (b) have higher capacity than images such as in Figure 2.5 (a).

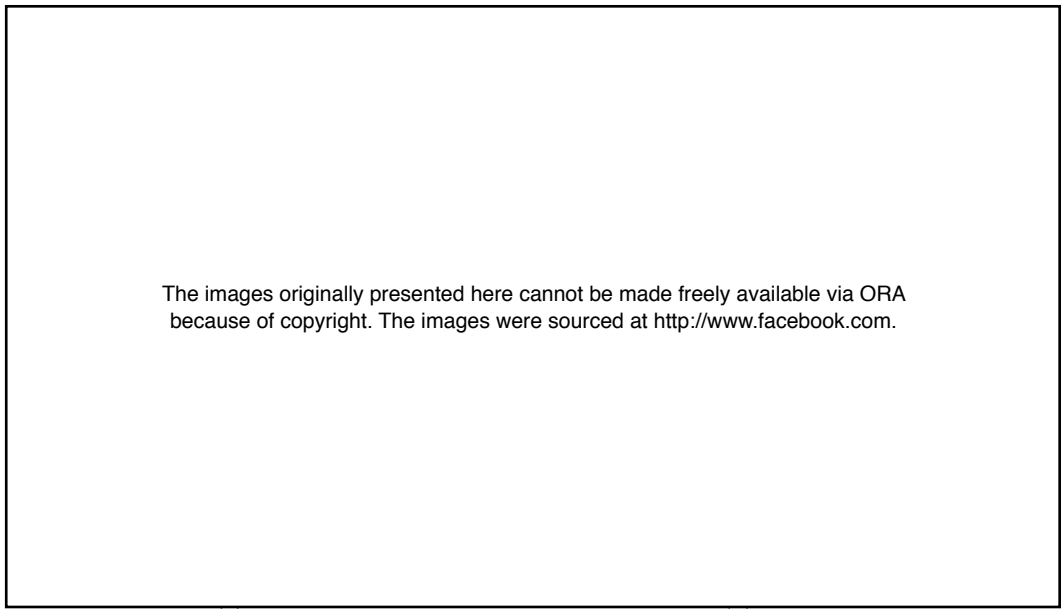


Figure 2.5: Example images for two extremes of PQ capacity from the Facebook data set: minimum (left) and maximum (right).

Risk of JPEG compression library leak

It is known that a classifier's accuracy in an experimental environment can be artificially boosted by difference in the compression libraries that were used to create cover and stego images [67]. In the case of sPQ this is not an issue because we are forced to re-compress to a lower quality factor to create both cover and stego images from a given precover for training and therefore the effects of the JPEG compression library will be the same for both classes. It is true, however, that if one was to inspect images created by such an algorithm in practice, the non-standard quantisation table, which can be found from the JPEG headers or estimated, may itself serve as a leak and be treated as a warning sign that the image may have been tampered with using steganographic embedding.

In the case of nsF5 no re-compression is performed because we are altering already quantised coefficients and therefore no library leak will occur. This can be easily checked by performing a zero-operation embedding (i.e. just rewriting all DCTs with their original values) and comparing the output file to the original.

We therefore conclude that there should be no risk of boosting the classifier performance through detecting the differences in JPEG compression in addition to the differences in presence or absence of a stego payload.

Any residual effects of the original JPEG library used at image creation (and in some cases perhaps two or more JPEG libraries if the social networking website re-compressed the images) should be erased.

2.6.3 Usage in experiments

In our experiments we focus on the two embedding schemes which were covered in detail above: the nsF5 embedding scheme [41] and the PQ embedding scheme. In all experiments we are using simulated versions of their embedding operations (labelled "nsF5" and "sPQ") with two embedding rates each: 0.05 and 0.1 bpnc, and 0.2 and 0.4 bpuc respectively. We chose these rates because they gave the most meaningful detection figures in our preliminary experiments. If the accuracy is too high or

too low, then the observed difference in the detection rates may be dominated by the noise introduced by embedding of random payloads and other factors affecting estimation (see Section 5.3.3 for more details) instead of the true differences in image sources.

Chapter 3

Features

Features engineering is the main area of research in steganalysis. In this chapter we first give an overview of the common characteristics of steganalysis feature sets, followed by brief literature review. We conclude with a detailed description of the CC-C300 feature set which was adapted for experiments in this thesis.

3.1 Introduction

Early steganalysis detectors were targeted to specific steganography algorithms and were derived analytically from an elaborate analysis of statistical artefacts introduced by early steganography schemes. They were often based on a single discriminating statistic, which was a function of images or their transforms taking certain values in the case of stego images, and other values in the case of innocent cover images [56]. An example of such a targeted attack was the Sample Pairs detector [54] which was based on known flaws of LSB-replacement steganography algorithm - the symmetry that it introduced into the histogram of pixel values. Many detectors in this style have been proposed, including but not limited to Sample Pairs Analysis (SPA) and its generalisations [54], Weighted Stego-image analysis (WS) [15] and the Chi-square attack [119]. In rare occasions, e.g. LSB-replacement, they remain to be state-of-the-art, because the weakness they exploit is so prominent.

A different approach is to use a collection of more general statistics, called features, and to train a machine learning classifier to distinguish the features corresponding

to cover and stego classes respectively. In practice features may take different forms - from image quality metrics to unions of second-order statistics of DCT coefficients appearing in specific pairs of modes. By construction, such features are more general in scope than the features used in targeted steganalysis and can therefore be used to detect one of many different steganography algorithms. In the majority of the research literature that uses this approach the modus operandi is to train a binary classifier using a set of examples represented as features, with each example corresponding to one of the two classes - typically cover and a class of images embedded with a specific (known) steganography algorithm, fixed payload and other control variables that comprise a source as discussed[1] in Chapter 1.

Naturally, features for spatial-domain and JPEG-domain images differ although it has been shown that combinations of features from both domains tend to improve classifiers' accuracy in JPEGs [74, 71].

Features should provide as full a statistical description of the cover objects as possible [58]. However it has been argued that features that perfectly model the cover objects from one source do not exist [15]. On the other hand it has also been shown that capturing a wider variety of different statistics allows us to build highly accurate detectors. Ultimately the aim of features engineering is to find the best possible separation between classes, however as we show this may not be enough for practical steganalysis of real-world sources.

3.2 Main concepts

Given a mapping (feature extraction) function ϕ, an image $\mathcal{I}$ from source $\boldsymbol{\chi}$ is mapped to a d-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^d$. The dimensionality d is independent of the size of $\mathcal{I}$. One of the reasons steganalysis is difficult is because the embedding changes are very small in their magnitude, and their count, compared to the image content. For this reason for many feature sets the mapping ϕ oper-

[1] We will continue this discussion throughout this thesis.

ates not on the image itself but on its transform[2]. Several transformations can be used to improve the signal-to-noise ratio between the stego signal and the content noise which aids the separation between classes. They are filtering and Cartesian calibration and are discussed in more detail in the next sections.

3.2.1 Filtering

In the JPEG domain, the features are calculated either directly from quantised coefficients or from their residuals after simple directional filtering. An example of a horizontal (x-axis) filtering used in many feature sets including Subtractive Pixel Adjacency Matrix (SPAM) [94] and JPEG Rich Model (JRM) [71] is the following simple kernel:

$$K_h = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

More complex kernels, up to 5-by-5 in size, are used in the spatial-domain. A filtered image is found by means of convolution between the kernel and the image. In some instances, mostly in the spatial domain, the resulting filtered image is subtracted from the original to compute residuals:

$$X' = K * X - X \ , \tag{3.1}$$

where $*$ denotes the convolution operation and X denotes the image. The output residuals X' are then used to compute the necessary features. In the JPEG domain, filters are applied to quantised coefficients and the result of this operation, the filtered coefficients, are typically used directly for subsequent computing, unlike in Equation 3.1 where a difference is taken with the original.

Several modern feature sets employ this technique. For example, the JPEG Rich Model [73] uses as many as six different filters for parts of its feature set.

3.2.2 Cartesian calibration

Another technique that is often used in JPEG-domain steganalysis to improve signal-to-noise ratio is called Cartesian calibration. Cartesian calibration is a popular technique for improving steganalysis features by providing each feature with a calibrating

[2]N.B. this is not a classical image transform such as the discrete cosine transform, but a more general operations such as filtering or cropping; see further text for details.

feature calculated from a transformation of the target image. It was originally introduced to steganalysis in [37].

The procedure of calibration is straightforward. Let X' be a stego image created from a cover X. The aim is to create calibrated image X'_c such that it is a reference of the cover X. This can be achieved in the JPEG domain by desynchronising the DCT blocks which largely deletes the stego noise which was added to X with the embedding changes when creating X'. Desynchronisation is often done by applying small transformations to a decompressed X' such as cropping or rotation. One of the popular transforms for calibrating JPEGs is a horizontal and vertical crop by 4 pixels in both directions. The cropping moves the 8×8 DCT grid out-of-sync with the content and therefore desynchronises it from the previous compression. Cropping is performed in the spatial domain, i.e. involving decompression, cropping transform and subsequent re-compression. Technically we can crop between 1 and 7 pixels (or any multiple thereof if applicable), but 4 is the most commonly used value, although it has been shown that other values such as 2 can sometimes yield much better results, e.g. in [67].

Although it has been shown [69] that the features $x'_c = \phi(X'_c)$ rarely correspond to features calculated from the original image $x = \phi(X)$, it has been used in a large number of publications with positive results. Early works used the difference $x - x'_c$ as features, but the positive impact of calibration on the accuracy of the detector was found to be especially pronounced [69] when the reference features extend the original feature vector calculated from the non-calibrated image therefore increasing the dimensionality and predictive power of the original feature vector two-fold by using the union of the two sets of features.

Most modern JPEG-domain feature sets use calibration, which is marked by "CC-" prefix to their name.

3.2.3 Histograms, co-occurrences and estimates of conditional probability mass functions

The feature vector itself typically consists of unions of models which are generally based around counting statistics such as histograms, co-occurrence matrices or estimates of conditional probability mass functions. Pevny writes [93]: "The rationale behind [using adjacency histograms] is that neighbouring pixels in digital images are correlated, which is caused by the smoothness of our world and by the usual image processing". These statistics can be calculated either directly from pixels/DCT coefficients or from their residuals after filtering as discussed above.

Let us take the co-occurrence matrix as an example of such a model. It measures an estimate of the distribution of co-occurring values in a given pair of relative pixel positions or DCT modes. For a pair of modes X and Y and the set $\mathcal{S}^{XY}$ of all instances of this pair of modes in an image[3] $\mathcal{I}$ we can define a bin h_{ij} of the co-occurrence matrix h as:

$$h_{ij} = \sum_{(x,y)}^{\mathcal{S}^{XY}} \delta \text{ s.t. } \begin{cases} \delta = 1, & \text{if } i = \min(|x|, T) \times \operatorname{sgn}(x),\ j = \min(|y|, T) \times \operatorname{sgn}(y) \\ \delta = 0, & \text{otherwise,} \end{cases} \tag{3.2}$$

where the size of the co-occurrence matrix is restricted to a small number by a parameter T, for the reasons of keeping small dimensionality (the number of bins is $(2T+1)^2$) and/or keeping the bins well populated. The resulting co-occurrence matrix is often normalised such that the sum of all bins is 1.

The selection of pairs of modes of interest differs from one features extraction algorithm to another. Some, like the CC-C300 features set, restrict it to a pre-designed set of pairs, which aims to capture only the most significant dependencies between two modes. Others use a very dense collection of modes, such as taking all possible pairs of adjacent modes which appear in the low frequency sub-bands of the DCT block (the JRM feature set).

[3]The size of $\mathcal{S}^{XY}$ depends on the relative position of the two modes. If they appear in a single block (intra-block dependency) then this is equal to the number of blocks; otherwise (inter-block dependency) there may be up to two less blocks per row (or column) that contribute.

It is straightforward to see how features in this style get costly rapidly in terms of both the size of the feature set and the time required to compute it. Many domain-specific heuristics have been devised to keep the dimensionality down. One was shown in Equation (3.2) as T, which is the truncating parameter that keeps the size of the co-occurrence matrix small. On the other hand, when T is small, features are less complete. Other methods exist, including the assumption of symmetry about zero that generally holds in the histogram of certain residuals from a single block , which allows to averaging over the two "sides" of the symmetry [51].

The general aim of using different co-occurrence matrices (using, for example, different residuals) is to achieve greater diversity of the statistics they capture, which allows for building richer models of images.

3.3 Challenges in steganalysis data

There exist many challenges in steganalysis feature design. In this section we give a brief overview based on two major groups: general challenges and those directly related to cover source mismatch.

3.3.1 General

First, a major challenge for steganalysis that uses machine learning is in the fact that we are trying to learn from *something that is meant to be hidden*. The signal for the positive class is very small in magnitude and length compared to the image content. Filtering and cartesian calibration are both used to improve signal-to-noise ratio (stego signal to content noise), which appears to help by increasing overall accuracy, therefore the progress in this direction is needed.

The second challenge comes with the uncertainty about the positive class. Whilst it is generally acceptable to assume that the target embedding algorithm and payload size are known, it is unlikely to be the case in many real-world situations. Several solutions have been proposed including quantitative steganalysis (regression) to account for different sizes of payload [97], one-class [85, 92] and multiclass [92, 80, amongst others] steganalysis, banks of experts [103] and unsupervised classification

techniques [60].

Another challenge that has been discovered recently, with the introduction of large feature sets, concerns the speed of their extraction and their usage in training classifiers commonly used in steganalysis such as the kernel Support Vector Machine. The latter has been approached with divide-and-conquer techniques such as ensemble classifiers [70]. In this thesis we go one step further in complexity reduction and use linear classifiers with their obvious speed advantages, which is desirable, as we show later, when dealing with real-world data (and can be improved further with techniques for parallelisation such as Delayed Stochastic Gradient Descent [77] (not employed in our experiments)).

Finally, there is the challenge of features correlation. No single steganalysis feature appears to carry significant discriminative information and as such they need to work together to allow for good performance. Many methods that assume the opposite such as feature selection, L1-sparsification and decision-trees-based methods may not work very well, especially with the more modern feature sets.

3.3.2 From cover source mismatch

Steganalysis features are also known to be very sensitive to the cover source. Several authors reported the negative effect that this may have on steganalysis [4, 20, 40, 59, 94, 44, 62]. The "source" in this context appears to be defined not only by the image acquisition scene (content), but also by camera model, image processing operations and other factors. Some of these have been found to suppress the noise that exists in digital images (e.g. blurring, denoising, lossy compression) whilst others have no effect on noise, but may introduce statistical artefacts such as spikes and valleys in the histogram (e.g. contrast adjustment and gamma correction) [35]. Some of these affect statistical dependencies between pixels which in turn affects steganalysis features.

In steganalysis, it is common to think of image content as "noise" and steganographic embedding as "signal". Based on this the features are often said to need

to be sensitive to steganographic embedding but not to the content (which can be thought of as a function of the above discussed properties). More precisely they should suppress the "noise" (content) and increase the signal-to-noise ratio. Despite filtering and cartesian calibration (which are largely designed to remove the effects of image content and consequently the dependence on source) as well as some efforts in controlling for external variables such as image size and quality factor the image content (and hence source due to its intrinsic influence on the content) still has a strong effect on steganalysis features.

Several authors have studied the effects of individual variables on steganalysis features [92, 15, 20, 36, amongst others]. One of the best understood is the effect of cover size on its capacity (and thus also steganalysis) which was devised to be governed by the Square Root Law [55]; the capacity of a cover scales with the square root of its size. Some basic results have been found for other isolated cover properties. For example in [72] Kodovsky et al. study the effects of different image resizing algorithms on detection performance and devise a predictor which is based on the cover size and the scaling factor and other properties of the resizing algorithm. It is easy to see how such a predictor can be used to choose a matching detector. The success of such a framework rests on the assumption that bias introduced by a particular cover property is predictable and that it is possible to isolate a particular property responsible for such bias. This may or may not be possible to do for all properties of covers - as mentioned earlier, covers are known to be one of the most difficult parts of steganalysis system to control for.

It was recently proposed [62] that different sources may have different centres in the feature space. A geometric technique for source mismatch correction based on centering and calibrating for the direction and rate of feature movement under payload was shown to help with overcoming the cover source mismatch problem. This style correction requires sufficient data from the target source in case of centering or from a number of cover sources (not necessarily including target) in case of direction and rate to estimate parameters for the correcting transformations. Its advantage

is that it does not require the knowledge of bias introduced by individual characteristics of sources.

Whilst these are very encouraging results, not many results like these exist yet and progress in this direction is slow. It is easy to see that the real-world images come with an abundance of different properties that may affect steganalysis. Practical detectors must be robust to these - they must generalise well to unseen sources. In this thesis (Chapters 6 and 7) we study how to measure the general effects of cover source mismatch and simple ways to mitigate them.

3.4 Literature review: summary of features sets

The dimensionality of steganalytic features has been increasing over years. Whilst early attacks were using as little as one feature, modern detectors use models built from tens of thousands of features. Despite the large dimensionality, the process of features design involves careful manual feature engineering. Numerous feature sets have been proposed for both JPEG and spatial domains.

Extending the excellent table from [15] with the newest developments in steganalysis features we can clearly see the continuing trend for larger feature sets (Table 3.1). The table shows a selection of features sets for both spatial and JPEG domains. It is by no means exhaustive and is given with the aim to demonstrate the trends (towards larger features sets) and basic building blocks (counting statistics). Several combinations of previous feature sets have also been proposed and proven to work well in practice. Notable examples of this are CC-PEV features [96] which combined Markov [108] and DCT features [34] and are still used to date. CDF features [74] combined spatial-domain features (SPAM [94]) with JPEG-domain features (CC-PEV [96]). J+SRM [71] combined versions of CC-JRM and SRM features.

With a clear trend towards capturing more different statistics one can envisage even larger, more elaborate feature sets in the future. In this thesis we employed the first feature set that arguably started this trend, the CC-C300 features from [70], details of which are given in the next section.

Table 3.1: Selected steganalysis detectors for image steganography. Copied verbatim from [15] and extended.

year	author	description	domain	num. of features
2001	Avcibas et al. [3]	Spatial domain and spectral quality metrics	Spatial	26
2003	Lyu and Farid [84]	Moments of Fourier transform coefficients and size of predictor error	Spatial	72
2003	Harmsen and Pearlman [46]	Smoothness of histogram	Spatial	3
2004	Fridrich [34]	DCT histogram measures, blockiness, coefficient co-occurrence	JPEG	23
2006	Goljan et al. [43]	Moments of residuals of wavelet coefficients after Wiener filter denoising ("WAM")	Spatial	27
2006	Shi et al. [108]	Intra-block difference histograms of absolute DCT coefficients	JPEG	324
2007	Pevny and Fridrich [96]	DCT-based global and local histograms, inter-block co-occurrences, blockiness measures and based on features from [108] and [34] ("PEV")	JPEG	274
2008	Chen and Shi [24]	Inter- and intra-block conditional-probability-based features	JPEG	486
2010	Pevny et al. [94]	Conditional probability estimates of triples of adjacent pixel differences along eight directions ("SPAM")	Spatial	686
2011	Liu [79]	Calibrated inter- and intra-block co-occurrences	JPEG	216
2011	Kodovsky et al. [70]	Inter- and intra-block co-occurrence based on top 300 pairs of modes selected using an estimate of mutual information ("CC-C300")	JPEG	48 600
2012	Kodovsky and Fridrich [71]	JPEG-domain Rich Model ("JRM") built from co-occurrences calculated on filtered DCT blocks using dense selection of pairs of modes	JPEG	22 510
2012	Kodovsky and Fridrich [71]	Spatial-domain Rich Model ("SRM"): co-occurrences from many different filtered residuals of an image	Spatial	34 671
2012	Kodovsky et al. [73]	Compact Rich Model: symmetrised version of [71] ("CF*")	JPEG	7 850
2013	Holub et al. [51]	Projection SRM: histograms of random convolutions applied to filtered images	Spatial	12 870 / 34 320

3.5 Example feature set: C300 features

The CF family of features was first presented in [70]. It is based on the idea of finding pairs of coefficients that predict each other in cover images; those dependencies may be broken by embedding. Technically, this is done by capturing statistical dependencies between DCT coefficients that exhibit high mutual information. Mutual information (MI) is a measure of the amount of information that one random variable provides about another random variable and can be calculated according to Equation (3.3). This selection does not depend on the embedding method as only cover images are used.

The features are found in two steps. First is a design step, which is similar in its procedure to filter-based feature selection methods used in Machine Learning to reduce the size of a feature vector [6]. Given an image $\mathcal{I}$, we truncate all of its quantised coefficients such that any values outside the range $[-T, T]$ are replaced with $-T$ or T, whichever is closer. Let X and Y represent a pair of DCT modes where coefficients take values x and y, $x, y \in [-T, T]$. Then we can calculate an estimate of mutual information $I(X, Y)$ for that pair as follows (symmetrical for X and Y):

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} \hat{p}(x,y) \log \left(\frac{\hat{p}(x,y)}{\hat{p}(x)\,\hat{p}(y)} \right) , \tag{3.3}$$

where $\hat{p}(x)$ and $\hat{p}(y)$ are probability estimates of coefficients X and Y taking values x and y respectively and $\hat{p}(x, y)$ is an equivalent estimate of their joint probability. All probability estimates $\hat{p}$ are computed from a sample of cover images or their random crops. A pair of modes (X, Y) is said to have a high dependency if we can measure a high mutual information $I(X, Y)$ for them over a large enough sample of images. This can be repeated for all possible pairs of modes capturing both inter- and intra-block dependencies. Due to the fact that MI is symmetric it is safe to assume that pairs such as ([1,2],[9,2]) and ([9,2],[1,2]) are identical.

The feature dimensionality is controlled using several techniques. First, the DC mode is not used as most embedding methods avoid it. Second, an assumption is made that the mutual information between DCT modes decreases with the increase

in the spatial distance between them, therefore the search can be restricted to a smaller neighbourhood, e.g. a 3-by-3 neighbourhood of DCT blocks. Finally, all pairs of modes are ranked by their MI score and a set $\mathcal{P}$ of top F pairs is selected for the final feature set[4]. To follow in line with previous work [70, 73] we implement the C300 features using 300 pairs ($F = 300$).

The second step is feature extraction. Here, the feature vector $\boldsymbol{x} \in R^d$ for an image $\mathcal{I}$ is calculated using the top F pairs from $\mathcal{P}$. Each pair $(X, Y) \in \mathcal{P}$ provides one co-occurrence matrix of dimension $(2 \times T + 1)^2$ which is calculated according to Equation (3.2). Often used values for T are 2, 3 or 4 and we use $T = 4$ as per the original publication [70]. When $F = 300$ and $T = 4$ this is a C300 feature set of 24300 dimensions. Its calibrated version has been used as benchmark in [70, 73].

It is possible to use metrics other than MI to select most informative pairs (or triples, etc) of modes. In [70] the authors suggest the Fisher Linear Discriminant criterion or the Maximum Mean Discrepancy (MMD) as alternatives. It is also possible to estimate MI from different sets of images, perhaps targeting a specific source. We performed basic experiments to measure the mutual information on different data sets and cross-referenced the performance of those features across those data sets. Even though the sets of features differed quite dramatically between different data sets, the classifier performance did not change significantly. This is interesting because it points to the trend in modern steganalysis feature design which is based on the idea of capturing many different statistics (see [51] and [71] for examples). However, it has also been argued, [71] that the *diversity* of such statistics is as important, if not more important, than the overall number of features that that are based on the *same* statistic.

This feature set strikes a good balance between training a sensitive detector of non-adaptive embedding schemes and the speed with which it can be extracted. At the time when our early experiments for this dissertation were conducted it gave

[4]N.B. Once the set $\mathcal{P}$ has been found from a sample of cover images, it remains fixed for all images and we can reuse it for images from other sources (see further text).

the most sensitive detector of nsF5 (and many other embedding algorithms) compared to all other state-of-the-art features [70]. More sensitive feature sets have been proposed since, at rate of roughly one every six months. They are usually grouped under the umbrella of large features that provide "rich" models and generally follow in the same direction capturing as many different statistics as possible. However, their current implementations are prohibitively slow to incorporate them into our experiments[5]. Two notable feature sets are the JPEG domain Rich Model (CC-JRM) and the Projection Spatial Rich Model (PSRM, also works for JPEG images) due to Kodovsky et al. [71] and Holub et al. [51] respectively. CC-JRM involves additional filtering and calculating more different co-occurrence matrices using a denser selection of pairs of modes. Unlike other feature sets, PSRM applies a random filter (through convolution with a random kernel, the height and width of which is determined uniformly on the interval [1..8] and entries are Gaussian) to image residuals before calculating the histogram; this may pick up long range dependencies as a large neighbourhood of pixels is considered. Both feature sets yield better classification but slower feature extraction - it has been noted [57] that, for example, PSRM features require on the order of one million floating point operations per pixel.

[5]For example, it takes approximately 6 seconds to extract JRM features from one image in our data set, which makes it infeasible for us to use them for experiments with millions of images.

Chapter 4

Classifiers

The aim of a steganalytic classifier is to learn to distinguish features from cover objects from features from stego objects. Its accuracy depends on features (how well they model the separation between classes) and its expressive power (the class of functions that can be represented by it). Features provide a fixed representation. Many machine learning algorithms are designed to work in this given representation, these are often linear algorithms, but also include some more powerful non-linear variants such as Neural Networks and Nearest Neighbour classifiers. Other classifiers, such as kernel-based algorithms, are capable of changing the given representation and mapping it to an embedded higher-dimensional space. Such algorithms, in particular the Support Vector Machine based on the Gaussian kernel, have been used extensively in steganalysis. It is a common belief that non-linear algorithms perform better in most binary steganalysis applications. However, non-linearity comes at a cost of increased complexity. It may therefore be beneficial to compare the performance of different types of classifiers in practice. For this thesis we consider several algorithms from both groups.

4.1 Binary classification

Let $\boldsymbol{x}_1 \dots \boldsymbol{x}_n$ be a set of n training examples drawn as an independent identically distributed sample from a given domain[1], $\boldsymbol{\chi}$. These examples belong to one of the

[1]We assume that all of Alice's cover and stego images come from this domain.

two classes and are given labels $y_1 \dots y_n$. Here the two classes correspond to instances of cover and stego images with payload from a known embedding operation and of a known size. Each class has a distribution over the domain $\boldsymbol{\mathcal{X}}$ denoted P_c and P_s respectively.

Given a fixed hypothesis class (classifier) $\boldsymbol{\mathcal{H}}$ the learning task is to find a hypothesis $h \in \boldsymbol{\mathcal{H}}$ which given an unseen test example $\boldsymbol{x}$ outputs its prediction $\hat{y}$ of the true label y.

Many classifiers can be described as the triplet $\langle h(\boldsymbol{x}), \mathcal{L}(\boldsymbol{\theta}), \mathcal{A} \rangle$ which is composed of a decision function (which can be either an intermediate or the final hypothesis) $h(\boldsymbol{x})$, an objective function $\mathcal{L}(\boldsymbol{\theta})$ and an optimisation algorithm $\mathcal{A}$. Mathematically the triplet can take either the *primal* or the *dual* form, depending on how the decision boundary is expressed. We will use this convention in the forthcoming discussion where possible to describe the classification algorithms used in this thesis.

Any classifier explicitly or implicitly defines a decision boundary which partitions the example space into classes. For example, linear classifiers define the decision boundary as a hyperplane whose position in the feature space spanned by the data is controlled by a set of parameters $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^T \boldsymbol{x} + b = 0 \; . \tag{4.1}$$

The parameters $\boldsymbol{\theta}$ need to be estimated from the training data. The idea is to learn $\boldsymbol{\theta}$ such that our triplet $\langle h(\boldsymbol{x}), \mathcal{L}(\boldsymbol{\theta}), \mathcal{A} \rangle$ produces maximally accurate predictions on unseen data.

Each prediction that is made by the model defined by the parameters $\boldsymbol{\theta}_t$ at time t can be associated with a cost (loss). We can therefore write a loss function as a function of three arguments: $\ell(\boldsymbol{x}, y, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ uniquely defines the hypothesis h and y is the true class label of example $\boldsymbol{x}$, often being modelled as the 0/1 or -1/1 binary response for negative/positive examples depending on the mathematical con-

venience[2]. An example of a very simple loss function is the 0-1 loss, which assigns the cost of 0 to each correct decision and the cost of 1 to each incorrect decision:

$$\ell_{0\text{-}1}(\boldsymbol{x}, y, \boldsymbol{\theta}) = \begin{cases} 0 & \text{if } \overbrace{h(\boldsymbol{\theta}, \boldsymbol{x})}^{\text{decision}} = \overbrace{y}^{\text{true label}} \\ 1 & \text{otherwise .} \end{cases}$$

Objective function $\mathcal{L}(\boldsymbol{\theta})$ provides a strategy for choosing parameters $\boldsymbol{\theta}$. The strategy of Empirical Risk Minimisation looks for values of $\boldsymbol{\theta}$ which minimise the average error (also called empirical risk) over the training set:

$$\mathcal{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \ell_{0\text{-}1}(\boldsymbol{x_i}, y_i, \boldsymbol{\theta}) \ . \tag{4.2}$$

It is however impractical (NP-hard) to compute this minimisation due to the combinatorial nature of the problem and the fact that 0-1 loss is non-differentiable [90]. Instead we generalise the above objective to minimise over the average of some more general loss function which approximates the 0-1 loss. We will discuss examples of such functions in greater detail below.

Simply following the empirical risk minimisation strategy often leads to overfitting and therefore the objective function usually takes the form of a weighted sum of two terms - the average loss and a penalty (regularising) term (λP):

$$\mathcal{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{x_i}, y_i, \boldsymbol{\theta}) + \lambda P \ . \tag{4.3}$$

Many convex objective functions exist which can be minimised with popular convex optimisation algorithms such as the gradient descent (the $\mathcal{A}$ part of our triplet). Others, like some versions of the SVM, require more complex optimisation algorithms that take into account certain constraints that bind the solutions of the problem to a specified range, which is parametrised by the user.

Both functions $h(\boldsymbol{x})$ and $\mathcal{L}(\boldsymbol{\theta})$ can be written in primal or dual forms. Each form

[2] All classifiers implemented in this thesis use $y \in \{-1, 1\}$ labels; other classifiers use $y \in \{0, 1\}$, for example Logistic Regression, which estimates $\hat{y}$ as class-belonging probability.

dictates how $\boldsymbol{\theta}$ is expressed. In the primal form $\boldsymbol{\theta}$ is a vector $\boldsymbol{w}$, called the weight vector. The primal version of the decision function often takes the simple form:

$$h(\boldsymbol{x}) = \text{sgn}(\boldsymbol{w}^T\boldsymbol{x}) \ ,$$

where $(\boldsymbol{w}^T\boldsymbol{x})$ is the inner (dot) product between the weight vector $\boldsymbol{w}$ and feature vector $\boldsymbol{x}$ describing the input example and $\text{sgn}(\cdot)$ is a sign function[3].

For the dual form the Representer theorem [47] allows us to express $\boldsymbol{\theta}$ as a weighted linear combination of all n training points:

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \ ,$$

where $\alpha_i \in \mathbb{R}$ represents some weight assigned to example $\boldsymbol{x}_i$. In the dual representation the decision function takes the following form:

$$h(\boldsymbol{x}) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i (\boldsymbol{x}_i^T\boldsymbol{x})\right) , \tag{4.4}$$

which is essential for enabling non-linear classification boundaries. This is due to the fact that we can define a function $k(\cdot,\cdot)$, which replaces the dot product $\boldsymbol{x}_i^T\boldsymbol{x}$ in the dual representation:

$$h(\boldsymbol{x}) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i k(\boldsymbol{x}_i^T\boldsymbol{x})\right) \ . \tag{4.5}$$

The function k is called the kernel and in the case of a linear classifier it is simply the dot product: $k(\boldsymbol{x}_i, \boldsymbol{x}) = \boldsymbol{x}_i^T\boldsymbol{x}$. Examples of non-linear kernels are given in Section 4.3.1.

Whilst in theory primal and dual forms provide "equivalent ways of reaching the same result" [22] they have their advantages and disadvantages when it comes to practical applications. The primal form is generally used in linear binary classifiers and is good for a small memory footprint and quick computation of the decision

[3]N.B. We drop the bias term b (Equation 4.1) by assuming extra w_0 and x_0 terms in $\boldsymbol{w}$ and $\boldsymbol{x}$ respectively and that $x_0 = 1$.

function (predictions). The dual form can be faster when the solution is sparse (i.e. the solution is computed from a subset of the training data, usually support vectors). More precisely, the complexity of computing $\boldsymbol{w}$ grows (approximately quadratically) with the number of features, whilst the complexity of computing $\boldsymbol{\alpha}$ grows (also approximately quadratically) with the number of training examples.

Below we describe our linear classifiers using the primal form. For kernel-based algorithms, solving the problem in primal form is often an approximation of the explicit kernel solution provided by the dual form. We discuss this point in more detail at the end of Section 4.3.1 and the algorithms will be described using the dual form.

Those learners, which can be expressed in primal form and have convex objective functions expressed as a sum of differentiable loss functions (as per Equation 4.3), can be optimised with stochastic gradient descent (SGD). SGD approximates the regular gradient descent update by taking the gradient of the loss function at the current example. The convexity guarantees convergence and the differentiability allows for calculating the gradient of the loss function[4]. In SGD, the gradient of the loss function $\nabla_{\boldsymbol{w}}\ell(\boldsymbol{w})$ with respect to $\boldsymbol{w}$ forms the training update:

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \nabla_{\boldsymbol{w}}\ell(\boldsymbol{w}_i, \boldsymbol{x}_i, y_i) \ , \tag{4.6}$$

where i is the index of a current training example. If we loop over all training examples using this update our stochastic optimisation of the objective function will eventually approach convergence [17], having learnt the "optimal" weight vector $\boldsymbol{w}$.

In the following sections we will use the triplet $\langle y(\boldsymbol{x}), \mathcal{L}(\boldsymbol{\theta}), \mathcal{A}\rangle$ to simplify the discussion of the algorithms employed in this thesis.

Regularisation of the objective function

Minimising the loss function alone often leads to overfitting. Overfitting is the condition when the classifier trains a model which is too specific to the training data

[4]N.B. Some loss functions that we will study will not be continuously differentiable but mathematical tricks such as subgradients or approximations of those functions can be used instead.

set and will fail to generalise well. This is especially true of training on smaller data sets and manifests itself in large values being assigned to the parameters ($\boldsymbol{\theta}$) in the trained model. Regularisation is normally used as the primary defence against overfitting. This is done by including a penalty term in the loss function (shown as λP in Equation 4.3). There exist a number of different regularisation functions, some of which are more suited to some situations than others. An example of an often used penalty is based on the function $\|\boldsymbol{w}\|_p$, a p-norm of the weight vector $\boldsymbol{w}$ [11]:

$$\|\boldsymbol{w}\|_p = \left(\sum_i |w_i|^p \right)^{\frac{1}{p}} . \tag{4.7}$$

We refer to $\|\boldsymbol{w}\|_2$ as the ℓ^2 norm and it is an often used regulariser (sometimes written as $\|\boldsymbol{w}\|$). It has the effect of keeping the absolute values of the weights of the classifier from increasing in proportion to the number of iterations and hence overfitting the training data. Likewise, $\|\boldsymbol{w}\|_1$ is the ℓ^1 norm and works by a similar principle with a useful side-effect of causing some weights that correspond to unimportant features to be exactly zero, hence providing sparser solutions. It is possible to use one or both of these regularisers to improve generalisation capabilities of a classifier, for example in the following manner [76]:

$$\mathcal{L}(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} \left(\ell(\boldsymbol{w}, \boldsymbol{x_i}, y_i) + \lambda_1 \|\boldsymbol{w}\|_1 + \frac{\lambda_2}{2} \|\boldsymbol{w}\|_2^2 \right) , \tag{4.8}$$

where λ_1 and λ_2 are hyperparameters set by the user.

4.2 Linear classifiers

In this section we give some basic details of the classifiers employed for most of the experiments presented in this thesis.

4.2.1 Average Perceptron

The Average Perceptron [31] is a version of a classifier known as the Perceptron with a weight vector averaging step which improves its stability through acting as a regulariser. By construction this is an online classifier, which means it can be trained one example at a time.

The Perceptron, which was first introduced by Rosenblatt in [31], aims to minimise the total number of misclassified training examples. Following our framework from before we define the loss function, the objective function and the update rule for the Perceptron as follows.

Perceptron loss

$$\ell_{Perceptron} = \max\left(0, -y(\boldsymbol{w}^T\boldsymbol{x})\right) \tag{4.9}$$

Perceptron objective function

$$\mathcal{L}_{Perceptron} = \arg\min_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n} \max\left(0, -y_i(\boldsymbol{w}^T\boldsymbol{x}_i)\right), \tag{4.10}$$

which is approximated with a stochastic update at each step (example) i:

Perceptron update

$$\begin{aligned}\boldsymbol{w}_{i+1} &= \boldsymbol{w}_i - \nabla_{\boldsymbol{w}} \max(0, -y_i(\boldsymbol{w}_i^T\boldsymbol{x}_i)) \\ &= \begin{cases}\boldsymbol{w}_i + y_i\boldsymbol{x}_i & \text{if } y_i(\boldsymbol{w}_i^T\boldsymbol{x}_i) < 0 \\ \text{unchanged} & \text{otherwise .}\end{cases}\end{aligned} \tag{4.11}$$

The Perceptron mostly follows the normal SGD update[5] from Equation 4.6. The update happens when a new input example (indexed by i) is assigned the wrong label. On linearly-separable data the Perceptron is guaranteed to converge, that is it will find a set of parameters $\boldsymbol{w}$ that minimises its objective function $\mathcal{L}_{Perceptron}$. If the classes overlap in the original representation used to encode our examples $\boldsymbol{x}$ then convergence is not guaranteed. It is therefore essential to monitor the progress of its training by testing on validation data. It is straightforward to see that when stopped the Perceptron's decision boundary may not be optimal. This problem is overcome using the Average Perceptron. In the Average Perceptron the update includes the regularisation step, where the average weight vector is updated at every input example:

$$\boldsymbol{w}_{avg} = \boldsymbol{w}_{avg} + \boldsymbol{w}_i \ . \tag{4.12}$$

[5] A simplification is made here - we assume the derivative of $\ell_{Perceptron}$ at point $y_i(\boldsymbol{w}_i^T\boldsymbol{x}_i)) = 0$ is equal to zero, whilst its actual value is undefined.

It was shown [31] that such a weighted combination of weight vectors is guaranteed to produce better classification than the regular Perceptron algorithm in data which is not linearly-separable.

The vector $\boldsymbol{w}_{avg}$ is used in the final decision function to predict the label of test example $\boldsymbol{x}$:

$$h(\boldsymbol{x}) = \text{sgn}(\boldsymbol{w}_{avg}^T \boldsymbol{x}) \ .$$

In its simplest form the online version of Averaged Perceptron does not have parameters. If trained in batch setting it has the implicit parameter of the number of iterations needed for early stopping.

4.2.2 Linear Support Vector Machine

The Support Vector Machine is a classic classifier that was developed by Vapnik as a consequence of the analysis of the Empirical Risk Minimisation principle [114]. It guarantees convergence to an optimal decision boundary, one that gives the maximum margin, for a given training data set assuming the correct choice of regularisation parameters. To achieve this SVM finds a maximum margin separator given a set budget for misclassification[6]. The margin can be taken as $\frac{2}{\|\boldsymbol{w}\|_2}$, but the objective is instead normally written in terms of $\frac{1}{2}\|\boldsymbol{w}\|_2^2$ which makes it a convex function in $\boldsymbol{w}$. Using hinge loss in the objective from Equation 4.8 and omitting the ℓ^1 norm gives us the soft margin SVM in primal form. The hinge loss encodes the logic for sparsity and the soft margin (see Section 4.3.1). The update takes place when the training example $\boldsymbol{x}_i$ violates the functional margin $y_i(\boldsymbol{w}^T\boldsymbol{x}_i) = 1$. Otherwise only regularisation is performed. We present here the hinge loss with the associated objective function and online training update:

Hinge loss

$$\ell_{SVM} = \max\left(0, 1 - y(\boldsymbol{w}^T\boldsymbol{x})\right) \ , \tag{4.13}$$

[6]The budget is specified using a hyperparameter, normally expressed as C (see Section 4.3.1, but the regularisation parameter λ used in the hinge loss effectively does the same job

SVM objective using hinge loss

$$\mathcal{L}_{SVM} = \arg\min_{\boldsymbol{w}} \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{n}\sum_{i}^{n} \max\left(0, 1 - y(\boldsymbol{w}^T\boldsymbol{x})\right) \ , \tag{4.14}$$

which can be approximated with SGD update at each step (example) i:

SVM update

$$\begin{aligned} \boldsymbol{w}_{i+1} &= \boldsymbol{w}_i - \eta\left(\lambda\boldsymbol{w}_i + \nabla_{\boldsymbol{w}} \max\left(0, 1 - y_i(\boldsymbol{w}_i^T\boldsymbol{x}_i)\right)\right) \\ &= \boldsymbol{w}_i - \begin{cases} \eta(\lambda\boldsymbol{w}_i - y_i\boldsymbol{x}_i) & \text{if } y_i(\boldsymbol{w}_i^T\boldsymbol{x}_i) < 1 \\ \eta\lambda\boldsymbol{w}_i & \text{otherwise} \ , \end{cases} \end{aligned} \tag{4.15}$$

where λ is the regularisation parameter. We note that a certain degree of regularisation happens even when the current example is classified correctly. This loss function, being a maximum function, does not have a derivative at $1 - y(\boldsymbol{w}_i^T\boldsymbol{x}_i) = 0$ therefore $\nabla_{\boldsymbol{w}} \max\left(0, 1 - y_i(\boldsymbol{w}_i^T\boldsymbol{x}_i)\right) = 0$ is found using the subgradient method.

4.2.3 Fisher Linear Discriminant

Training Fisher's Linear Discriminant is a batch procedure and involves estimating the mean vectors ($\boldsymbol{\mu}_{y=-1}$ and $\boldsymbol{\mu}_{y=1}$) and the covariance matrices ($\boldsymbol{\Sigma}_{y=-1}$ and $\boldsymbol{\Sigma}_{y=1}$) for the two classes using the training data. The decision boundary is given by $\boldsymbol{w}$:

$$\boldsymbol{w} = \left(\boldsymbol{\Sigma}_{y=-1} + \boldsymbol{\Sigma}_{y=1}\right)^{-1}\left(\boldsymbol{\mu}_{y=-1} - \boldsymbol{\mu}_{y=1}\right) \ , \tag{4.16}$$

where superscript -1 means matrix inverse. The decision function $h(\boldsymbol{x})$ is calculated as:

$$h(\boldsymbol{x}) = \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{c}) \ , \tag{4.17}$$

where c is calculated as:

$$\boldsymbol{c} = \frac{1}{2}(\boldsymbol{\mu}_{y=-1} + \boldsymbol{\mu}_{y=1}) \ .$$

In addition to the standard implementation we also implemented a pseudo-online version of this algorithm which estimates the covariance matrices and the means in an online fashion therefore allowing us to process larger training sets. We refer to it as pseudo-online due to the fact that costly operations such as the covariance-matrix inversion are delayed until all training examples have been visited.

We only use the FLD classifier as a baselearner in the ensemble (see Section 4.3.2).

4.3 Non-linear classifiers

On their own linear classifiers can only model linear decision boundaries in the given feature space. There exist a multitude of algorithms for non-linear classification. We discuss two common approaches here. The first uses the kernel trick to enable linear classifiers to work in an expanded representation in which the class separation is better. The second combines classifiers into an ensemble such that non-linear boundaries can be learnt in the original representation.

It is often assumed that steganalysis problems have non-linear decision boundaries. One classifier that became the benchmark for binary steganalysis classification is the kernel Support Vector Machine, which we discuss in the following section and use as a benchmark in our experiments. KSVM is indeed widely accepted as a robust tool for non-linear classification of many different problems. However when the training data and features become too large KSVM becomes inefficient for practical experiments. For this reason we consider other non-linear classifiers which are capable of faster and more memory-efficient computation.

We start this section with an introduction to the kernel SVM. Next, the ensemble classifier is discussed, for which we propose an improvement using online updates. Finally, we discuss a selection of alternatives which are not implemented in this project.

4.3.1 Kernel Support Vector Machine

We give a standard criterion for training kernel Support Vector Machines. First, reformulate the hinge loss to introduce slack variables ξ:

$$\xi = \begin{cases} 0 & \text{if } y(\boldsymbol{w}^T\boldsymbol{x}) \geq 1 \\ 1 - y(\boldsymbol{w}^T\boldsymbol{x}) & \text{otherwise ,} \end{cases}$$

which is equivalent to writing

$$\begin{aligned} y(\boldsymbol{w}^T\boldsymbol{x}) &\geq 1-\xi \\ \xi &\geq 0 \end{aligned} \tag{4.18}$$

simultaneously. Along with reformulating λ as $\frac{1}{C}$ we obtain a constrained SVM objective function (cf. Equation (4.14)):

$$\mathcal{L}_{SVM} = \arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_i^n \xi_i \ ,$$

subject to constraints from (4.18), $\forall i$.

This objective function gives rise to the standard kernel formulation of the SVM based on the dual form. Making use of the Equations (4.1) and (4.5) and introducing Lagrange multipliers $\boldsymbol{\alpha} \in R^n$ and simplifying, we arrive at the dual form of the kernel SVM objective function:

$$\mathcal{L}_{SVM} = \arg\max_{\boldsymbol{\alpha}} \sum_i^n \alpha_i - \frac{1}{2}\sum_{j=1}^n\sum_{i=1}^n \alpha_i\alpha_j y_i y_j k(\boldsymbol{x}_i^T\boldsymbol{x}_j) \ , \tag{4.19}$$

subject to constraints

$$\begin{aligned} \sum_{i=1}^n \alpha_i y_i &= 0 \\ 0 \leq \alpha_i &\leq C, \forall i. \end{aligned}$$

The decision function takes the classic dual form shown previously in Equation 4.5. The kernel SVM was designed to produce high accuracy from small data through the use of the kernel trick and the soft margin. These make it very suitable for a typical steganalysis problem, as we know it, i.e. a binary classification problem with very limited training data. The size of training data is limited because of $O(n^2)$ training.

Throughout our experiments we use the radial basis function kernel (see below), following the example of most steganalysis literature. In this configuration, the kernel SVM has two parameters to optimise, cost C and kernel width γ. The parameter

C defines the trade-off between the penalty of misclassification and the size of the margin between classes [11], which essentially balances training accuracy with the generalisation capabilities of the model. The γ parameter is discussed below.

For our KSVM tests we used an established open source library `libsvm` [21] with its extensive toolkit for parallelisation and other tools to control space and time trade-offs.

Radial basis function kernel

In the absence of any prior knowledge about the problem one might choose to use a universal[7] kernel such as the RBF kernel. It is by far the most often used kernel in steganalysis. To fall in line with the previous research we also employ the RBF kernel in our experiments. It is defined as:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\gamma||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2\right) \ . \tag{4.20}$$

Together with the regularising cost parameter C, γ needs to be found using a grid-search or a similar procedure which allows finding a balance between the complexity of the model and its generalisation accuracy.

Kernels like this are used to compute the dot products in a representation which is different to the original representation defined by the features. This representation is determined by the complexity of the kernel. This technique is used to find better separation between classes to that which is allowed by the current representation.

Working in the expanded space implicitly through the use of a kernel function was thought of as saving time and space - consider the following argument. For polynomial kernel of degree d and feature vector of length n there are $O\binom{n+d}{d}$ features in the embedded space. For higher degrees of this kernel and other more complex kernels, like the RBF kernel, this might be too large (or in some cases impossible - the exponential expands into an infinite series) to calculate explicitly.

[7] RBF kernel was shown to be capable of universally approximating any arbitrary function to small error ϵ [47].

In contrast to using the dual form representation in conjunction with a non-linear kernel function k, we can explicitly transform the feature vector into an embedded space. There exist kernel approximations which compute quick approximations of the expansion using ideas from compressed sensing. These approximations are much shorter ($O(\log \binom{n+d}{d})$ cf. $O(\binom{(n+d)}{d})$) and allow for using kernels together with the primal form of a linear classifier. Several references to specific examples of such algorithms are given in Section 4.3.3.

4.3.2 Ensemble classifiers

A different approach to non-linear classification is to use ensembles of classifiers, such as the voting ensembles of randomly-projected linear FLDs which are used in this dissertation.

There exist many variants of combining classifiers into an ensemble. Some are rather complex and are known for their performance guarantees, for example, the AdaBoost algorithm. Others are simpler, such as the ensemble used here. The ensemble classifier used here is our version of the classifier first introduced to steganalysis in [70] as an alternative to SVMs with the promise of more efficient computation and thus better scalability to rich features.

This ensemble strategy is often referred to as the *random subspace method*. The ensemble consists of multiple base learners which learn from the same data, each using different (sub-)features. The sub-features are random slices from the original feature set. As a result, the ensemble allows for training a non-linear classifier on *large features* in reduced time.

Many different classifiers can play the role of a baselearner. The final decision is typically calculated by taking the majority vote from all baselearners, but more advanced strategies exist involving weighing the contributions of different votes based on the output values of the baselearners' decision functions.

An ensemble classifier will be online-capable if it uses online algorithms as base

learners. Using the Average Perceptron as the base learner in [81] and here allowed us to train the ensemble on large data sets.

4.3.2.1 Ensemble FLD

Two versions of this classifier are used in our experiments. The first version is our own reimplementation of the classifier first introduced by Kodovsky [70] which is a batch algorithm. The second version is based on the quasi-online method we proposed in Section 4.2.3. The two will be used interchangeably in our experiments depending on the computational constraints and we refer to both as EFLD throughout this thesis[8].

In this configuration, L Fisher Linear Discriminant base learners are trained on L different subsets of k dimensions drawn at random from a feature set. The decision function combines the predictions from individual base learners using the (binary) majority vote:

$$h(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \sum_{l=1}^{L} \operatorname{sgn}\big(\boldsymbol{w}_l^T(\boldsymbol{x} - \boldsymbol{c}_l)\big) > 0 \\ -1 & \text{otherwise}\,, \end{cases}$$

where $\boldsymbol{w}$ and c are the parameters of the l's Fisher's discriminant baselearner, δ is the Kronecker delta and y is the label of training example $\boldsymbol{x}$.

This method has shown very promising results using the CC-C300 feature set [70] and similar large feature sets [73, 51] and is currently considered to be a state-of-the-art classifier for binary steganalysis. A major advantage was shown to be the speed of training and the reduced complexity of parameter optimisation.

Chaumont et al. [23] have demonstrated a weighted version of EFLD which was shown to improve its performance on typical steganalysis tasks.

[8] Whilst the two algorithms may in practice produce different results, in our experience the difference was small enough for us to assume their performance equal and use them interchangeably.

4.3.2.2 Ensemble Average Perceptron

We introduced another version of this classifier in [81]. This is the online ensemble Average Perceptron (OEAP), which fuses the advantages of the stochastic nature of online learners with the non-linearity of ensembles classifiers. The set up remains identical to the ensemble FLD, with the same number of learners, same number of randomly-chosen features per learner (discussed in Chapter 5), and the majority vote which establishes the final decision. The main difference is in the choice of baselearner, which in this case is the Average Perceptron.

4.3.3 Other non-linear methods

Only a few non-linear classifiers are capable of training with stochastic updates. Even fewer of these have been widely studied and/or have available implementations. One might consider k-Nearest Neighbour classifier as being part of this group, but more sophisticated algorithms exist. One of these algorithms is the kernel approximation which was mentioned above (see one of [120, 100, 78] for examples). A similar approach using explicit preprocessing (mapping) of the data is described in [25]. Another example of an online non-linear algorithm is the online version of the Learning Vector Quantisation [16]. Some other, perhaps more exotic, variants exist, such as the Locally Linear SVM due to Ladicky [75] or the kernelised stream SVM [101]. These algorithms have not yet been considered for steganalysis and we do not consider them in our study.

4.3.4 Analysis

Finding the right classifier for a particular learning task is partially a problem of balance. When data is scarce one might want to use the most powerful classifier possible. Therefore based on the historically-defined view of steganalysis being a binary classification problem in images from a single (small) source the KSVM might be a sound choice. On the other hand when data is abundant we want to use as much of it as possible to reduce the generalisation error. It is common knowledge in machine learning that for ERM-like methods (most classifiers used in steganalysis to

date) large training samples are required for good generalisation[9]. In such a situation a sensible strategy might be to improve performance using more data. Once no further improvement is possible one might consider using better representations and hence more complex models. But, as we show, it may not always be computationally feasible to train those models on enough data to achieve same or better levels of generalisation accuracy. We consider this point in more detail in the upcoming chapters.

4.4 Conclusion

This chapter presented a selection of linear and non-linear classifiers that will be used in our experiments. In many ways these classifiers are very similar, if not identical, to those used in contemporary steganalysis research literature. Some of these classifiers, such as the kernel SVM have been considered to be state-of-the-art in many publications. Others, such as the ensemble classifier have challenged that point of view and shown equivalent or better performance when combined with the modern rich feature representations. One contribution of this thesis to the field is a demonstration that with the new rich features we can achieve better classification accuracy by training simpler models on larger data sets instead of using non-linear models and being computationally restricted in the size of the training data. Our view is that such classifiers may not only yield better models if trained on large enough data such that a well trained, if not converged, model is produced but may also facilitate our attempt to overcome some of the problems that arise when steganalysis is viewed as a domain adaptation problem and the associated cover source mismatch occurs. Some domain adaptation techniques such as importance weighting [26] (see Chapter 7) often use linear classifiers to achieve them.

[9]This is a fundamental result of Statistical Learning Theory which will be discussed in detail in Chapters 6 and 7.

Chapter 5

Designing experiments

Having discussed our selection of features and classifiers, in this chapter we will look at what other ingredients go into our experiments and their overall design. We start with discussion of the data. Our aim is to create a set of realistic testing conditions for classifiers of binary steganalysis problems. This will enable us to compare, in practical terms, how such conditions differ from the often used laboratory conditions from the point of view of their ability to influence the classifiers' accuracy.

5.1 Data

Images acquired from real-world sources are best suited to our study due to their variety and abundance. They are not required to share some common properties, in particular the properties that may affect steganalysis statistics - we are interested in testing exactly the opposite. In order to acquire such data we needed a source of public images. The Internet contains many such images and they are often easily discoverable and downloadable using a Web crawler.

All images for the experiments presented here were acquired by my supervisor, Dr. Andrew Ker, at the University of Oxford. The data set consists of albums of photographs made publicly-visible by Facebook users from the Oxford University network. In total over 4 million JPEG photographs were acquired, although in this thesis we restrict ourselves to two smaller subsets. For the first subset we selected all users with at least 4000 images. There were 26 such users in total. The second

subset consists of 1711 users with at least 500 images each. All downloaded content was anonymised, but the natural split into one subset per user was preserved. A sample from this data set is given in Figure 5.1.

These images have some important characteristics. Two characteristics were enforced by Facebook automatically, through resizing and recompression at upload, and are therefore shared between all images; their size (approximately 1 megapixel but not fixed) and the JPEG quality factor (85). The latter makes it feasible to use these images in our study[1]. All other characteristics of these images are variable between users, including those that are related to their acquisition (camera model, focal length, exposure, ISO factor, scene and more) and those that are related to their processing (resampling, original size and JPEG compression, denoising, colour filtering, added synthetic content (date/comments) and more). At the time of writing it is largely unknown exactly how these characteristics, individually or in combination, affect steganalysis features, we are thus required to treat them as unknown.

There is consensus, however, that these different characteristics have a negative effect on the overall reliability of steganalytic detectors [4, 20, 40, 59, 94, 44], which manifests itself as the problem known as cover source mismatch. This means that even the conventional approach of training and testing on images from the same data set (in our case - the same actor) should be expected to yield variable performance from one data set to the next. Furthermore there are a significant number of outliers - images that are not completely natural photographs; they may be montages, have captions, or be entirely synthetic. We view this as further contributing to the realism and, perhaps, the difficulty of this data set.

The only pre-screening that was performed on these images was aimed at the removal of images with little or no content and those with a non-standard compression quality factor. This was done by deleting any image with a file size smaller than 5KB and any image with a non-standard quality factor. Less than 1% of all downloaded

[1]State-of-the-art steganalysis features are incompatible with varying JPEG quality factors (see Section 3.3).

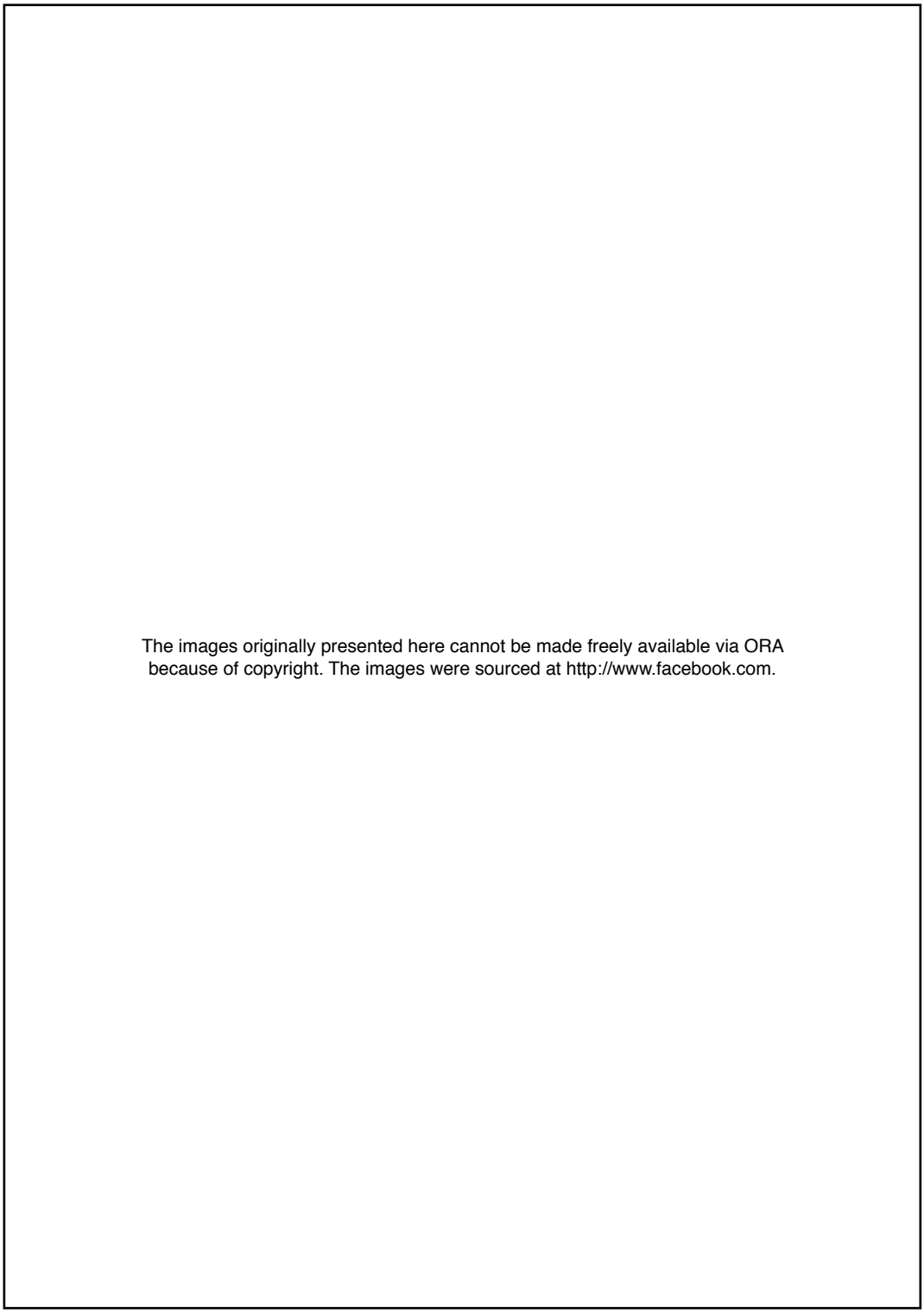


Figure 5.1: Sample images from the Facebook data set, anonymised.

images were affected.

We will treat the different users (also called "actors" here) as different sources. Onwards, the terms "user", "actor" and "source" will be used interchangeably, except in the domain adaptation context when we may refer to source domains as those users for which we have labelled data and which can be used for training (similarly target domain will be used to describe testing data from one or more users).

Two assumptions have to be made about these images, because we have no means of guaranteeing the correctness of any test for alternatives using the modern tools. First, let us consider the question of ground truth. Unless we have perfect detectors or have been informed though other means, there is always a danger that a real-world source of images will be contaminated with steganography. We like to think that at present the likelihood of steganography being used by the members of the Oxford University network on Facebook is very small[2]. We thus assume that this data is free of steganography and label all downloaded images as covers. Second, with the absence of reliable metadata there is a possibility that an image set from one user may be composed of more than one source (if, for example, they use different cameras or different post-processing). Whilst no formal check has been performed we assume that the vast majority of actors provide a homogeneous or near-homogeneous set.

5.2 Experimental modelling of classification-based steganalysis scenarios

What is particularly special about this data set is that it allows us to create multiple distinct models of the steganalysis problem. In particular, if we recall Ker's categorisation of steganalysis scenarios from Chapter 1, three of those are of special interest because we can model them here: in scenario (2) we do not know exact characteristics of Alice's cover source but have examples to learn from, (3) - we don't have examples from Alice's source but can learn from similar examples and

[2]This is also considering that the acquisition process took place before the existence of Secretbook [19].

(4) - we know nothing about Alice's cover source. Each scenario provides different challenges for training a classifier which we explore in our experiments. Some of these are better studied than others.

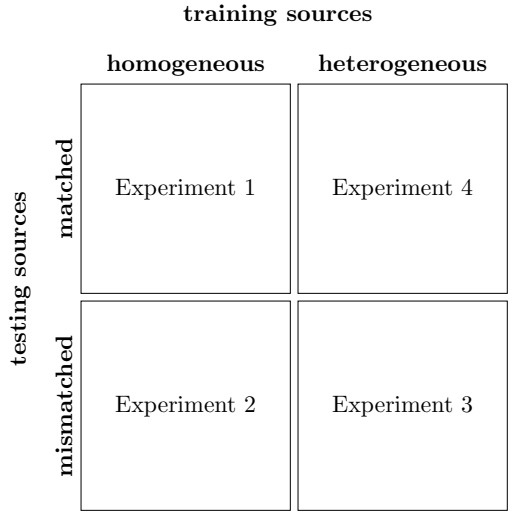
The most obvious way to model Alice here is to choose an actor at random from our data set. However, the choice of training data to use in order to train a good detector for Alice's (testing) data is non-trivial. It depends on which of the above categories we are trying to model. Take category (2) as an example. In this situation we assume the possession of training data that matches Alice's source. Whilst an unlikely scenario, it is widely studied in steganalysis literature as a testbed for creating new features or breaking new steganography schemes. It is widely considered to be "optimal" for training steganalysis detectors. We will refer to it as a homogeneous-matched data scenario under "laboratory" conditions. It is matched because training and testing data come from the same source and it is homogeneous because there is only one source of data that we are interested in. For this scenario, the Facebook data set provides 26 different actors with 4000 cover images each. For the binary problem studied in Chapter 5 we can split each of these 26 data sets into 3000 training pairs of images and 1000 testing pairs of images.

In categories (3) and (4), the source of testing images is unknown and we may or may not have unlabelled testing data at training time. The literature that tested steganalysis in this scenario used training and testing data from different homogeneous data sets. We call this a homogeneous-mismatched data scenario and can simulate their results on the data from the aforementioned homogeneous-matched scenario by taking any two mismatched training and test sets. There are up to 650 such combinations, although we restrict ourselves to 26 for the reasons described in Chapter 5. A different strategy is to pick as diverse a training set as possible. With the lack of understanding of what makes a source, the preferable option is to assume that the diversity of sources should generally come with their quantity. Whilst it has been argued that this guarantees mismatch between training and test sets [58], we see two distinct situations arising. In the first situation, which corresponds to

Ker's category (3), we are able to find training data that matches the target actors. We call this a heterogeneous-matched data case. It is not clear whether this is generally possible and techniques for this are yet to be developed. A possible solution is discussed in Chapter 7. Whilst this situation is not formally tested in the present thesis, an optimistic variant, which assumes a successful match, appeared in our previous work [81]. The second situation corresponds to category (4), the heterogeneous-mismatched scenario, which will be compared directly to the two scenarios based on homogeneous data discussed above. We reserve a fixed set of 100 actors selected randomly to be testing sources. Each of these actors provides 500 cover images and we aim to test our detectors on each of them independently. The full binary testing set is composed of 50 000 images. The remaining 1611 actors will be used for training which provides for a heterogeneous-mismatched case by design.

We summarise the above into the following definitions. A **homogeneous** data set is composed of images from one actor. Most images in such a data set are likely to share common characteristics, such as originating from the same camera. A **heterogeneous** data set is composed of images from multiple actors. A heterogeneous training set may be highly diverse, containing small number of images from many different actors, or less diverse, in which case only a small number of actors contribute many of their images. **Matched** data means training and testing sources are the same. **Mismatched** data means training and testing data come from different sources.

The matched/mismatched and homogeneous/heterogeneous properties of training and testing data sets make it less or more difficult for feature-based steganalysis. We test each scenario in a separate experiment in Chapters 5 and 6. Let us define the following matrix for convenience:



The matrix shows that[3]:

Experiment 1 tests the matched-homogeneous situation with 26 different data sets.

Experiment 2 instantiates the mismatched-homogeneous situation by building upon experiment 1 and testing the models trained in Experiment 1 against mismatched testing data.

Experiment 3 corresponds to the mismatched-heterogeneous situation - training and testing data comes from a number of mismatched sources and hence is heterogeneous and mismatched.

Experiment 4 (see [81], also reproduced in Figure B.1 in the appendix) is concerned with an intermediate situation where the data comes from an unknown mixture of sources and where the information about their origin is unknown.

Experiment 1 replicates the conditions studied in most steganalysis work to date. Experiment 2 has also been studied but to a lesser extent. The added advantage of using 26 different image sets in both of these experiments is that it allows us to reason about the detectors' accuracy from a statistical point of view by looking at means, standard deviations, minimums and maximums of their performance. Experiment 3

[3] A more detailed description of these experiments is given in the next chapter.

provides a new approach to training a steganalysis detector, whilst conditions similar to Experiment 4 were studied in [63].

5.3 Methodology

When designing experiments for steganalysis, as well as any other discipline where statistical modelling/machine learning is involved, one must take into account many possible sources of error. Furthermore, there exist certain conditions that are specific to steganalysis which have been shown to impact the performance of a classifier. In this section we discuss these in more detail.

The data and the scenario models discussed in Section 5.2 allow for testing steganalysis in new conditions. These are closer to a simulation of real-world steganalysis classification than ever before. To be able to make inferences about the real-world from such simulations we must account for sources of potential error. Borrowing from the statistical literature we can describe the following four sources of error, each of which is discussed in further detail in this section:

1. **Model error** - concerned with the classifier choice.
2. **Calibration error** - how the classifier is trained and how the parameters are chosen.
3. **Computational constraints** - to do with finite time and space budgets to train the classifier.
4. **Uncertainty quantification** - metrics, statistical tests and their interpretation.

The following subsections provide a closer look at some of these points with examples. But first let us consider how the performance will be evaluated.

5.3.1 Performance measures

Our performance measure is the classification accuracy, which is defined as the proportion of correctly classified examples to the total number of examples in the

data set in question:

$$\begin{aligned} \text{accuracy} &= 1 - \frac{P_{FP} + P_{FN}}{2} \\ &= 1 - E \ , \end{aligned} \tag{5.1}$$

where P_{FP} and P_{FN} are probability estimates of having a false positive and a false negative respectively. The data will often be split into three non-overlapping data sets. The training set will be used to train the classifier. The validation set will be used to evaluate the performance of different hyperparameters during their optimisation (e.g. in a grid-search). The test set will be used to evaluate the classifier's performance using the final hyperparameter values. A classifier's performance can be measured on the training set (E_{train}), on the validation set (E_{val}) or on the testing set (E_{test}).

Another measure of performance, that is often reported in steganalysis literature, is the testing error achieved by the classifier with its threshold adjusted to some optimal value (in contrast to E_{test} which is measured using the default threshold of 0 or 0.5 depending on the classifier). The idea behind this is that by the means of adjusting a classifier's threshold to minimise its E_{train} or E_{val} it is possible to minimise the probability of it misclassifying future data based on the assumption that the test data comes from the same distribution as training or validation data respectively:

$$P_E = \min \frac{P_{FP} + P_{FN}}{2} \ , \tag{5.2}$$

In the literature the threshold is selected to minimise the number of false positives and false negatives when the classifier is tested on training data. Whilst this may be a good strategy in the case of conventional steganalysis when training and testing data is matched, we deviate from this for two reasons. First, in some of our experiments the training set may be prohibitively large to use for adjusting threshold. Second, when the test data comes from a mismatched source the same distribution assumption may not hold any longer. We therefore adjust it based on the validation set. In Chapter 6 we report $1 - P_E$.

The multitude of target sources that we can test on in Chapter 6 allows us to measure the classifiers' performance in a statistical manner. Several statistics will be displayed. Formally, if $E^1_{test}...E^n_{test}$ are errors on each of n sources then the average detection accuracy of a classifier, displayed as "μ", will be calculated as:

$$\mu = 1 - \frac{1}{n}\sum_{i=1}^{n} E^i_{test} \ .$$

Similarly for σ, the standard deviation:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\left(1 - E^i_{test}\right) - \mu\right)^2} \ .$$

Both μ and σ will be serving as a general measure of a detector's robustness. μ tells us about its generalisation - if we were to pick an image from a random actor this is an empirical estimate of how accurate our classifier's prediction will be on that image. We are also interested in σ, which allows us to reason about the detector's variability in different sources. Higher μ and lower σ scores should indicate a better classifier. It is important to note, however, that even though we often measure steganalysis security in terms of the detectors generalisation accuracy (μ), there are real-world situations where other measures may be considered equally important. For example, let us consider a situation where a legal decision has to be made based on predictions from our binary detector. If the accuracy of the classifier varies between different sources (as expected based on the current state-of-the-art), then it is the worst case detection accuracy that is likely to be decisive when choosing the best detector. It may be difficult to measure the true worst case scenario performance from a fixed sample of sources. It can be shown that as the number of examples in each test source tends to infinity the generalisation accuracy of a classifier tends to the worst case. We will be using the minimum accuracy (displayed as "min") as a rough estimate.

5.3.2 Model error

Model error relates to the choice of statistical model, or in our case, the choice of classifier and whether it is capable of capturing the patterns that exist in the data, whilst allowing for desired computational cost given a fixed representation (features).

The theory of statistical learning applies here and we discuss it in further detail in Chapter 7. To account for model error we propose to test multiple classifiers of different complexity - linear and non-linear ensemble and kernel-based classifiers.

5.3.3 Calibration error

Calibration error may arise from data-related and classifier-related issues. Some examples are given below.

5.3.3.1 Data-related issues

To minimise the effects of calibration error due to data format and handling we perform a number of steps, most of which are used regularly in steganalysis research. Many of these are self-explanatory, others are discussed in more detail.

- All final testing is performed on testing data which are disjoint from the training or the validation data (obvious in mismatched cases but also important for matched tests).

- The data sets used for validation and testing should be large enough such that the measurement error is low for our statistical tests to be viable. In other words the standard error should be smaller than the expected standard deviation between different testing sets. The smallest testing set for us is composed of 1000 test examples and with the expected accuracy of 85% the confidence interval is $\pm 2.2\%$ at 95% level[4]. For the larger matched data set it is 1.6%.

- Throughout all experiments we assume a binary problem with equal prior probabilities on the two classes, i.e. all data sets contain an equal split between cover and stego images. In addition, no cover-stego pair is split between training and testing data sets. Preserving cover-stego pairs has been shown [106, 67] to be important for training binary classifiers for steganalysis, particularly during hyperparameter optimisation because estimation procedures,

[4]For the binary case this is calculated as C.I. $= z\sqrt{\frac{p(1-p)}{n}}$, where $z = 1.96$ (for 95% level), $p = 0.85$ and $n = 1000$.

such as cross-validation, may be sensitive to this and it would otherwise lead to finding a suboptimal set of hyperparameter values. It was shown [67] to affect smaller payloads to a greater extent where the separation between classes is worse. Whilst we have not formally tested whether this problem occurs when optimising on larger data sets, most of our hyperparameter optimisation is performed on data sets of similar size to [67] and as such we follow this guideline. A testing data set composed of cover-stego pairs may introduce dependencies between false positives and true positives (and between false negatives and true negatives), which may slightly affect performance statistics such as accuracy and the receiver operating characteristic curve. In this thesis we use paired testing following the convention in steganalysis literature.

- The images creating cover-stego pairs are drawn at random from the data set and the sequence in which they are visited is randomised.

- Normalisation of the data is performed to zero mean and unit variance. Ordinarily the normalisation factors (the mean and the standard deviation) are calculated on the training set. Whilst this may be practical for smaller data sets, it was not for our mismatched training set. This is especially problematic if a training set is large enough to require online-only training in practice. One might consider approximating the normalisation factors as training examples arrive, but we decided to use the validation data set instead. We calculate multiple sets of normalisation parameters - one for each different type of payload.

- The training and testing actors' selection is preserved between the tests of different classifiers, however some noise may be present in the results due to the different random payloads embedded.

- When the computational requirements of a classifier restrict us to training on a subsample of the full training set we perform five such samples in order to minimise the chance of a bad sample producing a biased model and affecting the final testing results. The mean, standard deviation, minimum and maximum metrics in these cases are calculated over all five runs.

- For the situation when the classifier cannot be trained on all of the training data it is important to ensure good sampling. We compared two strategies: a) a more diverse data sampling strategy in which we sample images at random from from a mixture of training sources; and b) a less diverse data sampling strategy where we select a small set of sources and use all of their images for training. In [82] we tested both strategies and include the summary table here for completeness (consider the second column of Table 5.1). It was found that the more diverse data sampling strategy is statistically better - it produces models which are more stable when tested on our mismatched testing set of 100 actors. This is reflected in the reduced standard deviation (σ^2) figures, where the reduction tested to be statistically significant ($p \ll 0.001$) as shown in the second column of Table 5.1. With this in mind the heterogeneous-mismatched experiment (Experiment 3) of the next chapter is based on using this sampling strategy.

5.3.3.2 Classifier-related issues

Many classifiers allow for tuning their learning through the use of hyperparameters. They are often explicitly stated and have well-defined effects on the learning algorithm in theory. In practice, their values must be found empirically by performing a search through the hyperparameter space. This allows us to maximise the chances of selecting good parameters for each learning problem. Several techniques automating this search have been proposed, for example the adaptive learning rate used by the Vowpal Wabbit (VW) implementation [76] (see Section 5.4). In many situations simpler techniques are preferred such as five-fold cross-validation (KSVM) and out-of-bag error estimate (ensemble classifiers). These procedures are straightforward to implement and allow us to explore the parameters space by sampling from discrete points using training and validation data only. Simplicity equates to transparency and ease-of-use, although it also leads to a substantial increase in the computational cost. For example, the grid search needs to visit $i \times j$ points, where i and j are the number of distinct values tried for the two hyperparameters respectively. In our case we often explored grids of 9-by-9 or larger. At each point the measurement had to be as precise as allowed by the available data hence the need for tools such as five-fold

Table 5.1: Comparison of the more diverse (subscripted with "1") and the less diverse (subscripted with "2") training data sampling strategies (see text for details). The two strategies are compared for two classifiers, KSVM and EFLD, using the Student t-test (μ_1 vs μ_2) and the F-ratio test (σ_1^2 vs σ_2^2). A higher mean (μ) means better accuracy on average, whilst a lower variability (σ^2) means better robustness. The last column shows the results. There is no statistically significant difference in the average performance (μ) for either of the two classifiers at 95% confidence level ($p > 0.5$ for both KSVM and EFLD). There is however statistically significant difference between the two sampling strategies in the variability (σ^2) in performance they yield for both classifiers (p-value was significantly lower than 0.001 for both KSVM and EFLD). The conclusion can be drawn that the more diverse training data sampling strategy produces more robust classifiers. Middle column is given for reference with the performance in matched experiments. Table reproduced from [82].

	Test	matched v more diverse	more diverse v less diverse
KSVM	$\mu_1 > \mu_2$	$t = 11.03$ $df = 62.80$ $p \ll 0.001$	$t = 0.54$ $df = 154.21$ $p > 0.5$
	$\sigma_1^2 < \sigma_2^2$	$F = 2.59$ $df = 99,\ 25$ $p < 0.05$	$F = 3.28$ $df = 99,\ 99$ $p \ll 0.001$
EFLD	$\mu_1 > \mu_2$	$t = 8.88$ $df = 57.20$ $p \ll 0.001$	$t = -0.29$ $df = 172.86$ $p > 0.5$
	$\sigma_1^2 < \sigma_2^2$	$F = 2.21$ $df = 99,\ 25$ $p < 0.05$	$F = 2.23$ $df = 99,\ 99$ $p \ll 0.001$

cross-validation, which are designed to facilitate the hyperparameter search when training and validation data is limited. Accounting for cross-validation, which was

performed when optimising for the homogeneous-matched experiments and some of the heterogeneous-mismatched experiments, a 9-by-9 grid search amounts to 405 runs of the classifier. The procedure may need to be repeated for each embedding payload as this may change the learning pattern of the classifier, and hence the optimal values of hyperparameters (more on this to follow below). For each classifier and each learning problem we perform an extensive search.

We continue this section with a description of different techniques used for hyperparameter optimisation. We first consider optimising algorithms trained in the batch/iterative setting. Here the minimum processing unit is a single iteration (or in the case of FLD and KSVM - the full training run) and the loss and training and validation errors are measured on per-iteration basis. Then online training will be discussed, where a processing unit is a single example.

Optimising linear classifiers

Early stopping is a simple technique which can be used to avoid overfitting in simple linear classifiers such as Average Perceptron or non-regularised Logistic Regression[5] and can be viewed as an implicit hyperparameter (number of iterations) when training in batch mode. Here, we train the classifier iteratively until convergence and evaluate its performance on a validation set at each iteration. The model that gives the highest accuracy is picked for the final test. Convergence happens when there is no further improvement in the value of the loss (or the *training* error E_{train} does not improve any further). The best model according to the early-stopping criterion is the best performing model on *validation* data (E_{val}) before or at convergence. Figure 5.2 shows an example graph generated from training AP on nsF5 0.05. Here the best model is selected from the region which is closest to the red $min(E_{val})$ line. If a classifier's objective function incorporates the ℓ_1 or ℓ_2 regularising terms, these are often controlled via hyperparameters, e.g. λ_1 and λ_2. The advantage of using this form of regularisation is that the final model, after some n iterations, will be as

[5]Although VW's implementation of LR includes l_1 and l_2 regularisation; in this project we use l_2.

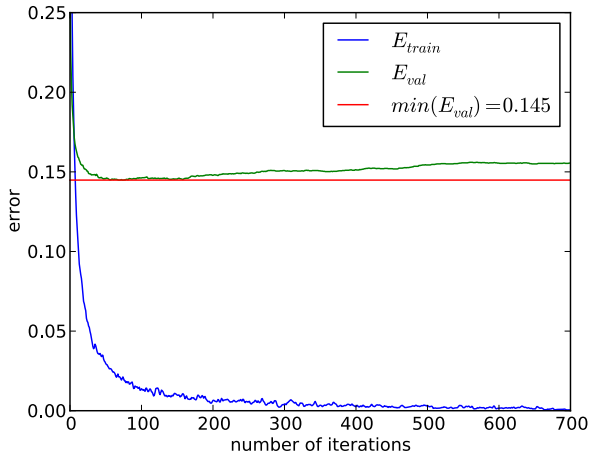


Figure 5.2: AP regularisation via early stopping.

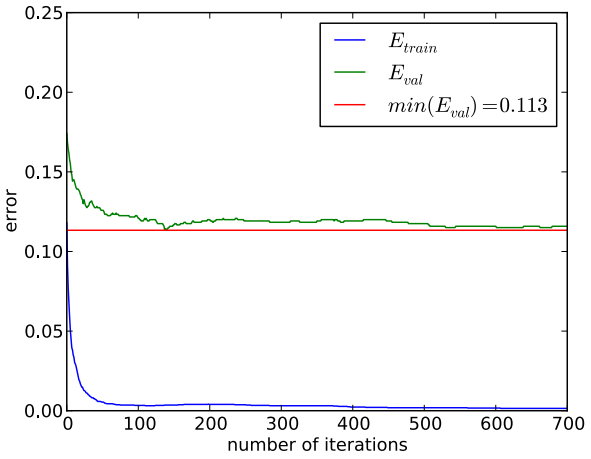


Figure 5.3: SVM regularisation using ℓ_2 norm.

close to the best model as allowed by the λ_1 or λ_2 parameter. The number of iterations n must be large enough for the model to converge. However the introduction of the extra λ_1 and/or λ_2 hyperparameters requires further parameter optimisation via, for example, a grid-search. Results from an example of such a grid search are shown in Figure 5.4.

Some linear classifiers, such as the Logistic Regression and the linear Support Vector Machine, naturally incorporate regularisation into their objective function.

Often different problems call for different values of hyperparameters, therefore the above graphs are only illustratory and would generally have the optimal regions at different points through the training. However, given enough iterations and the right value of the λ parameter the optimal region will be any point at or after convergence.

Optimising KSVM using the grid search

Grid search is one of the most popular methods to tune hyperparameters for the Kernel SVM. It is a straightforward but expensive procedure: the classifier is trained repeatedly with hyperparameters selected from a pre-defined grid of values. This

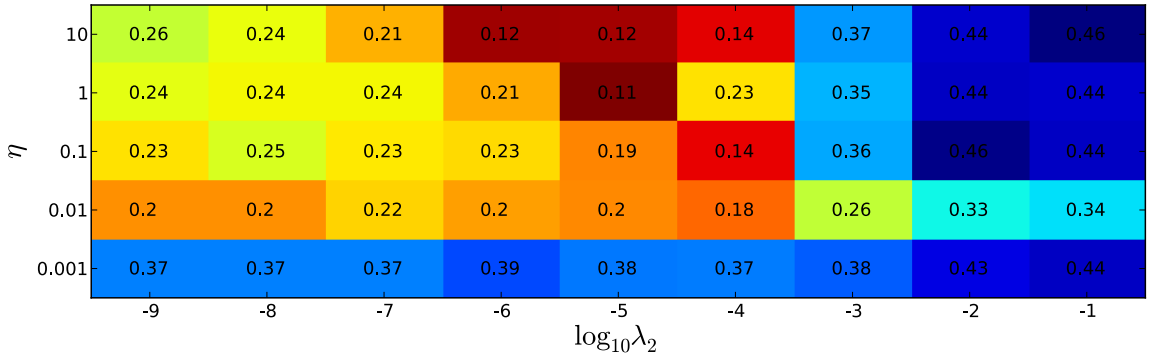


Figure 5.4: SVM regularisation via grid search. The figures in boxes show E_{val} values for different combinations of η and λ hyperparameters.

grid is typically on a log-scale, for example:

$$C \in \left\{2^i \mid i \in \{1, 3, ..., 21\}\right\} \quad \text{and} \quad \gamma \in \left\{2^j \mid j \in \{-23, -21, ..., 5\}\right\}$$

We also use a second finer grid around the region of interest. The combinations of parameter values that yield the best performance are selected for final training. For those experiments that involve the homogeneous data and therefore incur a limit on how much training and testing data is available for each run we pair this method with five-fold cross-validation. In heterogeneous cases where data is ample we create one training and one validation set of the same size for each experimental run.

One problem was found with the results of the cross-validated grid-search, which yielded unusual spikes on the contour-graph (see Figures A.1 and A.2 in the appendix). Under closer inspection these proved to be abnormal as the respective hyperparameter values produced vastly suboptimal test performance. We attribute this instability to one (or both) of the following factors: a) the training data set that was available was too small to produce reliable results under the grid search b) the folds were too small (perhaps a ten-fold cross-validation would work better) and the data within the overall training set not independent enough to train an unbiased model. In the heterogeneous cases when there was enough data to perform only a single training-test run for each point on the grid these problems did not occur. This is somewhat counter-intuitive because one would expect that the homogeneous

data, which is considered to be easier than the heterogeneous data, should yield more reliable results (especially given the grid search); however this was not the case. For an illustration of this issue compare Figures A.1 and A.2 in the appendix.

Figures 5.5 and 5.6 show the final results of a grid-search for KSVM on heterogeneous data with nsF5 payload embedded at 0.05 bpnc. For this we used two disjoint sets of 3000 cover-stego pairs, one for training and one for finding the validation error. It is evident that the classifier is more sensitive to the kernel's γ parameter than the cost parameter C (shown by the longer vertical ridges of the contour graph). We can see that relatively small values of the γ parameter, between 2^{-17} and 2^{-21}, provide best results which means that the classifier favours highly complex models. The situation is similar for other payloads and scenarios with the plot shifted towards the lower part of graph when payload is increased (i.e. the cost of misclassification can be lowered with the higher embedding payload).

Figure 5.6 depicts the second finer grid-search around the optimal region. It can be seen that it is relatively unstable with a general trend of declining performance towards the top left corner of the contour plot. However the instability is within the region of 0.5%. The conclusion from this is that a finer grid search is probably not required.

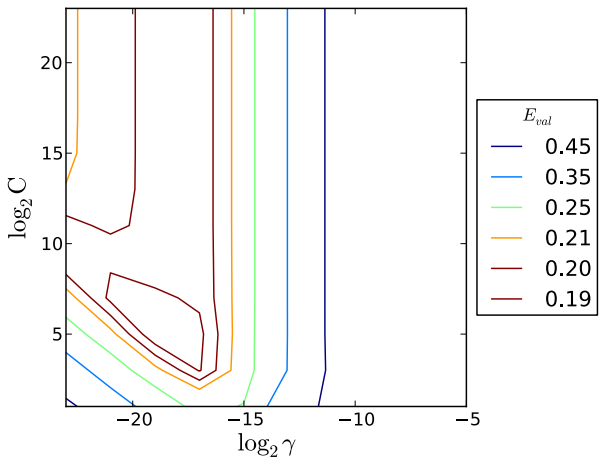


Figure 5.5: Kernel SVM grid search.

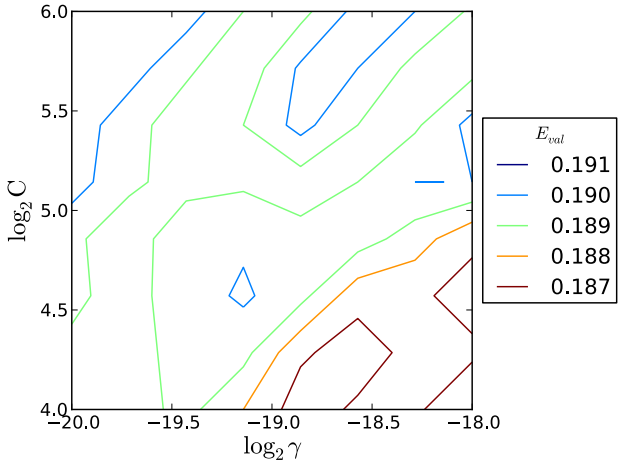


Figure 5.6: Zoomed-in on the region of interest.

Optimising the ensemble classifier with line search

The procedure used for optimising the hyperparameters of the ensemble classifier proposed in [70] is effectively a line search with extra steps around the critical values to localise the minimum. We adopt a similar procedure here. As per Section 4.3.2 there are two hyperparameters: the number of base learners L and the size of subfeatures space for each base learner k. From our experience the error converges to a stable "optimal" value once both hyperparameters' values exceed their respective critical points, say L_{opt} and k_{opt}, which mostly echoes the findings in [70]. In practice this means that given a large enough computation budget we can set $L > L_{opt}$ and $k > k_{opt}$ without the risk of increasing error. The crucial step is then to find L_{opt} and k_{opt}. We performed a line search first on L followed by a search on k.

In contrast to [67], where the author used the out-of-bag error to estimate the classifiers' parameters, we use the validation error as for all other tested classifiers. This is arguably a better estimate of the generalisation error for our experiments, especially those experiments where the source (training) and the target (testing) data are drawn from different data sets. This is due to the fact that out-of-bag error is measured on the training set and is therefore by definition biased towards the source data and not the target data as it would be preferable. Whilst the validation data used in our mismatched experiments is also drawn from a different set to the target data it is not biased to the training (source) data set and therefore should in theory allow for a better estimate of parameters.

The initial values for the two hyperparameters that were used in our experiments which appeared in [81] and [82] were chosen by this search. In experiments appearing in Chapter 6 (next chapter) we confirmed these by simple manual search to cater for the other payloads.

Hyperparameters and online training

Some simplification arises when training is done in an online as opposed to the batch setting. In this case a classifier is generally less prone to overfitting because each example is only seen once. On the other hand, the parameters such as the learning rate

and the λ_1 and λ_2 regularisation parameters have a direct impact on the model which is produced at the end of a one-pass run. For example, Vowpal Wabbit's implementations of LR and SVM allow for very fine control over the learning rate through the relationship which can be described with the following equation (reproduced from [76]):

$$\eta_t = \lambda d^k \left(\frac{t_0}{t_0 + w_t} \right)^p ,$$

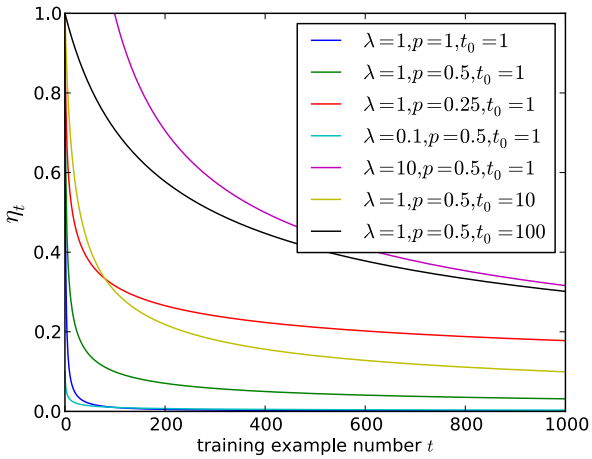


Figure 5.7: Tuning VW's learning rate.

where λ is the learning rate parameter, t_0 is the initial weight and w_t is the weight of example t. In our experiments these were optimised for both LR and SVM. In an online setting the d^k term is constant, where d is the learning rate decay parameter with k being an iterations counter. Then, the graph for η_t takes the general shape illustrated in Figure 5.7. The scale, the curvature and (the asymptotic properties of) the tails are controlled by the parameters λ, t_0 and p respectively. In our experiments these parameters were largely optimised by grid search with t_0 chosen using a manual search.

Other online-capable classifiers can be controlled in the same manner. From our experience setting the AP's learning rate to 1 appears to work well in practice.

Fine-tuning these parameters to data of particular difficulty and size requires skill and there is a plethora of heuristics available at a practitioner's disposal [18]. Regardless of the optimality of the parameters, one-pass-optimal performance may not necessarily match performance of a model trained iteratively. It has been argued

[17], that with a finite training set a classifier's optimal performance is not necessarily achievable in one-pass using standard first-order optimisation methods such as the SGD. The solution is to use second-order algorithms which have been shown to achieve near-optimal test set performance [17], however these are generally computationally expensive, which may or may not be justified by the problem given a fixed time and space budget. Whilst the above holds true for our experiments (see Table 5.3), most of the statistical results presented in the next chapter would still hold. We restricted our online experiments to first-order SGD-based optimisation.

When tuning a classifier for one-pass training it is important to look for convergence. Convergence can be measured using the validation error as we progress through the training set. Figure 5.8 shows the validation error graphs for four different online learners (Average Perceptron, Logistic Regression, SVM and online ensemble Average Perceptron), highlighting their online training progress through the 1.6M data set[6]. We can see that there are no signs of overfitting from any of the classifiers including the weak-regularised AP. This points to the decreased danger of overfitting when performing one-pass online training.Figure 5.9 also shows the region where the OEAP ensemble learner is starting to converge in terms of the number of baselearners needed to achieve the minimum error. For this graph the baselearners were selected in a greedy fashion using their individual validation error. It can be seen that a significant number of baselearners, approximately 100, is required for the ensemble to achieve this[7].

Other interesting observations that can be gleaned from this graph are as follows. The ensemble classifier approaches its minimum validation error faster than others, approximately 1/8th of the way into training[8]. This reiterates our previous findings where the FLD-based ensemble appears to converge early resulting in no further improvement with added training examples [81]. Whilst the linear classifiers

[6]Please note the non-linear x-axis.

[7]Please note that the x-axis is on non-linear scale.

[8]We depict the performance of the ensemble classifier as discreet points rather than a line to highlight the added cost of testing this type of classifier, however it is still in this instance trained online.

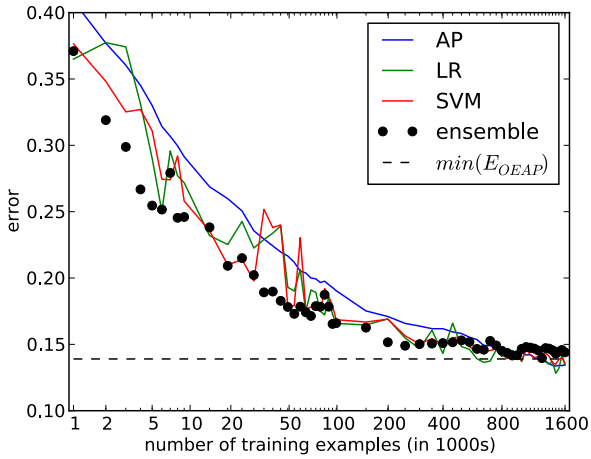


Figure 5.8: One-pass online training.

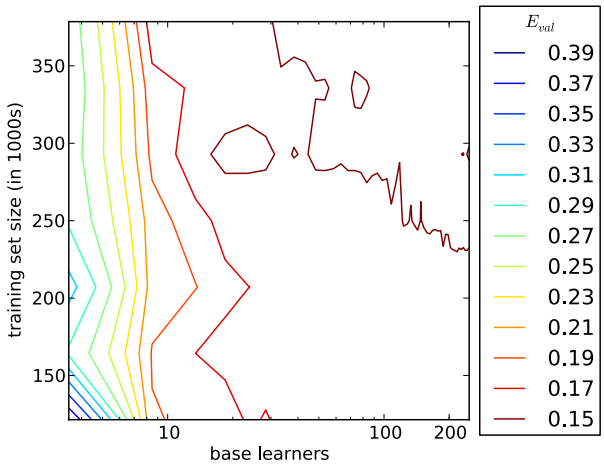


Figure 5.9: Forward greedy baselearner selection.

also demonstrate a tendency towards convergence in the later stages of the online one-pass training run they do not match the same detection performance as can be achieved using iterative training[9] (see Section 5.3.4.1 and Table 5.3 for details) which confirms our point above with regards to the optimality of one-pass training using first-order algorithms. We will revisit this point with regards to the linear classifiers in Section 5.3.4 where we compare their performance figures from a single one-pass (i.e. online training) and multiple iterations over the full training set. Please note that the relative instability in their validation accuracy that can be observed from the graph in Figure 5.8 in the range between 800 000 and 1600 000 is contained within 1% in most cases.

5.3.4 Computational constraints

Some of the classifiers have very different time and space complexities. There exist problems for which the complex algorithms such as the kernel SVM may be too slow and we will argue in the next chapter that steganalysis may be one such problem. Most optimisation algorithms associated with such classifiers currently offer only super-$O(n)$ training time complexity. In particular, KSVM's time complexity is often quoted to be between $O(mn^2)$ and $O(n^3)$, where m is the number of features

[9] We used the same SGD-based optimisation algorithms to train these classifiers in both one-pass and iterative experiments.

and n is the size of the training data. Taking into consideration the requirement of careful hyperparameter selection, the practical computation time of such algorithms increases even further. This limits the size of problems that can be solved with such algorithms. To illustrate this consider a situation when we have a high-bandwidth distributed learning system that trains at as many as 470M features per second - the bandwidth of one of the fastest "classifiers" tested to date [2]. Even at such high bandwidth if it uses a quadratic optimisation algorithm it will require years to train on our full data set of 1.6M training examples. This problem is exacerbated by the complexity of modern steganalytic feature sets, which contain tens of thousands of features. There is potential for even larger and perhaps sparse representations in the future which would render it impossible to use KSVM-like classifiers.

The problem of scaling steganalysis experiments to larger features has been encountered recently in [70], where KSVM was reported to scale poorly with the number of features given a fixed data size. This forced the designers of the first "rich" feature set to introduce the ensemble classifier based on random projections for their experiments [70]. A supporting result using the same features but scaling to larger data sets was given in [81], also favouring simpler classifiers.

What this means for the experiments presented in Chapter 6 is the following. The above complexity requirements raise the need to limit the size of training data for KSVM (and also the ensemble classifier, see discussion below). Before we discuss how we decide on the necessary limits let us consider the computational resources that are available at the time of writing.

The experiments for this thesis were run on a combination of computing resources, which are shown in Table 5.2. The majority of linear and ensemble experiments were performed on the first two systems, Arcus and Caribou, which are both part of the Advanced Research Computing (ARC), an Oxford University supercomputing facility formerly known as the Oxford Supercomputing Centre. Arcus was used extensively for parallelised grid-search and parallelised testing. Caribou was used to

set the timing benchmarks for AP and KSVM and for some of the most resource-intensive experiments using KSVM and iterative linear classifiers on large data sets.

Table 5.2: Computing resources used in our experiments

Owner	Name	Description	Memory
Andrew	T7500	12-core server	196 GB
Andrew	"the cluster"	8 + 1 4-core computing nodes	8×8 GB $+ 1 \times 24$ GB
ARC	Arcus	a cluster of 80 + 4 16-core nodes	80×64 GB $+ 4 \times 128$ GB
ARC	Caribou	64-core system	1 TB shared memory

Such computing resources allow for some very large scale experiments and it is important to give good consideration to the practical limits of running certain algorithms. Several different paradigms for training the linear and ensemble classifiers were explored - a fully online system with on the fly computation as well as iterating over the full data set loaded in memory. Unsurprisingly the fastest of our approaches was the second, however this was only possible on Caribou[10]. The speedup that can be gained from training linear classifiers (e.g. AP) with the full heterogeneous-mismatched training set (1 600 000 examples, approximately 580 GB) loaded in memory is as follows. For example, Average Perceptron takes approximately 12 minutes per iteration plus time to embed and extract the data and load it (approximately 30 hours if cover and stego examples are extracted in parallel or faster[11]). With all the processing being done sequentially and "on-the-fly" the same computation takes over five days. The throughput that was achieved using the faster approach was approximately 50M features per second (including the necessary I/O operations) which is only one order of magnitude slower than a distributed VW system with 1000 nodes that we mentioned earlier - one of the highest bandwidth learning systems tested to date [2].

[10]Caribou is a system shared by many users and sometimes has limited availability.

[11]This is an example of an embarrassingly parallel computing problem where each image can be processed independently of all others and the I/O bandwidth becomes the performance bottleneck.

For a fairer comparison a decision was made to allocate the classifiers to computing resources of the same order of magnitude, however we restricted ourselves to limiting time only and not space. The main question then concerned the heterogeneous-mismatched experiments where over 500 GB of training examples were available to train from. Specifically a decision needed to be made with regards to how much time we were to make available for training the KSVM based on the allowed resources. Kernel SVM's training and testing time is known to be dependent on the choice of hyperparameters and the complexity of the learning problem. The potential for training on multiple cores using LIBSVM's support for multithreading using OpenMP (Open Multi-Processing) complicates things even further. On Caribou we were able to run up to 64 parallel threads allowing for a significant speedup in training and testing. The link in reference [21] gives further details on where the parallelism is available in the LIBSVM implementation.

In addition it was decided to limit the time budget of the EFLD classifier because it was found that there was no accuracy to be gained from using more training examples as we have shown in [81] (and reproduced in Figure B.1), where it was found that a minimum of 20 000 training examples was sufficient [81]. This was supported by the observation we made earlier about the OEAP's convergence during one-pass training (see Figure 5.8).

The final decision was to limit the wall-clock time of KSVM to the same order of magnitude as that needed to train the fastest classifier in our tests - the Average Perceptron. This was found experimentally by a trial-and-error procedure using the final values for hyper-parameters from the grid-search and training sets of different sizes. All possible options for speedup were enabled including the OpenMP multithreading and the option for enlarged buffer[12] which regulates the proportion of the Gram matrix to be stored in cache.

An estimate was done using the following heuristic. Taking the problem of detecting nsF5 embedded at 0.05 bpnc as a baseline we were able to establish that AP

[12]This is controlled through the "-m" option on the command line.

required no more than 20 iterations to converge when trained on the full training set of 1.6 million examples. At 12 minutes per iteration this amounted to 4 hours of wall-clock time. It was measured that I/O was taking an additional 2 - 30 hours to load the data set into memory depending on the availability of resources. In comparison the wall-clock time used by KSVM when trained on 6 000 and 20 000 examples was 40 minutes and 20 hours respectively. A KSVM run using 40 000 examples was aborted after 30 hours of wall-clock time (and approximately 600 hours of CPU time). It was decided that 20 000 examples was a good compromise with a large enough safety time margin in case some of the more difficult problems such as detecting sPQ with 0.2 payload were to take longer. The runtime of all other classifiers was adjusted accordingly.

For ease of exposition, the post-factum training times on the binary problem of classifying cover versus nsF5 0.05 payloads are summarised under the relevant headings below[13]. Two benchmarks are provided: the total walltime and per-example training times. For iterative experiments the walltime is measured as training time until best model[14] - any additional time spent training is considered as part of optimisation. For one-pass classifiers both timings include on-the-fly embedding and feature extraction. The timings shown are for illustrative purposes only and should not be used as a formal benchmark.

Average Perceptron

Using iterative training and over 500 GB of memory AP took only 10 hours to train on the full training set. This amounted to per-example benchmark of 0.00045 seconds. In contrast a one-pass training run with minimum memory overhead required over 80 hours. The online update, which included our simulation of payload embedding as well as on-the-fly feature extraction required 0.18 seconds.

Support Vector Machine

The third-party implementation of SVM used 40 hours to train until conver-

[13]Please note that there is some noise in these measurements which is due the availability of resources/scheduling as well as different implementations of classifiers.

[14]Best model is found using validation error.

gence in iterative setting, which gives an average of 0.00113 seconds per update. The factor of four difference in walltime benchmarks is slightly inflated due to I/O, which was longer in the SVM's case due to scheduling. In practice the training updates were taking between 2 and 3 times longer.

Online training with the same system and a single pass through the data required approximately 120 hours in total or 0.27 seconds per update.

Ensemble Fisher Linear Discriminant

EFLD's training on 20 000 examples required approximately 10 hours and 1.8 seconds per example in the batch setting or 2.7 seconds per example in the pseudo-online[15] setting.

Online Ensemble Average Perceptron

OEAP was trained on the full training set of 1.6M examples in 320 hours or approximately 0.72 seconds per example[16].

Kernel Support Vector Machine

This could only be trained in batch setting and required 22 hours to train on a subset of 20 000 examples, which is approximately 3.96 seconds per training example.

5.3.4.1 One-pass or iterations until convergence?

Processing the linear classifiers in memory on the full mismatched training set allows for fast execution of multiple iterations. However doing this requires abnormal computational resources, in particular to store circa 580 GB of data in memory. To evaluate whether this yields a significant improvement over online training of the linear classifiers we performed several tests. An intermediate solution between these two cases would be to perform mini-batch training. We have not fully investigated this option except in [81] where some positive results were found (reproduced in this dissertation in Figure B.1 in the appendix).

[15] See Section 4.2.3 for more details.

[16] No iterative training was performed here because OEAP appeared to converge well within the progress of a single pass (see Figure 5.8).

In normal circumstances, when we have limited computational resources, we might be forced to do online or mini-batch training due to a fixed resource budget. In online training the speed is traded for the storage and memory resources. The empirical evidence of this trade-off can be seen in the differences in speed per example of the SGD trained linear algorithms (AP and SVM), which were given as "per example" training times in the previous section. In the fully-online training the extra time is taken by payload embedding and feature extraction both of which combined are up to 150 times slower than a classifier's training update. Table A.1 in the appendix gives a full breakdown of timing benchmarks for these operations for AP and EFLD - the fastest and the slowest of our online-capable classifiers.

To investigate how much performance is lost by having only a single pass some extensive experiments were performed to mirror those that appear in Chapter 6. The results are included in Tables C.1 and C.2 in the appendix for completeness.

We started by looking at the case of detecting nsF5 using a batch-trained AP, which yields approximately 1% gain in μ and min results over the model trained in one iteration. The question of whether the full 1.6M example training set was required for training our classifiers (those that were capable) requires careful consideration. A simple answer is provided in Table 5.3, but we will return to this issue in more detail in Chapter 7. The table shows that using less training data seems to have a negative impact on the success of AP (the μ column). Here the tests in which subsets of 16 000 and 160 000 examples (also shown in Figure 5.10) from the training set were used for iterating until convergence and were each repeated five times using a random subsample each time with the more diverse sampling strategy. In contrast in the tests marked with * only one training run was performed because no resampling is possible all examples were used for training. On a closer inspection, we can see that for sPQ the difference between one-pass and batch figures is rather large. In the subsequent experiments we therefore use batch training where possible.

Table 5.3: Heterogeneous mismatched results using AP in batch and online settings. In batch setting we continued iterating until early stopping criterion satisfied. A single pass was made in online setting.

Payload	Training regime	Training size	μ	Training accuracy
nsF5 at 0.05 bpnc	batch	16 000	0.7956	0.9322
		160 000	0.8375	0.8915
		1 600 000	0.8640	0.8755
	online	1 600 000	0.8530	–
nsF5 at 0.1 bpnc	batch	16 000	0.9087	0.9936
		160 000	0.9630	0.9793
		1 600 000	0.9701	0.9766
	online	1 600 000	0.9689	–
sPQ at 0.2 bpuc	batch	16 000	0.6437	0.8843
		160 000	0.6763	0.7638
		1 600 000	0.7085	0.7276
	online	1 600 000	0.6808	–
sPQ at 0.4 bpuc	batch	16 000	0.7856	0.9655
		160 000	0.8340	0.9093
		1 600 000	0.8758	0.8934
	online	1 600 000	0.8495	–

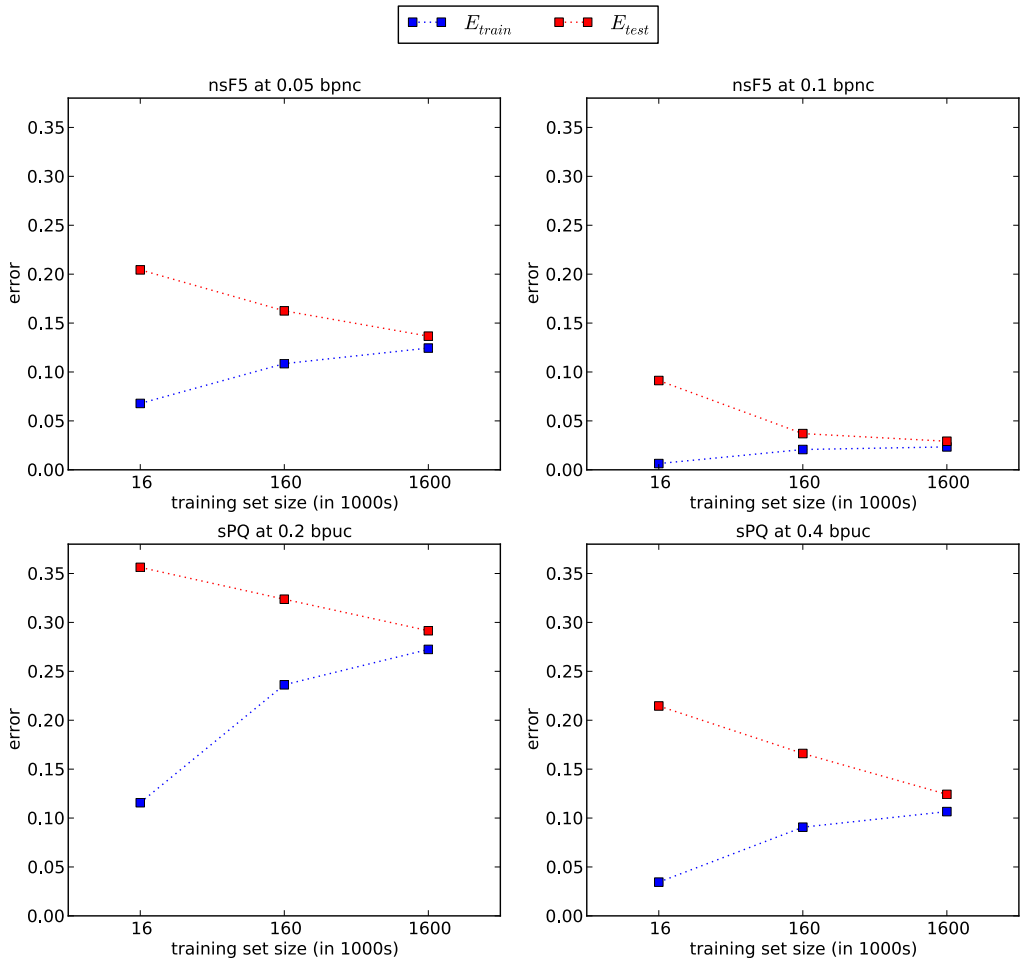


Figure 5.10: Convergence of AP as a function of increasing training set size. Please note the non-linear x-axis.

5.3.5 Uncertainty quantification (benchmarks)

In order to gain a solid understanding of the results all important comparisons are tested for statistical significance. We employ two statistical tests:

t-test is used for establishing if the difference between the means of two results is statistically significant. We employ Student two-sample t-test and, to account for unequal population variance, the Welch's degrees-of-freedom (df)-adjustment [102] is used. A classifier's accuracy on each actor's test data is taken as one data point.

F-ratio test is used to compare the variances. A classifier with lower variance is more stable, which is a desired property of any steganalysis detector and is a proxy for its tolerance to cover source mismatch.

It is unclear how we can measure the inherent amount of cover source mismatch that is present in a detection problem. In the experiments below we view this problem in terms of its affect on the classifiers' generalisation accuracy. We propose to measure it by statistically testing the difference of the variance of the *best* classifiers. This definition is somewhat analogous to the definition of steganographic security given in Chapter 1, where we say that the security of a steganography system is defined by the best detector. In Chapter 7 we give an example of a predictor of the cover source mismatch which is given as an empirical measure of discrepancy between the training and testing data.

5.4 Some implementation details

The code to perform our experiments incorporated many languages and libraries. We have the following stages in the processing pipeline, which can be done on a per-example basis in one-pass experiments or a per-data-set basis in batch experiments:

1. Payload embedding (own implementations of simulations in C and Python with interfaces to Independent JPEG Group's libjpeg library[17])

[17] `http://www.ijg.org`

2. Feature extraction (own reimplementation of Kodovsky's Matlab code[18] in C)

3. Classifier training update/test (own implementations of all classifiers in Python (with occasional interface to the Basic Linear Algebra Subprograms (BLAS) [12]) except for linear SVM and LR (Vowpal Wabbit [76]) and KSVM (LIBSVM [21]))

[18] `http://dde.binghamton.edu/download/feature_extractors`

Chapter 6

Experimental results

This chapter is devoted to introducing our experiments and their results. We aim to contribute to the field of steganalysis through a focus on real-world, large data to investigate cover source mismatch in a variety of settings, including some that have not previously been analysed. Specifically, we test a selection of classifiers in a series of three experiments, to examine whether choosing to train on large, mismatched data can have a positive effect on mitigating the cover source mismatch.

6.1 Background and motivation

There is a discrepancy between data used in steganalysis research and real-world data. One distinctive characteristic of real-world data is its abundance. Digital images are the most used type of media on the Internet and users upload millions of new pictures every day. The official figures from Facebook, from 2012, report that it has on average 300 million [28] photos being uploaded to it every day. Other major photo-sharing websites such as Flickr have comparable figures [1]. Unlike in other supervised image-based machine learning problems such as sentiment analysis where labels must often be annotated by hand, in steganalysis it is easy to acquire new examples; one simply downloads more images to get additional cover data and embeds a payload of choice to create stego data (assuming the source of images is known to contain only covers). The associated pipeline operations that are required for a typical experiment, namely payload embedding, feature extraction and class labelling, entail no manual processing and can be fast. With the right choice of clas-

sifier this, in theory, allows us to process virtually unlimited data. Many supervised classification algorithms apply, however simple linear learners are often associated with such problems owing to their fast training and straightforward convergence guarantees[1].

There are currently two caveats linked to real-world data. First, the data from a real-world sample, such as images from Facebook, contains examples from many different cameras, each example exhibiting a different mixture of characteristics that, as we established earlier, define a source (so images from Facebook would belong to many different sources). This creates difficulties for training a detector that generalises well to examples from a new source, as we show in the due course. Second, it is not generally possible to guarantee that all downloaded data are covers. We have to assume that it is and take it as the ground truth. Even if it is not, the likelihood of it having a visible effect on our results is very small as we do not believe that steganography is widespread enough on Facebook (or on the Internet in general) for our sample to be highly contaminated. Furthermore, given the fact that real-world data is generally noisy with images containing synthetic content and other anomalies, a small number of stego examples will simply compound that noise.

The alternative approach has underpinned the majority of steganalysis research to date. Steganalysis literature either implicitly or explicitly makes the assertion that an approach to steganalysis works if the classifier in question works on a data set from a single source (or a small collection of sources). Complex kernel-based classifiers are common throughout the literature [94, 70, 84, 44, amongst many others] and a typical testing scenario includes one or several image sets. These image sets typically come from a single cover source or a limited selection of sources. Often this is a personal photo library or a public image database such as the Natural Resources Conservation Service (NRCS) photo gallery [91]. One popular library that was first used in the Break Our Steganographic System (BOSS) contest in 2010 contains 10 000 images (version 1.0) from 7 different cameras [5]. In the literature,

[1]For example the Perceptron is guaranteed to find a separating hyperplane if one exists given long enough to train; SVM is guaranteed to find an optimal separator for a given training set, as its size tends to infinity; etc.

such a library is typically split into training and testing examples and the testing examples would serve as a proxy for the generalisation error of the detector. This setting is artificial not only because it is limited to one source (or a small selection of sources) but also because smaller data sets tend to produce undertrained models when using modern highly-dimensional feature representations. As we show further, there is a certain gap between the performance of detectors trained in this setting versus those that are trained on more realistic data.

Historically steganalysis was tested using this artificial approach, which is the reason why the steganalysis image libraries are relatively small (in the number of examples) compared to those used in other classification based machine learning related tasks. We show that this may be a problem for detectors based on machine learning classifiers. However it is generally difficult to acquire many images from a single source. One must take photographs with the same camera in a variety of different real-world conditions and process them using the same tools for compression, denoising and other operations which have also been found to contribute to the definition of a source. This is naturally a costly process and it would take an individual years before they could collect hundreds of thousands of "well-distributed" images. As it stands, the size of such individual homogenous data sets varies between 2000 (e.g. [59]) and 10 000 images; this figure is typically doubled if we consider the binary classification problem, because each image contributes two examples: one cover and one stego. Taking our sample of Facebook users as an example, out of more than 60 000 users visited only 26 had more than 3000 images and 1711 had more than 500 images. A similar situation was encountered in Flickr data.

In this artificial setting, many binary detectors have been shown capable of incredibly accurate performance [71, 51, and others], especially when detecting non-adaptive hiding schemes - furthermore they seem to be specialising to these conditions. However, in a realistic steganalysis situation the data is unlikely to be from a known source. Not knowing the source means that our classifiers need to be robust - they cannot be trained on data from the same distribution as the target data and a certain degree of performance degradation may be expected.

In steganalysis this problem is referred to as cover source mismatch and only a few publications have benchmarked the artificial setting against alternatives [4, 20, 40, 59, 94, 44]. The consensus amongst this research was for increased detection error in this setting, ranging from apparently significant to dramatic. It has been shown that combining data sets has a negative impact on the detection rates [43, 4, 94]. For example in [94] (Tables II and V) Pevny shows that training a KSVM model on images from four sets lead to the classification error increasing two-fold. A similar result was achieved in [4] with an FLD-based classifier.

As discussed in Chapter 3, the degradation in performance is often explained by the features being oversensitive to image content, e.g. Fridrich [35] suggests that steganalysis features like those used in [43, 4, 94] are sensitive to "microscopic" properties of the source. Because the content of images varies so widely and the stego signal is so weak, the features need to capture many local statistics about image pixels or their transform domain counterparts. These features have proven to be sensitive to many natural sources of variation that exist in images. It is known that different camera models, image processing tools (e.g. different implementations of the JPEG compression algorithm [67]) and even shooting conditions all have some effect on steganalysis features. It is still yet to be discovered how each of these factors (or perhaps some complex combination of them if they have some non-trivial interplay) influences the features. It is a challenging problem, because for many of these it is virtually impossible to vary them independently in order to measure their impact on steganalysis detection error. Furthermore, new feature sets have been proposed at a rate of one every six months which makes such evaluation problematic.

6.2 A review of solutions

There are several possible solutions to the cover source mismatch problem, however it is important to note that this problem remains largely unexplored [58]. Let us consider solutions which have been proposed previously in steganalysis literature.

First, one could gather or create new images from the correct source, however it

is unclear how to define a homogeneous source. As discussed above the physical situation which leads to it is dependent on many factors. Whilst it is possible to find a great deal of information about the target images using various image forensics tools such as extraction of the camera fingerprint [4, 83, 29, 39] it is not always possible to capture the full information about the cover source which therefore makes it very difficult to replicate the conditions in which the covers were created and thus to build a bespoke set of training images that represents exactly the target source distribution. This is referred to as forensics-aided steganalysis and it has not yielded improvement in a recent steganalysis competition [39].

Another solution [92] is to design a bank of experts system where an image in question would be first examined for known cover properties by an expert, such as a classifier trained to detect the presence of double compression and subsequently passed on to the next layer where a dedicated classifier trained on, say, double-compressed images only would make the final decision with regards to the presence of hidden payload of a known type. The bottleneck in this solution is in the requirement of the knowledge of what properties to test for in the first place - it is not yet well understood whether it is possible to identify and decorrelate all possible cover properties that may create mismatch.

An alternative solution is to design features that are less sensitive to cover source and therefore would not require training and test data to come from the same homogeneous source. This was largely the impetus behind the design of the 24 993-dimensional features which won the BOSS competition in [39]. Whilst in many cases (specifically the spatial domain) it has been shown that other modern high-dimensional steganalysis features (the "rich" models) improve accuracy in the situation of heterogeneous sources when compared to other features [39, 38, 51, 81], it is unclear whether this improvement is attributed to the general increased performance of such features or their reduced sensitivity to variations in the cover content. This idea is echoed by some recent theoretical studies, based on the theory of generalisation bounds for domain adaptation. Ben-David et al. [8] gives an upper bound on the generalisation error of a classifier trained and tested on data from two different

distributions - they discus how this bound depends on the features: "we see that a good representation [...] is one which achieves low values for both training error and domain $\mathcal{A}$-distance[2] simultaneously".

However, finding steganalysis features that provide a complete model of an empirical source is considered to be a hard problem. More importantly, if Alice was able to find such features there would be a danger of perfect steganography [15, 58].

Therefore, a more feasible solution perhaps would come from the field of machine learning, in particular the works that deal with the general problem of learning across domains and tasks. The plethora of relevant literature includes the topics of domain adaptation, transfer learning and multi-task learning amongst others. To our knowledge, the only published steganalysis work of this nature was [86], which was based on multi-task learning and yielded a statistical improvement in generalisation performance over the standard approach.

The main idea of domain adaptation (DA) is to build more robust models for the situation when training and test data are potentially generated by different sources. The success of a DA-based solution depends on the training (source) and the testing (target) data as well as their relatedness. The DA literature suggests that the choice of training data (matched or some mixture of mismatched sources or a combination of the two) is non-trivial [7, 87, 27] in contrast to the implicit assumptions in steganalysis literature [58], where matched data is favoured exclusively. For example, Theorem 5 from the domain adaptation work of Ben-David [7], which we study more closely in the next chapter, gives a uniform convergence bound for multi-source domain adaptation algorithms which minimise convex combinations of empirical source and target error. It makes explicit the trade-off between the size of data from the source domain (or a collection of sources) and the accuracy that can be achieved using target-only data. In some works [10] the authors go as far as saying that in the most optimistic scenario (i.e. under conditions that appear to be most favourable

[2]Domain $\mathcal{A}$-distance is a measure of distance between the distributions of training and test data - we define it more formally in Chapter 7 (shown as a more general disc_ℓ).

to DA) a DA approach will always[3] succeed in the limit of training data size [4]. Steganalysis is yet to be tested for most of these properties and assumptions.

Our work is perhaps the first step towards understanding of how ideas from domain adaptation can be applied to steganalysis. We look at the simplest solution which is sometimes referred to as conservative domain adaptation [99, 9, 10], but can also be thought of as not using domain adaptation (because we are not explicitly enriching the learner with any information about the test data). Formally, this approach entails training a maximally discriminating classifier on large quantities of mismatched training data. Several questions are apparent: Does this improve detectors' performance? What does "large" data mean - do we need hundreds of thousands or millions of training examples? Does the choice of classifier matter? We try to answer these questions here with the help of three experiments: 1) using homogeneous-matched data as a reference point for the "perfect" case; 2) using homogeneous-mismatched data for a simulation of the standard setting in which cover-source mismatch has been encountered; and 3) heterogeneous-mismatched data to test the proposed strategy. We argue that approach 3 is better than 2 but also can match or exceed the performance of 1 *in some cases*. The added difficulty to our analysis arises with consideration of classifiers of different complexity, because we cannot ultimately train $O(n^2)$ or slower classifiers on millions of training examples given modern tools (and normal practical constraints).

6.3 Hypothesis

We argue that the advantage of training a classifier on mismatched data many orders of magnitude larger than is normally possible in steganalysis in the matched case, may outweigh the loss caused by the mismatch between such data and the testing data. This is motivated by a fundamental result from the theory of statistical learning which tells us that a loss-minimising classifier is most likely to converge to the optimal solution in the limit of the training data[5] [114]. We show that this holds

[3] Or rather with high probability.
[4] More details to be discussed in next chapter.
[5] We will revisit this point more closely in the next chapter.

even for classifiers of different complexity and in Chapter 7 provide an analysis of why this might be the case.

We will work with the following hypotheses:

Hypothesis 1 : linear classifiers are more robust than non-linear classifiers to cover source mismatch in real-world data (because they can be trained on more data).

Hypothesis 2 : linear classifiers are so robust that given sufficiently large mismatched training data they can equal the performance of any classifier trained on small matched data.

We test our two hypotheses using three experiments. Hypothesis 1 will be supported by our results, whilst Hypothesis 2 will be invalidated. In turn, the sections below present the three experiments each providing empirical evidence either in favour or against the above hypotheses based on real-world data. In Experiment 1 we evaluate the classifiers' performance on matched data from one source. The assumption here is that each source is small. In Experiment 2 we test models from 1 on unseen sources - this provides the commonly accepted benchmark for cover source mismatch. Experiment 3 extends this to large data where the training data is a mixture of sources and the testing data is a collection of targets. In each experiment, the quality of the classifiers' predictions will be benchmarked against what is considered to be a state-of-the-art detector - a large feature set ("rich" model) coupled with a complex ensemble- or kernel-based classifier.

6.4 Experiment 1: Steganalysis in "laboratory" conditions

In this first experiment, we aim to replicate the typical laboratory conditions for steganalysis research, where the performance of classifiers is evaluated on a simple binary problem and training and test images come from the same data set. We deviate from the literature in two (small) ways. First, we repeat it for twenty-six different data sets instead of the one or two that would be typical. Second, we

consider only a small number of different problem parameters such as the embedding method and size of payload. Otherwise, this experiment is in line with many previous works appearing in the steganalysis literature. The preliminary results from this section were published previously in [82].

6.4.1 Procedure for Experiment 1

Here we are testing the fully-matched scenario which requires images in the training and test sets to be drawn from the same source, therefore one actor provides samples for both training and test sets. We assume that the image source is homogeneous for each actor but because this data was collected from real-world users there is a small chance of some users' images representing a mixture of sources.

The fully-matched data set $\boldsymbol{B}$ is composed of 26 actors having 4 000 cover images each. We follow the procedure described by Algorithm 3.

In this experiment the accuracy is measured on images T_e from the same actor $\boldsymbol{B_i}$ as the training images T_r. Training on T_r and testing on T_e involves creating cover and stego pairs of images from T_r and T_e respectively, i.e. stratified training. The experiment is repeated four times - on for each different payload: nsF5 with either 0.05 or 0.1 bits per non-zero DCT coefficient or sPQ with either 0.2 and 0.4 bits per usable DCT coefficient.

Algorithm 3 : Fully-Matched Data (Experiment 1)

1: **for** $\boldsymbol{B_i} \in \boldsymbol{B}$ **do**
2: Pick $T_r, T_e \in \boldsymbol{B_i}$, s.t. $|T_r| = 3\,000, |T_e| = 1\,000$ and $T_r \cap T_e = \emptyset$
3: Create pairs of cover and stego images from T_r and T_e
4: Randomise order of T_r
5: Train on T_r keeping track of estimated threshold t^*
6: Test on T_e using t^* or default t
7: **end for**

6.4.2 Results for Experiment 1

The results are given in Table 6.1, which shows the summary statistics across all 26 actors. The columns titled "μ" and "σ" show the average and the standard

deviation of detectors' testing accuracy across all actors. For more details on how these are calculated please refer to Section 5.3.1. The column "min" displays the worst testing accuracy achieved on any of the 26 actors and similarly for the "max" column. From the minimum and maximum figures we can see that the classifiers' accuracy varies up to 12% between different actors (compare the "min" and "max" columns in Table 6.1). This is a direct effect of the CC-C300 features' oversensitivity to variabilities in cover source and echoes our findings in much earlier spatial-domain WAM features[6] [59].

From the results, it is apparent that linear classifiers produced somewhat less accurate models when compared to KSVM and especially the EFLD. These results have been tested for statistical significance for two best classifiers in each group (AP and EFLD respectively), which is shown in the last two columns of Table 6.1. For all payloads the t-test shows a statistically significant advantage of EFLD over AP (all p-values are smaller than 0.01 which gives us 99% confidence level). At the same time, the two classifiers appeared to produce equally stable performances based on the F-test which showed no statistical significance in the difference of their variances[7].

In the unlikely scenario when we have access to larger training data set under the same conditions as used in this experiment (i.e. training and test data generated by the same distribution) there is potential for improving the performance of individual classifiers. However under the assumption that the data from the same source is likely to be limited it is safe to conclude that the results hold. We will use these figures as a benchmark when analysing the loss in generalisation accuracy which is due to cover source mismatch.

[6] Although it must be noted that the difference there was as large as 40% (cf. 12% for CC-C300 features) between some of the data sets

[7] Here and throughout this chapter we use the following null hypothesis: the true ratio of variances is equal to 1. If the resulting p-value is less than, say, 0.05 we can reject the null hypothesis at the 95% confidence level.

Table 6.1: Results from Experiment 1 comparing classifiers' accuracy in matched-homogeneous data using nsF5 at embedding rate of 0.05 and 0.1 bpnc and sPQ at embedding rate of 0.2 and 0.4 bpuc.

Problem	Classifier	Training size	Testing size	μ	σ	min	max	t-test $\mu_{EFLD} > \mu_{AP}$	F-test $\sigma^2_{EFLD} < \sigma^2_{AP}$
nsF5 at 0.05 bpnc	AP	6 000 images	2 000 images	0.8640	0.0280	0.810	0.918	$t = 3.448$ $df = 49.63$ $p < 0.01$	$F = 1.187$ $df = 25, 25$ $p > 0.1$
	SVM			0.8646	0.0267	0.816	0.926		
	KSVM			0.8727	0.0262	0.818	0.934		
	EFLD			0.8902	0.0257	0.843	0.942		
nsF5 at 0.1 bpnc	AP	6 000 images	2 000 images	0.9549	0.0125	0.935	0.979	$t = 8.468$ $df = 46.95$ $p < 0.01$	$F = 1.660$ $df = 25, 25$ $p > 0.1$
	SVM			0.9571	0.0116	0.935	0.982		
	KSVM			0.9779	0.0092	0.958	0.993		
	EFLD			0.9820	0.0097	0.954	0.997		
sPQ at 0.2 bpuc	AP	6 000 images	2 000 images	0.6761	0.0227	0.635	0.739	$t = 2.809$ $df = 49.07$ $p < 0.01$	$F = 0.758$ $df = 25, 25$ $p > 0.1$
	SVM			0.6761	0.0227	0.635	0.739		
	KSVM			0.6817	0.0228	0.637	0.738		
	EFLD			0.6954	0.0260	0.635	0.761		
sPQ at 0.4 bpuc	AP	6 000 images	2 000 images	0.8120	0.0245	0.783	0.884	$t = 4.541$ $df = 49.99$ $p < 0.01$	$F = 1.032$ $df = 25, 25$ $p > 0.1$
	SVM			0.8164	0.0268	0.776	0.885		
	KSVM			0.8280	0.0266	0.781	0.899		
	EFLD			0.8504	0.0241	0.803	0.914		

6.5 Experiment 2: Cover source mismatch in "laboratory" conditions

A simple extension to the above experiment is to perform conservative DA over all actors - that is to see how the models from Experiment 1 generalise to new targets. Here the data from each actor is used as "source" in turn while the other 25 actors are used as "targets". This produces the total of 650 performance samples[8]. However for a stable F-test the performance samples must be independent therefore it is necessary to limit this experiment to 26 tests. In each test we train on actor i and test on actor $i + 1$. More formally this procedure is shown in pseudocode in

[8] Appendix D.1 includes the detailed 26-by-26 tables, which include the 25 matched tests.

Algorithm 4. This experiment models exactly what is most commonly referred to as cover source mismatch problem. The particular contributions of this experiment compared to literature are two-fold: a) in the number of different data sets tested and b) in the variety of metrics used to report the results. We present the summary statistics from this experiment in Table 6.2.

Algorithm 4 : Homogeneous-Mismatched Data (Experiment 2)

1: **for** $\boldsymbol{B_i} \in \boldsymbol{B}$ **do**
2: j = i + 1
3: Pick $T_r \in \boldsymbol{B_i}$, s.t. $|T_r| = 3\,000$
4: Pick $T_e \in \boldsymbol{B_j}$, s.t. $|T_e| = 1\,000$
5: Create pairs of cover and stego images from T_r and T_e
6: Randomise order of T_r
7: Train on T_r keeping track of estimated threshold t^*
8: Test on T_e using t^* or default t
9: **end for**

6.5.1 Results for Experiment 2

The extent of the cover source mismatch is apparent from the summary statistics shown in Table 6.2. In all cases the difference in average performance between Experiments 1 and 2 is between 6% and 7%, which is statistically significant as shown by the t-test in Table 6.3. The standard deviation ("σ" column in Table 6.2) increased nearly three-fold. The most significant result however is the extent to which the cover source mismatch affects the worst-case classification accuracy. Here the observed absolute loss in performance is as high as 35% compared to the mean accuracy and the discrepancy between highest and lowest score is as high as 40% (cf. 12% in Experiment 1). In some cases the classification decisions were close to near-random (see column titled "min" in Table 6.2). This holds true across the board for all classifiers tested.

The statistical tests shown in Table 6.2 (the last two columns) also pinpoint that the difference between classifiers does not disappear in this setting (at 90% confidence level - the largest p-value appears for sPQ payload at 0.4 bpuc) even though the means indicate they are very close - the AP is still not as good as EFLD statistically.

Table 6.2: Results from Experiment 2 comparing classifiers' accuracy in mismatched-homogeneous data using nsF5 at changerate of 0.05 bpnc and sPQ at 0.4 bpuc.

Problem	Classifier	Training size	Testing size	μ	σ	min	max	t-test $\mu_{EFLD} > \mu_{AP}$	F-test $\sigma^2_{EFLD} < \sigma^2_{AP}$
nsF5 at 0.05 bpnc	AP	6 000 images	2 000 images	0.7844	0.0737	0.542	0.885	$t = 2.291$ $df = 41.92$ $p < 0.05$	$F = 0.390$ $df = 25, 25$ $p < 0.05$
	SVM			0.7903	0.0741	0.536	0.889		
	KSVM			0.7864	0.0880	0.509	0.911		
	EFLD			0.8011	0.0845	0.550	0.911		
nsF5 at 0.1 bpnc	AP	6 000 images	2 000 images	0.8546	0.0720	0.637	0.951	$t = 2.285$ $df = 48.05$ $p < 0.05$	$F = 0.665$ $df = 25, 25$ $p > 0.1$
	SVM			0.8655	0.0721	0.698	0.958		
	KSVM			0.9038	0.0850	0.613	0.983		
	EFLD			0.9066	0.0883	0.658	0.990		
sPQ at 0.2 bpuc	AP	6 000 images	2 000 images	0.6371	0.0295	0.539	0.687	$t = 1.958$ $df = 48.93$ $p < 0.1$	$F = 0.742$ $df = 25, 25$ $p > 0.1$
	SVM			0.6371	0.0295	0.539	0.687		
	KSVM			0.6369	0.0287	0.537	0.682		
	EFLD			0.6547	0.0342	0.543	0.702		
sPQ at 0.4 bpuc	AP	6 000 images	2 000 images	0.7555	0.0528	0.558	0.827	$t = 1.724$ $df = 48.79$ $p < 0.1$	$F = 0.727$ $df = 25, 25$ $p > 0.1$
	SVM			0.7588	0.0556	0.550	0.830		
	KSVM			0.7717	0.0407	0.678^{+}	0.848		
	EFLD			0.7835	0.0618	0.572	0.861		

Analysing the related tables in Appendix D.1, which give a breakdown of performance scores by actor for different classifiers and payloads, we can see that the columns of the tables are clearly more stable than the rows - this means that easy testing data will be classified well by most models, whilst a given model will not generalise well to many sources. Moreover a good performance by a model on matched data does not imply good generalisation (and vice versa). It may therefore be counterproductive to try to select the "best" model from a pool of single-source-trained models. A better strategy would perhaps be finding a source that is "closest" in terms of features to a given target. This is, however, problematic when the dimensionality of features is high because the distances in high dimensions are likely to be

unstable[9].

This experiment represents the most conservative domain adaptation - we tried adapting to a new domain without informing the model about it in any way. We argue however that by adopting the simplest solution which does not entail changing the training regime we can improve on the results from Experiment 2 (and also from Experiment 1 to some degree). Our approach entails training on a sufficiently large and diverse mismatched data set therefore allowing the classifier to estimate better parameters (as more data is given). The diversity of data provides further advantages, which we discuss in Chapter 7. Experiment 3 follows to test this.

Table 6.3: Comparing classifiers' performance on matched-homogeneous and mismatched-homogeneous data (Experiment 1 (μ_1) v Experiment 2 (μ_2)).

Payload	AP $\mu_1 > \mu_2$	EFLD $\mu_1 > \mu_2$
nsF5 0.05	$t = 9.084$ $df = 38.03$ $p < 0.001$	$t = 5.040$ $df = 29.57$ $p < 0.001$
nsF5 0.1	$t = 6.862$ $df = 26.51$ $p < 0.001$	$t = 4.233$ $df = 25.63$ $p < 0.001$
sPQ 0.2	$t = 5.243$ $df = 46.88$ $p < 0.001$	$t = 4.731$ $df = 46.66$ $p < 0.001$
sPQ 0.4	$t = 5.476$ $df = 35.32$ $p < 0.001$	$t = 5.039$ $df = 32.44$ $p < 0.001$

[9]Several solutions were tested without success (see "Future work" section in Chapter 8).

6.6 Experiment 3: Real-world data: linear and non-linear classification in mismatched data sources

Here we use data from the same data set as in the previous two experiments but expand it to incorporate the total of 1600 training and 100 testing actors. Each actor is represented by a set of 500 cover images. This allows for as many as 1.6M training examples for binary classification which is several orders of magnitude larger than any steganalysis data set used previously with the exception of [63] where up to several hundreds of thousands of images were used for training. For this experiment we introduce a second ensemble-based non-linear classifier - the online ensemble Average Perceptron (OEAP), which has the advantage of speed over the other two non-linear classifiers and can therefore be applied to the full data set in a reasonable time.

6.6.1 Procedure for Experiment 3

Algorithm 5 describes the general procedure employed in this experiment. Unlike in the other two experiments where training data varied with each training actor (hence potentially providing a training set of different relative difficulty), here the training set is fixed (all 1.6M examples) for all linear classifiers and the OEAP classifier. For KSVM and EFLD we used subsets of this data for training for the reasons of computational complexity. For these classifiers the experiment was repeated five times to eliminate any possibility of noise in the results due to an unlucky sample. As before, we test four different binary classification problems, in each case detecting one of the following payloads amongst covers at even proportion: nsF5 embedding operation with 0.05 and 0.1 bpnc and sPQ embedding operation with 0.2 and 0.4 bpuc. A fixed testing set of 100 actors is used and is disjoint from the set of training actors. This procedure is described in pseudocode in Algorithm 5.

6.6.2 Results for Experiment 3

We will use this section to briefly discuss the results that are local to this experiment and perform a formal comparison between all three experiments in the next section.

Table 6.4: Results from Experiment 3 showing the summary statistics from classifiers' performance on mismatched-heterogeneous data.

Experiment	Classifier	Threshold	Training Size	Testing Size	μ	σ	min	max
linear classifiers nsF5 0.05 bpnc	AP	default	1 600 actors × 1 000 images	100 actors × 1 000 images	0.8640	0.0361	0.658	0.912
		adaptive			0.8647	0.0354	0.657	0.914
	SVM	default			0.8756	0.0382	0.658	0.918
		adaptive			0.8644	0.0432	0.646	0.931
non-linear classifiers nsF5 0.05 bpnc	KSVM	default	5 × 20 000 images	100 actors × 1 000 images	0.8381	0.0392	0.648	0.906
		adaptive			0.8384	0.0383	0.646	0.907
	EFLD	default	5 × 20 000 images		0.8400	0.0384	0.637	0.892
		adaptive			0.8370	0.0384	0.633	0.902
	OEAP	default	1 600 actors × 1 000 images		0.8482	0.0427	0.648	0.908
		adaptive			0.8497	0.0379	0.648	0.900
linear classifiers nsF5 0.1 bpnc	AP	default	1 600 actors × 1 000 images	100 actors × 1 000 images	0.9701	0.0271	0.771	0.991
		adaptive			0.9701	0.0271	0.770	0.990
	SVM	default			0.9764	0.0266	0.778	0.996
		adaptive			0.9769	0.0257	0.781	0.996
non-linear classifiers nsF5 0.1 bpnc	KSVM	default	5 × 20 000 images	100 actors × 1 000 images	0.9682	0.0272	0.798	0.994
		adaptive			0.9685	0.0270	0.800	0.993
	EFLD	default	5 × 20 000 images		0.9688	0.0301	0.759	0.994
		adaptive			0.9687	0.0303	0.757	0.994
	OEAP	default	1 600 actors × 1 000 images		0.9711	0.0278	0.790	0.993
		adaptive			0.9714	0.0278	0.778	0.992
linear classifiers sPQ 0.2 bpuc	AP	default	1 600 actors × 1 000 images	100 actors × 1 000 images	0.7085	0.0345	0.575	0.773
		adaptive			0.7145	0.0305	0.589	0.763
	SVM	default			0.7171	0.0336	0.581	0.770
		adaptive			0.6839	0.0350	0.541	0.765
non-linear classifiers sPQ 0.2 bpuc	KSVM	default	5 × 20 000 images	100 actors × 1 000 images	0.6685	0.0257	0.577	0.739
		adaptive			0.6689	0.0254	0.578	0.578
	EFLD	default	5 × 20 000 images		0.6855	0.0280	0.574	0.742
		adaptive			0.6762	0.0289	0.565	0.730
	OEAP	default	1 600 actors × 1 000 images		0.6627	0.0364	0.550	0.741
		adaptive			0.6824	0.0299	0.576	0.746
linear classifiers sPQ 0.4 bpuc	AP	default	1 600 actors × 1 000 images	100 actors × 1 000 images	0.8758	0.0376	0.692	0.937
		adaptive			0.8760	0.0372	0.697	0.929
	SVM	default			0.8781	0.0412	0.684	0.930
		adaptive			0.8755	0.0422	0.667	0.932
non-linear classifiers sPQ 0.4 bpuc	KSVM	default	5 × 20 000 images	100 actors × 1 000 images	0.8142	0.0394	0.654	0.894
		adaptive			0.8142	0.0394	0.654	0.894
	EFLD	default	5 × 20 000 images		0.8325	0.0420	0.686	0.889
		adaptive			0.8266	0.0424	0.659	0.901
	OEAP	default	1 600 actors × 1 000 images		0.8358	0.0464	0.638	0.907
		adaptive			0.8399	0.0444	0.654	0.901

Algorithm 5 : Heterogeneous-Mismatched Data (Experiment 3)

if not using full training set **then**
  for iteration 1...5 **do**
    Split $\boldsymbol{D}$ into training actors $\boldsymbol{T}$ and testing actors $\boldsymbol{A}$
    Pick $T_r \subset \bigcup \boldsymbol{T}$, s.t. $|T_r| = 10\,000$
    Pick $\boldsymbol{A'} \subset \boldsymbol{A}$, s.t. $|\boldsymbol{A'}| = 10$
    $T_e = \bigcup \boldsymbol{A'}$
    Randomise order of T_r
    Train on T_r, keeping track of estimated threshold t^*
    Test on T_e using t^* or default t
  end for
else
  Split $\boldsymbol{D}$ into training actors $\boldsymbol{T}$ and testing actors $\boldsymbol{A}$
  $T_r = \bigcup \boldsymbol{T}$
  Pick $\boldsymbol{A'} \subset \boldsymbol{A}$, s.t. $|\boldsymbol{A'}| = 100$
  $T_e = \bigcup \boldsymbol{A'}$
  Randomise order of T_r
  Train on T_r, keeping track of estimated threshold t^*
  Test on T_e using t^* or default t
end if

Table 6.4 summarises the results. As before we compare linear and non-linear classifiers' performance. The mean and the minimum accuracy figures indicate that in this mismatched setting the non-linear classifiers did not produce the best models unlike in the other experiments. Based on the two metrics the most accurate models overall were produced by SVM and OEAP in the linear and non-linear groups of classifiers respectively.

The speed advantage of linear classifiers allowed for performing multiple iterations over the full training data which resulted in an extra performance advantage. We provide one-pass accuracy figures of linear classifiers in Tables C.1 and C.2. Whilst it might have been possible to train the ensemble classifiers in the same manner with extra time allowance, we expect this would not have yielded better performance figures because, as was shown in the previous chapter (Figure 5.8), they appear to converge 1/8th of the way into a one-pass training run.

Statistically neither of the three non-linear classifiers matched the results of the linear classifiers. This is highlighted by the following results from statistical tests (Table 6.5) where the comparison figures are given for the two best classifiers from both groups. SVM statistically outperforms ensemble classifiers in terms of averages (t-test at 99.9% confidence level) in all payloads, except for the nsF5 with 0.1 bpnc. The same cannot be said about the F-test results, which showed no significance in the difference of their variances across all four payloads.

The above discussion goes in line with our first hypothesis and we can conclude that we found no evidence to reject it. This means that in accordance with the given figures the linear classifiers perform better than non-linear classifier in mismatched data when using modern large feature sets and detecting some of the commonly-studied payloads. We leave a closer analysis of why this might be the case until the next chapter.

Table 6.5: Comparing the performance of the best performing classifiers from linear and non-linear groups of classifiers on data from mismatched-heterogeneous sources: SVM (μ_1, σ_1^2) v ensembles (μ_2, σ_2^2).

	nsF5		sPQ	
Test	0.05	0.1	0.2	0.4
	SVM v OEAP	SVM v OEAP	SVM v EFLD	SVM v OEAP
$\mu_1 > \mu_2$	$t = 4.7486$ $df = 195.561$ $p < 0.001$	$t = 1.3819$ $df = 197.646$ $p > 0.1$	$t = 7.1882$ $df = 191.76$ $p < 0.001$	$t = 6.7773$ $df = 195.187$ $p < 0.001$
$\sigma_1^2 < \sigma_2^2$	$F = 0.7991$ $df = 99, 99$ $p > 0.1$	$F = 0.9188$ $df = 99, 99$ $p > 0.1$	$F = 1.4402$ $df = 99, 99$ $p < 0.1$	$F = 0.7856$ $df = 99, 99$ $p > 0.1$

6.7 Reconciling results across the three experiments

The variability of the detection accuracy between actors is an indicator of the extent of the penalty incurred by our classifiers due to cover source mismatch, which is what we are ultimately interested in in this part of the thesis. This penalty is referred to as adaptation loss. As was hinted at by the brief analysis in the results sections of the first two experiments two metrics reflect it best: the absolute minimum and the standard deviation over multiple targets (test actors). By design our reference point is the performance figures from the best detector trained and tested on matched data (i.e. Experiment 1), which is the best-case scenario for training a detector. Our comparison utilises the figures from the three experiments in straightforward F-tests of the variances of the best models.

We start with the comparison of the results from Experiments 2 and 3. Given a fixed classifier by training on a mixture of mismatched sources, as per Experiment 3, we consistently gain between 12% and 15% in minimum accuracy compared to a straightforward mismatch from Experiment 2 (see columns "min" in Tables 6.2 and 6.1 respectively). In addition, the standard deviation ("σ" column) decreases dramatically. The significance of this performance improvement is shown in Table 6.6. Using an F-test on AP and EFLD as an example (all others follow the same pattern) we show that for both classifiers the reduction in variance is statistically significant - all p-values are smaller than 0.05 which gives us a 95% confidence level, with the exception of the sPQ 0.2 case. This, however, is probably due to the F-test not being strong enough for this case - it is clear from the figures that the classifier from Experiment 3 is superior. Similarly, this also holds for the EFLD classifier.

More interesting, perhaps, is the difference in classifiers performance between Experiments 1 and 3 because it serves as the first indicator of whether our second hypothesis holds: the classifiers trained on large heterogeneous-mismatched data can be as good as classifiers trained on small homogeneous-matched data. As before we take AP and EFLD as example classifiers. In contrast to our hypothesis, Table 6.7 shows that at 95% confidence level ($p < 0.05$) the models trained on

Table 6.6: Comparing classifiers' performance on mismatched-homogeneous and mismatched-heterogeneous data (Experiment 2 (σ_1^2) v Experiment 3 (σ_2^2)).

Classifier	nsF5 0.05	nsF5 0.1	sPQ 0.2	sPQ 0.4
AP $\sigma_1^2 < \sigma_2^2$	$F = 2.2021$ $df = 25, 99$ $p < 0.01$	$F = 7.2909$ $df = 25, 99$ $p < 0.001$	$F = 0.7528$ $df = 25, 99$ $p > 0.1$	$F = 2.0237$ $df = 25, 99$ $p < 0.05$
EFLD $\sigma_1^2 < \sigma_2^2$	$F = 4.9964$ $df = 25, 99$ $p < 0.001$	$F = 8.8796$ $df = 25, 99$ $p < 0.001$	$F = 1.5401$ $df = 25, 99$ $p > 0.1$	$F = 2.2298$ $df = 25, 99$ $p < 0.01$

heterogeneous-mismatched data produced significantly more variability in their performance that those trained on homogeneous-matched data. As with the previous discussion of Table 6.6, here we also have outliers (AP tested on nsF5 at 0.05 bpnc payload and EFLD tested on sPQ at 0.2 bpuc payload).

Finally, the test for statistical significance of the difference in the variability of the best models from Experiments 1 and 3, is given in Table 6.8. This allows us to test our second hypothesis in full. Shown are the results of F-test comparing the variance of linear SVM trained on large heterogeneous-mismatched data with the variance of the non-linear EFLD classifier trained on small homogeneous-matched data. Again they contradict our second hypothesis (all p-values are smaller than 0.05, with the exception of the outlier in the sPQ 0.2 tests), which leads us to conclude that there is sufficient evidence to reject it. In addition to the Table 6.8, Figure 6.1 provides violin plots for the ease of exposition of our results. A violin plot is similar to a histogram - it provides a view of the shape of an estimate of the probability distribution of the data (in our case data points are accuracy values). The solid horizontal line shows the median and the top and bottom dashed lines show the 75th- and 25th-percentile respectively. A notable feature in these plots is the "tails" introduced by the cover source mismatch tested in Experiments 2 (red

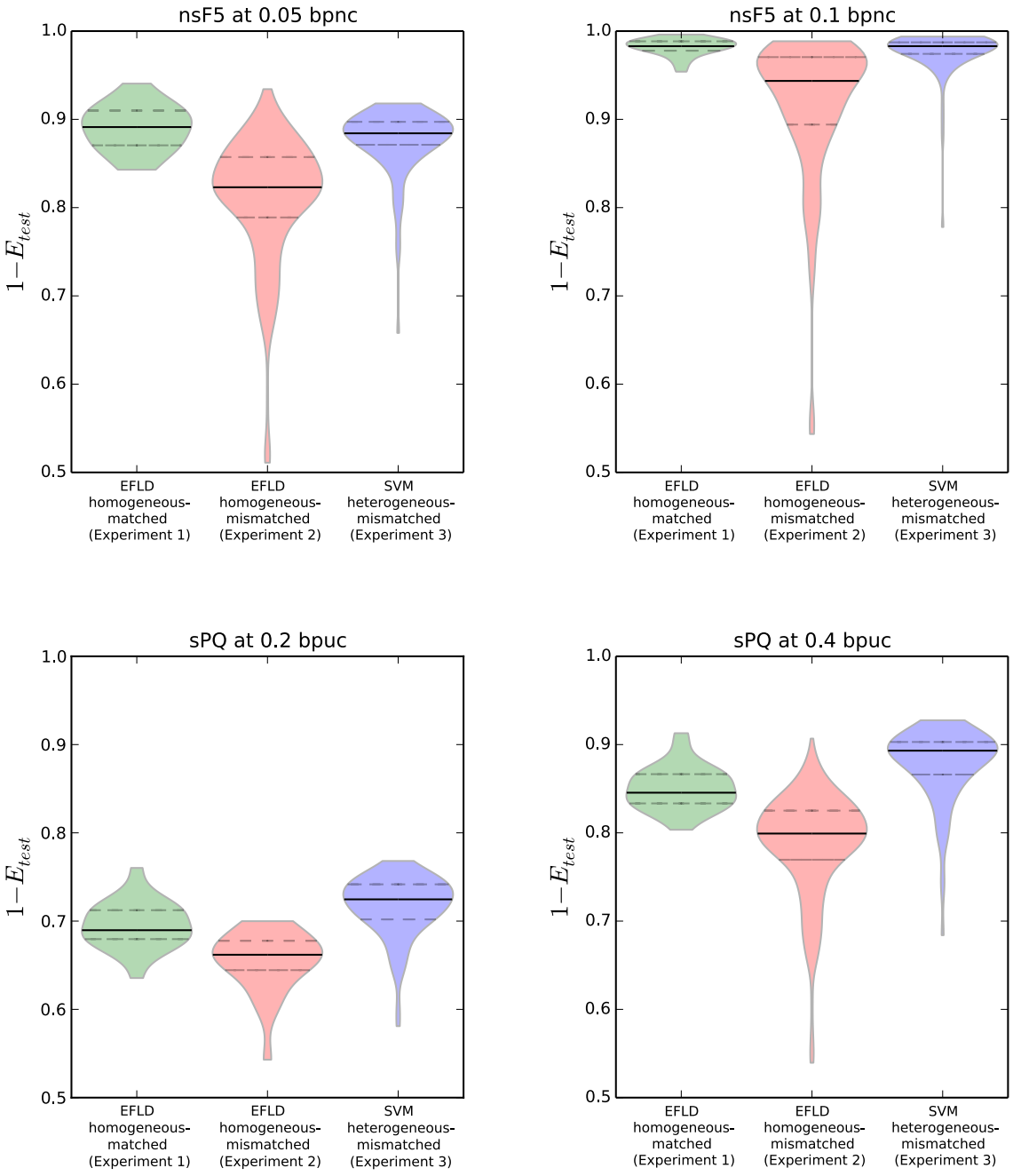


Figure 6.1: Violin plots for the best classifier in each of the three experiments (red, green and blue) presented in this chapter; grouped by payload type and size. The red violins are based on 650 performance samples each (see Section 6.5) for details.

Table 6.7: Comparing classifiers' performance on matched-homogeneous and mismatched-heterogeneous data (Experiment 1 (σ_1^2) v Experiment 3 (σ_2^2)).

Classifier	nsF5 0.05	nsF5 0.1	sPQ 0.2	sPQ 0.4
AP $\sigma_1^2 < \sigma_2^2$	$F = 0.6194$ $df = 25, 99$ $p > 0.1$	$F = 0.2198$ $df = 25, 99$ $p < 0.001$	$F = 0.4441$ $df = 25, 99$ $p < 0.05$	$F = 0.4373$ $df = 25, 99$ $p < 0.05$
EFLD $\sigma_1^2 < \sigma_2^2$	$F = 0.4604$ $df = 25, 99$ $p < 0.05$	$F = 0.1072$ $df = 25, 99$ $p < 0.001$	$F = 0.8902$ $df = 25, 99$ $p > 0.1$	$F = 0.3396$ $df = 25, 99$ $p < 0.01$

Table 6.8: Comparing performance of the best classifier in matched-homogeneous experiment and the best classifier in mismatched-heterogeneous experiment (Experiment 1 v Experiment 3).

Classifier	nsF5 0.05	nsF5 0.1	sPQ 0.2	sPQ 0.4
$\sigma_{EFLD}^2 < \sigma_{SVM}^2$	$F = 2.154$ $df = 99, 25$ $p < 0.05$	$F = 7.295$ $df = 99, 25$ $p < 0.001$	$F = 1.618$ $df = 99, 25$ $p > 0.1$	$F = 2.824$ $df = 99, 25$ $p < 0.01$

violins) and Experiment 3 (blue violins) and the rather large difference between them. It is easy to see how the performance in the latter is far less variable.

Consider the differences between the top two plots and the bottom two plots. In the top two plots the performance of the detector trained on small homogeneous-matched data (green violins) is notably superior to that of the detector trained on large heterogeneous-mismatched data (blue violins), which is also supported by the mean figures and the F-tests. In the bottom two plots this is also the case, which is made transparent by the violin plots whilst being masked by the mean scores from

the tables.

Three main results emerged from the comparison between Experiments 1, 2 and 3:

1. It is straightforward, from the figures and the plots, that detectors from Experiment 3 are better than the detectors from Experiment 2. Instead of having an enormous loss due to cover source mismatch in the case of the mismatched-homogeneous setting (Experiment 2) we now have a much smaller loss under the mismatched-heterogeneous setting (Experiment 3). Table 6.6 shows statistical tests to support this[10].

2. Despite what the μ scores might suggest (especially in the case of detecting sPQ), the models from Experiment 3 are still statistically worse than the models from Experiment 1 (at least until we can fix the minimum accuracy via some DA-based means - to be discussed in Chapter 7). This is reflected by the violin plots and the F-test in Table 6.8. As the discussion above shows it is not just the mean of detection accuracy/error that matters. The average sometimes masks poor performance, which is visible in the violin plots given in Figure 6.1. If a robust detector was needed and a choice of small homogeneous-matched and large heterogeneous-mismatched data was given then one should choose to use the small matched data for training, providing there is enough data to train a good model - a point which we discuss in more detail in the next chapter. We propose that detectors' performance should be measured on individual sources and displayed, for example, using a violin plot.

3. There is a discrepancy in the ranking of linear and non-linear classifiers between Experiments 1 to 2 and Experiment 3. There are two potential reasons for the lower relative performance of the non-linear classifiers in the third experiment: a) ensembles do not work well in the mismatched setting, which is reflected

[10]We opted for not performing statistical tests on the figures for minimum performance for two reasons: a) because we cannot expect the minimums to be comparable if one sample size is four times larger than the other sample size; and b) because such tests are likely to be unstable as they are very sensitive to the underlying distribution which we do not know. Therefore only the F-test is given in this case.

in the lack of performance improvement beyond certain point during their training; and b) kernel classifiers are undertrained on this data due to the computational constraints put upon them. We will explore this point further in the next chapter.

All three points support our argument that there is value in training on more data as it tends to produce better models. We can conclude that we found no evidence to reject our first hypothesis as per point 3 above. However our second hypothesis must be rejected based on the evidence from the F-test. The test which is shown in Table 6.8 indicates a statistically significant difference in the variance of the best models from each of the experiments in question in favour of Experiment 1 which modelled the matched setting.

The vast improvement in results between Experiments 2 and 3 as well as the improvement in the mean accuracy scores between some of the tests (sPQ) in Experiments 1 and 3 suggests that there is promise in DA-based solutions given that we have only tried the most conservative DA regime in which the classifier was not informed with any information about the targets. Consider the following argument. Imagine we are prepared to accept the same level of minimum accuracy as that which appeared in the worst-case scenario from the best classifier (EFLD) from the *matched* setting of Experiment 1. Then we can see how often the best classifier (SVM) from Experiment 3 yields lower performance in the heterogeneous-mismatched scenario. It turns out that the levels are violated 14 out of 100 times in the nsF5 0.05 test, 11 out of 100 times in the nsF5 0.1 test and only 4 out of 100 times in both sPQ 0.2 and 0.4 tests. All other results are admissible under the above conditions. This comes in contrast with the results from Experiment 2 where the admissible versus violating counts are reversed. If we were able to find some special treatment for the anomalous cases we would have a state-of-the-art detector for binary steganalysis which uses *mismatched*-only data. We review some possible solutions in the next chapter.

6.8 Conclusions and critical assessment

Whilst 6000 training examples in the matched case might seem like an arbitrary number, in many applications we cannot reasonably expect to be provided with a matched training set of any particular size and almost certainly not of the same order of magnitude as the mismatched data set used in Experiment 3. Two scenarios come to mind in which it might be possible: a) folders which are explicitly labelled as containing cover/stego images were found and another folder containing images with unknown labels and all of them appear to have been taken by the same camera in similar conditions, etc. b) the source camera got seized so we can take as many pictures as we like with it (and the shooting conditions/other parameters for the target pictures can be replicated as well). In these particular applications large matched training data may be available, but one should not expect it to be a standard feature.

It must be noted that the link between our simulation and the real-world is still somewhat tenuous. There are certain assumptions that have been made, such as the equal distribution of cover and stego examples in the test sets; a fixed change rate; our perfect knowledge of the embedding schemes and some others, which may produce a different kind of mismatch and thus require more elaborate training regimes. However we believe our simulations have been valuable in the sense that they came closer than ever to the real-world situation and helped us understand the value of larger training sets. We performed experiments in order to observe the cover source mismatch when the classifiers are trained and tested on different users and proposed a selection of different ways to measure it. We have shown that looking at variability and the worst-case accuracy is invaluable when considering the cover source mismatch because the average accuracy can be misleading. We have shown some evidence in favour of the hypothesis that, in practice, linear classifiers work better with the modern rich steganalysis features than the more complex non-linear classifiers. A basic DA-based method to reduce the impact of cover source mismatch was proposed. It was based on the idea that training over a large data set composed of a mixture of mismatched sources may produce a well-generalising classifier if the training sources are "similar enough" to the testing sources. This idea has

good theoretical foundations as we see in the next chapter, albeit with some strong assumptions about the relationship between the training (source) and the testing (target) data. It was shown to have particular promise owing to it yielding an increased performance relative to other settings at least in terms of some of the metrics.

We can summarise the findings with this feature set as follows.

- Given a fixed feature set and similar time constraints, linear classifiers are capable of producing better performance than complex classifiers.
- Linear and complex classifiers produce equally good models from small matched and large mismatched training data if evaluated on average scores, but statistically different if evaluated on minimum scores.
- In steganalysis the worst-case performance is crucial.
- Sampling: in a mismatched scenario more diversity between training sources is better.

Chapter 7

Analysis

In the previous chapter we showed that our classifiers, when trained on a large data set composed of different mismatched sources (Experiment 3), were capable of producing performance which was equivalent to, or better than, the matched case (Experiment 1) *for the majority of testing sources*. This is contrary to previous findings in the steganalysis literature as well as our result from Experiment 2 which employs largely the same strategy. We consider all those cases as examples of successful domain adaptation and this chapter is devoted to explaining this result. Statistical learning theory and its recent developments with respect to domain adaptation are employed.

We begin this chapter with a brief introduction of the necessary concepts from statistical learning theory. This is followed by the presentation of a major theoretical result from Ben-David et al. [7]: a learning bound for conservative domain adaptation based on the measures of relatedness between the source (training) and target (testing) data. We show how this theory explains some of our findings from Chapter 6. This is followed by a second major result from [7]: an extension to the first learning bound, which considers a simple combination of labelled source and target data. Some general deliberations of what this result entails for steganalysis and steganography follow. This chapter leads to many open questions for steganalysis.

The training regime described and tested in Experiment 3 is commonly referred

to in the literature as *conservative* domain adaptation [99]. In this setting the training data will be referred to as the source domain (or a collection of source domains) and the testing data as the target domain (each actor is a different target domain). In contrast, *adaptive* domain adaptation uses prior information about the target domain such as, for example, a (large) set of unlabelled examples from the target domain with the aim of improving on the conservative approach. Examples of adaptive learners include numerous heuristics which are largely aimed at minimising the discrepancy between the domains. Examples include importance weighting [26] and pivot features [13]. In this chapter we will concern ourselves with only conservative domain adaptation.

7.1 Statistical learning theory for steganalysis

7.1.1 Introduction

As stated in Section 1.3, our motivation for this study comes from the idea that more (labelled) data helps with training a better model. This may be especially important in adversarial conditions that may be natural to steganalysis when we might not have any prior knowledge, or have reservations about the correctness of our prior knowledge, of the distribution of the target data. The setting in which we assume no prior knowledge is called agnostic learning. This idea is deeply rooted in statistical learning theory.

Statistical learning theory is concerned with several theoretical models of learning including empirical risk minimisation [114]. It makes the assumptions of the training data being independent and identically distributed, same distribution for training and testing data (as per Experiment 1) which is also fixed over time. Many machine learning classifiers, including regularisation-based algorithms such as Support Vector Machines, as well as ensembles such as presented here fall under this theoretical framework.

A major contribution of statistical learning theory is the definition of generalisation bounds, which are central to our analysis. Before we can discuss generalisation

bounds we need to introduce several important concepts.

Conservative domain adaptation is the empirical risk minimisation (ERM) over the available training data. Therefore, it describes exactly the regime that was used in Experiment 3 of Chapter 6. Statistical learning theory will not only help us analyse the results, but will also put steganalysis in the framework of theoretical machine learning, specifically comparing the regular training to domain adaptation based settings.

7.1.2 Formalising concepts

Recall from Section 4.1 that we can think of a classifier in terms of the class $\boldsymbol{\mathcal{H}}$ of all hypotheses h it can represent. As stated, we assume that both the training and the test data are drawn as independent identically distributed samples from the same domain $\boldsymbol{\mathcal{X}}$ and that the joint probability distribution P of example-label pairs $(\boldsymbol{x}_i, y_i) \in (X, Y)$ over $\boldsymbol{\mathcal{X}}$ is fixed.

In agnostic learning, the learning task is to use the training examples in search of a hypothesis $h \in \boldsymbol{\mathcal{H}}$ with minimum expected risk over any examples generated by P, including all test examples. In classification the *expected risk* of some hypothesis h can be defined using 0-1 loss:

$$R(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}\left(\ell_{\text{0-1}}\left(\boldsymbol{x}, y, h\right)\right).$$

Recall that 0-1 loss assigns a fixed cost of 1 to each wrong prediction and 0 to each correct prediction. It is clear that a hypothesis h is better than another hypothesis g if $R(h) < R(g)$. Given P we can define the hypothesis h_{Bayes} that gives the minimum risk $R(h_{\text{Bayes}})$ as the so called Bayes-classifier [115]:

$$h_{\text{Bayes}}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \Pr(Y = 1 | X = \boldsymbol{x}) \geq 0.5 \\ -1 & \text{otherwise.} \end{cases}$$

Therefore given n training examples $\boldsymbol{x_1}...\boldsymbol{x}_n$ and a fixed classifier $\boldsymbol{\mathcal{H}}$, our task is to find h_{Bayes} if it is contained in $\boldsymbol{\mathcal{H}}$ or find an h_n such that is as close to it as possible in terms of their expected risks, i.e.:

$$h_n = \underset{h\in\boldsymbol{\mathcal{H}}}{\arg\min}\left(R\left(h_n\right) - R\left(h_{\text{Bayes}}\right)\right)$$

If we look closer at the quantity $R(h_n) - R(h_{\text{Bayes}})$, it can be decomposed into two terms [115]:

$$R(h_n) - R(h_{\text{Bayes}}) = \underbrace{\Big(R(h_n) - R(h_{\boldsymbol{\mathcal{H}}})\Big)}_{\text{estimation error}} + \underbrace{\Big(R(h_{\boldsymbol{\mathcal{H}}}) - R(h_{\text{Bayes}})\Big)}_{\text{approximation error}}, \tag{7.1}$$

where $h_{\boldsymbol{\mathcal{H}}}$ is the best function in $\boldsymbol{\mathcal{H}}$:

$$h_{\boldsymbol{\mathcal{H}}} = \arg\min_{h \in \boldsymbol{\mathcal{H}}} R(h).$$

Assuming a perfect model and infinite data we should be able to find a fit such that both estimation and approximation error terms are reduced to zero. This is impossible in practice in most cases, however we should aim to find the right balance between the complexity of the model and the amount of data we can train it on. The estimation error depends largely on our algorithm for finding the best h from some fixed class $\boldsymbol{\mathcal{H}}$ and the available training data (say of size n) as well as various limiting resources such as space and time. The approximation error depends on how well functions in $\boldsymbol{\mathcal{H}}$ can approximate the true solution. This depends on the capacity of a classifier, the given representation (power of features) and the underlying distribution P which is often unknown (and which may change with a change in representation).

Generalisation bounds provide us with some insights into what is necessary for the quantity $R(h_n) - R(h_{\text{Bayes}})$ to be ϵ-small with some high probability for all distributions P:

$$\Pr\Big(R(h_n) - R(h_{\text{Bayes}}) > \epsilon\Big) \to 0 \text{ as } n \to \infty. \tag{7.2}$$

A classifier that consistently produces such h_n on any P is called *universally Bayes-consistent* [115]. This means that with high probability the function h_n that was found by such an algorithm from some training set of size n will be equal or close to a Bayes-classifier as n approaches infinity. It requires minimising both estimation and approximation error terms which appear on the right hand side of the Equation (7.1). Minimising the approximation error (second term of Equation (7.1)) is however a hard problem [115] and statistical learning theory as well as its domain adaptation

extensions mainly concern themselves with minimising the estimation error (first term of Equation (7.1)) and hence look for an algorithm that satisfies the property of *universal consistency* instead of (7.2):

$$\Pr\Big(R(h_n) - R(h_{\boldsymbol{\mathcal{H}}}) > \epsilon\Big) \to 0 \text{ as } n \to \infty. \tag{7.3}$$

The quantities h_{Bayes}, $R(h_{\text{Bayes}})$ or even $R(h)$ for some h cannot generally be evaluated because we do not have access to P. Instead we use the notion of *empirical risk* for the generalisation bound or (in practice) its approximations. The empirical risk is used in many classification algorithms and for binary classification was shown in Equation (4.2) as:

$$R_{emp}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell_{0\text{-}1}(\boldsymbol{x_i}, y_i, h)\ . \tag{7.4}$$

The empirical risk minimisation principle states that to find the optimum h we simply minimise the empirical error (as it is effectively the only (approximately) measurable quantity available at the time of training):

$$h = \arg\min_{h \in \boldsymbol{\mathcal{H}}} R_{emp}(h). \tag{7.5}$$

Vapnik and Chervonenkis [114] proved that the following condition which uses the empirical risk $R_{emp}(h)$ is *necessary and sufficient* for universal consistency (see (7.3)) of the ERM principle:

$$\Pr\Big(\sup_{h \in \boldsymbol{\mathcal{H}}} |R(h) - R_{emp}(h)| > \epsilon\Big) \to 0 \text{ as } n \to \infty, \tag{7.6}$$

The use of supremum here means we are looking for the worst case. A major result of statistical learning theory is that this probability can be bound using the following general form (rewriting (7.6) as a *uniform convergence bound* and using the concept of classifier's capacity) [13]:

$$R(h) \leq R_{emp}(h) + \underbrace{O\left(\sqrt{\frac{\text{capacity}(\boldsymbol{\mathcal{H}})}{n}}\right)}_{\text{complexity term}} + \underbrace{O\left(\sqrt{\frac{\log(\frac{1}{\delta})}{n}}\right)}_{\text{confidence}}, \tag{7.7}$$

which holds with probability at least $1 - \delta$ for all hypotheses h that are found by a learning algorithm, including h_n found by ERM (7.5) or any of its improvements

such as regularised algorithms some of which were presented in Chapter 4. The third term is non-negative and is related to our confidence in the bound - it grows with smaller values of δ because, intuitively, for more confidence in our bound it needs to be looser. The second term (labelled as "complexity term") in the right hand side of (7.7) explains the conditions required from $\boldsymbol{\mathcal{H}}$ and n such that the ERM principle is consistent. It is clear that consistency is achieved when the complexity term converges to 0, which requires a small capacity or large n. Too small a capacity, however, leads to underfitting and thus a high approximation error. For the approximation error to decrease we often want high capacity. On the other hand, with the ratio in the complexity term too large we experience overfitting. This is why in practice we often need to employ regularisation (with a higher regularisation parameter λ for smaller n) which restricts the number of eligible hypotheses and can thus be thought of as a technique for controlling the capacity of $\boldsymbol{\mathcal{H}}$ (and why, for example, KSVM with Gaussian kernel has a finite VC-dimension in practice [105]).

Many bounds exist which differ in constants and the capacity terms. The capacity can take the form of the shattering coefficient, VC-dimension [114], Rademacher complexity [115] or something more specific to the class of functions $\boldsymbol{\mathcal{H}}$, e.g. a function of the margin in SVMs. All of these bounds have their advantages and disadvantages. Two bounds used in theoretical domain adaptation work which we recall in the next section are based on the VC-dimension [7] and the Rademacher complexity [87].

The VC-dimension of $\boldsymbol{\mathcal{H}}$ is the maximum number of examples that can be separated in all possible ways by some $h \in \boldsymbol{\mathcal{H}}$, i.e. h must be able to assign any arbitrary labelling to these examples by separating them with its decision boundary. When the decision boundary is a hyperplane (linear classifiers), the VC-dimension can be shown to be $d + 1$ where d is the (operational) dimensionality of the problem.

Using VC-dimension as capacity and denoting it d the uniform convergence bound

becomes [13]:

$$R(h) \leq R_{emp}(h) + O\left(\sqrt{\frac{d\log(\frac{n}{d}) + \log(\frac{1}{\delta})}{n}}\right) . \tag{7.8}$$

7.1.3 Link to steganalysis

One of the consequences of this bound is that the size of training data is important and to build detectors with minimal risk of misclassification we should explore the possibility of training with a large n. Previously, much focus in steganalysis was directed towards features engineering which effectively minimised the approximation error. The exception to this is when dimensionality was considered and reduced - this additionally minimises the estimation error through reducing the classifier's capacity, which is a non-negative additive term in the uniform convergence bound. The same focus perhaps also explains the often-used KSVM as the classifier of choice in steganalysis. Early in this thesis we motivated our study with the idea that using more training data will lead to better classifiers, which is an alternative to the previously mentioned works. It is easy to see that this idea is a direct consequence of the above bound.

An example of how this bound can be instantiated was shown in Figure 5.10 in Chapter 5. There the AP classifier and the CC-C300 features in conjunction provided a fixed hypothesis class $\boldsymbol{\mathcal{H}}$ and a (largely) fixed capacity. If we assume no cover source mismatch for the moment then we can see that the empirical risk ($R_{emp}(h)$ shown as training error E_{train}) converges to as close as 1.2% away[1] from the expected risk ($R(h)$ estimated by the mean test error shown as E_{test}) at 1.6M training examples. The 1.2% "gap" may be solely due to adaptation error, which we discuss in the next section, or may possibly be improved further with a larger n; nevertheless it is small enough (relative to the magnitude of μ) for us to say that μ, in this case, is a good proxy for $R(h_{\boldsymbol{\mathcal{H}}})$, which means (given necessary and sufficient conditions for universal consistency) empirical risk (and estimation error) has been effectively minimised. The same does not hold for when we train on 16 000

[1]This is the difference between μ and E_{train} as measured by AP on the problem of detecting nsF5 embedded at 0.05 bpnc, but similarly small figures were achieved for other problems as can be seen from Figure 5.10 and the complimentary Table 5.3.

or 160 000 examples - the graph in Figure 5.10 shows the gap between training and test error being considerably larger. With this in mind we can reason about the difference in test error from the point of view of domain adaptation in the next section.

It is known that smaller payloads and more advanced embedding methods create smaller separation between classes, which is given by the joint probability distribution P. In practice, we have seen that this is also reflected in our estimates of the AP's convergence rates on the different problems - it is evident from Figure 5.10 that it converges faster on larger payloads (compare the left two figures to the right two). The uniform convergence bound given in terms of the concept of VC-dimension (Equation (7.8)) is not dependent on P because the VC-dimension does not capture this [115]. Other capacity concepts, such as Rademacher complexity, which by definition depends on n and P [115, 87], will lead to better bounds which reflect how the convergence rate changes with the difficulty of the classification problem, which in our case is governed by the embedding rate or embedding method.

As discussed previously simply looking at the means does not give us the full picture because of the problem of cover source mismatch. Mismatch between the distributions of training (source) and testing (target) calls for extra error terms in the convergence bounds.

7.2 Domain adaptation bounds for classification

Theoretical analysis of domain adaptation can be viewed as an extension of the standard statistical learning theory. It is devoted to studying the additional error terms that occur due to adaptation, the nature of those terms and ways to minimise them. The same assumptions hold, e.g. data from source and target domains are both generated as independent and identically distributed samples. Formally, a labelled source domain sample S will be generated by its distribution P_S with the labels assigned by its labelling function Y_S. Similarly, for the target domain we have a sample T from a domain defined by distribution P_S and labelling Y_S. Without loss of generality we can assume that a collection of source domains can be treated as one source [7].

Several generalisation bounds using 0-1 loss (classification) have been proposed for conservative domain adaptation including the works of Ben-David et al. [8], Blitzer et al. [14], Mansour et al. [87] and more, which we discuss here. They take the following general form (cf. Equation (7.7) from the previous section):

$$R^T(h) \leq R^S(h) + \underbrace{\text{disc}_\ell(P_S, P_T)}_{\substack{\text{distance} \\ \text{between} \\ P_S \text{ and } P_T}} + \underbrace{\lambda}_{\substack{\text{distance} \\ \text{between} \\ Y_S \text{ and } Y_T}} . \tag{7.9}$$

Note that, unlike the standard bound, here *both* left-hand-side and right-hand-side risk terms R are expressed as expected risk. Both $\text{disc}_\ell(P_S, P_T)$ and λ are non-negative measures of relatedness between domains and are relative to (i.e. dependent on) $\boldsymbol{\mathcal{H}}$. Just like the capacity term from the uniform convergence bound presented in Equation (7.7), several definitions for $\text{disc}_\ell(P_S, P_T)$ and λ exist.

Let us have a closer look at the additional error terms. First is the discrepancy distance between source and target distributions for the 0-1 loss, which is given by [87]:

$$\text{disc}_\ell(P_S, P_T) \ = \max_{h^S, h^T \in \boldsymbol{\mathcal{H}}} |R_T(h^S, h^T) - R_S(h^S, h^T)| \ ,$$

for any two hypotheses h^S and h^T in $\boldsymbol{\mathcal{H}}$ trained on source and target domains respectively. Just like $R_T(h)$ measuring the expected risk (loss) with respect to the true labels of the target domain (as assigned by the labelling function Y_T), the quantity $R_T(h^S, h^T)$ measures the expected loss between two labelling assignments on source domain examples - one given by hypothesis h^S and the other by hypothesis h^T. Similarly for $R_S(h^S, h^T)$. The loss function ℓ can take the form of 0-1 loss or one of any more general loss functions. Blitzer [13] shows that $\text{disc}_{\ell_{0\text{-}1}}(P_S, P_T)$ can be estimated from unlabelled samples from two domains by training a maximally discriminating classifier to separate between them. Obviously this estimate depends on the classifier and the size of two samples and is susceptible to its own estimation and approximation errors. Approximation error in this case is exactly the subject of interest because it is a proxy for the divergence between domains that the classifier in question will experience. Cortes et al. [27] point to a problem with the

$\text{disc}_\ell(P_S, P_T)$ when measured using 0-1 loss - there exists a possibility for it being zero even when the source and target distributions are different. This is the topic of current research in domain adaptation.

The second error term λ is the notion of the difference in the labelling functions. Ben-David et al. [7] define it as:

$$\lambda = R^S(h_{\boldsymbol{\mathcal{H}}}^{S\cup T}) + R^T(h_{\boldsymbol{\mathcal{H}}}^{S\cup T})$$

where $h_{\boldsymbol{\mathcal{H}}}^{S\cup T}$ is some optimal hypothesis which is a minimiser for both target and source risks. λ is assumed to be small otherwise (i.e. if such single joint good hypothesis does not exist), conservative adaptation will not succeed.

We will refer to the two terms simultaneously as adaptation error terms.

Using these definitions the following gives uniform convergence bound for conservative domain adaptation similar to the standard bound based on the VC-dimension shown previously in Equation (7.8) [7]:

$$R^T(h) \leq R^S(h) + \frac{1}{2}\text{disc}_{\ell_{0\text{-}1}}(P_S, P_T) + 4\sqrt{\frac{2d\log(n) + \log(\frac{2}{\delta})}{n}} + \lambda. \tag{7.10}$$

Although it is known that this bound does not reduce (as it should) to the standard uniform convergence bound when the source and target data comes from the same domain (Mansour et al. [87] provide an alternative which does, using Rademacher complexity and other definitions for empirical and adaptation errors) we view it as sufficient for the purposes of exposing the basic ideas.

7.2.1 Analysis of experimental results

In this section we provide several observations about the results we reported in Chapter 6, using the theory. We attempt to decompose the observed error of various classifiers which was given in the tables in Chapter 6 into approximation, estimation and adaptation error terms.

Observation 1

The results from experiments 1 and 3 are both attempting to provide a reference point. The aim of the former is to give an estimate of $R(h_n)$ where n is perhaps a typical size of a matched steganalysis training set and the latter attempts to give an upper bound on $R^T(h_{\boldsymbol{\mathcal{H}}})$ where T is an unknown target.

Let us first focus on the μ figures from Experiment 3. In the previous section we argued that in the case of linear classifiers μ is a good proxy for $R^S(h_{\boldsymbol{\mathcal{H}}})$. Making the same observation as before with regards to the closeness of training error to test error and using Equation (7.10), we can say that linear classifiers' μ given in Experiment 3 is not only a good proxy for $R^S(h_{\boldsymbol{\mathcal{H}}})$ but also a good proxy for $R^T(h_{\boldsymbol{\mathcal{H}}})$ where T is an *average* target (assuming all targets are equally likely).

The theory also says that for non-linear algorithms, which have higher capacity than the linear algorithms, the rate of convergence[2] is slower (higher capacity-to-n ratio), at least in the worst case. They may therefore require a larger n for empirical risk to converge to the expected risk and thus also $R^S(h_{\boldsymbol{\mathcal{H}}})$ and $R^T(h_{\boldsymbol{\mathcal{H}}})$ (i.e. more training examples are needed to obtain the minimum estimation error, such that only approximation and adaptation error terms are left). Therefore, similarly strong conclusions cannot be drawn about KSVM because estimation error may not have been minimised.

Observation 2

This observation concerns classifiers' variability in Experiment 3. Having established that for linear classifiers the estimation error is near-zero and that μ (i.e. average $R^T(h)$) is a good proxy for the approximation error, we can now make the following statement: the observed variability in linear classifiers' performance in Experiment 3 is due to the adaptation error terms.

As before, we cannot make a similarly strong statement about the error of

[2]Here the term "rate of convergence" is used to denote how fast the right hand side of Equation (7.7) converges to the left hand side not how fast a specific algorithm such as SGD find the minimum.

those classifiers which are less likely to have converged given the practical computing limits.

Observation 3

The new bound is good for providing an exposition for what is required for domain adaptation to succeed. The smaller the discrepancy between the source and target domains ($\text{disc}_\ell(P_S, P_T)$), as well as their labelling functions (λ), the closer $R^T(h)$ will be to $R^S(h)$. An obvious strategy would be to attempt to minimise $\text{disc}_\ell(P_S, P_T)$ and/or λ. Whilst in this thesis we mainly focused on the other (estimation) error term, in some circumstances using a large diverse source can also be thought of as minimising the discrepancy between source and target distributions. Consider the difference in variability (σ) between Experiments 1, 2 and 3.

The difference in difficulty between sources which is depicted in the variability in Experiment 1 is likely to be amplified by the estimation error. The same error contributes to the huge variability shown in Experiment 2, where it is not only the difficulty of the source that varies but also the target. The full tables where the results are broken down by actor (see the Appendix D.1) provide an intuitive explanation; the variability within each row is a proxy for changes in $\text{disc}_\ell(P_S, P_T)$[3], whilst the variability between the average of rows tells us about the variability in estimation error.

Additionally, compared to Experiment 3, where we know that the only significant source of variability is adaptation error, the variability in Experiment 2 is also amplified due to the fact that we are likely to have higher adaptation error (discrepancy and λ terms) than in Experiment 3. This is due to the fact that in Experiment 2 we have a single source domain, which may or may not match well to the target domain, and in Experiment 3 the training data is comprised of a large collection of source domains, which appear to provide small discrepancy on average. This is why we can think of conservative DA

[3] As mentioned earlier, to practically compute a proper estimate of disc_ℓ we would need to follow the procedure outlined in [13].

additionally as a strategy that minimises the discrepancy. This is in line with our previous findings [82] that more diverse sampling from the source domain yields better results than less diverse sampling (please refer back to Table 5.1 for more details on this result).

This leads directly to the following observation.

Observation 4

In order to measure the adaptation error, we have to minimise the estimation error. This, as we have shown, is easier to do with lower capacity which means either a) smaller features or b) simpler classifiers.

Observation 5

There are several possible explanations as to why some target domains (actors) yielded a qualitatively lower accuracy in Experiment 3. In Section 6.7 of Chapter 6 we argued that there were 14 out of 100 actors in the case of the problem of detecting nsF5 embedded at 0.05 bpnc stego payload amongst covers using linear SVM that required special treatment due to a lowered accuracy. The observed difference in accuracy between them and the average may be due to an inherent discrepancy between sources or simply due to a higher number of outlier[4] images present. For example, in nsF5 at 0.05 bpnc the difference between the accuracy of $\mu = 0.8756$ and $\min = 0.658$ in 500 images is approximately 110 images, which means that the difference is more likely to be due to the former than the latter explanation. Whilst no formal forensic analysis was performed this was supported by the following observation. We found that if for each of these outlier target domains we measure the discrepancy distance ($\mathrm{disc}_{\ell_{0\text{-}1}}(P_S, P_T)$) between it and the training data, we get a perfect or near-perfect separation which indicates a large divergence between the source and the target data. Ben-David [9] argues that small $\mathrm{disc}_{\ell_{0\text{-}1}}(P_S, P_T)$ distance and small λ are necessary and sufficient to guarantee successful domain adaptation. This explains why in these situations conservative domain adaptation

[4]The meaning of outliers is defined in Section 5.1.

was not successful. Furthermore, simple adaptive strategies such as importance weighting, where weights are often based on the same quantities that are used for estimating $\text{disc}_{\ell_{0\text{-}1}}(P_S, P_T)$, will not work (this was confirmed by simple spot checks). An important question then is whether we can always identify the target domains where such a strategy will not succeed. Can we use $\text{disc}_{\ell_{0\text{-}1}}(P_S, P_T)$ or ($\text{disc}_{\ell}(P_S, P_T)$ for any more general loss such as hinge loss) as a predictor for when we should not attempt DA?

7.3 Can we do any better?

The discussion above provides us with some understanding of what happens when the discrepancy between the source and the target is too large - we get a qualitative drop in performance, which is what happened with those (few) actors that created the "tails" that can be seen in Figure 6.1. What about λ? Does there always exist a classifier that performs well when the data is combined? Here we had to make the assumption that this is the case. It has been argued [13] that λ cannot be measured effectively for a DA problem (if it could then there would be no need for adaptation in the first place - there would be enough examples with labels from the target domain to train a good classifier). Our previous findings [81], which we omit from this thesis for the purpose of brevity (main results are reproduced in Figure B.1 in the appendix), have shown that λ appears to be small on average.

Conservative domain adaptation, as per Experiment 3, is one of the simplest possible solutions to the cover source mismatch problem. It may or may not work in some situations. We have shown that in steganalysis it appears to work in significantly more situations than it does not. We briefly discussed that its success largely depends on a) the relatedness between source and target data; and b) the size of labelled data available from the source and target domains. Many adaptive DA learners are based on minimising the discrepancy and/or utilising the labelled target domain data - in other words their practical value rests on the availability of labelled or unlabelled images from the target domain. In this section we give further details on some conditions for which conservative domain adaptation may be suboptimal. We start with going back to the matched case.

7.3.1 Agnostic learning in target domain > domain adaptation

To understand when learning solely from the target domain will consistently provide superior results to using some of the conservative domain adaptation strategies, such as the strategy we used in Experiment 3, Ben-David et al. [7] consider a learner which minimises a convex combination of empirical source and target risks:

$$R^{\alpha}_{emp}(h) = \alpha R^{T}_{emp}(h) + (1-\alpha) R^{S}_{emp}(h)$$

for some weighting factor $\alpha \in [0, 1]$. They show how it is possible to derive the optimal value for α which then leads to an important observation which states that α takes the value of 1 (which indicates that it is optimal to use only target data for training) when the following holds:

$$n_T \geq \frac{d}{A^2} \,, \tag{7.11}$$

where, as before, d is some measure of capacity[5] of our hypothesis class $\boldsymbol{\mathcal{H}}$ and A is an empirical estimate of the divergence ($\mathrm{disc}_{\ell}(P_S, P_T)$) between source and target data and is dependent on $\boldsymbol{\mathcal{H}}$. This means that unless we have as many as n_T labelled examples from the target domain we run a risk of training a suboptimal detector (of capacity d) under the fact of existence of some other labelled (source) data which yields the given A.

It points towards the same tradeoffs that we have discussed earlier between the sizes of the available labelled target and source data sets, the capacity of a given classifier and the divergence between source and target data as perceived by that classifier. Following up on our discussion from Section 7.1.3, it is important to note that the capacity measure d will in practice also depend on the distribution P of the two classes (or in other words the convergence rate of the classifier on a particular problem), so different values of n_T may be required for different problems. For example, based on our estimation shown in Figure 5.10, AP's convergence rates are faster on nsF5 embedded at 0.1 bpnc than at 0.05 bpnc. This means that the minimum number of training examples from the target domain (n_T) required to train a

[5] N.B. As before, capacity here can be thought of as a combination of operational (i.e. true) feature dimensionality and the classifier's complexity.

matched classifier will be smaller for the former than the latter problem.

What is remarkable about this bound is that it provides us with an insight into how much training data is required for matched and mismatched (domain adaptation) cases. In steganalysis it is not clear what is a reasonable assumption on n - the size of a matched training data set (i.e. the size of a target domain training sample), if any, may be available to train a detector for a real-world situation. It is a rather philosophical question for which we have no empirical evidence, except for the previously-mentioned fact that amongst more than 60 000 Facebook users we visited, only 26 of them had more than 3000 images each. Hence we leave it to the reader to decide on the magnitude of n for themselves.

7.3.2 Further works on domain adaptation

We have only touched the surface of the vast literature that exists on domain adaptation, and related topics, that may be applicable to steganalysis (transfer learning, multitask learning, etc). There is no doubt that the ideas of DA are inherent to many steganalysis problems, such as cover source mismatch, and many more general problems which arise due to some other mismatch between training and testing data, such as dealing with different quantisation tables. We have taken the first steps towards understanding the basics from the point of view of some theory, but in addition to the theoretical work discussed above, the empirical DA literature (and its related topics) is rich with propositions of heuristics to deal with different scenarios of domain adaptation, including many adaptive methods. Reference [99] provides a good survey. Many of these propositions, which often fit well with the theory presented here, were tested as solutions to specific domain adaptation applications such as sentiment analysis or visual object recognition amongst others. A better understanding is yet to be developed of whether some of these may be applicable to steganalysis.

7.4 Consequences for steganalysis

This chapter represents the first step towards understanding how domain adaptation ideas can be applied to steganalysis. We framed cover source mismatch as a

problem of domain adaptation and discussed the tradeoffs that exist in it, using statistical learning theory and its domain adaptation extensions. We analysed the error introduced by cover source mismatch and provided a robust methodology for estimating it, which involves minimising classifier's estimation error. That allowed us to demonstrate that the choice of classifier for steganalysis is non-trivial as is the choice of training data or features representation. Specifically, an analysis was given as to why a simpler (smaller capacity) classifier may outperform a more complex (large capacity) classifier, which reiterated our empirical results from the previous chapter. We have argued that steganalysis features need to be designed such that they are maximising the separation between classes and minimising the discrepancy between sources, at the same time. We have shown that there exists a DA-based framework which can be used for approximately measuring the discrepancy. Lastly, we have shown empirically and explained in theory that using only matched data for training may not necessarily be the optimal option for a steganalyst. A known framework from DA literature was introduced to help one reason about this. Many open questions remain, which need to be investigated in order to complete our understanding of which DA solutions may lead to the optimal choice of strategy when tackling cover source mismatch and similar steganalysis problems.

Chapter 8

Conclusions

The security of a steganography system is defined by our ability to detect it. It is of no surprise then that steganography and steganalysis both depend heavily on the accuracy and robustness of our detectors. This is especially true when real-world data is considered, due to its heterogeneity.

Until now it has been asserted that matched data (which means same source and no heterogeneity, i.e. all characteristics are identical and/or near-identical) is strictly necessary to train accurate and robust detectors. Several problems can be identified with this approach. First, in reality, situations may occur when matched training data is unavailable. Second, one must have absolute certainty in the ground truth (i.e. perfect knowledge of the properties of the source), otherwise the cost is high (performance drop due to cover source mismatch is enormous) as the researchers have repeatedly encountered and we confirmed experimentally. Even when the first two issues are resolved, this thesis shows that in light of modern tools - large feature sets coupled with complex non-linear classifiers, this still may not be the best strategy due to elevated estimation and adaptation error (if applicable). Our empirical results motivate this observation, which goes in line with modern theory of domain adaptation.

Motivation for this thesis initially came from the idea that using more training data could improve steganalysis detectors. We investigated this through two hypotheses, first, that linear classifiers are more robust than non-linear classifiers to cover source mismatch (in the view of the presence of normal limiting factors such as space and

time) and second, that linear classifiers are so robust that given sufficiently large mismatched training data they can equal the performance of any classifier trained on small matched data. We have shown that our first hypothesis was supported by our results and theory. Whilst our second hypothesis was not supported by empirical evidence we have shown that a non-trivial situation[1] may arise ($n_T > d/A^2$), which happened in a large number of tests on mismatched actors. In conjunction with several domain relatedness conditions, which are known to be necessary and sufficient, this allows us to reason about when conservative domain adaptation is applicable to steganalysis.

With the help of real-world data we have shown that the penalty that is often associated with cover source mismatch may, in fact, be a combination of two error terms. The first of these is the estimation error, which is due to small training data sets and a high capacity classifiers-features combination. The second is the adaptation error, which is due to mismatch between training and test data and (for a fixed representation) depends on the divergence between underlying distributions that generate the training and test data and the choice of classifier. We argue that a simple strategy of training on an unweighted combination of a large number of different sources allows us to approximately minimise both terms. The success of such a strategy was shown to (in theory) depend on the relatedness between the domains from which training and test data were generated respectively. It was confirmed that for difficult domains, for which our proposed method did not work very well, the relatedness was low. In-depth analysis was provided, raising many open questions. To our knowledge, this is the first work in steganalysis framing it in terms of theoretical domain adaptation.

8.1 Open questions

Can we predict when conservative DA will succeed?

The success of domain adaptation is a function of relatedness between source

[1]By trivial situation, here we mean that the size of matched training data is prohibitively small for training.

and target domains (small disc_ℓ and λ are required) and, in the case of ERM-like training, the size of source domain data available for training. It was tested that disc_ℓ was large for those sources for which our detectors showed lowered performance. This raises the immediate question: is disc_ℓ a good predictor for steganalysis cover source mismatch? If the answer is yes then we can always construct a state-of-the-art classifier for *mismatched* sources which yield small enough disc_ℓ using a simple conservative DA strategy (empirical risk minimisation over a large collection of sources). Obviously, the measure of disc_ℓ for a given target domain is dependent on the classifier and features, as well as what source domain data is available.

Similarly, there might be an impact on steganography. If a steganographer can measure the discrepancy between sources then they might want to choose to design a source (without raising suspicion in any other way - i.e. images must still be plausible) in such a way that the discrepancy is maximised which would in turn yield a larger adaptation error for the potential detector. Such process would be costly, but one needs to evaluate the consequences of this.

Can we do better than conservative DA?

For all other sources (those for which disc_ℓ is large), one might choose to use a different strategy for steganalysis - perhaps one that would allow us to explicitly minimise the discrepancy between training and test data and it is yet to be discovered which domain adaptation method is best for steganalysis (many exist) and if one would always succeed.

Many general-purpose adaptive DA heuristics exist such as importance weighting and pivot features. Some of these may or may not be applicable to steganalysis and it is yet to be fully understood which ones would be. Our preliminary results have shown that importance weighting, for example, might not work. This is a topical area of research in domain adaptation, which will undoubtedly yield better solutions in time.

Other approaches to tackle cover source mismatch, which are not based on

solutions from the theory and practice of domain adaptation exist. Some of them are concerned with predictable bias such as in [72, 62] and this is a topic of ongoing research in steganalysis.

Can non-linear algorithms lead to smaller disc_ℓ distance?

Faster, non-linear classification algorithms exist, which were not evaluated in this study. A natural extension to our thesis would be to see whether it is possible to effectively minimise their estimation error on typical steganalysis problems. What will happen to approximation and adaptation errors in this case? We predict that the adaptation error would increase (based on the intuition about how the discrepancy distance can be approximated - a better algorithm should better discriminate between examples from source and target distributions). If they do make the adaptation error smaller (or it stays the same) then this contradicts our previous claim [82] that non-linear classifiers overfit the source. Further investigation is required.

How can we use the convergence rates?

Theory says that the rate of convergence of ERM (and ERM-like algorithms, such as regularised classifiers) on different problems depends on the capacity of a classifier (its complexity coupled with the complexity of features) and the complexity of the problem (not according to VC-dimension but according to Rademacher complexity for example) with respect to that capacity. This goes in line with our observations.

An important open question is whether steganalysts (or even steganographers) can use the information about convergence rates to their advantage?

8.2 Limitations

Finally we would like to discuss some potential limitations that may be associated with the present work:

Features

Only one set of features was tested. Subsequently to undertaking the experiments presented in this thesis, several versions of similar large features sets

have been proposed which use filtered residuals of DCT coefficients instead of using their values directly. They have been shown to improve detection performance in general. As of yet, however, there is no evidence to suggest that such features (specifically the filtering) reduce the adaptation penalty. It has been shown [80] that earlier feature sets that also used filters were still susceptible to it, for example the SPAM feature set. It is unclear whether the same holds for JRM, PSRM or other similar new-style feature sets. We have no reason to believe they are very different to other feature sets in terms of how much they are affected by the adaptation error. We leave the investigation of this issue for future work.

Classifiers

Other classifiers may have allowed for better results. This is likely to require non-linear decision boundaries and efficient training (perhaps kernel approximation methods). For example, in [10] k-Nearest Neighbour classifier was shown to exhibit desirable properties when applied to a model domain adaptation problem. Further understanding of how classifier complexity affects the discrepancy distance between source and target is required.

Time and space constraints

Different allowances for training time and/or smaller or larger data sets would have yielded different figures for the results tables given in Chapter 6. This may have led to a different ranking of classifiers and/or training regimes (standard or with domain adaptation) in our experiments. However, we have shown a general technique to minimise the estimation error and what it entails in terms of the trade-offs that exist between data size, classifier capacity, training time and space and the presence and severity of data mismatch.

Better priors

The thesis was concerned with the agnostic learning setting, for which regularised ERM-like methods are optimal. This allows us to build a model from given data and if data is limited so is our model through the estimation error. Bayesian methods allow for a better usage of data by encompassing prior knowledge into the model. Training then allows us to correct that model for

any bias based on the data. If such priors can be found for steganalysis this will inevitably lead to faster rates of convergence and thus smaller requirements from the point of view of data availability.

8.3 Further work

Cover source mismatch can be thought of as a particular instance of a more general problem of model mismatch that is inherent to steganalysis. Recall Ker's categorisation of steganalysis scenarios from Chapter 1. The lettered and numbered scenarios can be dealt with via different means. In this thesis we made an often-used simplification about the knowledge about the embedding (category (A) (payload size and embedding method are known)) which meant that the binary problem was fixed. This allowed us to focus on cover source mismatch which we showed has close connections with problems from domain adaptation. Categories (B) (embedding algorithm known, payload size unknown) and (C) (neither is known) represent a different type of model mismatch and might require different tools[2]. Transfer learning or multitask learning may be applicable here. Other instances of mismatch may also arise, for example when the JPEG quantisation tables differ, and are likely to require special treatment. Furthermore a combination of the above may also occur. Many of these problems are yet to be solved, tools from machine learning are likely to be applicable but this area remains largely unexplored.

In the light of the open questions presented at the beginning of this chapter, we can conclude that there is without doubt more work needed for us to understand the full extent of the impact domain adaptation ideas may have, not only on cover source mismatch, but also on many related steganalysis problems. We hope that the first step we have made towards that goal is valuable moving towards future research in this area.

[2]These have previously been approached with either regression or multiclass classification algorithms respectively.

Appendix A

A.1 KSVM grid search on homogeneous and heterogeneous data

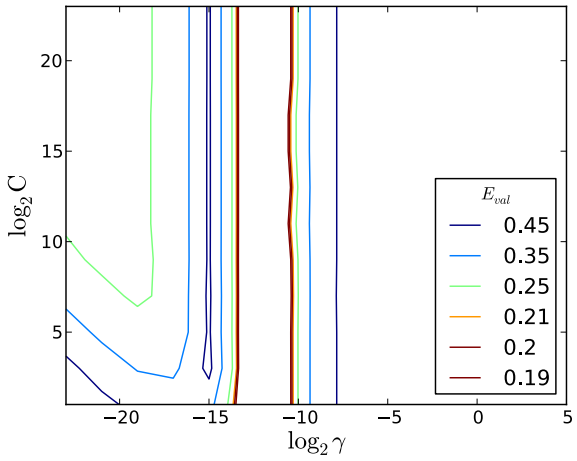


Figure A.1: KSVM grid search on homogeneous data using five-fold cross-validation on 6000 examples.

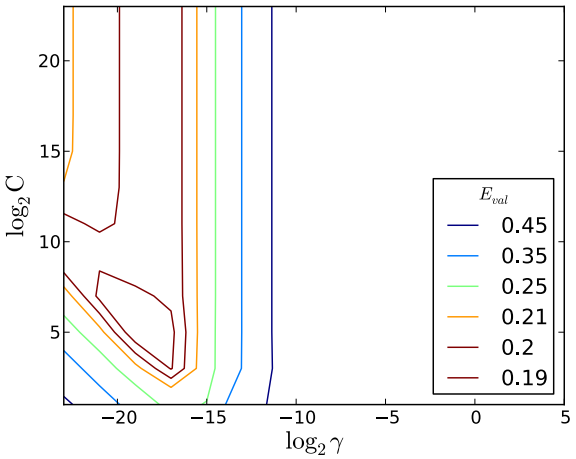


Figure A.2: KSVM grid search on heterogeneous data without cross-validation on 6000 examples.

Table A.1: Benchmarking steganalysis pipeline using online AP and FLD.

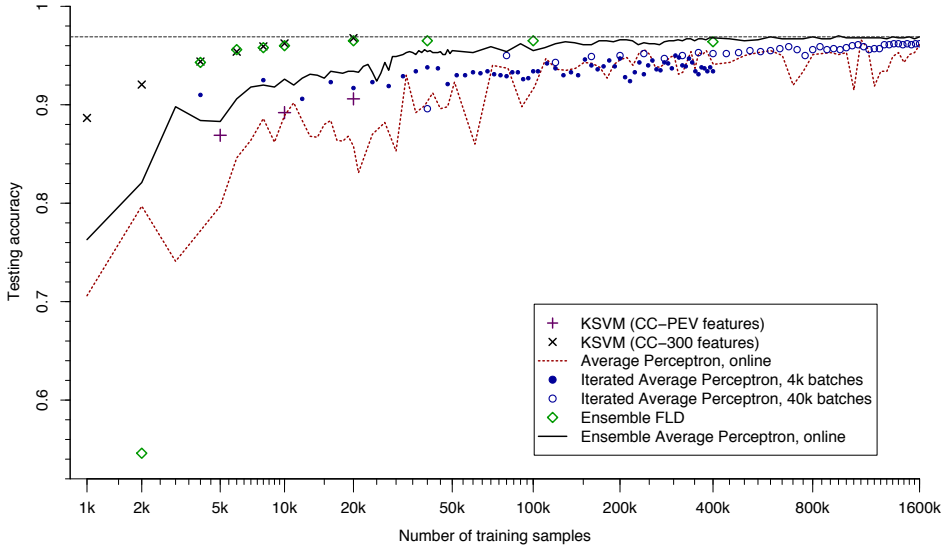
	EMBEDDING FEATURES			
	FB	Flickr	FB	Flickr
nsF5	11	112	136	1540
sPQ	7	174	170	1800

	CLASSIFIER UPDATE								
	$+_{BLAS}$	$+_{numpy}$	$\langle \cdot, \cdot \rangle_{BLAS}$	$\langle \cdot, \cdot \rangle_{numpy}$	$\otimes^{M}_{BLAS}$	$\otimes^{M}_{numpy}$	$cast_{BLAS}$	$TOTAL_{BLAS}$	$TOTAL_{numpy}$
AP	0.008	0.02	0.012	0.015	—	—	0.031	0.101	0.098
oFLD	2×0.01	$*2 \times 0.011$	—	—	17.651	119.739	0.026	17.816	119.879

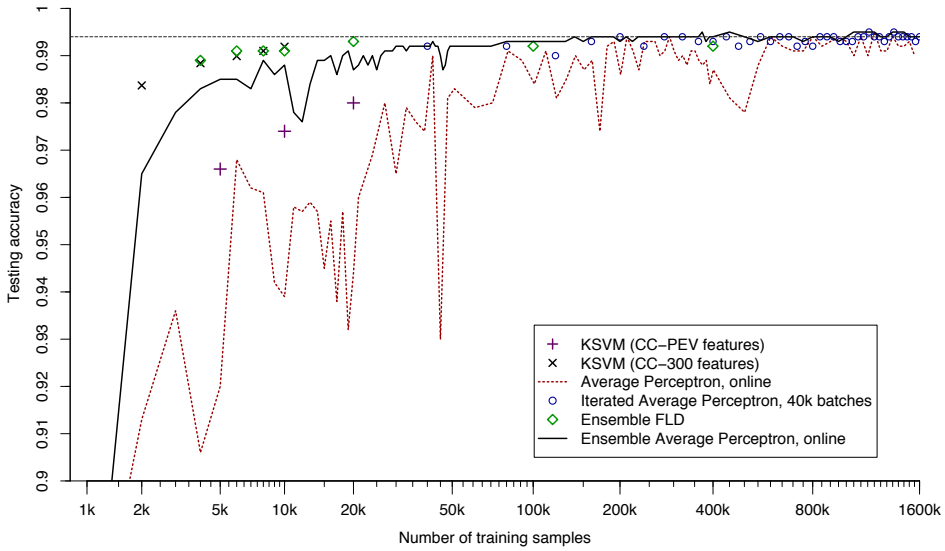
"M" means matrix operation, includes addition. * means scaling also included in operation (i.e. it's x = a + cb where c is some scaling factor). "—" means operation not required. All figures are in milliseconds.

The measurement is based on 1000 random images from 1000 random actors with all I/O operations performed in RAM disk. These figures are per one training step with features vector length of 4000. The fastest speed measured in a production run was just under 0.5 milliseconds per example, which is approximately five times slower on the full feature vector (48600 features). N.B. however that the overall training speed for AP also depends on the difficulty of the training set, because of the mistake-driven update rule. So the more difficult the data is, the more mistakes will be made the more updates will have to happen. So there is up to ±50% error in our time estimates. In contrast, for online FLD the timings are constant, however they are still considerably larger those of AP.

Appendix B



(a)



(b)

Figure B.1: *Testing* accuracy for all classifiers in the heterogeneous-matched experiment as a function of training set size. Above, payload of 0.1 bpnc. Below, payload of 0.2 bpnc. Note the nonlinear scale on the x-axis. Linear classifiers and ensemble AP trained in one-pass. Reproduced from [81]. For more details refer to the full paper.

Appendix C

C.1 Results from the final models when the classifiers are trained online under the normal computational constraints

Table C.1: Comparison of classifiers trained in one-pass on mismatched data with nsF5-embedding.

Experiment	Classifier	Threshold	Training Size	Testing Size	μ	σ	min	max
linear classifiers nsF5 @ 0.05	AP	default	1 600 actors × 1 000 images	100 actors × 1 000 images	0.8530	0.0381	0.648	0.907
		adaptive			0.8549	0.0371	0.646	0.907
	LR	default			0.8539	0.0386	0.652	0.912
		adaptive			0.8495	0.0402	0.648	0.910
	SVM	default			0.8526	0.0427	0.646	0.917
		adaptive			0.8502	0.0434	0.648	0.914
non-linear classifiers nsF5 @ 0.05	OEAP	default	1 600 actors × 1 000 images		0.8482	0.0427	0.648	0.908
		adaptive			0.8497	0.0379	0.648	0.900
linear classifiers nsF5 @ 0.1	AP	default	1 600 actors × 1 000 images	100 actors × 1 000 images	0.9689	0.0284	0.768	0.992
		adaptive			0.9672	0.0282	0.760	0.992
	LR	default			0.9699	0.0252	0.796	0.991
		adaptive			0.9699	0.0252	0.796	0.991
	SVM	default			0.9756	0.0267	0.776	0.995
		adaptive			0.9757	0.0266	0.776	0.995
non-linear classifiers nsF5 @ 0.1	OEAP	default	1 600 actors × 1 000 images		0.9711	0.0278	0.790	0.992
		adaptive			0.9714	0.0278	0.778	0.992

Table C.2: Comparison of simple and complex classifiers trained in one-pass on mismatched data with sPQ-embedding.

Experiment	Classifier	Threshold	Training Size	Testing Size	μ	σ	min	max
linear classifiers sPQ @ 0.2	AP	default	1 600 actors × 1 000 images	100 actors × 1 000 images	0.6805	0.0322	0.566	0.746
		adaptive			0.6949	0.0281	0.588	0.744
	LR	default			0.7010	0.0307	0.580	0.764
		adaptive			0.7016	0.0302	0.586	0.764
	SVM	default			0.6995	0.0320	0.578	0.753
		adaptive			0.7007	0.0329	0.581	0.754
non-linear classifiers sPQ @ 0.2	OEAP	default	1 600 actors × 1 000 images		0.6627	0.0364	0.550	0.741
		adaptive			0.6824	0.0299	0.576	0.746
linear classifiers sPQ @ 0.4	AP	default	1 600 actors × 1 000 images	100 actors × 1 000 images	0.8495	0.0410	0.681	0.913
		adaptive			0.8511	0.0405	0.695	0.909
	LR	default			0.8486	0.0421	0.677	0.905
		adaptive			0.8442	0.0431	0.660	0.907
	SVM	default			0.8527	0.0468	0.667	0.914
		adaptive			0.8480	0.0473	0.652	0.913
non-linear classifiers sPQ @ 0.4	OEAP	default	1 600 actors × 1 000 images		0.8358	0.0464	0.638	0.907
		adaptive			0.8399	0.0444	0.654	0.901

Appendix D

D.1 Combined tables for homogeneous-matched (major diagonal, highlighted in bold) and homogeneous-mismatched (off-diagonal entries).

Table D.1: Accuracy rates for AP tested on nsF5 embedded at 0.05 bpnc

training actor \ testing actor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	**0.877**	0.701	0.803	0.758	0.870	0.876	0.513	0.843	0.843	0.867	0.797	0.903	0.793	0.808	0.803	0.842	0.850	0.803	0.788	0.766	0.786	0.781	0.854	0.864	0.829	0.787
2	0.668	**0.846**	0.783	0.864	0.810	0.863	0.796	0.802	0.812	0.841	0.835	0.856	0.802	0.825	0.794	0.825	0.841	0.774	0.774	0.768	0.857	0.782	0.714	0.861	0.787	0.801
3	0.656	0.753	**0.855**	0.727	0.822	0.842	0.524	0.787	0.808	0.839	0.781	0.862	0.776	0.775	0.807	0.812	0.808	0.819	0.781	0.824	0.758	0.801	0.697	0.859	0.804	0.784
4	0.720	0.746	0.771	**0.875**	0.834	0.858	0.810	0.802	0.827	0.836	0.840	0.870	0.791	0.829	0.805	0.829	0.832	0.782	0.805	0.741	0.862	0.784	0.731	0.857	0.795	0.786
5	0.671	0.729	0.778	0.740	**0.880**	0.866	0.517	0.794	0.811	0.848	0.791	0.898	0.798	0.805	0.792	0.835	0.836	0.795	0.788	0.725	0.784	0.761	0.708	0.850	0.816	0.782
6	0.682	0.690	0.770	0.751	0.827	**0.883**	0.542	0.795	0.839	0.839	0.797	0.887	0.783	0.799	0.776	0.830	0.843	0.775	0.751	0.711	0.789	0.762	0.723	0.846	0.798	0.793
7	0.644	0.724	0.710	0.778	0.746	0.743	**0.867**	0.723	0.697	0.748	0.763	0.773	0.733	0.776	0.765	0.721	0.739	0.736	0.757	0.764	0.764	0.753	0.658	0.805	0.756	0.726
8	0.832	0.748	0.733	0.860	0.834	0.856	0.782	**0.849**	0.799	0.843	0.841	0.879	0.785	0.805	0.800	0.836	0.831	0.782	0.792	0.757	0.846	0.770	0.821	0.862	0.803	0.794
9	0.709	0.685	0.801	0.755	0.877	0.887	0.510	0.800	**0.884**	0.885	0.795	0.920	0.798	0.818	0.789	0.844	0.854	0.793	0.777	0.717	0.793	0.773	0.736	0.841	0.809	0.799
10	0.686	0.672	0.811	0.750	0.863	0.875	0.513	0.804	0.871	**0.901**	0.812	0.910	0.788	0.816	0.831	0.845	0.841	0.828	0.810	0.772	0.790	0.794	0.741	0.887	0.832	0.804
11	0.730	0.773	0.783	0.866	0.842	0.874	0.788	0.812	0.815	0.834	**0.860**	0.875	0.797	0.833	0.822	0.832	0.840	0.791	0.801	0.803	0.854	0.806	0.724	0.872	0.819	0.799
12	0.680	0.739	0.805	0.756	0.855	0.880	0.510	0.794	0.836	0.857	0.801	**0.914**	0.805	0.809	0.796	0.833	0.835	0.811	0.781	0.754	0.790	0.791	0.714	0.875	0.824	0.783
13	0.663	0.794	0.744	0.824	0.823	0.836	0.743	0.797	0.806	0.838	0.806	0.873	**0.821**	0.802	0.788	0.811	0.811	0.782	0.773	0.733	0.818	0.761	0.706	0.849	0.798	0.778
14	0.658	0.783	0.731	0.846	0.831	0.837	0.793	0.782	0.824	0.814	0.824	0.870	0.785	**0.843**	0.791	0.820	0.824	0.782	0.793	0.730	0.846	0.747	0.693	0.829	0.790	0.776
15	0.666	0.718	0.782	0.858	0.823	0.852	0.782	0.804	0.779	0.837	0.830	0.867	0.799	0.819	**0.846**	0.834	0.830	0.832	0.793	0.836	0.849	0.833	0.699	0.873	0.831	0.796
16	0.654	0.783	0.784	0.832	0.839	0.853	0.784	0.802	0.763	0.817	0.825	0.857	0.797	0.811	0.827	**0.845**	0.822	0.820	0.808	0.810	0.848	0.812	0.689	0.861	0.821	0.780
17	0.669	0.775	0.703	0.848	0.808	0.831	0.774	0.778	0.792	0.807	0.823	0.861	0.784	0.805	0.777	0.815	**0.823**	0.782	0.778	0.777	0.824	0.774	0.706	0.860	0.821	0.772
18	0.663	0.762	0.785	0.725	0.844	0.859	0.522	0.784	0.750	0.825	0.786	0.861	0.805	0.789	0.828	0.818	0.816	**0.884**	0.803	0.854	0.795	0.830	0.696	0.863	0.835	0.790
19	0.682	0.759	0.733	0.806	0.817	0.805	0.687	0.778	0.781	0.810	0.807	0.848	0.769	0.806	0.789	0.807	0.815	0.781	**0.810**	0.758	0.820	0.787	0.691	0.844	0.789	0.768
20	0.689	0.705	0.748	0.747	0.833	0.837	0.517	0.787	0.810	0.837	0.799	0.873	0.794	0.794	0.817	0.818	0.820	0.834	0.776	**0.890**	0.780	0.853	0.702	0.883	0.844	0.800
21	0.695	0.733	0.767	0.877	0.817	0.858	0.805	0.798	0.806	0.816	0.840	0.872	0.774	0.832	0.806	0.823	0.826	0.790	0.788	0.776	**0.867**	0.788	0.709	0.867	0.819	0.783
22	0.696	0.655	0.762	0.740	0.832	0.869	0.512	0.803	0.820	0.842	0.801	0.875	0.764	0.797	0.839	0.838	0.843	0.848	0.768	0.867	0.782	**0.871**	0.721	0.869	0.851	0.819
23	0.849	0.671	0.782	0.745	0.839	0.883	0.514	0.829	0.854	0.864	0.800	0.902	0.779	0.808	0.800	0.819	0.851	0.797	0.767	0.768	0.787	0.797	**0.870**	0.872	0.825	0.799
24	0.690	0.713	0.806	0.751	0.848	0.882	0.509	0.816	0.822	0.861	0.816	0.916	0.802	0.819	0.837	0.854	0.849	0.849	0.777	0.811	0.803	0.827	0.713	**0.918**	0.851	0.800
25	0.676	0.715	0.793	0.742	0.830	0.858	0.511	0.790	0.826	0.837	0.797	0.888	0.786	0.794	0.812	0.829	0.840	0.833	0.786	0.851	0.787	0.855	0.711	0.881	**0.874**	0.790
26	0.649	0.728	0.776	0.836	0.807	0.827	0.770	0.802	0.804	0.828	0.825	0.862	0.771	0.816	0.814	0.821	0.824	0.799	0.791	0.823	0.834	0.826	0.708	0.852	0.818	**0.814**

Table D.2: Accuracy rates for EFLD tested on nsF5 embedded at 0.05 bpnc

training actor \ testing actor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	**0.892**	0.704	0.826	0.766	0.910	0.901	0.514	0.873	0.850	0.890	0.832	0.929	0.814	0.822	0.837	0.871	0.873	0.826	0.815	0.808	0.818	0.822	0.873	0.909	0.863	0.832
2	0.678	**0.866**	0.809	0.879	0.853	0.883	0.805	0.820	0.810	0.851	0.861	0.887	0.836	0.836	0.831	0.856	0.858	0.813	0.808	0.829	0.881	0.814	0.726	0.889	0.834	0.811
3	0.663	0.782	**0.879**	0.744	0.844	0.866	0.531	0.807	0.768	0.863	0.799	0.879	0.809	0.782	0.835	0.851	0.836	0.853	0.812	0.881	0.785	0.847	0.709	0.871	0.828	0.809
4	0.729	0.767	0.777	**0.900**	0.867	0.885	0.825	0.842	0.819	0.857	0.861	0.897	0.811	0.854	0.837	0.854	0.854	0.805	0.800	0.797	0.891	0.812	0.740	0.895	0.837	0.812
5	0.679	0.724	0.750	0.756	**0.910**	0.889	0.516	0.821	0.752	0.828	0.810	0.928	0.827	0.833	0.823	0.873	0.854	0.816	0.785	0.758	0.810	0.769	0.715	0.885	0.839	0.792
6	0.707	0.703	0.770	0.771	0.869	**0.913**	0.550	0.819	0.776	0.861	0.831	0.911	0.814	0.802	0.812	0.853	0.847	0.799	0.758	0.729	0.812	0.772	0.732	0.886	0.820	0.812
7	0.592	0.726	0.699	0.749	0.667	0.628	**0.889**	0.648	0.567	0.667	0.685	0.707	0.700	0.710	0.736	0.671	0.675	0.726	0.738	0.766	0.710	0.706	0.593	0.714	0.685	0.657
8	0.848	0.757	0.734	0.877	0.856	0.879	0.790	**0.871**	0.762	0.849	0.861	0.895	0.811	0.833	0.818	0.857	0.850	0.805	0.812	0.776	0.873	0.786	0.835	0.892	0.815	0.805
9	0.717	0.686	0.809	0.769	0.913	0.915	0.511	0.823	**0.922**	0.895	0.823	0.939	0.812	0.832	0.798	0.871	0.887	0.805	0.794	0.708	0.811	0.777	0.760	0.854	0.841	0.823
10	0.705	0.691	0.823	0.767	0.896	0.905	0.516	0.841	0.880	**0.918**	0.845	0.927	0.810	0.840	0.859	0.877	0.869	0.871	0.825	0.842	0.816	0.858	0.760	0.933	0.889	0.846
11	0.746	0.781	0.784	0.883	0.867	0.897	0.797	0.842	0.771	0.859	**0.888**	0.904	0.823	0.835	0.855	0.855	0.858	0.818	0.821	0.839	0.883	0.815	0.740	0.903	0.848	0.818
12	0.688	0.764	0.794	0.763	0.893	0.880	0.514	0.826	0.769	0.853	0.825	**0.931**	0.829	0.819	0.834	0.858	0.852	0.836	0.795	0.793	0.814	0.803	0.711	0.917	0.861	0.800
13	0.684	0.816	0.751	0.836	0.853	0.870	0.723	0.826	0.832	0.857	0.828	0.893	**0.864**	0.822	0.810	0.838	0.839	0.797	0.792	0.767	0.842	0.782	0.722	0.888	0.824	0.803
14	0.674	0.792	0.699	0.879	0.852	0.851	0.804	0.816	0.817	0.833	0.850	0.885	0.801	**0.865**	0.816	0.844	0.859	0.792	0.821	0.760	0.860	0.776	0.704	0.854	0.814	0.805
15	0.693	0.726	0.774	0.877	0.856	0.874	0.790	0.825	0.754	0.852	0.859	0.880	0.811	0.842	**0.877**	0.863	0.853	0.855	0.813	0.868	0.876	0.842	0.714	0.875	0.844	0.832
16	0.653	0.794	0.795	0.848	0.859	0.869	0.760	0.815	0.690	0.821	0.832	0.837	0.806	0.818	0.849	**0.871**	0.842	0.853	0.823	0.822	0.867	0.797	0.688	0.809	0.805	0.793
17	0.673	0.773	0.715	0.865	0.855	0.863	0.779	0.804	0.802	0.828	0.831	0.894	0.799	0.819	0.825	0.848	**0.853**	0.822	0.816	0.830	0.857	0.812	0.714	0.879	0.846	0.797
18	0.656	0.720	0.790	0.738	0.865	0.896	0.533	0.804	0.688	0.821	0.819	0.869	0.815	0.815	0.852	0.857	0.830	**0.912**	0.821	0.872	0.815	0.837	0.706	0.869	0.843	0.818
19	0.674	0.788	0.746	0.839	0.836	0.842	0.728	0.791	0.760	0.836	0.822	0.863	0.793	0.826	0.824	0.830	0.823	0.816	**0.843**	0.823	0.841	0.829	0.696	0.852	0.819	0.799
20	0.672	0.678	0.750	0.750	0.845	0.859	0.526	0.798	0.825	0.875	0.806	0.895	0.795	0.802	0.840	0.847	0.833	0.859	0.805	**0.908**	0.790	0.867	0.710	0.898	0.873	0.828
21	0.720	0.747	0.767	0.883	0.851	0.887	0.804	0.827	0.717	0.835	0.861	0.892	0.816	0.841	0.858	0.856	0.838	0.838	0.799	0.826	**0.892**	0.788	0.725	0.891	0.836	0.818
22	0.686	0.666	0.792	0.764	0.883	0.905	0.516	0.827	0.871	0.895	0.830	0.895	0.781	0.827	0.871	0.865	0.872	0.879	0.798	0.900	0.815	**0.904**	0.734	0.884	0.883	0.850
23	0.866	0.688	0.805	0.764	0.879	0.901	0.516	0.857	0.869	0.889	0.828	0.924	0.809	0.824	0.823	0.847	0.869	0.822	0.801	0.793	0.804	0.829	**0.890**	0.911	0.861	0.832
24	0.704	0.694	0.808	0.768	0.898	0.909	0.514	0.841	0.764	0.885	0.839	0.933	0.828	0.817	0.868	0.877	0.850	0.875	0.792	0.824	0.828	0.832	0.730	**0.942**	0.866	0.824
25	0.675	0.728	0.819	0.762	0.867	0.897	0.522	0.817	0.850	0.859	0.813	0.910	0.814	0.820	0.850	0.858	0.867	0.855	0.821	0.888	0.801	0.871	0.732	0.905	**0.900**	0.828
26	0.656	0.740	0.812	0.869	0.851	0.871	0.783	0.819	0.829	0.847	0.845	0.859	0.796	0.827	0.861	0.856	0.859	0.853	0.820	0.847	0.869	0.873	0.720	0.841	0.850	**0.843**

Table D.3: Accuracy rates for KSVM tested on nsF5 embedded at 0.05 bpnc

training actor \ testing actor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	**0.885**	0.712	0.799	0.743	0.881	0.882	0.516	0.860	0.844	0.889	0.834	0.910	0.794	0.819	0.829	0.856	0.860	0.832	0.814	0.818	0.796	0.815	0.862	0.906	0.857	0.816
2	0.677	**0.858**	0.795	0.858	0.825	0.870	0.795	0.815	0.818	0.838	0.859	0.884	0.828	0.820	0.828	0.837	0.842	0.786	0.788	0.793	0.856	0.799	0.719	0.888	0.816	0.808
3	0.659	0.748	**0.864**	0.714	0.836	0.828	0.524	0.770	0.800	0.842	0.790	0.872	0.792	0.774	0.829	0.819	0.825	0.843	0.786	0.843	0.765	0.830	0.706	0.881	0.839	0.792
4	0.726	0.746	0.782	**0.880**	0.836	0.865	0.802	0.822	0.820	0.833	0.851	0.886	0.779	0.838	0.814	0.828	0.843	0.796	0.809	0.772	0.868	0.786	0.729	0.881	0.818	0.811
5	0.694	0.734	0.800	0.741	**0.896**	0.861	0.513	0.805	0.812	0.864	0.822	0.908	0.812	0.805	0.830	0.855	0.851	0.833	0.806	0.814	0.794	0.802	0.720	0.904	0.852	0.800
6	0.682	0.705	0.797	0.744	0.844	**0.897**	0.509	0.806	0.843	0.856	0.818	0.903	0.792	0.801	0.811	0.839	0.848	0.795	0.773	0.773	0.797	0.803	0.734	0.895	0.830	0.819
7	0.618	0.705	0.729	0.746	0.687	0.619	**0.876**	0.639	0.623	0.678	0.674	0.714	0.703	0.730	0.721	0.666	0.679	0.730	0.737	0.738	0.712	0.716	0.594	0.706	0.723	0.645
8	0.841	0.747	0.768	0.855	0.857	0.863	0.734	**0.861**	0.822	0.844	0.856	0.875	0.801	0.815	0.821	0.848	0.844	0.787	0.807	0.765	0.838	0.780	0.827	0.876	0.820	0.806
9	0.694	0.711	0.814	0.749	0.874	0.899	0.509	0.818	**0.887**	0.881	0.831	0.910	0.806	0.818	0.828	0.857	0.868	0.813	0.788	0.761	0.802	0.803	0.755	0.890	0.837	0.821
10	0.698	0.694	0.804	0.738	0.873	0.863	0.514	0.807	0.851	**0.900**	0.819	0.913	0.787	0.824	0.840	0.846	0.837	0.847	0.811	0.809	0.790	0.828	0.743	0.908	0.856	0.815
11	0.723	0.753	0.800	0.858	0.849	0.869	0.791	0.818	0.814	0.840	**0.872**	0.882	0.792	0.838	0.831	0.841	0.830	0.817	0.804	0.831	0.867	0.800	0.730	0.890	0.839	0.806
12	0.693	0.723	0.801	0.741	0.874	0.877	0.512	0.807	0.815	0.868	0.819	**0.924**	0.810	0.805	0.839	0.838	0.845	0.832	0.805	0.805	0.795	0.818	0.719	0.915	0.862	0.796
13	0.682	0.785	0.815	0.749	0.853	0.868	0.534	0.803	0.826	0.870	0.816	0.891	**0.853**	0.797	0.812	0.842	0.834	0.804	0.796	0.801	0.787	0.812	0.720	0.908	0.837	0.803
14	0.675	0.765	0.774	0.860	0.835	0.836	0.805	0.792	0.815	0.830	0.832	0.874	0.791	**0.849**	0.821	0.829	0.850	0.785	0.806	0.802	0.857	0.781	0.694	0.861	0.804	0.799
15	0.685	0.731	0.801	0.852	0.844	0.861	0.753	0.803	0.789	0.831	0.853	0.875	0.789	0.812	**0.864**	0.833	0.838	0.856	0.807	0.848	0.856	0.827	0.711	0.875	0.851	0.817
16	0.684	0.738	0.801	0.841	0.862	0.859	0.719	0.801	0.771	0.847	0.839	0.868	0.799	0.817	0.835	**0.861**	0.844	0.829	0.817	0.842	0.841	0.820	0.703	0.880	0.831	0.803
17	0.692	0.730	0.771	0.825	0.857	0.824	0.688	0.771	0.790	0.832	0.805	0.870	0.776	0.798	0.814	0.831	**0.830**	0.808	0.799	0.800	0.821	0.789	0.711	0.881	0.818	0.771
18	0.664	0.712	0.774	0.700	0.846	0.822	0.525	0.772	0.730	0.834	0.772	0.842	0.784	0.775	0.823	0.816	0.796	**0.871**	0.819	0.834	0.769	0.818	0.662	0.829	0.832	0.768
19	0.668	0.789	0.762	0.824	0.831	0.804	0.763	0.762	0.754	0.823	0.809	0.857	0.780	0.822	0.813	0.826	0.806	0.817	**0.833**	0.810	0.841	0.790	0.673	0.851	0.803	0.768
20	0.668	0.711	0.730	0.700	0.845	0.748	0.531	0.759	0.745	0.825	0.762	0.852	0.754	0.748	0.822	0.795	0.802	0.815	0.802	**0.898**	0.727	0.826	0.653	0.886	0.834	0.750
21	0.691	0.760	0.783	0.849	0.848	0.836	0.782	0.797	0.756	0.823	0.822	0.853	0.789	0.815	0.824	0.817	0.810	0.832	0.817	0.816	**0.852**	0.805	0.697	0.855	0.828	0.787
22	0.681	0.706	0.757	0.719	0.859	0.842	0.519	0.779	0.805	0.875	0.797	0.859	0.773	0.780	0.847	0.820	0.820	0.877	0.798	0.857	0.775	**0.875**	0.696	0.844	0.855	0.817
23	0.859	0.704	0.801	0.738	0.873	0.881	0.517	0.852	0.851	0.888	0.824	0.907	0.779	0.803	0.837	0.838	0.854	0.833	0.805	0.827	0.789	0.835	**0.873**	0.911	0.855	0.811
24	0.692	0.724	0.810	0.741	0.851	0.883	0.511	0.812	0.846	0.870	0.827	0.897	0.793	0.807	0.838	0.843	0.851	0.846	0.791	0.816	0.802	0.836	0.726	**0.934**	0.871	0.822
25	0.662	0.757	0.788	0.722	0.864	0.832	0.520	0.779	0.802	0.841	0.785	0.895	0.794	0.802	0.835	0.820	0.817	0.852	0.819	0.859	0.780	0.841	0.692	0.884	**0.879**	0.777
26	0.648	0.725	0.785	0.777	0.823	0.829	0.741	0.778	0.778	0.819	0.828	0.821	0.770	0.764	0.831	0.807	0.807	0.839	0.790	0.842	0.808	0.828	0.698	0.828	0.815	**0.818**

Table D.4: Accuracy rates for AP tested on sPQ embedded at 0.4 bpuc

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	**0.809**	0.701	0.680	0.727	0.792	0.789	0.607	0.765	0.759	0.789	0.740	0.827	0.749	0.751	0.771	0.790	0.797	0.783	0.762	0.755	0.745	0.773	0.772	0.810	0.763	0.748
2	0.721	**0.784**	0.745	0.764	0.781	0.796	0.729	0.774	0.770	0.801	0.768	0.818	0.769	0.764	0.777	0.778	0.786	0.768	0.757	0.789	0.790	0.773	0.714	0.825	0.740	0.768
3	0.666	0.720	**0.830**	0.697	0.790	0.794	0.540	0.752	0.772	0.801	0.748	0.815	0.741	0.727	0.788	0.774	0.784	0.818	0.767	0.820	0.740	0.805	0.683	0.836	0.779	0.765
4	0.704	0.734	0.669	**0.818**	0.771	0.810	0.771	0.760	0.740	0.743	0.777	0.808	0.734	0.784	0.776	0.767	0.789	0.770	0.732	0.772	0.798	0.747	0.699	0.790	0.711	0.743
5	0.664	0.676	0.658	0.723	**0.834**	0.803	0.554	0.752	0.756	0.795	0.740	0.831	0.735	0.780	0.746	0.805	0.800	0.779	0.767	0.692	0.737	0.734	0.705	0.784	0.709	0.743
6	0.657	0.695	0.661	0.728	0.818	**0.833**	0.557	0.754	0.776	0.780	0.766	0.846	0.750	0.784	0.774	0.801	0.812	0.784	0.760	0.732	0.762	0.767	0.717	0.819	0.732	0.767
7	0.679	0.687	0.655	0.796	0.757	0.734	**0.823**	0.724	0.719	0.736	0.743	0.781	0.715	0.782	0.758	0.753	0.764	0.772	0.761	0.720	0.787	0.723	0.684	0.744	0.703	0.723
8	0.766	0.738	0.658	0.802	0.793	0.788	0.760	**0.796**	0.771	0.776	0.780	0.807	0.752	0.779	0.777	0.790	0.795	0.764	0.781	0.733	0.798	0.743	0.771	0.805	0.739	0.760
9	0.677	0.694	0.721	0.724	0.816	0.829	0.548	0.774	**0.802**	0.824	0.769	0.861	0.765	0.764	0.802	0.810	0.820	0.821	0.776	0.822	0.752	0.820	0.709	0.848	0.779	0.778
10	0.664	0.679	0.720	0.721	0.789	0.806	0.538	0.757	0.786	**0.832**	0.746	0.848	0.756	0.764	0.786	0.801	0.795	0.815	0.775	0.764	0.753	0.798	0.692	0.833	0.776	0.751
11	0.720	0.741	0.678	0.804	0.778	0.819	0.762	0.767	0.767	0.775	**0.804**	0.827	0.745	0.791	0.788	0.794	0.803	0.800	0.755	0.807	0.812	0.789	0.712	0.840	0.745	0.791
12	0.693	0.688	0.689	0.736	0.809	0.820	0.536	0.765	0.788	0.777	0.754	**0.877**	0.755	0.785	0.783	0.809	0.818	0.809	0.760	0.745	0.751	0.789	0.706	0.843	0.767	0.760
13	0.737	0.740	0.709	0.760	0.793	0.799	0.677	0.781	0.774	0.809	0.758	0.835	**0.799**	0.783	0.787	0.795	0.807	0.775	0.771	0.750	0.766	0.764	0.736	0.829	0.758	0.768
14	0.702	0.690	0.625	0.787	0.780	0.752	0.745	0.747	0.722	0.759	0.743	0.810	0.723	**0.806**	0.752	0.781	0.773	0.750	0.769	0.691	0.792	0.721	0.706	0.761	0.716	0.725
15	0.745	0.710	0.691	0.797	0.782	0.791	0.766	0.781	0.778	0.776	0.774	0.825	0.745	0.791	**0.828**	0.784	0.799	0.802	0.767	0.794	0.807	0.782	0.731	0.830	0.773	0.777
16	0.685	0.710	0.651	0.788	0.782	0.790	0.737	0.760	0.761	0.787	0.771	0.840	0.745	0.779	0.782	**0.806**	0.789	0.767	0.760	0.749	0.781	0.762	0.700	0.813	0.758	0.770
17	0.732	0.711	0.659	0.792	0.801	0.794	0.743	0.769	0.770	0.800	0.779	0.827	0.749	0.782	0.781	0.795	**0.815**	0.780	0.772	0.757	0.781	0.765	0.729	0.810	0.752	0.759
18	0.659	0.673	0.686	0.717	0.805	0.818	0.545	0.755	0.783	0.793	0.767	0.834	0.748	0.754	0.803	0.803	0.810	**0.840**	0.763	0.836	0.752	0.823	0.691	0.853	0.790	0.790
19	0.759	0.678	0.649	0.790	0.782	0.763	0.753	0.766	0.762	0.765	0.762	0.807	0.723	0.786	0.758	0.774	0.784	0.770	**0.783**	0.747	0.793	0.753	0.739	0.792	0.739	0.738
20	0.659	0.690	0.705	0.727	0.800	0.799	0.572	0.755	0.778	0.780	0.762	0.826	0.739	0.751	0.790	0.789	0.802	0.808	0.748	**0.843**	0.754	0.824	0.696	0.848	0.782	0.791
21	0.701	0.718	0.655	0.814	0.798	0.807	0.781	0.774	0.757	0.763	0.798	0.835	0.755	0.794	0.794	0.792	0.802	0.779	0.758	0.771	**0.824**	0.763	0.712	0.817	0.732	0.748
22	0.694	0.696	0.732	0.739	0.803	0.831	0.578	0.779	0.804	0.800	0.772	0.843	0.757	0.780	0.805	0.811	0.804	0.818	0.762	0.844	0.759	**0.833**	0.716	0.865	0.788	0.796
23	0.764	0.712	0.676	0.724	0.782	0.789	0.602	0.771	0.775	0.762	0.750	0.845	0.742	0.770	0.781	0.794	0.786	0.779	0.741	0.766	0.743	0.771	**0.795**	0.819	0.740	0.762
24	0.668	0.689	0.705	0.734	0.791	0.835	0.535	0.774	0.802	0.800	0.766	0.851	0.752	0.773	0.803	0.817	0.817	0.823	0.749	0.823	0.760	0.823	0.714	**0.884**	0.791	0.791
25	0.698	0.663	0.654	0.735	0.777	0.778	0.638	0.755	0.775	0.777	0.761	0.835	0.733	0.774	0.789	0.781	0.790	0.791	0.759	0.810	0.747	0.810	0.710	0.838	**0.816**	0.772
26	0.708	0.733	0.693	0.790	0.780	0.812	0.738	0.762	0.794	0.787	0.786	0.819	0.751	0.779	0.806	0.795	0.789	0.796	0.753	0.819	0.798	0.802	0.699	0.841	0.762	**0.785**

Table D.5: Accuracy rates for EFLD tested on sPQ embedded at 0.4 bpuc

training actor \ testing actor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	**0.820**	0.726	0.682	0.740	0.806	0.798	0.607	0.793	0.796	0.827	0.772	0.850	0.767	0.758	0.801	0.817	0.810	0.830	0.783	0.831	0.779	0.820	0.787	0.873	0.794	0.758
2	0.722	**0.803**	0.759	0.804	0.798	0.827	0.758	0.797	0.806	0.833	0.805	0.838	0.799	0.797	0.821	0.811	0.815	0.813	0.785	0.829	0.818	0.820	0.724	0.869	0.778	0.800
3	0.653	0.744	**0.859**	0.702	0.793	0.802	0.547	0.752	0.783	0.834	0.763	0.815	0.780	0.740	0.811	0.788	0.794	0.827	0.786	0.844	0.760	0.819	0.662	0.836	0.808	0.781
4	0.704	0.758	0.686	**0.853**	0.822	0.828	0.794	0.782	0.770	0.788	0.815	0.837	0.774	0.819	0.806	0.802	0.820	0.803	0.772	0.822	0.844	0.796	0.708	0.852	0.757	0.790
5	0.658	0.692	0.653	0.744	**0.868**	0.850	0.556	0.770	0.771	0.823	0.778	0.843	0.761	0.794	0.778	0.829	0.830	0.797	0.780	0.726	0.770	0.766	0.717	0.804	0.731	0.776
6	0.671	0.718	0.651	0.752	0.843	**0.863**	0.572	0.786	0.780	0.820	0.799	0.867	0.782	0.795	0.799	0.844	0.837	0.813	0.787	0.722	0.794	0.776	0.727	0.813	0.747	0.791
7	0.662	0.736	0.656	0.817	0.768	0.774	**0.859**	0.754	0.752	0.752	0.782	0.791	0.739	0.795	0.793	0.773	0.780	0.779	0.785	0.802	0.820	0.770	0.676	0.793	0.741	0.740
8	0.794	0.758	0.661	0.834	0.812	0.822	0.773	**0.833**	0.795	0.803	0.802	0.846	0.773	0.808	0.809	0.816	0.818	0.800	0.797	0.769	0.824	0.774	0.802	0.843	0.766	0.786
9	0.681	0.710	0.702	0.742	0.838	0.846	0.547	0.790	**0.842**	0.858	0.783	0.873	0.793	0.782	0.834	0.828	0.837	0.857	0.816	0.863	0.783	0.869	0.707	0.910	0.832	0.788
10	0.681	0.698	0.698	0.744	0.839	0.836	0.549	0.803	0.825	**0.869**	0.791	0.881	0.786	0.798	0.821	0.839	0.832	0.851	0.803	0.835	0.771	0.837	0.720	0.888	0.810	0.803
11	0.742	0.752	0.676	0.829	0.826	0.847	0.785	0.797	0.801	0.812	**0.840**	0.856	0.784	0.808	0.827	0.829	0.828	0.840	0.805	0.859	0.837	0.826	0.718	0.881	0.790	0.786
12	0.692	0.692	0.683	0.748	0.843	0.837	0.541	0.784	0.806	0.840	0.797	**0.897**	0.774	0.798	0.820	0.840	0.838	0.853	0.791	0.845	0.785	0.843	0.720	0.892	0.826	0.779
13	0.755	0.779	0.707	0.783	0.818	0.841	0.695	0.793	0.804	0.856	0.799	0.875	**0.836**	0.800	0.803	0.820	0.829	0.810	0.795	0.799	0.787	0.812	0.738	0.869	0.788	0.781
14	0.696	0.696	0.635	0.809	0.803	0.805	0.767	0.772	0.748	0.786	0.779	0.839	0.746	**0.827**	0.793	0.797	0.803	0.780	0.792	0.734	0.814	0.760	0.699	0.802	0.741	0.758
15	0.758	0.719	0.675	0.830	0.808	0.841	0.786	0.807	0.810	0.819	0.812	0.852	0.773	0.824	**0.848**	0.815	0.831	0.830	0.806	0.841	0.839	0.829	0.729	0.867	0.804	0.809
16	0.685	0.715	0.653	0.819	0.820	0.833	0.769	0.788	0.785	0.806	0.809	0.863	0.766	0.793	0.818	**0.837**	0.824	0.807	0.804	0.801	0.810	0.792	0.702	0.853	0.788	0.787
17	0.742	0.728	0.657	0.818	0.825	0.829	0.764	0.797	0.796	0.824	0.820	0.861	0.773	0.812	0.823	0.822	**0.845**	0.821	0.804	0.807	0.818	0.800	0.740	0.861	0.788	0.803
18	0.667	0.667	0.681	0.730	0.840	0.833	0.542	0.781	0.817	0.835	0.787	0.858	0.772	0.768	0.836	0.831	0.823	**0.868**	0.800	0.861	0.786	0.863	0.704	0.884	0.811	0.792
19	0.769	0.718	0.653	0.823	0.806	0.791	0.778	0.795	0.774	0.806	0.801	0.846	0.756	0.817	0.804	0.816	0.809	0.817	**0.824**	0.824	0.824	0.802	0.733	0.842	0.788	0.774
20	0.658	0.696	0.689	0.715	0.809	0.779	0.573	0.751	0.805	0.808	0.773	0.842	0.754	0.743	0.807	0.799	0.804	0.839	0.787	**0.877**	0.754	0.846	0.676	0.880	0.813	0.772
21	0.696	0.730	0.659	0.846	0.830	0.837	0.800	0.785	0.775	0.796	0.824	0.854	0.778	0.806	0.828	0.816	0.824	0.820	0.792	0.818	**0.863**	0.799	0.726	0.847	0.775	0.791
22	0.692	0.695	0.715	0.740	0.833	0.838	0.574	0.790	0.834	0.833	0.793	0.859	0.762	0.786	0.831	0.826	0.836	0.861	0.802	0.880	0.786	**0.869**	0.717	0.896	0.817	0.810
23	0.794	0.725	0.673	0.773	0.820	0.828	0.618	0.815	0.801	0.814	0.796	0.874	0.775	0.792	0.811	0.822	0.829	0.821	0.790	0.789	0.770	0.806	**0.827**	0.861	0.771	0.783
24	0.670	0.689	0.696	0.744	0.838	0.839	0.539	0.782	0.822	0.841	0.789	0.865	0.780	0.772	0.843	0.847	0.833	0.863	0.794	0.861	0.787	0.856	0.698	**0.914**	0.821	0.795
25	0.678	0.671	0.680	0.742	0.809	0.804	0.601	0.769	0.821	0.830	0.779	0.866	0.758	0.788	0.828	0.807	0.818	0.836	0.782	0.854	0.762	0.843	0.717	0.880	**0.844**	0.786
26	0.699	0.730	0.674	0.820	0.807	0.841	0.775	0.789	0.819	0.835	0.827	0.847	0.777	0.802	0.854	0.826	0.829	0.838	0.800	0.848	0.843	0.830	0.726	0.871	0.802	**0.829**

Table D.6: Accuracy rates for KSVM tested on sPQ embedded at 0.4 bpuc

training actor \ testing actor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	row mean
1	**0.815**	0.721	0.693	0.727	0.792	0.788	0.615	0.782	0.769	0.810	0.750	0.828	0.759	0.755	0.779	0.795	0.799	0.792	0.770	0.814	0.753	0.804	0.773	0.840	0.784	0.755	**0.772**
2	0.716	**0.781**	0.745	0.774	0.788	0.806	0.735	0.772	0.773	0.802	0.772	0.823	0.772	0.770	0.799	0.792	0.795	0.780	0.768	0.816	0.788	0.796	0.703	0.833	0.765	0.768	**0.778**
3	0.658	0.731	**0.844**	0.691	0.778	0.783	0.545	0.740	0.767	0.812	0.740	0.795	0.755	0.724	0.777	0.761	0.767	0.823	0.777	0.818	0.738	0.802	0.657	0.809	0.780	0.763	**0.755**
4	0.712	0.750	0.678	**0.826**	0.790	0.811	0.770	0.772	0.767	0.760	0.797	0.832	0.751	0.798	0.788	0.795	0.802	0.787	0.772	0.792	0.813	0.782	0.708	0.832	0.734	0.750	**0.776**
5	0.667	0.717	0.682	0.729	**0.840**	0.817	0.553	0.772	0.777	0.804	0.764	0.845	0.753	0.779	0.768	0.816	0.802	0.793	0.771	0.786	0.760	0.784	0.716	0.843	0.758	0.761	**0.764**
6	0.668	0.720	0.678	0.735	0.821	**0.843**	0.568	0.776	0.801	0.800	0.787	0.855	0.768	0.787	0.791	0.829	0.820	0.805	0.778	0.777	0.765	0.806	0.727	0.848	0.772	0.780	**0.773**
7	0.674	0.727	0.678	0.797	0.753	0.728	**0.837**	0.722	0.738	0.762	0.754	0.789	0.731	0.782	0.775	0.754	0.773	0.783	0.771	0.791	0.808	0.787	0.675	0.793	0.754	0.732	**0.757**
8	0.770	0.757	0.662	0.816	0.790	0.802	0.761	**0.811**	0.777	0.785	0.780	0.813	0.759	0.786	0.780	0.798	0.791	0.786	0.772	0.762	0.799	0.754	0.774	0.823	0.753	0.750	**0.777**
9	0.670	0.720	0.728	0.727	0.816	0.822	0.558	0.766	**0.807**	0.822	0.774	0.858	0.773	0.763	0.807	0.802	0.817	0.827	0.781	0.845	0.760	0.831	0.700	0.865	0.806	0.771	**0.777**
10	0.668	0.710	0.722	0.720	0.813	0.807	0.549	0.766	0.793	**0.845**	0.768	0.856	0.780	0.775	0.808	0.804	0.797	0.826	0.777	0.823	0.758	0.821	0.698	0.868	0.802	0.781	**0.775**
11	0.734	0.757	0.688	0.812	0.794	0.820	0.770	0.783	0.785	0.791	**0.817**	0.838	0.768	0.790	0.803	0.802	0.816	0.822	0.782	0.828	0.807	0.802	0.718	0.855	0.768	0.782	**0.790**
12	0.684	0.717	0.701	0.742	0.819	0.813	0.539	0.772	0.810	0.824	0.782	**0.880**	0.775	0.790	0.810	0.812	0.818	0.827	0.777	0.817	0.760	0.821	0.713	0.877	0.794	0.770	**0.779**
13	0.730	0.750	0.718	0.764	0.797	0.807	0.671	0.782	0.786	0.813	0.772	0.848	**0.807**	0.789	0.797	0.801	0.816	0.789	0.776	0.793	0.768	0.794	0.718	0.845	0.778	0.780	**0.780**
14	0.708	0.706	0.637	0.800	0.785	0.770	0.770	0.760	0.757	0.772	0.757	0.823	0.733	**0.808**	0.768	0.770	0.782	0.780	0.775	0.745	0.812	0.757	0.703	0.790	0.744	0.740	**0.760**
15	0.753	0.736	0.693	0.812	0.799	0.807	0.772	0.794	0.795	0.780	0.797	0.830	0.762	0.786	**0.836**	0.797	0.805	0.813	0.772	0.818	0.815	0.802	0.739	0.843	0.790	0.774	**0.789**
16	0.703	0.740	0.681	0.797	0.803	0.799	0.733	0.772	0.777	0.796	0.784	0.846	0.750	0.780	0.798	**0.806**	0.798	0.797	0.767	0.787	0.786	0.784	0.704	0.836	0.781	0.770	**0.776**
17	0.728	0.720	0.670	0.801	0.807	0.809	0.748	0.777	0.780	0.802	0.782	0.828	0.764	0.787	0.800	0.797	**0.826**	0.789	0.784	0.788	0.782	0.783	0.733	0.843	0.773	0.780	**0.780**
18	0.669	0.688	0.701	0.714	0.817	0.817	0.541	0.768	0.799	0.805	0.780	0.843	0.755	0.768	0.816	0.812	0.811	**0.853**	0.774	0.847	0.762	0.832	0.680	0.861	0.809	0.789	**0.774**
19	0.757	0.725	0.658	0.794	0.785	0.765	0.766	0.768	0.763	0.777	0.772	0.810	0.738	0.793	0.780	0.780	0.786	0.777	**0.798**	0.787	0.811	0.787	0.720	0.804	0.758	0.763	**0.770**
20	0.656	0.711	0.723	0.703	0.810	0.763	0.561	0.745	0.784	0.799	0.755	0.807	0.750	0.720	0.791	0.780	0.795	0.815	0.778	**0.867**	0.740	0.832	0.672	0.868	0.794	0.762	**0.761**
21	0.708	0.748	0.678	0.828	0.815	0.823	0.792	0.775	0.771	0.772	0.810	0.844	0.778	0.807	0.810	0.798	0.817	0.812	0.780	0.815	**0.831**	0.792	0.717	0.849	0.770	0.765	**0.789**
22	0.682	0.710	0.728	0.734	0.807	0.816	0.574	0.772	0.806	0.815	0.768	0.837	0.757	0.772	0.810	0.816	0.817	0.816	0.783	0.855	0.755	**0.838**	0.708	0.862	0.792	0.795	**0.778**
23	0.782	0.731	0.685	0.740	0.803	0.800	0.594	0.780	0.782	0.782	0.760	0.850	0.762	0.774	0.785	0.805	0.810	0.799	0.757	0.788	0.757	0.792	**0.807**	0.843	0.760	0.761	**0.773**
24	0.664	0.724	0.713	0.732	0.818	0.818	0.541	0.772	0.816	0.818	0.772	0.837	0.773	0.756	0.819	0.818	0.811	0.833	0.776	0.846	0.761	0.845	0.703	**0.899**	0.802	0.788	**0.779**
25	0.679	0.687	0.672	0.730	0.795	0.775	0.617	0.752	0.785	0.810	0.757	0.834	0.745	0.777	0.800	0.773	0.799	0.794	0.767	0.819	0.750	0.820	0.701	0.836	**0.821**	0.771	**0.764**
26	0.721	0.742	0.693	0.797	0.784	0.802	0.750	0.780	0.792	0.799	0.788	0.813	0.764	0.782	0.810	0.795	0.798	0.796	0.778	0.822	0.807	0.806	0.706	0.838	0.783	**0.789**	**0.782**
column mean	**0.707**	**0.728**	**0.698**	**0.763**	**0.801**	**0.800**	**0.663**	**0.771**	**0.783**	**0.798**	**0.775**	**0.833**	**0.761**	**0.777**	**0.796**	**0.797**	**0.803**	**0.804**	**0.775**	**0.809**	**0.779**	**0.802**	**0.714**	**0.842**	**0.778**	**0.769**	

Bibliography

[1] Yahoo! Advertising. Flickr. `http://advertising.yahoo.com/article/flickr.html`[Last accessed December 2013].

[2] A. Agarwal, O. Chapelle, M. Dudík, and J. Langford. A reliable effective terascale linear learning system. *CoRR*, abs/1110.4198, 2011.

[3] I. Avcibas, M. Nasir, and B. Sankur. Steganalysis based on image quality metrics. In *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, pages 517–522, 2001.

[4] M. Barni, G. Cancelli, and A. Esposito. Forensics aided steganalysis of heterogeneous images. In *Acoustics Speech and Signal Processing*, Proc. IEEE ICASSP '10, pages 1690 –1693, 2010.

[5] P. Bas, T. Filler, and T. Pevný. Break Our Steganographic System: the ins and outs of organizing BOSS. In *Proc. 13th Information Hiding Workshop*, volume 6958 of *Springer LNCS*, pages 59–70, 2011.

[6] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *Trans. Neur. Netw.*, 5(4):537–550, July 1994.

[7] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, May 2010.

[8] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pages 137–144, 2006.

[9] S. Ben-David, T. Lu, T. Luu, and D. Pál. Impossibility theorems for domain adaptation. In *Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *JMLR Proceedings*, pages 129–136. JMLR.org, 2010.

[10] S. Ben-David and R. Urner. Domain adaptation – can quantity compensate for quality? *Annals of Mathematics and Artificial Intelligence*, pages 1–18, 2013.

[11] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[12] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28(2):135–151, June 2002.

[13] J. Blitzer. *Domain Adaptation of Natural Language Processing Systems*. PhD thesis, University of Pennsylvania.

[14] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems 21*, Cambridge, MA, 2008. MIT Press.

[15] R. Böhme. *Improved Statistical Steganalysis using Models of Heterogeneous Cover Signals*. PhD thesis, Technische Universität Dresden.

[16] L. Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. revised, oct 2012.

[17] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier and G. Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186. Physica-Verlag HD, 2010.

[18] L. Bottou. Stochastic gradient descent tricks. In G. Montavon, G. Orr, and K. Mller, editors, *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 421–436. Springer Berlin Heidelberg, 2012.

[19] O. Campbell-Moore. An error-resistant steganography algorithm for communicating secretly on Facebook. Undergraduate Thesis, University of Oxford, 2013.

[20] G. Cancelli, G. Doërr, I. Cox, and M. Barni. A comparative study of ±1 steganalyzers. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, pages 791–796, 2008.

[21] C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[22] Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

[23] M. Chaumont and S. Kouider. Steganalysis by ensemble classifiers with boosting by regression, and post-selection of features. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 1133–1136, 2012.

[24] C. Chen and Y.Q. Shi. JPEG image steganalysis utilizing both intrablock and interblock correlations. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 3029–3032, 2008.

[25] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011*, volume 15 of *JMLR Proceedings*, pages 215–223. JMLR.org, 2011.

[26] C. Cortes, Y. Mansour, and M. Mohri. Learning bounds for importance weighting. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 442–450. 2010.

[27] C. Cortes and M. Mohri. Domain adaptation in regression. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory*, ALT'11, pages 308–323, Berlin, Heidelberg, 2011. Springer-Verlag.

[28] Facebook. Big data whiteboard. `http://www.scribd.com/doc/103621762/big-data-whiteboard-082212`. [Last accessed September 2013].

[29] T. Filler, J. Fridrich, and M. Goljan. Using sensor pattern noise for camera model identification. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 1296–1299, 2008.

[30] T. Filler, J. Judas, and J. J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3-2):920–935, 2011.

[31] Y. Freund and R. E. Schapire. Large margin classification using the Perceptron algorithm. *Mach. Learn.*, 37:277–296, December 1999.

[32] J. Fridrich, M. Goljan, P. Lisonek, and D. Soukal. Writing on wet paper. *Signal Processing, IEEE Transactions on*, 53(10):3923–3935, 2005.

[33] J. Fridrich, M. Goljan, and D. Soukal. Perturbed quantization steganography. *Multimedia Systems*, 11(2):98–107, 2005.

[34] J. J. Fridrich. Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In *Proceedings of the 6th International Conference on Information Hiding*, IH'04, pages 67–81, Berlin, Heidelberg, 2004. Springer-Verlag.

[35] J. J. Fridrich. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, 2009.

[36] J. J. Fridrich. Effect of cover quantization on steganographic fisher information. *Information Forensics and Security, IEEE Transactions on*, 8(2):361–373, 2013.

[37] J. J. Fridrich, M. Goljan, and D. Hogea. Steganalysis of JPEG images: Breaking the F5 algorithm. In *Revised Papers from the 5th International Workshop on Information Hiding*, IH '02, pages 310–323, London, UK, UK, 2003. Springer-Verlag.

[38] J. J. Fridrich and J. Kodovský. Rich models for steganalysis of digital images. *Information Forensics and Security, IEEE Transactions on*, 7(3):868–882, 2012.

[39] J. J. Fridrich, J. Kodovský, V. Holub, and M. Goljan. Breaking HUGO – the process discovery. In *Information Hiding, 13th International Workshop, Lecture Notes in Computer Science*, 2011.

[40] J. J. Fridrich, J. Kodovský, V. Holub, and M. Goljan. Steganalysis of content-adaptive steganography in spatial domain. In *Proc. 13th Information Hiding Workshop*, volume 6958 of *Springer LNCS*, pages 102–117, 2011.

[41] J. J. Fridrich, T. Pevný, and J. Kodovský. Statistically undetectable JPEG steganography: dead ends challenges, and opportunities. In *Proc. 9th ACM workshop on Multimedia and Security*, ACM MM&Sec '07, pages 3–14, 2007.

[42] T. Gloe. Forensic analysis of ordered data structures on the example of JPEG files. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*, pages 139 –144, dec. 2012.

[43] M. Goljan, J. Fridrich, and T. Holotyak. New blind steganalysis and its implications. In *Security, Steganography and Watermarking of Multimedia Contents VIII*, volume 6072 of *Proc. SPIE*, pages 0101–0113, 2006.

[44] G. Gul and F. Kurugollu. A new methodology in steganalysis: Breaking highly undetectable steganography (HUGO). In *Proceedings of the 13th International Conference on Information Hiding*, IH'11, pages 71–84, Berlin, Heidelberg, 2011. Springer-Verlag.

[45] L. Guo, J. Ni, and Y. Q. Shi. An efficient JPEG steganographic scheme using uniform embedding. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*, pages 169–174, 2012.

[46] J. J. Harmsen and W. A. Pearlman. Steganalysis of additive-noise modelable information hiding. In *Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents V*, volume 5020 of *Proc. SPIE*, pages 131–142, 2003.

[47] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, July 2003.

[48] S. Hetzl and P. Mutzel. A graph-theoretic approach to steganography. In J. Dittmann, S. Katzenbeisser, and A. Uhl, editors, *Communications and Multimedia Security*, volume 3677 of *Lecture Notes in Computer Science*, pages 119–128. Springer Berlin Heidelberg, 2005.

[49] V. Holub and J. Fridrich. Designing steganographic distortion using directional filters. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*, pages 234–239, 2012.

[50] V. Holub and J. J. Fridrich. Digital image steganography using universal distortion. In *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '13, pages 59–68, New York, NY, USA, 2013. ACM.

[51] V. Holub, J. J. Fridrich, and T. Denemark. Random projections of residuals as an alternative to co-occurrences in steganalysis, 2013.

[52] N. Hopper, L. von Ahn, and J. Langford. Provably secure steganography. *Computers, IEEE Transactions on*, 58(5):662–676, 2009.

[53] A. D. Ker. Improved detection of LSB steganography in grayscale images. In *Proceedings of the 6th International Conference on Information Hiding*, IH'04, pages 97–115, Berlin, Heidelberg, 2004. Springer-Verlag.

[54] A. D. Ker. A general framework for structural steganalysis of LSB replacement. In *Proceedings of the 7th International Conference on Information Hiding*, IH'05, pages 296–311, Berlin, Heidelberg, 2005. Springer-Verlag.

[55] A. D. Ker. The square root law in stegosystems with imperfect information. In *Proc. 12th Information Hiding Workshop*, volume 6387 of *Springer LNCS*, pages 145–160, 2010.

[56] A. D. Ker. Advanced Security: Information Hiding. Lecture Notes, Hilary Term 2013.

[57] A. D. Ker. Implementing the projected spatial rich features on a GPU. In *Media Watermarking, Security, and Forensics 2014*, volume 9028 of *Proc. SPIE*, pages 1801–1810. SPIE, 2014.

[58] A. D. Ker, P. Bas, R. Böhme, R. Cogranne, S. Craver, T. Filler, J. Fridrich, and T. Pevný. Moving steganography and steganalysis from the laboratory into the real world. In *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '13, pages 45–58, New York, NY, USA, 2013. ACM.

[59] A. D. Ker and I. Lubenko. Feature reduction and payload location with WAM steganalysis. In *Media Forensics and Security XI*, volume 7254 of *Proc. SPIE*, pages 0A01–0A13, 2009.

[60] A. D. Ker and T. Pevný. A new paradigm for steganalysis via clustering. In *Media Watermarking, Security, and Forensics III*, volume 7880 of *Proc. SPIE*, pages 78800U–78800U–13. SPIE, 2011.

[61] A. D. Ker and T. Pevný. Batch steganography in the real world. In *Proceedings of the on Multimedia and Security*, MM&Sec '12, pages 1–10, New York, NY, USA, 2012. ACM.

[62] A. D. Ker and T. Pevný. A mishmash of methods for mitigating the model mismatch mess. In *Media Watermarking, Security, and Forensics 2014*, volume 9028 of *Proc. SPIE*, pages 1601–1615. SPIE, 2014.

[63] M. Kharrazi, H. T. Sencar, and N. Memon. Benchmarking steganographic and steganalysis techniques, 2005.

[64] Y. Kim, Z. Duric, and D. Richards. Modified matrix encoding technique for minimal distortion steganography. In *Proceedings of the 8th International Conference on Information Hiding*, IH'06, pages 314–327, Berlin, Heidelberg, 2007. Springer-Verlag.

[65] A. B. King. *Website Optimization*. O'Reilly Media, 2008.

[66] J. Kodovský. Perturbed quantization - a MATLAB implementation. `http://dde.binghamton.edu/download/pq/`[Last accessed December 2013].

[67] J. Kodovský. *Steganalysis of Digital Images Using Rich Image Representations AND Ensemble Classifiers*. PhD thesis, Binghamton University, SUNY, 2012.

[68] J. Kodovský and J. Fridrich. Influence of embedding strategies on security of steganographic methods in the JPEG domain.

[69] J. Kodovský and J. Fridrich. Calibration revisited. In *Proceedings of the 11th ACM Workshop on Multimedia and Security*, MM&Sec '09, pages 63–74, New York, NY, USA, 2009. ACM.

[70] J. Kodovský and J. J. Fridrich. Steganalysis in high dimensions: fusing classifiers built on random subspaces. In *Media Watermarking, Security, and Forensics III*, volume 7880 of *Proc. SPIE*, pages 0L01–0L13, 2011.

[71] J. Kodovský and J. J. Fridrich. Steganalysis of JPEG images using rich models, 2012.

[72] J. Kodovský and J. J. Fridrich. Steganalysis in resized images. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2857–2861, 2013.

[73] J. Kodovský, J. J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7(2):432–444, 2012.

[74] J. Kodovský, T. Pevný, and J. J. Fridrich. Modern steganalysis can detect YASS. In *Media Forensics and Security XII*, volume 7541 of *Proc. SPIE*, pages 0201–0211, 2010.

[75] L. Ladický and P. Torr. Locally linear support vector machines. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 985–992, New York, NY, USA, 2011. ACM.

[76] J. Langford, L. Li, and A. Streh. Vowpal Wabbit open source project. Technical report, Yahoo!, 2007.

[77] J. Langford, A. Smola, and M. Zinkevich. Slow learners are fast. *Journal of Machine Learning Research*, 1(2099):1–9, 2009.

[78] Q. V. Le, T. Sarlós, and A. J. Smola. Fastfood - computing hilbert space expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, volume 28, pages 244–252, 2013.

[79] Q. Liu. Steganalysis of DCT-embedding based adaptive steganography and YASS. In *Proceedings of the Thirteenth ACM Multimedia Workshop on Multimedia and Security*, MM&Sec '11, pages 77–86, New York, NY, USA, 2011. ACM.

[80] I. Lubenko and A. D. Ker. Steganalysis using logistic regression. In *Media Watermarking, Security, and Forensics III*, volume 7880 of *Proc. SPIE*, pages 0K01–0K11, 2011.

[81] I. Lubenko and A. D. Ker. Going from small to large data in steganalysis. In *Media Watermarking, Security, and Forensics 2012*, volume 8303 of *Proc. SPIE*, pages 0M01–0M10. SPIE, 2012.

[82] I. Lubenko and A. D. Ker. Steganalysis with mismatched covers: do simple classifiers help? In *Proceedings of the on Multimedia and security*, MM&Sec '12, pages 11–18, New York, NY, USA, 2012. ACM.

[83] J. Lukas, J. J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *Information Forensics and Security, IEEE Transactions on*, 1(2):205–214, 2006.

[84] S. Lyu and H. Farid. Detecting hidden messages using higher-order statistics and support vector machines. In F. A. P. Petitcolas, editor, *Proceedings of the 5th Information Hiding Workshop*, volume 2578 of *Lecture Notes in Computer Science*, pages 340–354. Springer Berlin Heidelberg, 2003.

[85] S. Lyu and H. Farid. Steganalysis using color wavelet statistics and one-class support vector machines. In E. J. Delp, III and P. W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306 of *Proc. SPIE*, pages 35–45, June 2004.

[86] J. Makelberge and A. D. Ker. Exploring multitask learning for steganalysis. In *Media Watermarking, Security, and Forensics*, volume 8665 of *Proc. SPIE*, pages 0N01–0N10, 2013.

[87] Y. Mansour, M. Mohri, and A. Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT 2009 - The 22nd Conference on Learning Theory.*

[88] Y. Miche. *Developing Fast Machine Learning Techniques with Applications to Steganalysis Problems.* PhD thesis, School of Science and Technology, Aalto University, 2010.

[89] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, and C. Suen. Unsupervised Feature Learning and Deep Learning Tutorial. `http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial` [Last accessed January 2014].

[90] T. Nguyen and S. Sanner. Algorithms for direct 0–1 loss optimization in binary classification. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1085–1093. JMLR Workshop and Conference Proceedings, May 2013.

[91] United States Department of Agriculture. Natural resources conservation service photo gallery. `http://photogallery.nrcs.usda.gov`. [Last accessed December 2013].

[92] T. Pevný. *Kernel Methods in Steganalysis.* PhD thesis, Binghamton University, SUNY, 2008.

[93] T. Pevný. Co-occurrence steganalysis in high dimensions. volume 8303, page 83030B. SPIE, 2012.

[94] T. Pevný, P. Bas, and J. J. Fridrich. Steganalysis by subtractive pixel adjacency matrix. In *Proc. 11th ACM workshop on Multimedia and Security*, ACM MM&Sec '09, pages 75–84, 2009.

[95] T. Pevný, T. Filler, and P. Bas. Using high-dimensional image models to perform highly undetectable steganography. In *Proceedings of the 12th International Conference on Information Hiding*, IH'10, pages 161–177, Berlin, Heidelberg, 2010. Springer-Verlag.

[96] T. Pevný and J. J. Fridrich. Merging Markov and DCT features for multi-class JPEG steganalysis. In *Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505 of *Proc. SPIE*, pages 03–14, 2007.

[97] T. Pevný, J. J. Fridrich, and A. D. Ker. From blind to quantitative steganalysis. *IEEE Transactions on Information Forensics and Security*, 7(2):445–454, 2012.

[98] N. Provos. Defending against statistical steganalysis. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, SSYM'01, pages 24–24, Berkeley, CA, USA, 2001. USENIX Association.

[99] J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[100] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, pages 1313–1320, 2008.

[101] P. Rai, H. Daumé, and S. Venkatasubramanian. Streamed learning: One-pass SVMs. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, pages 1211–1216, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[102] J.A. Rice. *Mathematical Statistics And Data Analysis*. Duxbury Advanced Series. Thomson/Brooks/Cole, 2007.

[103] B. M. Rodriguez and G. L. Peterson. Multi-class classification fusion using boosting for identifying steganography methods. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6974 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, March 2008.

[104] V. Sachnev, H. J. Kim, and R. Zhang. Less detectable JPEG steganography method based on heuristic optimization and BCH syndrome coding. In *Proceedings of the 11th ACM Workshop on Multimedia and Security*, MM&Sec '09, pages 131–140, New York, NY, USA, 2009. ACM.

[105] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[106] V. Schwamberger and M. O. Franz. Simple algorithmic modifications for improving blind steganalysis performance. In *Proceedings of the 12th ACM Workshop on Multimedia and Security*, MM&Sec '10, pages 225–230, New York, NY, USA, 2010. ACM.

[107] T. Sharp. An implementation of key-based digital signal steganography. In *Proceedings of the 4th International Workshop on Information Hiding*, IHW '01, pages 13–26, London, UK, UK, 2001. Springer-Verlag.

[108] Y. Q. Shi, Chunhua , and W. Chen. A Markov process based approach to effective attacking JPEG steganography. In *Proc. 8th Information Hiding Workshop*, volume 4437 of *Springer LNCS*, pages 249–264, 2007.

[109] G. J. Simmons. The prisoners' problem and the subliminal channel. In *Advances in Cryptology: Proceedings of CRYPTO '83*, pages 51–67. Plenum Press, 1983.

[110] K. Solanki, A. Sarkar, and B.S. Manjunath. YASS: Yet another steganographic scheme that resists blind steganalysis. In *Information Hiding*, volume 4567 of *Lecture Notes in Computer Science*, pages 16–31. Springer Berlin Heidelberg, 2007.

[111] K. Solanki, K. Sullivan, U. Madhow, B.S. Manjunath, and S. Chandrasekaran. Provably secure steganography: Achieving zero K-L divergence using statistical restoration. In *Image Processing, 2006 IEEE International Conference on*, pages 125–128, 2006.

[112] D. Upham. Steganographic algorithm JSteg. `http://zooid.org/~paul/crypto/jsteg`[Last accessed December 2013].

[113] M. van Dijk and F. Willems. Embedding information in grayscale images. In *Information and Communication Theory in the Benelux, Proceedings of the 22nd Symposium on*, pages 147–154, Enschede, The Netherlands, 2001.

[114] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[115] U. von Luxburg and B. Schölkopf. Statistical learning theory: Models, concepts, and results. *Handbook of the History of Logic*, 10:651–706, 2008.

[116] G. K. Wallace. The JPEG still picture compression standard. *Commun. ACM*, 34(4):30–44, April 1991.

[117] Y. Wang and P. Moulin. Perfectly secure steganography: Capacity, error exponents, and code constructions. *IEEE Transactions on Information Theory*, 54(6):2706–2722, 2008.

[118] A. Westfeld. F5 - a steganographic algorithm: High capacity despite better steganalysis. In *4th International Workshop on Information Hiding*, pages 289–302. Springer-Verlag, 2001.

[119] A. Westfeld and A. Pfitzmann. Attacks on steganographic systems. In *Proceedings of the Third International Workshop on Information Hiding*, IH '99, pages 61–76, London, UK, UK, 2000. Springer-Verlag.

[120] C. K. I. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS)*, pages 682–688, 2000.