Developing fast machine learning techniques with applications to steganalysis problems

Yoan Miche

DEVELOPING FAST MACHINE LEARNING TECHNIQUES WITH APPLICATIONS TO STEGANALYSIS PROBLEMS

YOAN MICHE

Dissertation for the obtention of the degree of Doctor of Science (D.Sc.) in Technology.

Department of Information and Computer Science

Faculty of Natural Sciences

Aalto University School of Science and Technology

Gipsa-Lab

Ecole Doctorale EEATS

Institut National Polytechnique de Grenoble

October 2010

DISTRIBUTION:

Aalto University School of Science and Technology
Faculty of Information and Natural Sciences
Department of Information and Computer Science
P.O. Box 15400
00076 Aalto
FINLAND
URL : http://ics.tkk.fi
Tel: +358 9 470 01
Fax: +358 9 470 23369
Email: series@ics.tkk.fi

ISBN: 978-952-60-3427-0
ISBN (ONLINE): 978-952-60-3428-7
ISSN: 1797-5050
ISSN (ONLINE): 1797-5069
URL: http://lib.tkk.fi/Diss/2010/isbn9789526034287

Multiprint
Espoo 2010

Yoan Miche: *Developing Fast Machine Learning Techniques with Applications to Steganalysis Problems,* Dissertation for the obtention of the degree of Doctor of Science (D.Sc.) in Technology to be presented with due permission of the Faculty of Information and Natural Sciences for public examination and debate in Auditorium AS1, at the Aalto University School of Science and Technology (Espoo, Finland) on the 2nd of November, 2010, at 12 noon, © October 2010

SUPERVISORS:
Olli Simula
Christian Jutten

INSTRUCTORS:
Amaury Lendasse
Patrick Bas

OPPONENT:
Tapio Seppänen

PRE-EXAMINERS:
Thomas Villmann
Andrew Ker

LOCATION:
Espoo

DATE:
October 2010

ISBN:
978-952-60-3427-0
ISBN (ELECTRONIC VERSION):
978-952-60-3428-7

With words,
priests and poets make into many
the hidden Reality which is but One.

— RigVeda, X, 114, 2 [1]

ABSTRACT

In the history of human communication, the concept and need for secrecy between the parties has always been present. One way of achieving it is to modify the message so that it is readable only by the receiver, as in cryptography for example. Hiding the message in an innocuous medium is another, called steganography. And the counterpart to steganography, that is, discovering whether a message is hidden in a specific medium, is called steganalysis. Other concerns also fall within the broad scope of the term steganalysis, such as estimating the message length for example (which is quantitative steganalysis).

In this dissertation, the emphasis is put on classical steganalysis of images first — the mere detection of a modified image — for which a practical benchmark is proposed: the evaluation of a sufficient amount of samples to perform the steganalysis in a statistically significant manner, followed by feature selection for dimensionality reduction and interpretability. The fact that most of the features used in the classical steganalysis task have a physical meaning, regarding the image, lends itself to an introspection and analysis of the selected features for understanding the functioning and weaknesses of steganographic schemes.

This approach is computationally demanding, both because of the feature selection and the size of the data in steganalysis problems. To address this issue, a fast and efficient machine learning model is proposed, the Optimally-Pruned Extreme Learning Machine (OP-ELM). It uses random projections in the framework of an Artificial Neural Network (precisely, a Single Layer Feedforward Network) along with a neuron selection strategy, to obtain robustness regarding irrelevant features, and achieves state of the art performances.

The OP-ELM is also used in a novel approach at quantitative steganalysis (message length estimation). The re-embedding concept is proposed, which embeds a new known message in a suspicious image. By repeating this operation multiple times for varying sizes of the newly embedded message, it is possible to estimate the original message size used by the sender, along with a confidence interval on this value. An intrinsic property of the image, the inner difficulty, is also revealed thanks to the confidence interval width; this gives an important information about the reliability of the estimation on the original message size.

KEYWORDS: Machine Learning, Steganography, Steganalysis, Extreme Learning Machine, Artificial Neural Networks, Feature Selection, Re-embedding.

RÉSUMÉ

Depuis que les Hommes communiquent, le besoin de dissimuler tout ou partie de la communication existe. On peut citer au moins deux formes de dissimulation d'un message au sein d'une communication: Dans le premier cas, le message à envoyer peut lui même être modifié, de telle sorte que seul le destinataire puisse le décoder. La cryptographie s'emploie par exemple à cette tâche. Une autre forme est celle de la stéganographie, qui vise à dissimuler le message au sein d'un document. Et de même que pour la cryptographie dont le pendant est la cryptanalyse visant à décrypter le message, la stéganalyse est à l'opposé de la stéganographie et se charge de détecter l'existence d'un message. Le terme de stéganalyse peut également désigner l'importante classe de problèmes liés à la détection de l'existence du message mais aussi à l'estimation de sa taille (stéganalyse quantitative) ou encore de son contenu.

Dans cette thèse, l'accent est tout d'abord mis sur le problème classique de stéganalyse (détection de la présence du message). Une méthodologie permettant d'obtenir des résultats statistiquement fiables dans ce contexte est proposée. Il sagit tout d'abord d'estimer le nombre d'échantillons (ici des images) suffisant à l'obtention de résultats pertinents, puis de réduire la dimensionalité du problème par une approche basée sur la sélection de variables. Dans le contexte de la stéganalyse, la plupart des variables obtenues peuvent être interprétées physiquement, ce qui permet une interprétation de la sélection de variables obtenue: les variables sélectionnées en premier réagissent vraisemblablement de façon importante aux changements causés par la présence du message. Leur analyse peut permettre de comprendre le fonctionnement et les faiblesses de l'algorithme de stéganographie utilisé, par exemple.

Cette méthodologie peut s'avérer complexe en termes de calculs et donc nécessiter des temps d'éxecution importants. Pour pallier à ce problème, un nouveau modèle pour le "Machine Learning" est proposé, l'OP-ELM. L'OP-ELM est constitué d'un Réseau de Neurones au sein duquel des projections aléatoires sont utilisées. Les neurones sont ensuite classés par pertinence vis à vis du problème, et seuls les plus pertinents sont conservés. Cette structure de modèle parvient à obtenir des performances similaires à celles de l'état de l'art dans le domaine du "Machine Learning".

Enfin, le modèle OP-ELM est utilisé dans le cadre de la stéganalyse quantitative, cette fois (l'estimation de la taille du message). Une approche nouvelle sur ce problème est utilisée, faisant appel à une technique de ré-insertion d'un message au sein d'une image considérée comme suspecte. En répétant ce processus de ré-insertion un certain nombre de fois, et pour des messages connus de tailles différentes, il est possible d'estimer la taille du message original utilisé par l'expéditeur. De plus, par l'utilisation de la largeur de l'intervalle de confiance obtenu sur la taille du message original, une mesure de la difficulté intrinsèque à l'image est présentée. Ceci permet d'estimer la fiabilité de la prédiction obtenue pour la taille du message original.

MOT-CLÉS: Machine Learning, Stéganographie, Stéganalyse, Extreme Learning Machine, Réseaux de Neurones, Sélection de caractéristiques, Ré-insertion.

Chaque génération éprouve le désir
toujours renouvelé de se former
en s'opposant à l'air du temps,
à l'esprit du lieu,
et le désir de s'épanouir à l'ombre —
ou plutôt à la clarté —
d'un maître exemplaire.

— Daniel Pennac, Chagrin d'Ecole.

ACKNOWLEDGMENTS

This dissertation is one of the end products of the research I have conducted at both the Department of Information and Computer Science (at Aalto University School of Science and Technology, Finland) and the Gipsa-Lab (Institut National Polytechnique de Grenoble, France). This work has been funded by both the INPG and the Aalto University, as well as the Helsinki Graduate School in Computer Science and Engineering (Hecse). Hence, many thanks to all the people behind the funding of this research, and more specifically to Erkki Oja, Pekka Orponen and also to Greger Lindén for all the work and dedication to the Hecse program.

I would like to thank again Erkki Oja and Pekka Orponen for providing such nice facilities and work environment at the ICS department. It truly helps a lot to have a good office and hardware to work with.

I am very grateful to my supervisors Christian Jutten and Olli Simula; first for accepting me as one of their students — it is certainly a privilege and honor to have you as supervisors — and second for providing comments and suggestions on some questions and problems I had very often not even thought of. You have been a great support and help in some key moments of this dissertation.

Amaury Lendasse got me into Machine Learning during a course I was not even supposed to take in the first place — and probably did not really understand then. After a short interview with him and Patrick Bas, they agreed to be my instructors even though I did not exactly give the best impression during this course. I truly learned a lot during the last four years, with your help and advising. And not only on the Machine Learning and Steganography subjects.

Thank you Amaury for putting up with me, for the encouragement and guidance during these years. And for bearing with my love of long, elaborate and novel-like sentences in scientific publications. I have strayed, but I shall redeem !

And thank you Patrick for interesting me with these security matters and for the "very-last-second" reviews and comments that I asked from you sometimes. Somehow, deadlines are always coming too quick. Thanks to their patience and constant help, I think I have started to grasp a bit of both subjects by now. Also, thank you both for being such good friends all along. I truly had a lot of fun with both of you.

I want to thank the EIML Group and all of its members, for their support, their everyday good mood and for all the good laughs. It really has been a pleasure to be around you and enjoy many good moments with you. Especially, thanks to Antti Sorjamaa and Emil Eirola for always having some advice, comments, ideas — including of course funny videos to watch — and also helping so much with my Finnish-related issues.

It has been nice to work around the members of the whole ICS department, and I would like to thank them all and especially Mari-Sanna Paukkeri for her unalterable joy.

I thank Tapio Seppänen for the honor of having him as an opponent, and also the members of the French jury — which contains the pre-examiners Thomas Villmann and Andrew Ker. Thank you for insightful comments and suggestions on the dissertation.

My family also deserves many thanks, for their support, believing in what I do and the choices I have made, and most of all, for never asking the dreaded question of doctoral students: “But when are you getting a real job?”.

Finally, this dissertation would not have been the same — if at all — without the help and support of Bénédicte during the intensive two weeks it took to write it. Thank you so much, Ma Dame.

Yoan Miche,
Espoo, October 2010.

CONTENTS

$\|\cdot\|$ A norm

$\|\cdot\|_2$ The Euclidean norm

$\hat{\cdot}$ The estimate of a quantity

A Number of non-zero AC coefficients

A_o The original number of non-zero AC coefficients

Acc The accuracy (defined with true positives/negatives)

B The number of JPEG sub-blocks in the image i

B_1, B_2 The first and second order blockiness of the image

b A bias

$\mathbf{b}$ The modified cover image bits (wet paper codes)

C Embedding capacity

$\mathbf{C}$ The co-occurrence matrix of neighboring DCT coefficients

C_{Tot} Total embedding capacity

D The inner image difficulty

$\mathbf{D}$ A pseudo-random binary matrix

D_1, D_2 Amounts of distortions caused to a medium

d A number of modifications (context of coding-based steganography)

E Number of embedding changes

$E(\cdot)$ The expectation

E_o The original number of embedding changes

E_j The number of re-embedding changes

E_j^O The number of embedding changes for the first embeddings

$f(\cdot)$ A function

$f_{risk}(\cdot,\cdot)$ A risk function

$f_{MSE}(\cdot,\cdot)$ A Mean Square Error risk function

F A functional (context of feature extraction)

F_h, F_v, F_d, F_m The difference arrays of DCT coefficients for horizontal, vertical, main diagonal and minor diagonal directions

FP, FN False positives, false negatives

FS, FS_{sub} A feature set/ a subset of the feature set FS

$G(\mathbf{i})$ The smoothness of the group of pixels $\mathbf{i} = (i_1, \ldots, i_n)^T$

$g_{x,y}(I_{dh})$ The dual histogram of DCT value I_{dh} at the (x, y) DCT coefficient

$H(\cdot)$ The entropy

$\mathbf{H}$ The global histogram $\mathbf{H} = (H(1), \ldots, H(R))^T$

H_c^j, H_s^j The histograms of pixel values j for cover and stego images I_c and I_s

$h_{x,y}(r)$ The individual histogram of DCT value r for DCT coefficient (x, y)

$\mathbb{I}$ The sample space of the cover images for random variable I_c

$\mathbf{I}$ The identity matrix

I The DCT coefficients representation of the image i

I_c The random variable representing the cover images

I_{dh} A DCT value

I_s The random variable representing the stego images

$I(x, y)$ The DCT coefficient at coordinates (x, y) in I

$I^{(k)}(x, y)$ The DCT coefficient at coordinates (x, y) in the JPEG sub-block k

I_α, I_β The random variables corresponding to the parts of the image i_α, i_β

I_{row}, I_{col} The vectors of sub-blocks indices while scanning the image i by rows or columns, resp.

i An image

i_α, i_β Parts of the image i

i_j Chunks of the host image (Steghide algorithm)

i_m A part of the image i that holds the message $\mathbf{m}$

K A stego-key

$\mathcal{K}$ A kernel function (SVM); a non-linear activation function (ANN)

k The number of Nearest Neighbors in the k-NN algorithm; also a number of bits

$\mathcal{M}$ A model

M The total number of features, variables

M_f The dimensionality of the space

$\mathbf{M}_h, \mathbf{M}_v, \mathbf{M}_d, \mathbf{M}_m$ The Markov transition probability matrices for horizontal, vertical, main diagonal and minor diagonal directions

$MI(\cdot)$ The mutual information

$\mathcal{M}$ A model

$\mathbf{m}$ The message to embed

m_j Bits of the message $\mathbf{m}$ to embed

$\mathbb{N}$ The ensemble of natural numbers

N The total number of images, samples

N_{00}, N_{01}, N_{11} Measures based on the co-occurrence matrix of DCT coefficients

N_P, N_N Number of total positive/negative instances

n A number of elements

$P(\cdot)$ A probability

$\mathbf{P}$ A projection matrix

P_I A probability distribution of the random variable I (modeling the images)

QF_h Quality factor for JPEG compression (YASS algorithm)

$Q(x, y)$ The (x, y) component of the quantization matrix Q

$\mathbb{R}$ The ensemble of real numbers

$R^{(1)}, R^{(2)}$ Embedding rates

R_o The original embedding rate (in the $R^{(1)}$ sense)

R_j The re-embedding rate (in the $R^{(1)}$ sense)

R_j^O The embedding rate for the first embeddings

$S(\mathbf{m})$ Length of the embedded message (in bits)

s_j Bits of the image in which to embed the message bits m_j

$\mathbf{s}(k)$ The Reservoir internal state at step k

$SF(\cdot)$ A scoring function (filtering approach for feature selection)

T A threshold

TP, TN Amount of true positive/negative

V The variation (context of calibrated DCT features)

$\mathbf{w}$ A hyperplane/ normal to a hyperplane (in LDA or SVM); the weights of an ANN

$\mathbf{x}$ A matrix containing the input data

$\mathbf{x}(k)$ A time variant process input data at step k

$\mathbf{x}_l, \mathbf{x}_v, \mathbf{x}_t$ A matrix containing the learning/ validation/ test data

$\mathbf{y}$ A matrix containing the output data

$\mathbf{y}(k)$ A time variant process output data at step k

$\hat{\mathbf{y}}$ An estimation by a model of the output data

$\boldsymbol{\beta} = (\beta_1, \ldots, \beta_N)^T$ The output layer weights in an ANN

χ^2 A chi-square statistic

$\delta(\cdot, \cdot)$ The Kullback-Leibler divergence

$\delta_{\cdot,\cdot}$ The Kronecker symbol

ε, ϵ An error

$\varepsilon_{FS}, \varepsilon_{FS_{sub}}$ The error obtained using a feature set/ using a subset of the feature set FS

$\varepsilon_r, \varepsilon_c$ A regression/ classification error

$\varepsilon_l, \varepsilon_v, \varepsilon_t$ A learning/ validation/ test error

$\Gamma(\cdot)$ The Gamma function $\Gamma(y) = \int_0^\infty t^{y-1} e^{-t} dt$

ϕ The output layer function (usually linear) in an ANN

$\boldsymbol{\Sigma}$ A covariance matrix

σ, σ_0 The standard deviation (context of multivariate Gaussian or RBF)

θ_j A model parameter

Θ A set of model parameters θ_j

ξ_j Slack variables in the Support Vector Machines formulation

ACRONYMS

AIC Akaike's Information Criterion

ANN Artificial Neural Network

BIC Bayesian Information Criterion

BOWS2 Break Our Watermarking Scheme 2

BSS Blind Source Separation

CV Cross-Validation

DCT Discrete Cosine Transform

ELM Extreme Learning Machine

EM-ELM Error-Minimized Extreme Learning Machine

GLVQ Generalized Learning Vector Quantization

GP Gaussian Processes

GRLVQ Generalized Relevance LVQ

HQ Hannan-Quinn (criterion)

ICA Independent Component Analysis

J-L Johnson-Lindenstrauss

JPEG Joint Photographic Experts Group

JPHS JPEG Hide and Seek

KL Kullback-Leibler (divergence)

KNN k-Nearest Neighbors

LARS Least Angle Regression

LDA Linear Discriminant Analysis

LOO Leave-One-Out

LSB Least Significant Bit

LS-SVM Least-Squares Support Vector Machines

LVQ Learning Vector Quantization

MDS Multi-Dimensional Scaling

MLP Multi-Layer Perceptron

MMD Maximum Mean Discrepancy

MMx Modified Matrix Encoding

MRSR Multi-Response Sparse Regression

MSE Mean Square Error

NMSE Normalized Mean Square Error

OLS Ordinary Least Squares

OP-ELM Optimally-Pruned Extreme Learning Machine

OP-KNN Optimally-Pruned k Nearest Neighbors

PCA Principal Component Analysis

QIM Quantization Index Modulation

RBF Radial Basis Function

RBFN Radial Basis Function Network

RC Reservoir Computing

RGB Red Green Blue

RLVQ Relevance Learning Vector Quantization

ROC Receiver Operating Characteristic

SLFN Single-Layer Feedforward Network

SMO Sequential Minimization Optimization

SOM Self-Organizing Map

SPAM Substractive Pixel Adjacent Matrix

SVM Support Vector Machine

VQ Vector Quantization

YASS Yet Another Steganographic Scheme

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 SCOPE OF THE DISSERTATION

A typical problem in Machine Learning relates to the growing amount of data and the ways of handling it. In this dissertation, we explore this problem from a particular point of view: the steganalysis problem.

In the same way that cryptanalysis is the counterpart of cryptography, steganalysis attempts to uncover steganography. The primary goals are rather different from the cryptography ones, though. The art of steganography relates to information hiding: a pair of communicating parties, sender and receiver, attempt to pass a message hidden in an innocuous medium. A usual setup is to have a potential eavesdropper who has access to the transmission channel on which the medium between sender and receiver is exchanged — for example, access to the picture sharing website that sender and receiver use to exchange pictures (Flickr, Picasa...). This eavesdropper is supposed to be passive, in this dissertation, and only wishes to identify whether the concerned medium has been tampered with or if it is genuine.

Steganalysis is then the work perpetrated by this eavesdropper, typically. In the classical sense of steganalysis, the eavesdropper only identifies the medium as tampered — for example to report it to authorities — but other classes of steganalysis can go beyond a simple detection task: one may also wish to gather information about the hidden message, such as its length, how it has been hidden (i.e. by which steganography process)...

In the unlikely event that the sender is being careless about the way of hiding the message, the eavesdropper might even realize that the medium has been modified by simply looking at it with his own eyes. With the coming of digital age and more elaborate steganographic techniques, the task has become much more challenging, though.

One way of performing steganalysis — and nowadays the most widely used — is to extract some specific characteristics from the suspicious image, known as features, and compare them to the ones obtained on other original images. We can see here a supervised binary classification problem arise: to compare the features of a suspicious image and of genuine images, it is possible to use a machine learning model previously trained to recognize the differences between genuine and tampered.

Due to the large number of features typically used in steganalysis (in the order of magnitude of hundreds), the machine learning task is not straightforward and requires specific models able to deal with such dimensionality of the data.

In this dissertation, we propose a new machine learning model, the OP-ELM, based on random projections and neuron selection, to obtain a sufficiently good performance/ speed ratio; it is then possible to conduct more numerous and elaborate experiments related to steganalysis problems. We then develop a methodology for steganalysis, to obtain a practical benchmark for a steganographic algorithm in a reliable manner. Finally, we address the

problem of the estimation of the message length and its reliability in a novel manner, by the use of an approach using re-embeddings.

1.2 SCIENTIFIC CONTRIBUTIONS OF THE DISSERTATION

The present dissertation contains the following scientific contributions:

- A new machine learning model based on random projections and Artificial Neural Networks is proposed, the Optimally-Pruned Extreme Learning Machine. It is based on an existing scheme, the Extreme Learning Machine, by Huang [69] and addresses one of its original weaknesses: its sensitivity to irrelevant features in a data set. Using a large number of neurons in the neural network built by the ELM, ranking them using the MRSR algorithm [117] and selecting only the most relevant ones by a Leave-One-Out criterion, the OP-ELM reduces greatly the sensitivity of the original ELM, while retaining very low computational times.
- The original version of the OP-ELM is modified in the place of the neuron selection criterion. Instead of using the Leave-One-Out criterion, we propose to use an information theoretic based one, the Hannan-Quinn criterion [62]. This modification is also proposed for another model using a structure similar to that of the OP-ELM, the OP-KNN [136], and enables to increase the speed of the original OP-ELM by three to four folds while retaining similar performances.
- Using the OP-ELM as a machine learning tool, a methodology for reliable steganalysis is proposed, which aims at estimating a sufficient number of images required for the results to be statistically significant. This methodology illustrates that for all cases of steganographic algorithms, an insufficient number of images to perform steganalysis gives results with a large variance. Regarding the number of features this time, the second part of the methodology for reliable steganalysis uses feature selection to reduce the computational time and complexity of the problem, while keeping the same performances. These selected features can then be analyzed to reverse-engineer the steganographic algorithm used.
- The problem of quantitative steganalysis is tackled by a novel approach using the re-embedding concept. By embedding new messages in multiple copies of the suspicious image, we propose to estimate the original embedding rate with a confidence interval on the value. In addition, the width of the obtained confidence interval is shown to be an estimator of the inner image difficulty.

1.3 PUBLICATIONS PRESENTED AND AUTHOR'S CONTRIBUTION

This dissertation consists in an introductory part, and the six following peer-reviewed publications.

A — *OP-ELM: Optimally-Pruned Extreme Learning Machine*, Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten and Amaury

Lendasse. In IEEE Transactions on Neural Networks, January 2010, Number 1, pp. 158–162 , Volume 21.

In this publication, the Optimally-Pruned Extreme Learning Machine (OP-ELM) is proposed. It is a methodology built around the original Extreme Learning Machine [69] with the aim of having a fast, efficient and more robust model (regarding the sensitivity of the original ELM to irrelevant data). The OP-ELM adds new kernels to the original ELM and uses a pruning of the neurons with the MRSR [117] algorithm and a Leave-One-Out criterion to remove the most irrelevant ones. In this work, the original idea was proposed by Amaury Lendasse and developed by the author and Amaury Lendasse. The experiments and writing of the publication have been carried out mainly by the author, with the help of Antti Sorjamaa. The other authors have provided useful suggestions and corrections to the original manuscript.

B — *A Faster Model Selection Criterion for OP-ELM and OP-KNN: Hannan-Quinn Criterion*, Yoan Miche and Amaury Lendasse. In ESANN'09: European Symposium on Artificial Neural Networks, April 2009, Michel Verleysen ed., published by d-side publications. pp. 177–182.

In this conference publication, we sought to replace the originally used Leave-One-Out criterion of the original OP-ELM by a faster one, and decided to use an information criterion for this task. Writing of the article and experiments using the OP-ELM and OP-KNN with the new criterion were carried out by the author.

C — *A Feature Selection Methodology for Steganalysis*, Yoan Miche, Benoit Roue, Patrick Bas and Amaury Lendasse. In MRCS'06, International Workshop on Multimedia Content Representation, Classification and Security, Istanbul (Turkey), B. Gunsel, A. K. Jain, A. M. Tekalp and B. Sankur eds., Lecture Notes in Computer Science, Springer-Verlag, 2006. Volume 4105, pp. 49–56.

This publication is the very first one dealing with feature selection for steganalysis and is the first publication of the author as a doctoral student. The idea is an improvement from the Master's thesis work carried out in the months before the beginning of the doctoral studies. The idea is to prove that by using a reduced set of features in steganalysis, one can obtain the same performances while reducing the computational time and gaining interpretability (from the selected features). The author conducted most of the experiments, helped by Benoit Roue, Patrick Bas and Amaury Lendasse. The conference paper was written by the author.

D — *Advantages of Using Feature Selection Techniques on Steganalysis Schemes*, Yoan Miche, Patrick Bas, Amaury Lendasse, Christian Jutten and Olli Simula. In IWANN'07: International Work-Conference on Artificial Neural Networks, San Sebastian, Spain, June 2007, Francisco Sandoval et al. eds., Lecture Notes in Computer Science, Springer Berlin / Heidelberg. Volume 4507/2007, pp. 606–613.

While performing more experiments using the methodology from the previous publication, the author realized that the statistical stability

of the results was affected by the number of samples used to perform the experiments. In this spirit, another methodology was devised, to infer a sufficient number of samples for the steganalysis task using a specific set of features. The idea and experiments were carried out by the author, as well as the writing. The advisors Patrick Bas and Amaury Lendasse provided very useful advice and helped correcting the original manuscript.

E — *Reliable Steganalysis Using a Minimum Set of Samples and Features*, Yoan Miche, Patrick Bas, Amaury Lendasse, Christian Jutten and Olli Simula. In EURASIP Journal on Information Security, March 2009. Hindawi Publishing Corporation. Volume 2009, Article ID 901381, pp. 1–13.

This publication is the result of the combination of the two previous ones, used on six different steganographic algorithms, for different embedding rates and on a large publicly available database of images. In addition, this version of the global methodology uses the OP-ELM consistently everywhere, while the previous publications were using k-Nearest Neighbors and Support Vector Machines at different stages, for speed or performance. The author carried out the experiments and wrote the article. Patrick Bas helped greatly with the interpretability of the selected features and Amaury Lendasse on the Machine Learning problems that arose. All authors have finally helped improving the quality of the manuscript by providing essential remarks.

F — *Using Multiple Re-embeddings for Quantitative Steganalysis and Image Reliability Estimation*, Yoan Miche, Patrick Bas and Amaury Lendasse. In TKK Reports in Information and Computer Science, June 2010, Espoo. Number TKK-ICS-R34. ISBN 978-952-60-3249-8 (Print).

The last publication is related to very recent work on quantitative steganalysis. The use of a novel approach using re-embedding enables to estimate more reliably the original message's length and also provides insights on the inner image difficulty. The use of re-embedding was devised during a fruitful discussion between the author, Tomàš Pevnỳ and Patrick Bas. The author carried out the experiments and the paper writing, with the help of Amaury Lendasse and Patrick Bas, especially on the concept of inner image difficulty.

In the rest of the dissertation, the included publications are referred by the capital letter used above, i.e. "Publication A" for the publication entitled "*OP-ELM: Optimally-Pruned Extreme Learning Machine*", by Yoan Miche *et al.*, January 2010.

1.4 STRUCTURE OF THE DISSERTATION

This dissertation is articulated in three main parts. The first one dwells with the field of steganography and steganalysis and aims at giving a short overview of the field in order to explicate the main results and contributions of this thesis. The second chapter proposes a review of steganography and its recent evolutions, toward digital media, and more specifically images. Some of the most widely used steganography algorithms are presented

and described, along with the main principles and important definitions. This chapter lays the foundation and motivation for the following one, about steganalysis. This third chapter describes the concept of steganalysis at length, with the variants of the steganalysis problem (quantitative, forensic...). A detailed presentation of the classical sets of features used to perform feature-based steganalysis is given to illustrate one of the problems faced in steganalysis today: the growing dimensionality of the data.

The second part of this dissertation proposes first a review of the machine learning field, in the fourth chapter, followed by the presentation of a novel machine learning method capable of handling large data sets in reasonable computational times. The fifth chapter details this novel model and the class of models it is built on, the random projections used in Artificial Neural Networks framework.

The third part lays two methodologies aimed at two different problems in steganalysis. The first one, in chapter six is directed toward obtaining reliable results in classical steganalysis setups, by the determination of a sufficient number of images required to perform the task, and then a study on the required features, to perform the task. The seventh chapter then deals with the quantitative steganalysis problem and uses a novel approach to obtain an estimate of the embedded message's length and estimate an inner characteristic of a considered image: its difficulty for a steganalysis task.

The dissertation is finally concluded in chapter eight.

Part I

BASICS ON STEGANOGRAPHY AND STEGANALYSIS

2 STEGANOGRAPHY

This chapter first defines globally the concept of *steganography*, with some historical examples. Some definitions in the steganography framework are then given such as the *capacity* and *embedding rate*, followed by details about the two main parts of steganography: the *model-based* and *coding-based* steganography. Finally, a non-exhaustive review of famous and widely used steganography algorithms and techniques is given.

2.1 WHAT IS STEGANOGRAPHY

While the idea of steganography dates back to ancient times (from the available records we have on it), it is only recently that the actual name has been devised, by Johannes Trithemius (1462–1516), in *Steganographia*. *Steganographia* is believed to be one of the very first works on cryptography and steganography, in which various primitive cryptography and steganography techniques are detailed. The steganography term itself was created from the Greek "*steganos*" — covered — and "*graphia*" — writing.

The third book of Steganographia was believed to be about occultism and magic until very recently [124]. When it got deciphered [109], it revealed to be also about steganography and cryptography.

In steganography, only the existence of the message is secret: the communication channel is considered as open and the message itself is not usually modified so as to resist an attacker by itself (although it can be encrypted, e.g.). The main achievement is hence to hide the message as well as possible in an innocuous content, so that any eavesdropper would have no suspicions. Figure 1 illustrates a case of steganography (LSB replacement, see section 2.4.2) for which the hiding of the message is invisible to the human eye.

It is important to distinguish *steganography* from *cryptography*, first: cryptography aims at modifying the message so that it becomes impossible to read to an eavesdropper. It is of no concern to cryptography that the encrypted message might look suspicious. Steganography does not alter the message but only hides it in a medium, so that it will not arise suspicions.

Steganography is different from cryptography and watermarking...

One also wants to make a difference between *steganography* and *watermarking*. In the latter, one of the main concerns is to be robust, not hidden (although it might be a secondary requirement). A watermark should resist to various transformations of the original content so that it does not get disrupted or destroyed. The simple example of a copyright mark placed in an image, should be resistant to most image transformations such as resizing, cropping, rotation or JPEG compression [78]... In a similar fashion, audio watermarking attempts to resist various transformations of the audio signal, for example MP3 compression [32].

Let us go through some historical examples of steganography. Trithemius describes only text-based steganography (text is the medium for the steganography) in his *Steganographia*, but in the history of human communications there have been numerous examples of steganography, in many different forms.

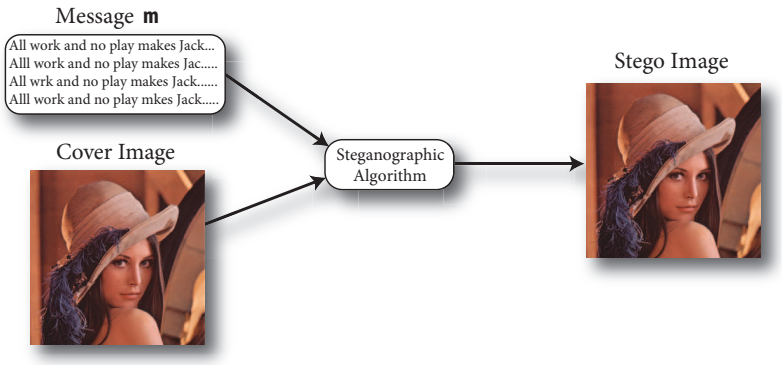

Figure 1: A simple illustration of steganography for an image: a *message* **m** is embedded in the *cover image* by the means of a *steganographic algorithm*. The resulting image (containing the message **m**), looking as similar as possible to the original cover image, is called *stego image.*

2.2 HISTORICALLY

Herodotus has many steganography examples: hiding a message in a belt buckle, earrings...

Since communication exists, the need for secrecy in this communication has been present, whether for benign privacy purposes or malicious ones. Some of the oldest known examples are related by Herodotus in his Histories [64]: Histiaeus, willing to regain his position of tyrant of Miletus, sent a message by means of steganography to his nephew Aristagoras, in order to instigate a revolt in the Ionia. The message was conveyed by a slave of Histiaeus whose head had been first shaved and tattooed with the content of the message. Once the hair grew back sufficiently to cover the message, the slave could be sent without any risk to Aristagoras, who only had to shave the head of the slave to recover the message.

Another example from the same source involves the use of a wooden tablet engraved with the message and then covered with wax. The recipients only had to scrape the wax to obtain the message, while throughout the many enemy hands the tablet passed, it only seemed like an innocuous wax tablet.

The coded message was also possibly included in the word processors of each minister.

More recently, when reprints of internal British cabinet documents were found repeatedly in the press, Margaret Thatcher allegedly had a small message (different for each copy) embedded in the documents given to each minister. The message was embedded in the spacing properties of the documents (word and line spacings) and enabled to identify a potential leak in the government [7].

Unfortunately all these examples rely on "non-official" sources and there are no concrete proofs of such steganography. Obviously, the best steganography cases will possibly never come detected or known...

2.3 NOWADAYS

In the last few years, the interest for steganography has been rising, as shows for example the number of publications related to steganography and watermarking subjects as reported by the IEEE [30].

Both newspapers *USA Today* (USA) and *Corriere della Sera* (Italy) claimed the use of steganographic means (using images as the medium) for Al Quaeda communications. Unfortunately, no evidence or actual article proving the allegations ever appeared.

Steganography is seldomly detected in real life, but is most likely used.

Nevertheless, research agencies and governments being apprehensive for the use of steganography have had a growing interest in developing and detecting it. The discovery of a section about steganography and covert communication in the *Technical Mujahid, a Training Manual for Jihadis*, has brought back these matters in the light of possible terrorism, which can partly explain this interest.

The possibility that it is used and not being detected worries many securities agencies. Especially regarding terrorism related communications.

In any case, the existence of steganography and the possibilities it opens makes it an interesting security-related problem worth being investigated.

The growth in numbers of digital objects publicly available has made digital steganography an easier task, in the sense that it has become difficult for the authorities to monitor all of the digital content being exchanged. Especially since this digital content can be of various nature. Indeed, one can rather easily find steganographic algorithms for media such as text (the British government example mentioned before being a simple but efficient one), sounds and images.

2.3.1 *Some steganography examples*

The cases of images and sounds are particular, since they can be compressed in a lossy way. This creates additional domains in which steganography can be performed. For example, MP3 compression for sounds is lossy in that it removes from the original recording the parts that are considered to be beyond standard human auditory resolution. This loss in data makes data hiding possible, for example by modifying slightly the MP3 compression algorithm so that it embeds the message bits by changing pseudo-randomly some parity bits in the compression process [95, 115]. Works on non-lossy objects or algorithms also exist, for example on Wave files by Least-Significant Bit modification [127].

While not the most popular form, audio steganography techniques exist.

In the following, and throughout the dissertation, the focus will be on image steganography, and more precisely, on JPEG images, for which some explanatory details are proposed in the next section 2.4. Image steganography seems to be the most popular kind of steganography, certainly because of the large amount of available images, but also because the widely used JPEG lossy compression algorithm gives many liberties for data hiding.

We focus only on image steganography and algorithms for it.

2.3.2 *The two main parts of steganography*

Before going into the details of some classical steganography techniques, it is necessary to point out the two main parts of steganography (intertwined for most of the algorithms): *Model-based* and *Coding-based*. In the following, we will go through a few definitions of steganography concepts first, and then present the Model and Coding based parts.

One can separate the steganography ideas into two categories: Model-based and Coding-based.

Definitions

In the following, we will refer to the medium as *the image*, and the message to hide in it as *the message*. A genuine image will be called "cover" or *cover image*, while an image containing a message — we say that the message has been *embedded* in the image — will be named *stego image*. Moreover, the term *stego* will be used in place of *steganographic*, for example in the terminology *stego algorithm* for a steganographic algorithm.

The stego capacity is a simple concept: how much can be embedded while satisfying low distortion constraints?

THE STEGO CAPACITY Let us now define a widely used quantity: the stego capacity. While it can be empirically described as "the maximum amount of data that can be hidden in the cover [image] while satisfying a set of constraints on the distortion", the stego capacity remains a quantity that is difficult to estimate.

In [27], an investigation of the general capacity definition for a system with side information at the embedder (which is the case of steganography) is proposed in the more specific case of steganalysis. A result from [57] states that the capacity C can be expressed as

$$C = \max_{P(U, I_s|I_c)} \left(MI(U, I_s) - MI(U, I_c) \right), \tag{2.1}$$

Capacity can be expressed using Mutual Information and Entropy.

where $MI(\cdot,\cdot)$ denotes the mutual information, U an auxiliary random variable and I_c and I_s the random variables corresponding to the cover image (respectively stego). Using then the fact that

$$MI(U, I_s) - MI(U, I_c) = H(I_s) - H(I_s|U) - MI(I_c, U), \tag{2.2}$$

where $H(\cdot)$ refers to the entropy, it appears that

$$MI(U, I_s) - MI(U, I_c) \leqslant H(I_c), \tag{2.3}$$

under the perfect steganography constraint, that is, in terms of entropy

$$H(I_c) = H(I_s). \tag{2.4}$$

Hence, under the assumption of a *perfect* stego algorithm, the upper bound on the capacity C is the entropy of the cover image.

For the matter of an *imperfect stego* algorithm (ε-secure with $\varepsilon > 0$ for example, see section 3.1 for the definition of the ε-security), Filler *et al.* in [42] propose the Theorem of the Square Root Law of steganography for Markov covers.

Under the assumptions that the cover images can be modeled by a first order (stationary) Markov Chain, and that the embedding process can be modeled as an independent substitution of one state to another (the LSB embedding, nsF5, JSteg and MMx stego algorithms presented in 2.4.2 respect this hypothesis), the capacity for an imperfect stego algorithm follows three main propositions (the results are valid in the limit case $n \to \infty$, with n the number of elements to embed in, and the reader is referred to the original publication [42] for more thorough presentation of the problem and results):

Under some assumptions, the stego capacity follows a square root law [42].

1. Using a capacity smaller than $\sqrt{n}$ — the square root of the number of elements that can be modified to embed the message — the embedding

can be arbitrarily secure (in terms of ε-security, it *can be* up to ε-secure with $\varepsilon > 0$), given enough elements n to embed the message in;

2. Using a capacity larger than $\sqrt{n}$ leads to a non-secure situation where the steganographer risks detection;

3. Using a capacity of the order of $\sqrt{n}$ leads to a *possibly* secure embedding (again, in terms of ε-security, this means that the embedding is ε-secure with a fixed ε — no more arbitrarily small).

Hence, if the communication is to remain secure between the two parties, one wants to be in the first proposition case and be careful about the *embedding rate* that is used, *vis a vis* the *secure* capacity (in the sense of proposition 1).

THE EMBEDDING RATE This second definition concerns the widely used quantity *embedding rate*. It is meant to measure a ratio between the amount of data embedded and the specifics of the cover image. Hence, it is highly dependent on the image considered: for a given stego algorithm, some images can contain more information than others, while remaining just as undetectable.

Two possible definitions for the embedding rate.

There are many definitions for it, although one seems to prevail nowadays: the embedding rate $R^{(1)}$ is defined as the ratio between the *number of embedding changes* E and the *number of non-zero AC coefficients* (of the cover image) A:

$$R^{(1)} = \frac{E}{A}. \tag{2.5}$$

Related to the embedding changes and non-zero AC coefficients...

The concept of AC coefficients is inherent to the JPEG format (see section 2.4) and hence, this measure is mostly used for JPEG steganography matters.

Another possible definition of the embedding rate (used in one of the publications related to this dissertation) is using the embedding capacity of the algorithm. It is therefore also related to the stego algorithm directly. The embedding rate $R^{(2)}$ using the capacity is defined as the ratio between the size $S(\mathbf{m})$ of the embedded message $\mathbf{m}$ (usually measured in bits) and the total embedding capacity C_{Tot} (also measured in bits), defined previously:

$$R^{(2)} = \frac{S(\mathbf{m})}{C_{Tot}}. \tag{2.6}$$

...or to the message size and capacity.

Again, the first definition $R^{(1)}$ is mainly used nowadays. The second definition $R^{(2)}$ can be most useful when the stego algorithm does not provide information on the number of embedding changes, though.

In the specific case of LSB embedding [49] (or for steganography on raw images, generally), the embedding rate can also be defined in *bits per pixel* (bpp), which consists in dividing the total amount of bits embedded by the number of pixels of the image.

Finally, it is worth noting that in the following, we will consider *natural images* only, meaning that synthetic images (entirely produced by a 3D rendering software or drawings, e.g.) are not part of this analysis. Natural images are for example outdoor scenes snapshots taken by a camera.

Follows the description of the two main parts of a stego scheme: Model-based and Coding-based steganography.

Model-based Steganography

Model-based aims at modeling the distribution of media to find where to embed to minimize the distortion.

Model-based steganography as introduced by Sallee [111] makes use of (part of) the knowledge of the medium's instances distribution in order to hide a message. If we consider that there exists a random variable I with probability distribution P_I modeling the images, and that we take a single realization i of I (i is an image), we can separate i in two parts, i_α and i_β (which are instances of the random variables I_α and I_β). The i_α part will remain intact while the i_β will be modified or totally replaced by the actual message i_m. See Figure 4 for an illustration of the idea of Model-based steganography for the stego algorithm MBSteg (section 2.4.2).

The inherent idea is that given the knowledge of P_I (or a sufficiently good approximation of it $\hat{P}_I$), it is possible to find i_β such that the composite (i_α, i_m) is correctly distributed *vis a vis* P_I (or its approximation $\hat{P}_I$). Meaning that we can estimate the distribution for multiple possible I_β conditioned on the current I_α: $\hat{P}_{I_\alpha|I_\beta}(I_\beta|I_\alpha = i_\alpha)$. If a i_m that respects this distribution is found, the composite image (i_α, i_m) respects all the properties of a cover image and cannot be distinguished in any sense from the other cover images (again, provided that our model approximation $\hat{P}_I$ is good enough to approximate cover images).

One obvious problem of this approach is to model P_I. While it is clear that modeling the ensemble of natural images is impossible, it remains possible to have a simplified model of P_I and cut i into i_α and i_β such that modifying i_β will satisfy a set of constraints (for example not being visible to the human eye).

This part of steganography really aims at having sufficient information (or a sufficiently good model) of the whole space of media (cover images here) such that the way of tempering with a cover image will make sure that it still follows the distribution of cover images.

Coding-based Steganography

Coding-based tries to insert the message using a special encoding so as to minimize the distortion.

The goal of the coding-based part of steganography is different, although as said before, both approaches are used at the same time in most usable stego algorithms. Here, the emphasis is on the way to code the information — the message to embed — such that the tempering of the cover image will be minimal. A famous example of coding-based scheme is the *Matrix Embedding* (or *Syndrome Coding*) approach proposed by Crandall in [31].

We give a simple example of this idea for the insertion of $k = 2$ bits of message, denoted as m_1 and m_2. Assume there are $n = 3$ bits available for insertion in the cover image (these bits have been devised previously, for example using Model-based steganography), denoted as s_1, s_2, s_3. If one wants to insert the $k = 2$ bits of information by changing only one bit among s_1, s_2, s_3, we have four possible cases to consider:

- $m_1 = s_1 \oplus s_3, m_2 = s_2 \oplus s_3 \Longrightarrow \text{no changes}$
- $m_1 \neq s_1 \oplus s_3, m_2 = s_2 \oplus s_3 \Longrightarrow \text{change} : s_1$

- $m_1 = s_1 \oplus s_3, m_2 \neq s_2 \oplus s_3 \Longrightarrow change : s_2$
- $m_1 \neq s_1 \oplus s_3, m_2 \neq s_2 \oplus s_3 \Longrightarrow change : s_3$

Therefore, when inserting two message bits in the host data, only one bit will be modified. In the general case, matrix encoding enables to find a set of n bits in the host data where k message bits can be embedded with less than d_{max} actual modifications; this found solution is usually denoted by the triplet (d_{max}, n, k). The F5 stego algorithm detailed in section 2.4.2 makes use of the matrix embedding approach.

Another famous coding-based technique is the named Wet Paper codes [52, 54], proposed by Fridrich *et al.* The concept is based on the exact solving of the linear system

Wet paper codes give good secrecy and allow to define areas of the image where not to embed the message.

$$\mathbf{Db} = \mathbf{m} \tag{2.7}$$

where $\mathbf{D}$ is a binary matrix shared by both sender and receiver, $\mathbf{b}$ represents the modified cover image bits (to be determined) and $\mathbf{m}$ contains the message bits. The difficulty of the problem lies in solving such a system exactly (or the message bits from $\mathbf{m}$ would be altered). Provided that the system 2.7 has a solution, though, this scheme is very secure since the binary matrix $\mathbf{D}$ (which can be brought back to a single stego-key K initializing a pseudo-random generator) is supposed to be known by sender and receiver only. More details about the specifics of this scheme can be found in the original publication [52] (this short presentation is overly simplified).

In this context of coding-based steganography, a widely used measure of the efficiency of the scheme used (such as matrix embedding) is the *embedding efficiency*, first defined in [132]. This quantity is measured by the expected number of message bits (supposed to be random, i.e. not having any dependence with the modified quantities in the cover image) embedded per embedding change. The reader is refered to [55] for a thorough review of the embedding efficiency for various types of matrix embedding schemes.

The concept of security for a steganographic scheme will be discussed in the next chapter 3, once steganalysis has been introduced.

2.3.3 *A future development: Batch Steganography*

A possibly more realistic setup for steganography: many images to embed the message in.

In [76], Ker proposed to consider a point which is often left aside, in steganography: the steganographer (sender) will most likely have access to more than just one cover image, and he will try to use this as an advantage. Ker poses the question and lays the foundation for what is called *batch steganography*, that is, finding the best possible way of embedding a predefined message in a set of cover images (possibly in a subset of them, actually).

Given a set of assumptions

- the number N of cover images is fixed beforehand;
- all cover objects have the same capacity;
- the sender chooses randomly the cover images in which to embed,

it is shown that the steganography is the most secure when the message $\mathbf{m}$ is divided in a small number of portions (and hence, embedded in a small

Theoretical results on batch stegano are counter-intuitive...

number of images), which is a rather non-intuitive idea. One would *a priori* think that the "safest" solution is to divide **m** into as many small parts as possible, and make a very small number of embedding changes to many images in the set.

This part of the steganography field is rather recent and has still not been widely investigated. Most algorithms and techniques considered to be state of the art are still working on single-image cases.

The following section proposes a non-exhaustive overview of the most used steganography algorithms. They are all publicly available.

2.4 CURRENT STATE OF THE ART TECHNIQUES

In order to present some of the most used stego algorithms, we first introduce some notations and definitions, especially about the JPEG image compression algorithm.

2.4.1 *JPEG basics*

The 6 main steps of the JPEG compression.

The acronym JPEG stands for Joint Photographic Experts Group, a committee which created the JPEG compression algorithm [28]. By extension, the JPEG name is used for images using this compression method, but the original name refers to the compression part only.

The JPEG algorithm is mostly known for its efficiency when used in a lossy way, but depending on the compression rate, one can also use it for non-lossy means (its performance in terms of compression is then outperformed by other algorithms).

There are 6 main steps in the JPEG algorithm:

Divide the image into blocks;

1. Block splitting: The original (raw) image is divided into square blocks, typically of 64 pixels (8×8, but it can be different). The following steps are then applied to each block separately.

change the color-space;

2. Changing color-space: The RGB original color-space is changed to a YCbCr. The RGB color-space is coding for the three basis colors it uses: Red, Green and Blue. The YCbCr color-space uses a luminance component (Y) and only two chroma components (Cb for blue-difference and Cr for red-difference).

sub-sample the colors;

3. Sub-sampling of chromas: The gain from the JPEG compression comes in part from this step. Since the human eye is more sensitive to luminance than to chromas, the Y component is left untouched while the sub-sampling is performed on the chromas.

DCT-transform the coefficients;

4. Discrete Cosine Transform: Each component (Y, Cb and Cr) of each block goes through a discrete cosine transform. For an 8×8 block denoted as $\{i(x,y)\}_{1 \leqslant x,y \leqslant 8}$, the DCT coefficients $\{I(u,v)\}_{1 \leqslant u,v \leqslant 8}$ are computed as

$$I(u,v) = \frac{1}{4} C(u) C(v) \left[\sum_{x=0}^{7} \sum_{y=0}^{7} i(x,y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \right],$$

(2.8)

with $C(u) = C(v) = 1/\sqrt{2}$ if $u = v = 0$ and $C(u) = C(v) = 1$ otherwise. The main idea behind this transformation is to get a large number of zero coefficients for the final coding.

quantize the coefficients;

5. Quantization: Using fixed quantization matrices (different for each channel), the low frequencies are preserved, while the high frequencies are getting close to zero (or are zeros). The rationale is again that the human eye is more sensitive to low frequencies than high ones.

order and compress.

6. Zig-Zag ordering and final lossless compression: These steps only aim at having an optimal way of compressing the remaining data (most of the coefficients of each block are zero) so that the final size is minimal.

Note that in the rest of this dissertation, we will mostly ignore the last step of this procedure, and consider that the JPEG image is a large array of DCT coefficients placed at the exact same place than the part of the image they are representing.

As discussed before, the lossy aspect of this algorithm makes it interesting for hiding data. Indeed, since most of the DCT coefficients are zeros or very small values, it seems likely that modifying the Least Significant Bit (LSB) of some coefficients will be visually (to the human eye, that is) undetectable. By modifying such bits (chosen pseudo-randomly thanks to a shared stego-key, e.g.), one can embed a message and avoid visual detection. This is the basis for LSB steganography. For large enough payloads, though, this becomes very easily detectable by statistical means [49, 133].

Modifying the LSB of DCT coeffs is a simple stego scheme eventually detectable visually.

Recent stego algorithms try to address not only the visual aspect of the distortion created by the embedding, but also many statistical aspects of the image (histograms of AC coefficients for example, that is, the low-value coefficients of the DCT-transformed matrix of values).

2.4.2 *A non-exhaustive overview of Stego algorithms*

The proposed overview of stego algorithms contains most of the considered state-of-the-art methods. It is by no means exhaustive and only aims at presenting the concepts of these algorithms so that future discussions on their behavior are made easier.

LSB replacement/ matching

As the name implies, the early schemes of LSB replacement [114] are embedding the information by replacing the LSB of an image's bytes directly (first separating the RGB components of the image), either in a completely sequential way, or by choosing the bytes to modify pseudo-randomly, thanks to a stego-key, for example.

LSB matching takes advantage of the LSB value.

The LSB matching (also called ± 1 embedding) is working in the same way, except that it compares the bit of the message to embed in a LSB and modifies it — by a random ± 1 modification on the LSB, except if it leads to a zero value — only if necessary (i.e. the message bit to embed and the LSB do not match).

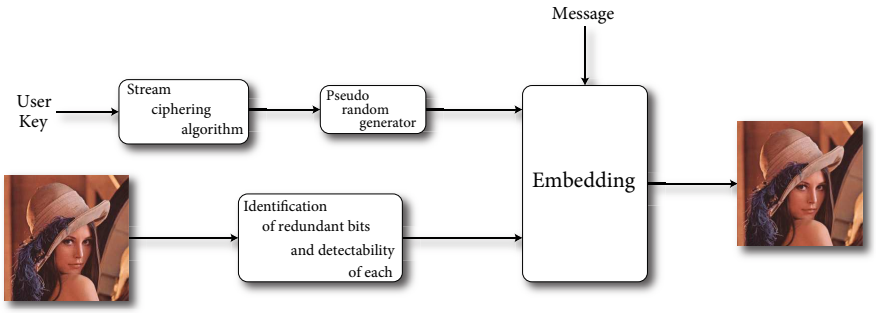

Figure 2: Schematic concept of the Outguess algorithm.

Outguess tries to avoid statistical attacks by "canceling" the changes caused by the embedding.

Outguess

Niels Provos in [106] proposes the Outguess algorithm, supposed to resist especially well to statistical attacks. The process has two main parts, detailed in the following and summarized on Figure 2.

- Identify the LSBs: these are chosen only among the LSB of DCT coefficients which are different from 0 and 1. This choice is made because most of the coefficients have 0 and 1 values and modifying them would result in a too visible change in the global histogram and be easily detectable. As some image parts may contain more information than the others, the LSBs are marked with their potential detectability. This enables the algorithm not to use these for message embedding (if possible).
- Select the LSBs to use. About half of them will be used to actually embed the message information, while the remaining half will be used to correct the statistical deviations created by the embedding.

Outguess embeds in half the identified LSBs and corrects the distortions using the other half.

First a stream cipher is initialized with a user key. A pseudo-random number generator initialization (first seed) is derived from this cipher. The algorithm hides another pseudo-random generator initialization state which will enable to recover the location of the modified LSBs, and the size of the embedded message.

After the seed and length of the message are embedded, the algorithm hides the message itself. In this embedding process, the algorithm tries to dispatch the message bits as much as possible in the image, by adapting the random values to the size of the message and of the available size in the embedding image.

During this process, a value of total detectability is incremented with the value of detectability of each used LSB, determined by a heuristic. The algorithm uses many try-outs and keeps the lowest possible total detectability value.

A correcting transform is then applied on the remaining LSBs (the other part of the redundant) to try to preserve as much as possible the statistics.

Retrieving of the message requires only the user key, to initialize the pseudo-random number generator and then obtain the second seed and length of the message from the image.

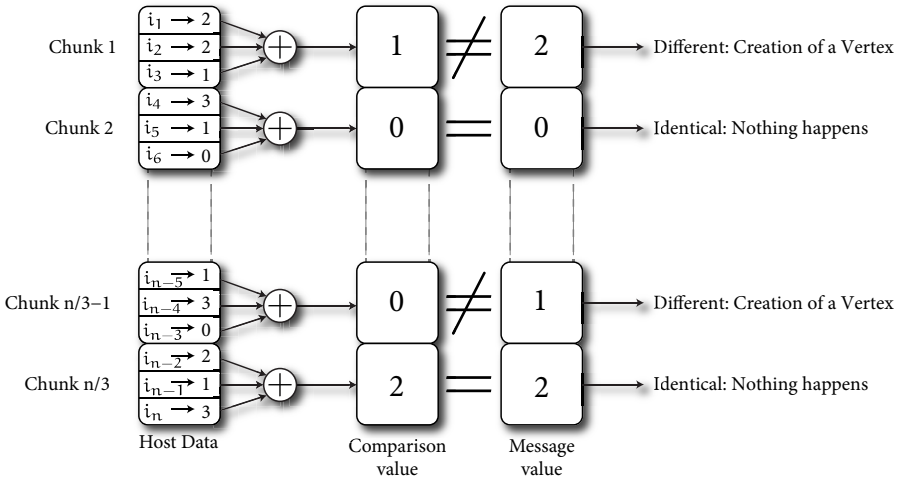

Figure 3: An overview of the vertices creation process for the Steghide algorithm. Sizes of chunk and modulo values are parameters of the algorithm and are respectively chosen as 3 and 4 in this example (arbitrary).

Steghide

Steghide uses sample flipping governed by an informed graph.

In [65], Stefan Hetzl and Petra Mutzel describe a graph-based algorithm for steganography which implementation is called Steghide. The embedding process is in fact mostly a sample flipping process, governed by a graph created while evaluating the samples of the host data and comparing them to the message to embed.

The host data is first abstracted as a set of samples (pixels for the case of images). Sets of samples will then be "evaluated" and compared to the data to embed. From the need to modify a sample of the host data in order to embed the message, a vertex is created. A similarity measure then enables to find potential samples in the host data to be exchanged with the one considered for embedding, and an edge is then created between these samples.

An example of the graph building in Steghide on Fig. 3.

As shown on Figure 3, host data (cover image) is divided into chunks of samples (i_j with three samples per chunk, in this example) for which a Comparison Value is calculated through addition modulo 4 in this case (these values of three samples per chunk and addition modulo 4 are algorithm parameters that are optimized for the considered embedding). The obtained values are compared with the message values. If different, a vertex is created for this pair host data chunk/message chunk. Parameters of the vertex are obtained from simple calculations on the chunk values.

Then, edges are created between the vertices, based on the similarity of the considered two vertices, for the actual embedding of the message is done through flipping of vertices, as mentioned above.

F5

F5 takes a permutation method and the matrix encoding scheme.

F5 has been proposed by Andreas Westfeld in [132] and claims to have an increased robustness compared to its predecessors, F3 and F4, as well as a much higher embedding rate.

It is based on two main ideas:

- A permutation method, allowing to scatter the embedded bits through all of the image instead of having the changes only in the beginning;
- Matrix encoding, imagined by Ron Crandall [31], enables to increase the embedding rate while minimizing the number of changes.

The F5 algorithm is largely based on matrix encoding; it uses $d_{max} = 1$ and is looking for the best k in order to insert the whole message. A main point about F5 is that it never directly modifies the LSB of DCT coefficients, but only decreases their values (except the ones that are already zero), thanks to the matrix encoding part. The so-called *shrinkage* effect happening in F5 is due to LSB decrease creating a zero value. Since the algorithm does not read (and does not use) the zero DCT coefficients, the information has to be embedded again in another LSB. This effect creates detectable changes in the histograms of the DCT coefficients.

-F5, nsF5,...

-F5 and nsF5 try to eliminate the shrinkage effect happening with F5.

The -F5 and nsF5 (for *no-shrinkage F5*) are derivations of the original F5 algorithm, which address some of its drawbacks. The *shrinkage* effect happens in F5 when the modification of the LSB by F5 leads to a zero. In [56], Fridrich *et al.* propose to alleviate this effect by increasing the value of the LSB, instead of decreasing it. This is called the -F5 algorithm.

The nsF5 algorithm makes use of the wet paper codes previously discussed. By doing so, it avoids putting LSBs of AC coefficients to zero without coding any information (the decoding part does not look for information in the zero LSBs), and hence, avoids the shrinkage effect. To describe this using the notations from 2.3.2, assume that the message $\mathbf{m}$ to embed is k bits long, and that there are n LSBs of AC coefficients (which have value either 0 or 1), out of which n_u are usable for modification (i.e. non-zero). The problem is then to find the vector of modified LSBs $\mathbf{b}$ such that the system

$$\mathbf{Db} = \mathbf{m}, \tag{2.9}$$

is satisfied, with $\mathbf{D}$ still being a pseudo-random binary matrix shared by both sender and receiver (via a secret key initializing a pseudo-random generator, for example).

MBSteg

MBSteg tries to model statistics of the image and keep them intact.

Sallee in [111] proposed the Model Based steganography, already presented in section 2.3.2. It tries to "adapt" the message to a part of the cover image, by entropy decoding. Figure 4 inspired from the original publication, illustrates the process for JPEG images.

The original cover image AC DCT coefficients are separated into i_α and i_β. i_α will be kept intact and serves to model the histograms of the individual AC DCT modes. The model $\hat{P}_{I_\alpha|I_\beta}$ is fed to an entropy decoder along with the encrypted/compressed message (so that it is close enough to random data). The result of this step is i_m which respects the cover image structure (thanks to the model) while embedding the message.

The interesting direct consequence from this scheme is that the histograms of individual AC DCT modes should be well preserved, along with the

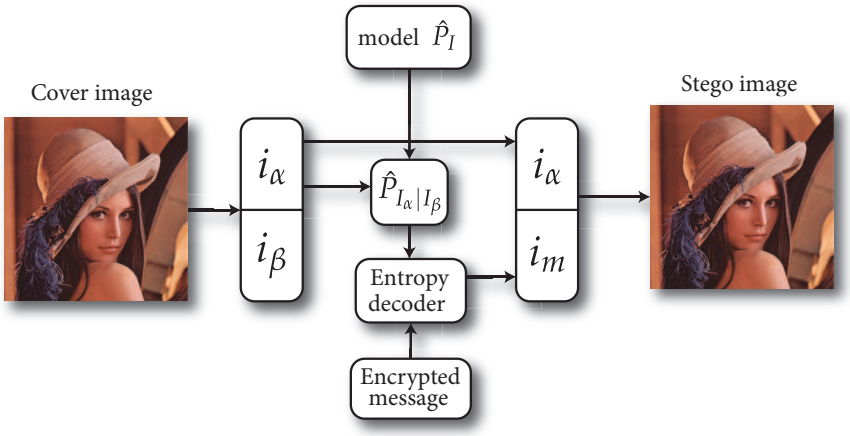

Figure 4: The MBSteg Algorithm.

global histogram, due to the "adaptation" through entropy decoding, of the message to embed.

MMx

MMx is a variation of the matrix encoding.

MMx — standing for Modified Matrix Encoding — is a variation of the matrix encoding algorithm (implemented in F5 by Westfeld [132]) proposed by Kim *et al.* in [79].

While F5 makes use of the triplet (d, n, k) with a fixed $d = 1$ for the matrix encoding, MMx uses a (d', n, k) triplet, with d' usually 2 or 3: this change makes it possible to have more than one bit change per block since d' modifications of the set of n bits (meant for embedding k message bits) are allowed for the matrix encoding.

JPHS

JPHS concept is not disclosed but seems to use LSB flippings.

The JPHide&Seek (also denoted JPHS) algorithm by Latham [84] is available on the web as binaries and source code, although its details have not been published currently. From experiments performed in [56], it seems that it mostly relies on LSBs flippings.

JSteg

JSteg is among the first stegano algorithms for JPEG.

JSteg by Upham [128] employs also LSB modifications, on quantized DCT coefficients. It is among the very first steganography algorithms and simply embeds the message bits directly into LSBs that are different from 0 and 1. The LSBs used for changes were at first chosen in a sequential fashion, but now use a pseudo-random path.

YASS

YASS uses a de-synchronization from the JPEG grid to embed the message.

Yet Another Steganography Scheme (YASS) has been proposed recently by Solanki *et al.* [119]. Its fame came from the fact that it was hardly detectable by any known means, at the time. Since then, many papers have shown that

it is detectable, for example by adapting the features used for the detection scheme (using un-calibrated features, see section 3.3.4). Let us describe shortly the concept of YASS.

The global idea is to embed the data in the spatial domain (before the actual JPEG compression) using error correction codes, to resist the JPEG compression. The de-synchronization *vis a vis* the typical 8×8 JPEG grid helps in hiding the message also. Five main steps describe the YASS (from the original paper [119]):

1. Coding of the message: using a repeat-accumulate code [35], to resist the JPEG compression.
2. Division into blocks: the cover image is divided into blocks of size $B \times B$, with $B > 8$, if 8 is the size of the JPEG blocks. This is for de-synchronizing the message embedding *vis a vis* the JPEG grid.
3. For each block (of size $B \times B$), a sub-block of size 8×8 is chosen (pseudo-randomly, with a secret key shared by the sender and receiver).
4. For each of the sub-blocks (of size 8×8), the 2D DCT transform is computed (as for the JPEG compression step 4 in 2.4) and divided by a quantization matrix for a specific quality factor QF_h. The actual message (encoded) is embedded in a predetermined band of low frequency AC coefficients with Quantization Index Modulation (QIM) [26].
5. The sub-blocks are then brought back to the spatial domain before the full image (which contains the message now) is finally compressed to JPEG.

2.5 CONCLUSION

In this chapter, we have described some of the concepts and notations used in the steganography field. Typically, stego algorithms use one of the two approaches mentioned — and sometimes both for improved security: the *model-based* steganography approach, in which one tries to estimate the global distribution of the media (images in this dissertation) and attempts to embed a message in a way that the resulting medium still belongs to the global distribution; and the *coding-based* approach where the emphasis is put on the manner of coding the message to embed such that the distortions caused to the medium by embedding are minimal.

We have then shortly described a promising extension of the typical steganography use: the batch steganography [76], in which one attempts to hide the message within a set of images, instead of just one.

We reviewed then the basic concept of the currently most used algorithms, in steganography, some of then used in the publications included in this dissertation (see publications C, D, E and F). Currently, all of these algorithms are well detected (for reasonable message sizes, at least) by the use of various sets of features extracted from the stego images.

The process of detecting whether an image is genuine (cover) or stego is called *steganalysis* and is described in more details in the next chapter.

3 STEGANALYSIS

Here we define the counterpart to steganography, *steganalysis*. We first discuss the concept of *security* in steganography/ steganalysis, in a theoretical way [23], which is unfortunately impossible to implement in reality at the moment. The current framework in which steganalysis is inscribed, based on an *empirical estimation* of the security through a specific setup, is discussed. We finally present the different possible *classes* of steganalysis with some of the classical steganalysis schemes used at the moment.

3.1 WHAT IS STEGANALYSIS

Before digital media appeared, the concept of *steganalysis* did not really exist. For the steganography example (chapter 2) of the tattoo on the head of Histiaeus' slave, steganalysis would have consisted in looking at the slave under all possible angles and try to guess (by the looks of it) whether there could be a tattoo hidden on his scalp.

Steganalaysis is not like cryptanalysis: only detection is at stake here.

The concept of steganalysis is again very different from that of cryptanalysis (as steganography differs from cryptography): in cryptanalysis, the aim is to "break the code" and then get the encoded message, simply put. Steganalysis does not aim at obtaining the message hidden in the cover medium, but only at detecting the mere presence of it. The original goal of steganalysis was hence to give a binary answer to the question "Is there a message hidden in this medium ?". Figure 5 illustrates this simple idea.

Later on with the coming of digital media and the growing importance of steganography, as discussed previously, the means of detecting steganography have been developed along with other kinds of steganalysis. For example one can cite the search for the stego-key used to embed the message, which relates closely to cryptanalysis [51]. Before proceeding with some of the different kinds of steganalysis that exist, let us remind why it is not an obvious task at all.

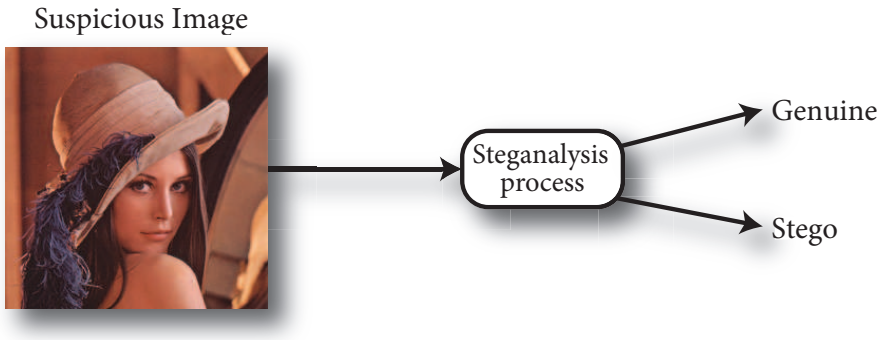

Figure 5: The classical steganalysis process: a suspicious image is processed by means of steganalysis to devise it genuine or stego (tampered).

3.1.1 *Kerckhoffs' principle*

As for cryptography, the Kerckhoffs principle can be applied to steganography. In [77], Kerckhoffs stipulates in substance, that a cryptographic algorithm should be able to withstand its principle being made public. That is, the only secret that can be considered acceptable is the secret key shared between sender and receiver. This key acts as a parameter of the cryptographic algorithm.

Steganalysis works on supposeably known stego algorithms.

The idea is the same for steganography: only the stego-key shared by sender and receiver can be considered as secret. The idea lying behind such a strong design principle is that a brute force attack — trying out all possible stego-keys to find the good one — against the stego scheme (to determine the presence of the message or not) would be infeasible in terms of time spent. If the space of stego-keys is large enough (and the stego algorithm has no obvious weaknesses), this attack has very low probability to succeed [51], due to the complexity of the search problem.

In this setup of a perfect stego algorithm (no weaknesses) and a large enough stego-key space, the stego algorithm would be considered as secure. We detail this concept of security for a stego algorithm in the following, as it is of the utmost interest when considering steganalysis.

3.1.2 *A definition of security for steganography*

Security in steganography is theoretically defined...

In the previous chapter 2 introducing steganography, we have brushed the problem of embedding a message in a cover image while minimizing the amount of distortions caused to it (see the section 2.3.2 about model-based and coding-based schemes, for example). The concept of security in steganography/steganalysis is related to this matter of the amount of distortions.

The following concept of security makes the assumption that the transmission channel does not introduce any distortion to the communicated content. This also means that a potential eavesdropper will not attempt to modify the content he monitors on the channel. This assumption on the eavesdropper is often refered to as *passive warden* (from the famous problem of the communication between two parties, typically Bob and Alice, and the warden or eavesdropper, Eve). The other approach consists in considering an active warden, who has the ability to modify the transmission and therefore send virtually any content to the other parties communicating. We do not consider this possibility in the rest of this dissertation.

With this assumption, Cachin in [23] defines the security of a stego algorithm as the amount of differences that exist between the distributions P_{I_c} and P_{I_s} of the two random variables I_c and I_s (respectively the random variable corresponding to the *cover image* and to the *stego image*) for an instance i. The sample space of the cover images for the random variable I_c is denoted $\mathbb{I}$. The "difference" between the distributions P_{I_c} and P_{I_s} is measured using a Kullback-Leibler (KL) divergence δ such that

...using a KL divergence between image distributions...

$$\delta(P_{I_c}, P_{I_s}) = \sum_{i \in I_c} P_{I_c}(i) \log \frac{P_{I_c}(i)}{P_{I_s}(i)}, \tag{3.1}$$

which is a measure of the relative entropy between the two distributions. With this definition, Cachin defines a stego algorithm S as ε-*secure* if $\delta(P_{I_c}, P_{I_s}) \leqslant \varepsilon$, with I_s being the random variable corresponding to the stego images from algorithm S. Therefore, the smaller the ε, the closer are the two distributions P_{I_c}, P_{I_s} and the harder it is to distinguish the suspicious image from a genuine one.

In the case where $\varepsilon = 0$, the stego-algorithm is said to be *secure*, since no difference can be made between the suspicious image and the genuine ones.

. . . but that definition is not practically usable.

This definition poses unfortunately many problems in practice. First, the size of the set of cover images $|\mathbb{I}|$ ($|\mathbb{I}|$ denoting the cardinal of the sample space of the variable I_c) is potentially infinite; it suffices to take a new picture with a camera to make it larger. Therefore, computing the KL divergence over the whole set of cover images is computationally infeasible.

Second, this definition requires estimating the distributions P_{I_c} and P_{I_s}, which is most likely impossible to perform, for example because of the nature of I_c.

These problems can be alleviated by the use of a set of features extracted from the images, which dramatically reduces the dimensionality of the problem and makes the marginals P_{I_c} and P_{I_s} possible to model.

Third, as pointed out by Pevnỳ in [97], there are no sufficiently good estimators of the KL divergence to perform the computations. Indeed, even though the size of the space of cover images $|\mathbb{I}|$ can be reduced, the KL divergence estimators (for example the Kraskov one based on k-Nearest Neighbors [83]) work properly for problems with dimensionality below 5.

MMD is a possible alternative to the KL divergence.

A possible solution is proposed in [101] by the use of the Maximum Mean Discrepancy (MMD), which measures the differences between two probability distributions by drawing samples from both and using a *witness function* behaving differently for samples of each distribution. The MMD is then obtained as the difference between the mean witness function values on samples of each distribution. The main advantage over the KL divergence being that the MMD behaves well in high-dimensional spaces (large number of features).

In order to assess the security (and thus the "quality") of a stego algorithm, practical steganalysis benchmarks are devised, through the use of features, for example. Some of the features used widely in steganalysis benchmarks are presented in section 3.3.

3.1.3 *Measuring security empirically: benchmarking*

We need to have practical benchmarking to assess the security of stego algorithms.

Setting up a practical steganalysis benchmark requires many choices in design, choices on which the benchmark will possibly highly depend, unfortunately.

RESTRICTING THE IMAGE SPACE First, since it is impossible to cover the set of all possible images, one has to restrict the space of cover images to a finite set. The choice of this set (both in terms of content and size of the set) might influence highly the results of the steganalysis, as demonstrated for example in chapter 6 and Publication E (section 4.2.1).

It is not possible to have all the existing images. . .

		True Value		
		True Positive (TP)	False Positive (FP)	N'_P
Prediction		False Negative (FN)	True Negative (TN)	N'_N
		N_P	N_N	

Table 1: Confusion matrix for a binary classification problem.

. . . nor to model images perfectly: we use features.

EXTRACTING FEATURES Second, the choice of the *characteristic features* (refered to as *features* in the following) extracted from each image. The goal of obtaining such features is to reduce the dimensionality of the space to analyze. Although extracting features from an image i is a destructive process (in that the projection from the image space to a lower dimensional feature space is destructive in terms of information), one can hope that the extracted features are descriptive enough that the most important information relevant to steganalysis is preserved.

MODEL A third parameter is the *model* chosen in order to discriminate between stego and genuine, based on the previously devised features. In most cases, a supervised machine learning model is used. Some of these models are presented in the next chapter 4.

How to measure the security through the benchmark?

PERFORMANCE MEASURE Finally, it is necessary to decide on what performance to report. The most common measure of the security of a stego algorithm is given by the ratio between the number $N_{\texttt{ident}}$ of correctly identified images (as being either stego or cover) and the total number N of images in the defined set. This is a particular measure known as the *accuracy*, which can be derived from the *confusion matrix* obtained after a binary classification problem (stego or cover). The classes of problems for machine learning models are described in more details in chapter 4. Let us describe the other possible practical measures of security in the framework of a steganalysis benchmark, of which the Receiver Operating Characteristic (ROC) curve is probably a good example.

The steganalysis problem can be assimilated to a binary classification, for which the outcome *stego* is the positive class and the *cover* outcome the negative class. Consider also that the set of $N = N_P + N_N$ images contains N_P instances of the positive class (stego) and N_N of the negative class (cover).

Denoting by N'_P and N'_N the number of positive instances (resp. negative) as classified by the model, the confusion matrix from Table 1 summarizes the results.

The accuracy is a proper measure of the performance. . .

The accuracy is therefore defined as $\frac{TP+TN}{N_P+N_N}$. Unfortunately, the accuracy does not account for the amount of false positives (or negatives) which can be of great importance in the steganalysis framework: one might prefer catching too many potential steganographers than missing them, for example.

The ROC curve plots the True Positive Rate $\frac{TP}{N_P}$ versus the False Positive Rate $\frac{FP}{N_N}$. Figure 6 gives an example of a ROC curve: the solid line indicates

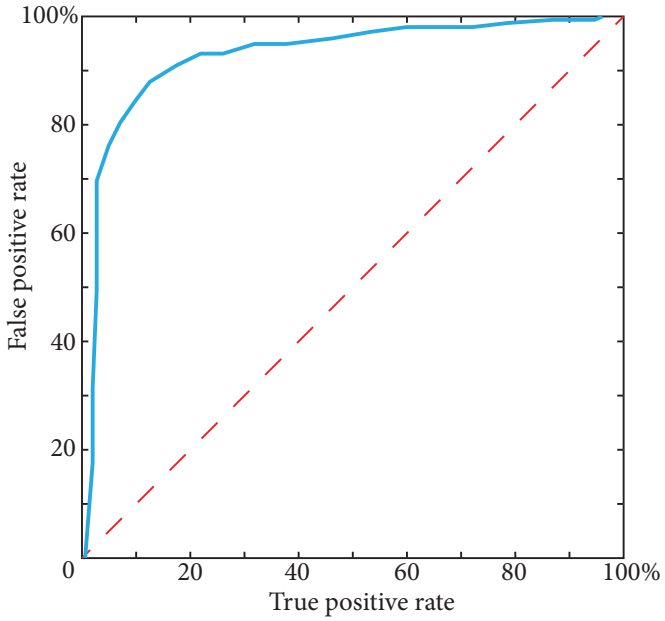

Figure 6: An example of ROC curve: the solid line represents the performance obtained using a model in a steganalysis benchmark for a stego algorithm S, while the dashed line is equivalent to random guess.

the evolution of the true positive rate when the false positive rate varies. For some applications, it is preferable to sacrifice the False Positive rate to obtain a better True Negative rate: it might be better for authorities to catch too many potential steganographers than let some of them go, for example. Some classification models allow this setting.

. . . although the ROC curve is more informative.

While the ROC curve and/or the confusion matrix describe in much more detail the outcome of the classification task in a steganalysis benchmark, the accuracy remains the most commonly used measure, which we will use in the following of the dissertation.

Now that the different parameters necessary for a typical steganalysis benchmark have been presented, let us introduce the different variations on steganalysis that currently exist. In the following, we will refer to a *steganalysis class* to describe the general type of that steganalysis, and to a *steganalysis scheme* for the practical way to perform the steganalysis (most of the time, by extracting specific information in the form of *features* from the suspicious image).

3.2 DIFFERENT CLASSES OF STEGANALYSIS

The original problem of steganalysis now has subdivisions.

Although the primary steganalysis goal is to detect the mere presence of a message in a suspicious medium, the field has evolved towards some refinements, derived from the original idea. The original form (the binary classification between cover and stego) could be qualified of *qualitative* steganalysis, although this terminology is not really used. Follow four other types of steganalysis, which have rather different goals than the qualitative

steganalysis. The next section 3.3 describes the practical *schemes* that can be applied to the following *classes*.

3.2.1 *Targeted steganalysis*

Here, the used stego scheme is known, following Kerckhoff's principle. This is a strong assumption.

The hypothesis of the targeted steganalysis is a strong one: the steganographic algorithm S used in known. This information is an important insight about how the steganalysis process should be designed. As discussed in the previous chapter, for example, simple stego algorithms changing directly the LSBs of DCT coefficients to embed the message (such as JSteg) will affect visibly the histograms of such coefficients (again, given that the message to embed is of sufficient size).

Most steganalysis schemes actually derive from this class: once a stego algorithm is known (assuming its functioning is made public also), a steganalysis scheme can be derived from the way the information is embedded, as for the example of the LSBs of DCT coefficients previously [48, 80].

3.2.2 *Blind steganalysis*

Blind steganalysis tries to infer the stego scheme used, only.

Also called *universal steganalysis*, it is the exact opposite concept to that of targeted steganalysis. This steganalysis also called *universal* aims at detecting any kind of steganographic algorithm. See for example [100] for a recent detector of most JPEG steganography schemes.

This concept was introduced in [9], and more recently, a specific steganalysis scheme made blind steganalysis much easier [46]. This scheme is described at length in 3.3. The work by Fridrich et al. in [98] uses this scheme for the elaboration of a blind steganalyzer (meant for JPEG images only) with high efficiency for the JPHide, F5, MBsteg and Outguess stego algorithms.

3.2.3 *Quantitative steganalysis*

This class of steganalysis estimates the length of the message.

The quantitative steganalysis approach differs again from the original qualitative steganalysis in that it predicts the length of the message that has been hidden in the cover medium. This is a very different problem from the classical binary classification one (cover or stego). The differences in terms of models used to perform quantitative steganalysis are detailed in chapter 4.

Quantitative steganalysis has been first introduced by Chandramouli [24], but rather few works have actually followed on this concept. In [50] a histogram of DCT coefficients approach is used, while the specific scheme meant for blind steganalysis [99] is used in [103] for this quantitative task and performs very well.

In the last publication present in this dissertation, publication F, this very same scheme is used through a particular methodology for quantitative steganalysis (see chapter 7).

3.2.4 *Forensic steganalysis*

Finally, the forensic steganalysis [53, 30] goes beyond the detection step of the classical steganalysis: obtaining the actual hidden message. There are

many possible reasons for which the eavesdropper would like to obtain the message. The most obvious one is if such an eavesdropper has control over the transmission channel and can decide of shutting it down if steganography has been detected. Instead of arousing suspicions from both sender and receiver by merely cutting the transmission channel, the eavesdropper might want to obtain the hidden message so that he knows exactly what is exchanged, without catching the attention of the communicating parties.

The most wide steganalysis class: find out everything about the message (presence, length, content...).

In this sense, forensic steganalysis is closer to cryptanalysis than steganalysis is. This specific part of steganalysis is not covered here, since it is rather different from the usual goals of steganalysis. Most likely, though, one of the very first steps a forensic steganalyzer would take is universal steganalysis, to determine which stego algorithm he is dealing with in the first place. If the identification of the stego algorithm works, it becomes either a matter of breaking the scheme by trying all possible stego-keys (if the stego algorithm requires one, this is the brute force approach), or use a weakness of the algorithm to obtain the message. Using quantitative steganalysis to have information about the length of the hidden message also gives important clues for obtaining it.

It possibly also relates to cryptanalysis.

Overall, forensic steganalysis makes use of the many other aspects of steganalysis, and possibly of some of the cryptanalysis ones.

3.3 PERFORMING STEGANALYSIS: SCHEMES

In order to perform in any of the classes of steganalysis presented, there is a need for a practical *scheme* on how to actually do it: visually, by analyzing the inner structure of the image, the statistics of the JPEG DCT coefficients... Follows a presentation of historical schemes along with some of the most recent and relevant ones, for the tasks of targeted, blind and quantitative steganalysis.

3.3.1 *Visual detection*

The title of this type of steganalysis is quite explicit: the human eye is here the model for classification. An image having visible discrepancies making it look suspicious or tampered, will be classified as stego. This simple form of steganalysis is still efficient for very simple stego algorithms, for example embedding the message by modifying largely the LSB of the DCT coefficients, for a JPEG image. The first version of JSteg (see 2.4.2) with a sufficiently large message would most likely create artefacts in the image that are visible to the human eye.

Use the human eye as a detector of discrepancies.

In order to make invisible changes — invisible to the human eye, that is — appear, Westfeld in [133] proposes to first filter a suspicious image (with filters specific to each stego algorithm). The resulting filtered image has a more obvious structure and renders the changes created by the embedding of the message visible. Figure 7 from [133] illustrates visually this concept. The left part shows the cover image (no message) after the filter (specific to the stego algorithm) and the right one the same image with a message embedded, also filtered. The visual difference is clear.

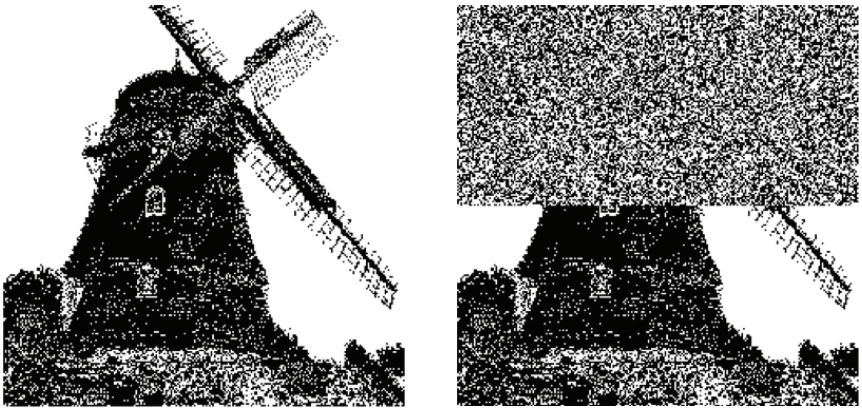

Figure 7: An example of the visual detection using a filter specific to the stego algorithm to reveal the modifications made to the cover image. Left is the genuine cover image, filtered and right is the same image with an embedded message, also filtered. From [133].

As mentioned in the previous chapter, though, most current algorithms do not create any visual discrepancies and are thus impervious to visual detection. Hence, one of the following statistics based steganalysis schemes should be preferred in the general case, to the simple visual detection, in order to obtain accurate detection. It should be noted, though, that careless steganographers might cause visual distortions while using recent steganography means, if the embedded message is too large, for example. In such cases, visual detection can at least alert the steganalyzer.

3.3.2 *First-order statistics based steganalysis*

The problem of visual detection is that it relies on the human eye and on the quality of the filter used, to identify steganography; as mentions Westfeld in [133], this approach fails for elaborate enough schemes. In addition, for cases not as obvious as the one in Figure 7, the reliability of the human eye can be controversial. The need for clean, reliable statistics on which to base the steganalysis decision initiated a trend in what is called *feature-based steganalysis*.

This scheme uses first order statistics of the image to discriminate...

Feature-based steganalysis, of which all following schemes are instances, relies on the extraction of certain characteristics of the suspicious image, followed usually by a comparison with the characteristics of a base of cover images. The base of cover images enables to obtain a "ground truth" about the cover images, in terms of the characteristics considered. The point is not to learn "what a cover image looks like", but more precisely "what a cover image looks like in terms of the characteristics chosen".

...for example histograms of pixel values.

This first scheme of steganalysis uses mainly histograms of pixel values, and is therefore named *Histogram attack*. As described before, in the case of LSB embedding, the LSBs are directly modified to embed the message. Consider that the image is in grayscale format, i.e. the values of its pixels range from 0 to 255 (8 bits to code the grayscale). In the case of LSB embedding, the pixels with an odd value are decreased (or unmodified if it leads to a

zero) and the pixels with an even value are increased (or unmodified if it leads to a 1).

In this setup, an even pixel value and the following odd pixel value $(2j, 2j+1)$ are flipped during the embedding of the message bits. Such a pair is called *Pair of Values* (PoV) and the whole histogram attack relies on these.

Denote by $H_c = (H_c^0, \dots, H_c^{255})^T$ the histogram values for all pixel values of the cover image I_c (which has N pixels), and by H_s the same histogram, for stego image I_s. Denote by k the number of bits of the message $\mathbf{m}$ that was embedded in I_s. Assuming that the message is composed of random bits (which can be achieved, by compressing it e.g.), it is possible to compute the expected number $E\left[H_s^{2j}\right]$ of pixels with value $2j$ in I_s,

$$E\left[H_s^{2j}\right] = H_c^{2j} - \frac{1}{2}\frac{k}{N}H_c^{2j} + \frac{1}{2}\frac{k}{N}H_c^{2j+1}, \tag{3.2}$$

since on average $\frac{k}{2}$ bits already have the proper value *vis a vis* the message $\mathbf{m}$ bits. The value for H_s^{2j+1} is obtained in the same fashion

In the case of embedding at full capacity, the steganalysis test with histograms is straightforward...

$$E\left[H_s^{2j+1}\right] = H_c^{2j+1} - \frac{1}{2}\frac{k}{N}H_c^{2j+1} + \frac{1}{2}\frac{k}{N}H_c^{2j}. \tag{3.3}$$

Then, in the obvious case where $k = N$, that is a message bit has been embedded in each pixel value, we have $E\left[H_s^{2j}\right] = E\left[H_s^{2j+1}\right]$, which gives $2E\left[H_s^{2j}\right] = H_s^{2j} + H_s^{2j+1}$ from Equations 3.2 and 3.3 and it becomes easy in this case to just test if $H_s^{2j} = H_s^{2j+1}$ to see if a message is embedded.

In less obvious cases ($k \neq N$), the use of a χ^2-test, for example Pearson's, permits by the computation of the p-value for the χ^2-test to check whether an image is stego or not, and to estimate the length of the message. By first calculating the χ^2 statistic X^2 as

$$X^2 = \sum_{j=1}^{d} \frac{\left(H_s^{2j} - E\left[H_s^{2j}\right]\right)^2}{E\left[H_s^{2j}\right]}, \tag{3.4}$$

with $d - 1 = 127$ degrees of freedom ($H_s = \left(H_s^0, \dots, H_s^{255}\right)^T$). From the previous analysis ($k = N$), the behaviour is here similar: a large value of X^2 means that the actual value of H_s^{2j} does not follow the expected value $E\left[H_s^{2j}\right]$ and therefore, that there is no message embedded. The reciprocal goes for a small value of X^2. A threshold has to be set on X^2 to make the final decision on the image being stego or cover.

Otherwise the use of a χ^2-test permits to perform it reliably.

The length of the message can then be determined by analyzing the statistical significance of X^2, by its p-value. As pointed out in [30], this only works if the "path" used for modifying the LSBs (i.e. the order in which they have been modified), is known. Since some algorithms were performing the LSB modifications in a predefined order, the following analysis can work. If the path is known only to the sender and receiver (by the use of a stego-key), the use of the p-value for the determination of the message length is not possible.

This approach enables to estimate the message length, with some assumptions.

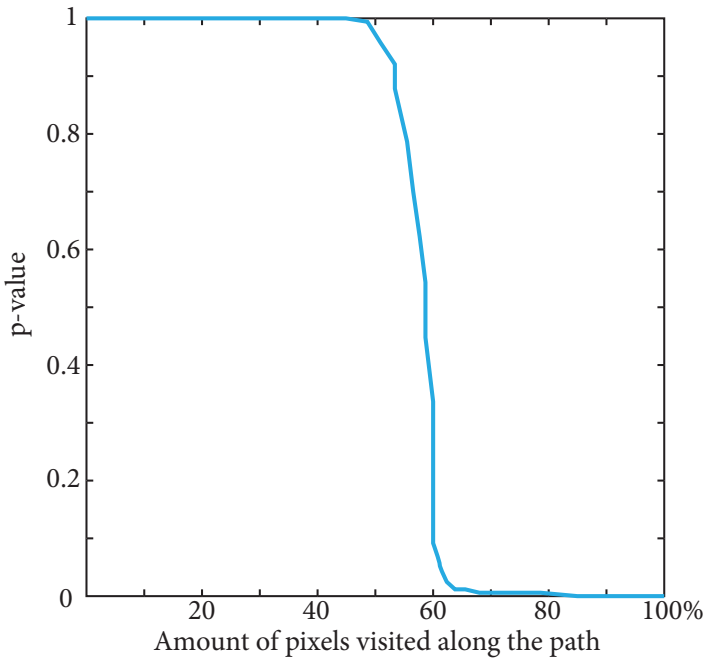

Figure 8: Example of the evolution of the p-value of a X^2 statistic for a LSB embedding stego scheme. Here the message was obviously embedded in the beginning of the "path". Inspired from [30].

The p-value for the statitic X^2 is computed as

$$p\left(X^2\right) = \left(2^{\frac{d-1}{2}}\Gamma\left(\frac{d-1}{2}\right)\right)^{-1} \int_{X^2}^{\infty} e^{-\frac{t}{2}} t^{\frac{d-1}{2}-1} dt, \tag{3.5}$$

(with Γ denoting the gamma function $\Gamma(y) = \int_0^\infty t^{y-1}e^{-t}dt$) which will be of value 0 if there is no message in the considered pixel, and going towards 1 if there is a message. Hence, by monitoring the p-value along the path of pixels, one gets the evolution depicted in Figure 8, for example (this is not obtained from a real simulation).

The p-value starts to decrease dramatically once the pixels containing message bits have been all visited, revealing hence the length of the original message.

Again, this attack only works in the case where the path is either trivial (the message is embedded into each LSB from the beginning of the image to the end) or if one knows the non-trivial path. In practical cases, this does not happen.

This scheme is mostly historical now. Stego schemes are more careful regarding the first order statistics.

This simple histogram-based and χ^2-test attack are only efficient for rather simple stego schemes, or schemes for which the embedding path can be devised — which is an important side-information. Recent stego schemes escape such detection means by preserving as much as possible the first-order statistics. The Outguess scheme, for example (see 2.4.2) tries to avoid this sort of attacks by only embedding in half of the available LSBs. The remaining half is meant to restore the distorted histograms of the DCT coefficients. As for visual detection, such statistics can be considered as superseded by the

following other schemes. Higher-order statistics have then been devised to address these new algorithms escaping first order statistics detection.

3.3.3 *RS steganalysis*

The concepts for *RS steganalysis* are presented in [48]. The name of RS is related to that of the pixel groups that are defined in this idea: Regular and Singular.

Here we use spatial dependencies between pixels to identify discrepancies.

The main idea is to extend the first-order statistics to spatial dependencies: the relationships between pixels (their values) are taken into account. Since the embedding process in the LSBs changes many values, it most likely increases the noise in the areas where a message is embedded. The goal is to measure the amount of noise created in the overall image, proceeding by small areas. For this purpose, the *smoothness* G of a set of pixels $\mathbf{i} = (i_1, \dots, i_n)^T$, with $i_j = i(x_j, y_j)$ (x_j and y_j describing the coordinates of the pixel $i(x_j, y_j)$ in the image i), is computed as

$$G(\mathbf{i}) = \sum_{j=1}^{n} \left| i_{j+1} - i_j \right|. \tag{3.6}$$

Obviously, the larger gets G, the noisier is the set of pixels $\mathbf{i}$ and the more likely it is to hold a hidden message. Then, the LSB flipping process is described by three functions, F_1, F_{-1} and F_0 such that (with pixel values in $[\![0, 255]\!]$)

$$\begin{cases} F_1 : & 0 \leftrightarrow 1, 2 \leftrightarrow 3, \dots, 254 \leftrightarrow 255 \\ F_0 : & 0 \leftrightarrow 0, 1 \leftrightarrow 1, \dots, 255 \leftrightarrow 255 \\ F_{-1} : & -1 \leftrightarrow 0, 1 \leftrightarrow 2, \dots, 255 \leftrightarrow 256 \end{cases}. \tag{3.7}$$

Using this notation, the potential pixel groups $\mathbf{i}$ are classified in three groups

$$\begin{aligned} &\mathbf{i} \text{ is } \texttt{regular} \text{ if} && G(F(\mathbf{i})) > G(\mathbf{i}) \\ &\mathbf{i} \text{ is } \texttt{singular} \text{ if} && G(F(\mathbf{i})) < G(\mathbf{i}), \\ &\mathbf{i} \text{ is } \texttt{unchanged} \text{ if} && G(F(\mathbf{i})) = G(\mathbf{i}) \end{aligned} \tag{3.8}$$

from which it is clear that a regular group most likely has a message embedded, compared to a singular group. Hence, after the embedding of a message, the relative amount of regular groups (divided by the total number of groups) R is larger than that of singular groups S. Estimates of the evolution of these two amounts can be obtained from experiments for varying amounts of LSB modifications.

This approach can also help estimating the length of the message, for quantitative steganalysis.

Comparing the R and S values obtained for a new suspicious image to the curves obtained experimentally, enables to identify the image as stego or cover (and have a message length estimate which is very accurate for some stego algorithms, according to [48]). One can refer to [38] for a general framework of RS steganalysis for LSB embedding.

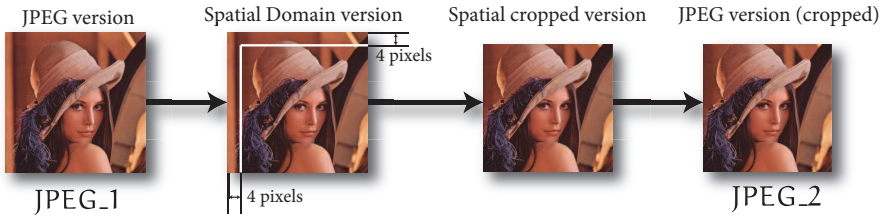

Figure 9: The calibration process as proposed in [46]: the considered image is first decompressed to spatial domain, cropped horizontally and vertically by 4 pixels, here, and then re-compressed using the very same quantization matrix and quality factor as that of the originally considered image.

3.3.4 *Calibration-based steganalysis*

The calibration process (decompress. crop the image and recompress) gives an estimate of the cover image's features.

One of the most useful information the steganalyzer could wish to get, for the steganalysis to be made easier, is the original cover image. From it, estimating whether the image has been tampered with would be very easy. While this never comes true in practice (except if the sender is not careful regarding the choice of the images), the cover image behavior and characteristics can be estimated, by the process of *calibration*, for example. In [46], Fridrich *et al.* crop the image by a certain number of pixels in both vertical and horizontal directions. The goal is to possibly "break" the inherent stego message lying in the image, and thus regain access to image characteristics that are close to that of the cover one. Figure 9 illustrates the idea: the suspicious image JPEG_1 is first decompressed to the spatial domain, and then cropped horizontally and vertically by $n = 4$ pixels (n can be different from 4). The resulting cropped image is then recompressed into JPEG_2 using the very same parameters (quantization matrix, quality factor) as that of JPEG_1.

Once both images are available, it is possible to compute a specific characteristic F (inherent to the image) on both of them, and compare the two values for example with the L_1 norm $\|F(\text{JPEG\_1}) - F(\text{JPEG\_2})\|_{L_1}$. With the calibration in mind, most the previously existing features (first order and higher order statistics, for example) are re-visited and deem significant improvements when put together. In [46] is proposed a set of overall 23 functionals F to use with the calibration. This set of features is meant for JPEG images exclusively, which are the only type of images considered in this dissertation.

A set of first and second order features is devised, to use with calibration.

For the following, let us denote by $I^{(k)}(x, y)$ the quantized DCT coefficient of coordinates (x, y) in the JPEG sub-block $I^{(k)}$ and the quantization matrix for this image i (of which I is the DCT coefficients representation) by $Q(x, y)$. For the considered case, we have blocks of size 8×8 and therefore $1 \leqslant x, y \leqslant 8$. Let us finally denote by B the total number of sub-blocks $I^{(k)}$, $1 \leqslant k \leqslant B$ and by $H(r)$ the histogram for DCT value r.

The first feature of this 23 feature set is the global histogram of DCT values $\mathbf{H} = (H(1), \ldots, H(R))^T$, with $R = \max_I(r)$.

As discussed before, the histogram attack can be countered (at least partially), for example by the Outguess approach, where the distortions caused to the global histogram of the LSBs are canceled by modifying the remaining LSBs (no message embedded in it, though). In order to try and detect

these modifications, individual histograms are used: the set of individual histograms $\{h_{x,y}(r)\}$ for DCT coefficient (x,y) is added to the set. Only a small number of these histograms is actually used, since only low frequency DCT coefficients have zeros to embed message bits. In the end, the set $\{h_{x,y}(r), (x,y) \in \{(1,2),(2,1),(2,2),(1,3),(3,1)\}\}$ is used.

A last feature of the first order type is included, the "dual histograms" $g_{x,y}(I_{dh})$, which are defined in [46] as the number of occurrences of the DCT value I_{dh} at the (x,y) DCT coefficient (for a fixed (x,y)) among all sub-blocks of the DCT array I,

$$g_{x,y}(I_{dh}) = \sum_{k=1}^{B} \delta_{I_{dh}, I^{(k)}(x,y)}, \tag{3.9}$$

where $\delta_{a,b}$ is the Kronecker symbol such that $\delta_{a,b} = 1$ iff $a = b$ and 0 otherwise.

Again, these dual histograms are only computed for certain DCT values, namely $-5 \leqslant I_{dh} \leqslant 5$.

These first-order features by definition only capture the dependencies within each of the sub-blocks. The goal of the introduced second-order features is to capture the dependencies *between* the sub-blocks of the image, with the rationale that the stego algorithm has probably distorted them, even if it managed to keep the inner sub-blocks dependencies rather similar to the cover.

For that matter, the *variation* V, two types of *blockiness* B_1 and B_2 and three measures N_{00}, N_{01} and N_{11} based on the *co-occurrence matrix of neighboring DCT coefficients*, are the final six features of the set.

The variation measures the dependencies between JPEG blocks.

The variation V capturing the dependency between neighboring sub-blocks is

$$V = \frac{1}{|I_{row}| + |I_{col}|} \left(\sum_{x,y=1}^{8} \sum_{k=1}^{|I_{row}|-1} \left| I^{(I_{row}(k))}(x,y) - I^{(I_{row}(k+1))}(x,y) \right| + \sum_{x,y=1}^{8} \sum_{k=1}^{|I_{col}|-1} \left| I^{(I_{col}(k))}(x,y) - I^{(I_{col}(k+1))}(x,y) \right| \right), \tag{3.10}$$

with I_{row} and I_{col} being the vectors of sub-blocks indices while scanning the image i by rows and columns, respectively.

The blockiness measures the inter JPEG block dependency over the whole image.

Following are the two blockiness measures, B_1 and B_2. B_j is defined on the *decompressed* (supposed gray-scale) JPEG image as

$$B_j = \frac{1}{N\lfloor (M-1)/8 \rfloor + M\lfloor (N-1)/8 \rfloor} \left(\sum_{y=1}^{M} \sum_{x=1}^{\lfloor (N-1)/8 \rfloor} |i(8x,y) - i(8x+1,y)| + \sum_{x=1}^{N} \sum_{y=1}^{\lfloor (M-1)/8 \rfloor} |i(x,8y) - i(x,8y+1)|^j \right), \tag{3.11}$$

Global Histogram	$\mathbf{H}/\|\mathbf{H}\|$
Individual histograms	$\frac{\mathbf{h}_{1,2}}{\|\mathbf{h}_{1,2}\|}, \frac{\mathbf{h}_{2,1}}{\|\mathbf{h}_{2,1}\|}, \frac{\mathbf{h}_{2,2}}{\|\mathbf{h}_{2,2}\|}, \frac{\mathbf{h}_{1,3}}{\|\mathbf{h}_{1,3}\|}, \frac{\mathbf{h}_{3,1}}{\|\mathbf{h}_{3,1}\|}$
Dual histograms	$\frac{\mathbf{g}(-5)}{\|\mathbf{g}(-5)\|}, \frac{\mathbf{g}(-4)}{\|\mathbf{g}(-4)\|}, \ldots, \frac{\mathbf{g}(4)}{\|\mathbf{g}(4)\|}, \frac{\mathbf{g}(5)}{\|\mathbf{g}(5)\|}$
Variation	V
Blockinesses	B_1, B_2
Co-occurrences	N_{00}, N_{01}, N_{11}

Table 2: The 23 DCT feature set [46].

where $i(x,y)$ denotes the gray-scale value (*not* a DCT value) of pixel (x,y) for the image i of dimensions $N \times M$.

Finally, the three last featuresN_{00}, N_{01} and N_{11} are computed out of the co-occurrence matrix $\mathbf{C}$ of neighboring DCT coefficients, of which element (u,v) is given by

$$\mathbf{C}(u,v) = \frac{1}{|I_{row}| + |I_{col}|} \left(\sum_{k=1}^{|I_{row}|-1} \sum_{x,y=1}^{8} \delta_{u,I^{(I_{row}(k))}(x,y)} \delta_{v,I^{(I_{row}(k+1))}(x,y)} + \sum_{k=1}^{|I_{col}|-1} \sum_{x,y=1}^{8} \delta_{u,I^{(I_{col}(k))}(x,y)} \delta_{v,I^{(I_{col}(k+1))}(x,y)} \right). \tag{3.12}$$

These measure the spreading of the values of the co-occurrence matrix.

The three features are then computed from $\mathbf{C}$ as

$$\begin{aligned}
N_{00} &= \mathbf{C}^{JPEG_1}(0,0) - \mathbf{C}^{JPEG_2}(0,0) \\
N_{01} &= \mathbf{C}^{JPEG_1}(0,1) - \mathbf{C}^{JPEG_2}(0,1) \\
&\quad + \mathbf{C}^{JPEG_1}(1,0) - \mathbf{C}^{JPEG_2}(1,0) \\
&\quad + \mathbf{C}^{JPEG_1}(-1,0) - \mathbf{C}^{JPEG_2}(-1,0) \\
&\quad + \mathbf{C}^{JPEG_1}(0,-1) - \mathbf{C}^{JPEG_2}(0,-1) \\
N_{11} &= \mathbf{C}^{JPEG_1}(1,1) - \mathbf{C}^{JPEG_2}(1,1) \\
&\quad + \mathbf{C}^{JPEG_1}(1,-1) - \mathbf{C}^{JPEG_2}(1,-1) \\
&\quad + \mathbf{C}^{JPEG_1}(-1,1) - \mathbf{C}^{JPEG_2}(-1,1) \\
&\quad + \mathbf{C}^{JPEG_1}(-1,-1) - \mathbf{C}^{JPEG_2}(-1,-1)
\end{aligned} \tag{3.13}$$

with $\mathbf{C}^{JPEG\_1}$ being the co-occurrence matrix of image JPEG_1 and similarly for JPEG_2.

This finally gives a set of 23 features, summarized in Table 2. Note that the histograms features are normalized.

Using such a set, most of the existing stego algorithms are detected. Thanks to the use of first and second order statistics and most of all, to the calibration process, this steganalysis is very successful.

Later on, in [99], the set is extended (by removing the normalization and taking all the values from the co-occurrence matrix $\mathbf{C}$) to a much larger 193 DCT features set.

This extended set is widely used in the presented publications in this dissertation. First, because the DCT features actually have a direct meaning, that is, they are not the result of a projection or transformation that would destroy the original physical sense. In this respect, they are interpretable, subject to feature selection, for example, as demonstrated in publications C (sections 5.3 and 5.4), D (sections 4.2 and 4.3) and E (sections 4.3 and 4.4). And second, because the large number of its features makes it a good candidate to such feature selection, which is important in steganalysis (see chapter 6 for details).

It is worth noting, finally, that a recent stego algorithm, the YASS, is not being properly detected by the DCT features if using the calibration process [80]. Using the un-calibrated version of the features though, provides better results (the calibration is called Cartesian calibration in [80], since it uses the Cartesian product of the features for JPEG_1 and JPEG_2). Overall, the calibration process is working especially well for stego algorithms which tend to respect the JPEG block structure. The calibration process "breaks" the original structure of the image and therefore the stego structure embedded. Algorithms such as YASS, for example, using de-synchronization regarding the JPEG blocks are not directly affected by the calibration process.

3.3.5 *Markov-based steganalysis*

The so-called *Markov-based features* are derived in [116] by the use of a Markov process to model the dependencies between the differences of the JPEG DCT values. From the original publication, this scheme clearly outperforms the previously presented calibrated DCT features. From the principle that many experts are better than just one, to make a decision, both approaches have recently been combined, into a large 324 features set, using both DCT and Markov-based calibrated features. The results outlined in [99] again outperform previous schemes.

The dependencies between pixels are modeled as Markov processes.

Let us present quickly the concept of the Markov-based features. In the original publication [116], the authors state and show empirically that there is a high correlation between the absolute values of DCT coefficients along the horizontal, vertical and diagonal directions. The goal of the Markov process is to model these dependencies and use the values of the transition probability matrix as features (actually a reduced subset of the transition probability matrix).

Denoting again $I(x,y)$ the value of the DCT coefficient (and therefore $|I(x,y)|$ its absolute value) at position (x,y) in the image i with N rows and M columns (i.e. $1 \leqslant x \leqslant N$ and $1 \leqslant y \leqslant M$), the so-called *difference arrays* of DCT coefficients for horizontal direction F_h, vertical direction F_v, main diagonal direction F_d and minor diagonal direction F_m are obtained as

$$\begin{aligned} F_h(x,y) &= |I(x,y)| - |I(x+1,y)|, \\ F_v(x,y) &= |I(x,y)| - |I(x,y+1)|, \\ F_d(x,y) &= |I(x,y)| - |I(x+1,y+1)|, \\ F_m(x,y) &= |I(x+1,y)| - |I(x,y+1)|, \end{aligned} \tag{3.14}$$

from which the actual transition probability matrices $\mathbf{M}_h, \mathbf{M}_v, \mathbf{M}_d$ and $\mathbf{M}_m$ for the four directions are

$$\begin{aligned}
\mathbf{M}_h(j,k) &= \frac{\sum_{x=1}^{N-2}\sum_{y=1}^{M}\delta_{F_h(x,y),j}\delta_{F_h(x+1,y),k}}{\sum_{x=1}^{N-1}\sum_{y=1}^{M}\delta_{F_h(x,y),j}},\\
\mathbf{M}_v(j,k) &= \frac{\sum_{x=1}^{N}\sum_{y=1}^{M-2}\delta_{F_v(x,y),j}\delta_{F_v(x,y+1),k}}{\sum_{x=1}^{N}\sum_{y=1}^{M-1}\delta_{F_v(x,y),j}},\\
\mathbf{M}_d(j,k) &= \frac{\sum_{x=1}^{N-2}\sum_{y=1}^{M-2}\delta_{F_d(x,y),j}\delta_{F_d(x+1,y+1),k}}{\sum_{x=1}^{N-1}\sum_{y=1}^{M-1}\delta_{F_d(x,y),j}},\\
\mathbf{M}_m(j,k) &= \frac{\sum_{x=1}^{N-2}\sum_{y=1}^{M-2}\delta_{F_m(x+1,y),j}\delta_{F_m(x,y+1),k}}{\sum_{x=1}^{N-1}\sum_{y=1}^{M-1}\delta_{F_m(x,y),j}},
\end{aligned} \tag{3.15}$$

of which only the central 9×9 part of each matrix is taken (since taking the full sized matrix would yield a too large number of features). The resulting set has hence $4 \times (9 \times 9) = 324$ features. As said, in [99], Fridrich proposes to apply the calibration procedure to these features, resulting in an improvement of the performances.

The Markov approach is rather successfull for classical stego algorithms, but would again most likely fail for the YASS case. In [86], Bin *et al.* indicate that Markov-based features would probably work on the original version of YASS for the location of the embedding blocks is not truly random and hence altered dependencies between DCT coeffcients could be detected. A version of YASS using truly random location for the blocks would on the contrary probably not be detectable by the Markov approach.

3.3.6 *SPAM features*

Markov processes are also used to model more directions in the image.

The last discussed steganalysis scheme is proposed by Tomàš Pevnỳ in [102] and named *SPAM* features, for *Substractive Pixel Adjacency Matrix*. The SPAM features are meant in the first place for the steganalysis of the YASS stego algorithm and globally for LSB embedding based schemes. The idea is closely related to that of the previously described scheme: modeling the difference arrays (this time in the spatial domain, though) with Markov processes, of first and second order. The main difference here is that the difference arrays are computed along "both" directions for horizontal (left and right), vertical (up and down), main diagonal and minor diagonal. As in the original publication, we denote by arrows each of these eight directions $\{\leftarrow, \rightarrow, \uparrow, \downarrow, \nwarrow, \searrow, \nearrow, \swarrow\}$.

Using the same notation as in the previous section, denote by $F_{\rightarrow}(x,y) = I(x,y) - I(x,y+1)$ the difference array for the first direction (the remaining seven are defined in the same fashion). Again, the first order Markov transition probability matrix is given by

$$\mathbf{M}_{\rightarrow}(j,k) = P\left(F_{\rightarrow}(x,y+1) = j | F_{\rightarrow}(x,y) = k\right), \tag{3.16}$$

Markov transition probability matrices are computed for first and second order transitions.

with the same sort of restriction on the values of j and k (restricted to a set $[\![-T,T]\!]$, with $T = 4$ for these first order in the original publication).

The second order one is then

$$\mathbf{M}_{\rightarrow}(j,k,l) = P\left(F_{\rightarrow}(x,y+2) = j | F_{\rightarrow}(x,y+1) = k, F_{\rightarrow}(x,y) = l\right), \quad (3.17)$$

also with j, k and l restricted to $[\![-3,3]\!]$ for these second order features. This restriction is again meant to reduce the dimensionality of the feature set (which is already 686 features for the second order case).

The final features sets of first order $\mathbf{F}^{(1)}$ and of second order $\mathbf{F}^{(2)}$ are obtained through

$$\mathbf{F}^{(j)} = \left[\frac{1}{4}\left(\mathbf{M}_{\rightarrow} + \mathbf{M}_{\leftarrow} + \mathbf{M}_{\downarrow} + \mathbf{M}_{\uparrow}\right); \frac{1}{4}\left(\mathbf{M}_{\nwarrow} + \mathbf{M}_{\searrow} + \mathbf{M}_{\nearrow} + \mathbf{M}_{\swarrow}\right)\right], \quad (3.18)$$

being the concatenation of the vectors of features. The first (resp. second) order term lies inside the $\mathbf{M}$ notation, for simplicity of notations here (as depicted in Equations 3.16 and 3.17).

The final feature set is a concatenation of direct directions and diagonal ones.

As pointed out by Pevnỳ in [102], the order of the Markov process and the threshold T control the extent of the space of features (the complexity of the Markov model). If one has a good classification model to discriminate by using such large number of features (and sufficient computational power, along with a *very* large database of images), it would be possible to go beyond the thresholds proposed here. These problems are presented in the next chapter 4 and experimentally illustrated in chapter 6 (referring to publications C, D and E).

3.3.7 *Undiscussed schemes*

As for the review of stego algorithms in the previous chapter, this presented list of steganalysis schemes is by no means exhaustive. It only attempts to cover some the most widely recognized and used steganalysis schemes, for their meaningfulness and innovative aspects. A more thorough covering of steganalysis schemes can for example be found in [47].

3.4 A PITFALL IN STEGANALYSIS

In the previous discussion, we have dealt with features sets of growing size: only the histogram of DCT values, for the histogram attack, then followed by the 23 features of the original calibrated DCT set, expanded to 193 later on; the merging of the Markov-based set of 324 features, with the 193 ones from the DCT set into a 517 features set (brought down to 274), and finally the SPAM ones, which lead to potentially very large feature sets (686 for a "restricted" second order Markov process).

All these feature sets are clearly getting larger, and the evolution is likely to be on the growing slope in the future: one wants to capture as many characteristics of the images in order to model them better and better, and thus make a "cleaner" difference between stego and cover — a better steganalysis. The recent trend — giving very promising results in terms of performances for the steganalysis — seems to be in combining the sets of features, in order

The growing size of feature sets can be a problem in Machine Learning...

to better cover the different domains in which the image characteristics are expressed. This will also lead to larger and larger feature sets to analyze.

Although this issue will be discussed more lengthily in the next chapter 4, the number of samples *has to* grow along with the number of features used. This problem is one of the famous *Curse of Dimensionality* and is not the only one related to the growth of the feature space.

Intuitively, we need to have larger number of features and larger number of samples, to accommodate the Curse. Therefore, the overall data set — the matrix holding in each column a different feature and in each row a different image — is growing exponentially (see section 6.1.1 for more details on high-dimensionality related problems).

One obvious problem is then to have a classification or regression model that can accommodate such large data sets in reasonable computational time: no one wants to wait weeks to extract the features out of a base of images, and then train a model to finally be able to predict on a few images, whether steganography has been performed.

3.5 CONCLUSION

The concept of security in steganography / *steganalysis* is classicaly defined in an information-theoretic sense, by measuring the "differences" between the cover image (original) and the stego one (with an embedded message). Since this theoretical definition contains quantities too difficult to estimate, the typical steganalysis framework reverts to an approximation for the estimation of the security. This estimation is most often based on the use of a certain amount of characteristics descriptive of the image considered, the *features*.

Using these features with a machine learning model enables to obtain an estimation whether an image is cover or stego. The drawback of the feature-based steganalysis is similar to the commonly encountered problem in Machine Learning: the growing *dimensionality* (corresponding to the number of features) — and therefore the global size of the data.

The next part (chapters 4 and 5) of this dissertation first presents a short state of the art in Machine Learning (mostly directed towards regression and classification problems, which are of interest for the steganalysis problem) in chapter 4, followed by the presentation of a new model for which training is fast enough to manage the large data sets coming from steganalysis, for example: the Optimally-Pruned Extreme Learning Machine (chapter 5 and publications A and B).

Part II

A FAST, EFFICIENT AND ROBUST MACHINE LEARNING TECHNIQUE: OP-ELM

4 A SHORT REVIEW ON MACHINE LEARNING

In an attempt to summarize the vast field of *Machine Learning* into a few key concepts most relevant to the steganalysis and steganography related problems, we propose in this chapter to first define precisely the steganalysis classes of problems in *machine learning terminology*. We refine the overview proposed in this chapter to *supervised binary classication* and *supervised regression*, which are the two very specific cases of machine learning problems at interest in this dissertation.

We also propose an established "procedure" to properly train, validate, build and test a machine learning model, on the general case. Finally, the machine learning models used in the publications of this dissertation are presented, with references to other famous related models.

4.1 LEARNING PROBLEMS

4.1.1 *What is Machine Learning*

It takes a new born between 6 and 24 months on average [91] to say its first (intelligible) words. The whole process of learning pronunciation, language structure, word construction and so on, takes even longer than this and is potentially a lifetime training and evolution. If we think about it, the amount of data to process for a baby's brain is tremendous. Only for the language part of the learning, it requires the processing of the mouth muscle movements (imitation of the surrounding people), remembering sequences of phonemes to create words, and associating these words with a context and content.

Machines probably lack the computational power — yet — to process enough data to be able to imitate the brain.

A human brain can apparently do this. At the time of this dissertation, a machine cannot. Although advances in voice synthesis and language modeling have enabled the creation of systems that seem to be human when asked questions, we are still very far from creating an actual structure able to learn a whole human-type language by itself.

The idea of mimicking human abilities such as the language, by machines is globally named *Artificial Intelligence*, of which *Machine Learning* is a sub-class. *Machine Learning* only aims at *learning*, that is to say, observe examples of a specific phenomenon in order to *model* its underlying process. If a sufficiently good model for this phenomenon is found, new examples can be used, either to make the model learn new insights on the phenomenon, or *predict* what happens for these specific examples.

The example given in the introductory part of [6] is interesting because it covers most of the aspects of the machine learning problem. Consider an important supermarket chain, having many shops throughout many countries. In order to advertise the right kind of products at the right time and to the right customer, they need to know what customers buy, when and possibly why. The cashiers' terminals are recording all this information about

what is bought, when and by which customer, and in the case of millions of customers everyday, this can create terabytes of data in a day or so.

Machine learning tries to extract an underlying generating process from the data.

In all this data, only a small amount of information is actually relevant and useful, for example for advertising purposes. The aim of machine learning (precisely *data mining*, in this case) is then to identify a pattern, an underlying process in the data: most likely indeed, customers are not buying the products at random. While it is obviously a very hard task (if possible at all) to model completely a customer's behavior based on this data, a good enough approximation of the behavior is sufficient already. And once we know that Mr. Muumipeikko buys chocolate-based products during the dark days of winter and beer and sausages when the first days of summer approach, it becomes easier to target the advertising and the offers for that specific customer.

In the steganalysis framework, the idea is to obtain a model of what an image looks like, in terms of the data we extract from it (the features). Provided that the model can learn the difference between a cover image and a stego image, in the classical qualitative steganalysis case, we can identify suspicious images as being stego properly.

4.1.2 *Classes of learning problems*

A problem is reduced to the data acquired from it, with samples and variables/ features.

In this dissertation, we consider that a problem is described by a *data set*, which takes the form of a matrix $\mathbf{x}$, called *inputs* (or *input data*). Other means of structuring the input data such as tensors or databases are not considered here. The typical formulation uses the rows of $\mathbf{x}$ as *samples* (examples of the observed phenomenon, being images in the steganalysis case), and columns as *variables* (or *features*, to refer to the steganalysis terminology). The data set is usually acquired from a specific source, either by measuring directly some quantities (steganalysis is in this case), or obtained from a supposeably reliable source.

Reliable and relevant data

In the following, we consider the data to be *reliable*. This means that we do not consider the possibility that part of the data is wrong (in the sense that it would describe the phenomenon improperly), or missing (some values missing for some samples).

The concept of missing values in machine learning is by itself a whole branch of the field and will not be discussed in this dissertation. One can refer for example to [89, 121, 90, 120] for a bibliography and insights on that specific problem.

The data is supposed to be reliable but not necessarily relevant.

The problem of having "wrong" data is difficult to define, but let us clarify the assumption made here. By "wrong", it is meant that the considered part of the data is trying to mislead voluntarily the model. For example, if two sensors are measuring the same phenomenon, but one of them has an inverted polarity compared to the other, we would try to model a phenomenon based on its behavior (first correct sensor) and the opposite of it (second inverted sensor).

Data that is *irrelevant* to the task (for example some random noise or the outdoor temperature for a steganalysis problem) is not considered as "wrong"

data. The problem of determining if data is relevant is brushed in section 6 about dimensionality reduction, which is a means of discarding such data from the original data set.

It is a reasonable assumption to suppose that we do not have "wrong" data in the steganalysis framework: the samples (the images) have to be properly selected so that they are natural images and not some pure random noise or some other digital object disguised as an image; and the features presented in chapter 3 are clearly sensible (although there might be redundancy in them, e.g.).

Given this setup, one can define two main classes of learning problems, in machine learning, the *unsupervised* and the *supervised* ones.

Unsupervised learning

This specific class of learning makes the assumption that only the data $\mathbf{x}$ is available, and tries to infer an underlying structure/behavior in this data.

Unsupervised learning aims at finding a structure in the data without any output information.

A classical example of this class is the problem of Blind Source Separation (BSS) [73], of which the so-called cocktail party problem is an instance. Imagine a cocktail hall where a party is being held, with many invitees speaking together at the same time. In the resulting "noise" from the simultaneous talking, separating each of the voices seems a very hard task (although it seems that humans can identify two voices simultaneously and separate them clearly from the rest [118]). In this case, the rows of the data matrix $\mathbf{x}$ are the recorded signals in the cocktail hall, while the columns correspond to the sampling of the sound recorded. The Independant Component Analysis (ICA) [73, 29] algorithm enables to isolate the sources $\mathbf{s}$ such that $\mathbf{x} = \mathbf{w} \cdot \mathbf{s}$ (with $\mathbf{w}$ being the mixing matrix, in the linear noiseless ICA case), provided that there are enough recorded signals.

Another use of unsupervised learning is *clustering*. This class of learning problem aims at separating the data as best as possible, so as to create *clusters*. In the previously mentioned example of the supermarket selling goods across many countries, it might be of interest to find groups of customers with specific behaviors to analyze: some "expected" groups with identified similar behaviors will be easier to target in terms of selling strategies, while unexpected groups formed in the clustering process may reveal the unexpected behaviour of a part of the population. Which this time would lead to a new, more adapted and specific selling strategy.

Supervised learning

This second class of learning problem is the one considered in the rest of this dissertation.

Supervised learning assumes the existence of a teacher.

In this case, the presence of a *teacher* (*supervisor*) is assumed, in addition to the data $\mathbf{x}$, so that the model can be trained using a reference for each of the samples in $\mathbf{x}$. This teacher usually takes the form of another matrix, of *outputs* denoted by $\mathbf{y}$.

For the context of qualitative steganalysis, for example, $\mathbf{y}$ is a vector constituted by elements depicting two *classes*: the first one coding for the image state "stego" and the second one for "cover". For quantitative steganalysis, it is a vector of positive real values, each representing the message size for each

sample (image). The difference between these two problems is discussed in the following subsection 4.1.3.

4.1.3 *Structure of the supervised learning problem*

With the assumption of a supervised learning problem, we can define two types of supervised learning, each of them with a different type and size of the output matrix $\mathbf{y}$.

The first type of supervised learning is called *regression*. In this case, the output $\mathbf{y}$ is a vector of generally real values, to be predicted. This means that the model $\mathcal{M}$ predicts values in $\mathbb{R}$:

$$\mathcal{M} : \mathbb{R}^{N \times M} \longrightarrow \mathbb{R}, \tag{4.1}$$

with N the number of samples and M the number of features (variables). This type is the one of quantitative steganalysis, for example. The problem is usually formulated as

The classical problem formulation in terms of approximation for regression.

$$\mathbf{y} = f(\mathbf{x}) + \mathcal{N}(0, \boldsymbol{\Sigma}), \tag{4.2}$$

with $\boldsymbol{\Sigma}$ the covariance matrix for the zero-mean noise $\mathcal{N}(0, \boldsymbol{\Sigma})$ and f the "process" underlying in the data $\mathbf{x}$. The model $\mathcal{M}$ hence tries to best approximate f.

The second one is *classification*, already mentioned in chapter 3 for the qualitative steganalysis problem. This time, the outputs $\mathbf{y}$ are *categorical* and can be of two different types: *nominal* or *ordinal*. *Nominal* outputs have no sense of order between them. For example, coding blue as 1, red as 2 and green as 3, one cannot sort the values $1, 2, 3$ with some standard ordering operation. The numerical attribute to code the meaning is here purely a commodity for manipulating the data.

There are two types of classification data: ordinal and nominal.

The *ordinal* outputs have a sense of order; for example, the age of a person, expressed in full years. In the end, both classification outputs (ordinal and nominal) $\mathbf{y}$ can be expressed as being in $\mathbb{N}$. In this dissertation, we only consider *nominal outputs* for the *classification problem* (qualitative steganalysis). Moreover, we restrict the nominal type to being only a binary classification output. The case of a multi-class classification problem can be rather complex and is not discussed here, although it is used in the blind steganalysis approach (see section 3.2). Eventually, it can be brought back to a binary classification problem using multi-outputs: consider a problem with classes in $[\![1, 3]\!]$, one can code each of the classes as following, using the binary choice of classes $[\![0, 1]\!]$:

$$\begin{aligned} 1 &\longrightarrow [0\,0\,1] \\ 2 &\longrightarrow [0\,1\,0] \\ 3 &\longrightarrow [1\,0\,0] \end{aligned} \tag{4.3}$$

It can be argued that predicting a single value (single output problem) in the $[\![1, 3]\!]$ range is different from predicting an output like $[0\,0\,1]$ for example (multi-output problem), since one predicts three outputs at once then. The possibilities of multi-output problems — $\mathbf{y}$ being a matrix and no longer a

vector — are not discussed in this dissertation and the previous example is merely meant to illustrate a possibility of dealing with multi-classes problem. Only binary classification ones are in the scope of this dissertation. One can refer to [104] and [5] for recent improvements in dealing with multi-class problems and to [6] for a state of the art and modifications of classical machine learning models for the multi-class case.

We only consider binary classification here.

4.1.4 *Building a model for the learning problem*

With the learning problem fixed (regression or classification) it is possible to build a model and train it on that particular data. The process of building a model requires at least four steps: selecting the *model class*, the *model structure*, *build* the model and finally *validate* these choices. An additional fifth step (if the data is available for it) can be used, to further ensure about the quality of the model on unused data; this is the *test* part. Each step is detailed in the following.

Four steps to build a model.

Choosing the model class

The model class selection is the first step and is primarily an informated choice on the user side: some models are meant for unsupervised learning and will obviously not be appropriate here; some other models are known to have strong limitations regarding irrelevant or correlated data (e.g., in the case where two or more features are describing the same part of the phenomenon in the very same fashion); and some models are known to be better than others on some specific problems for which they have been designed in the first place (see chapter 5 and Reservoir Computing for time-variant processes, for example).

In the end, the user has to decide *a priori* which model $\mathcal{M}$ to use among the large choice that exists. Section 4.3 gives a few examples of some famous model classes, and therefore, often used, especially in the framework of steganalysis.

An informated a priori decision is required to select the model class.

Model structure selection and error criterion

Choosing the model structure is intimately related with the next step, the model validation.

A model has usually a certain number of *hyper-parameters*, which relate to the model structure design choices (for example the order of a polynomial curve to fit to the data) and *parameters* which have to be determined using the data itself (the coefficients of the polynomial curve of fixed order).

We need to find the right hyper-parameters for the model...

The *parameters* of a model $\mathcal{M}$ are therefore part of the building of the model and their determination requires to use the data.

The goal of this step is to find a possible optimal set of l hyper-parameters $\Theta = (\theta_1, \ldots, \theta_l)$ such that the model $\mathcal{M}(\mathbf{x}, \Theta)$ makes the smallest *error* on the data $\mathbf{x}$, regarding the output $\mathbf{y}$. The *error* ε_r in regression can be defined by a risk function f_{risk} which will quantify how far the model's estimation $\hat{\mathbf{y}} = \mathcal{M}(\mathbf{x}, \Theta)$ for a specific fixed set of hyper-parameters Θ is from the real value $\mathbf{y}$

$$\varepsilon_r = f_{risk}(\hat{\mathbf{y}}, \mathbf{y}) = f_{risk}(\mathcal{M}(\mathbf{x}, \Theta), \mathbf{y}). \tag{4.4}$$

For the case of a regression learning problem (single output), the function f is typically a *Mean Squared Error* (MSE) risk function, that is

$$f_{risk}(\hat{\mathbf{y}}, \mathbf{y}) = f_{MSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{N}\sum_{j=1}^{N}\left(\hat{y}_j - y_j\right)^2, \tag{4.5}$$

. . . and for that we need a criterion to evaluate the model performance.

where $\hat{y}_j$ is the j-th component of the vector $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_N)^T$, the model's estimation and similarly for $\mathbf{y}$. For the following of the dissertation, we also define the *Normalized Mean Square Error* (NMSE) [58] as the MSE normalized by the variance of the output $\mathbf{y}$,

$$f_{NMSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{f_{MSE}(\hat{\mathbf{y}}, \mathbf{y})}{var(\mathbf{y})}. \tag{4.6}$$

Note that in the case where the prediction $\hat{\mathbf{y}}$ is the mean of $\mathbf{y}$, the NMSE is 1 meaning that the prediction is poor (note also that a NMSE can be larger than 1, for models giving a worse prediction than the mean of $\mathbf{y}$...).

In the same spirit, one can define an error for the binary classification problem, assuming that we are interested in having the highest *accuracy*, as defined in 3.1, by the ratio

$$Acc = \frac{TP + TP}{N_P + N_N}, \tag{4.7}$$

with TP and TN the amounts of true positive and negative, respectively, and N_P (and N_N) the total numbers of positive (respectively negative) instances. The error ε_c in classification can then be defined as

$$\varepsilon_c = 1 - Acc, \tag{4.8}$$

which can be expressed in percentage, and corresponds to the "percentage of incorrect classification" intuitively. Note that the problem of minimizing ε is very different for the regression and classification cases presented here. Actually, most models will preferably minimize the MSE ε_r, even for the classification case, and eventually report the classification error ε_c obtained using the same hyper-parameters $\Theta = \Theta_r$ that minimize the MSE. This can for example be due to historical reasons where the original model's algorithm was meant only for regression problems, minimizing the MSE and has later on been adapted to classification problems.

It is not a real issue to minimize the MSE for a binary classification problem, since there are only two classes in $\mathbf{y}$. Minimization of the MSE leads anyway to minimizing the discrepancies between $\hat{\mathbf{y}}$ and $\mathbf{y}$.

Building the model

This step is usually straightforward on the user side. With a fixed model class and set of hyper-parameters, the model can be built on the data $\mathbf{x}$ and is then ready to be used on other data: the model parameters are determined in this step, through the model's internal structure.

An important part of the model building requires to separate the original data set $\mathbf{x}$ so that *building* the model and *validating* it happen on two different

subsets of the original data. One could say that the hyper-parameters ("external" to the model) are optimized to minimize the validation error, while the parameters ("internal" to the model) are devised to minimize the training error, computed during the model building.

Validation of the model

Now that we have defined a way to measure the error (and hence, a criterion to obtain an optimal set of hyper-parameters for the model), let us have a practical approach on how to manage the data to evaluate the error ε for different sets of hyper-parameters Θ.

Originally, we can assume that we have a unique data set $\mathbf{x}$ containing all the available data about the phenomenon: in steganalysis, we have N rows, each containing the features (in the M columns) of the N images of the base of images.

The model should be validated on data not used for training then.

In order to estimate the model $\mathcal{M}$ for a fixed set of hyper-parameters Θ, we need *learning data* $\mathbf{x}_l$, which will be a subset of the whole data $\mathbf{x}$. The model is built on $\mathbf{x}_l$ and the *learning error* ε_l is computed $\varepsilon_l = \mathcal{M}(\mathbf{x}_l, \Theta)$. For most classes of models in machine learning, it is possible to have $\varepsilon_l > 0$ as small as possible, given enough time to find the correct set of hyper-parameters $\Theta = \Theta_l$ that minimizes sufficiently ε_l. The problem is then that the model has also learned the *noise* that is present in the data: real data, that is obtained from actual phenomena, always has noise in it — a part that should not be modeled. The over-fitting effect occurs, then. Figure 10 illustrates this problem: the model depicted by the solid line approximates the "idea" of the data, the global behavior of it, while the one represented by the dashed line only tries to fit to all the points, thereby losing the sense of the data itself.

This issue can be overcame by the use of a *validation set*. Instead of evaluating the performance of the model $\mathcal{M}$ on the learning data itself only, a different part of the data is used to *validate* the model, that is, evaluate its error on different data than that used for training.

Going back to our full data set $\mathbf{x}$, we need to divide it beforehand into two different sets $\mathbf{x}_l$ (of cardinal $N_l = |\mathbf{x}_l|$) for the learning part of the model, and $\mathbf{x}_v$ (of cardinal $N_v = |\mathbf{x}_v|$) to validate the model. The use of new data (i.e., never seen by the model during the training) helps in estimating the error of the model with such hyper-parameters Θ properly: an over-fitting of the model on the training data creates inevitably (if the validation data is different from the training one) a large error on the validation set. Hence, the set of hyper-parameters has to be modified so that the model *generalizes* well on the new data.

This approach has a strong drawback: it requires dividing the original data set into two subsets. Whenever the data is either scarce or costful to acquire, this can be a serious problem. In such cases, *cross-validation* can be used, to replace the standard validation.

For example, using k-fold cross-validation, the model is validated on unused data.

Cross-validation is usually performed in a k-fold way, where k determines the number of parts into which the data will be divided. Figure 11 presents a case of 3-fold cross validation scheme. The whole data $\mathbf{x}$ is divided into three equal sets, of which the first third (colored in the Figure) will be used as validation data $\mathbf{x}_v$, and the remaining two thirds for learning $\mathbf{x}_l$. The process is repeated $k = 3$ times, for this example, and the average error on the three

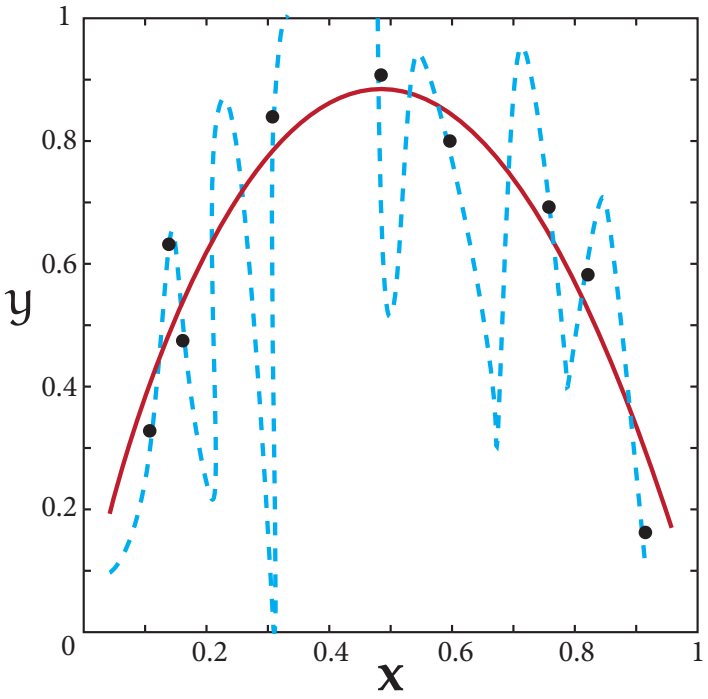

Figure 10: The over-fitting concept: the solid line depicts a model approximating the underlying phenomenon behind the data (black dots), while the dashed line just tries to fit the data completely.

runs is reported. Obviously, the number of times one needs to repeat the learning and the validation is equal to k which can be problematic for models that require large computational time.

Leave-One-Out is the limit case $k = N$ and gives a good estimate of the validation error.

In the limit case $k = N$ where the number of folds is the number of samples available in $\mathbf{x}$, the k-fold is called Leave-One-Out (LOO) cross-validation. Using this scheme enables to have a very good idea of the validation error that would be achieved on different data, since each point is in turn taken out of the learning set $\mathbf{x}_l$ and evaluated in validation. This is also the most costly approach to cross-validation, although in some cases, a closed form formula for the LOO error [94] can be devised and permits a fast evaluation (compared to running $k = N$ times the learning and validation).

Model test

Finally, the model $\mathcal{M}$ for which the best set of hyper-parameters Θ_v has been obtained (regarding the error criterion in validation) can be tested. It should be noted that the test data is again different from the learning and the validation data, and should be taken separately in the first place.

In the end, it is good to test the final model on totally unused data.

This test data can be taken from the original data set $\mathbf{x}$, to check if the model behaves well on totally new data (unseen before, both for validation and learning), or be data that has been acquired/obtained since. The test error ε_t is the proper error to be reported, if one refers to the performance of a model on some data set $\mathbf{x}$. Training and validation errors cannot be considered as a proper measure of performance of the model $\mathcal{M}$, even though the LOO

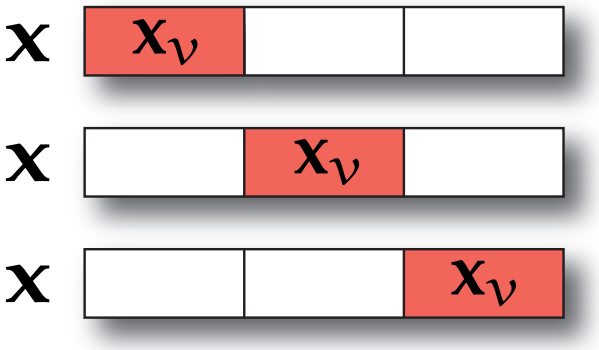

Figure 11: Example of 3-fold cross-validation: the whole data **x** is divided into three parts. The first part is used for validation (red) and the rest (white) for learning. Once done, the second part is used for validation and the first and third for learning...

error in validation can be considered to be a rough approximation of the real validation error — which would be obtained given an infinite validation set — if the data set is large enough ($N \longrightarrow \infty$). In [110, 33, 34] this claim is proved theoretically for some specific models (see [74] for a more thorough review and a broader view of the matter).

Bootstrapping is a concept similar to cross-validation.

Overall, the cross-validation idea is rather close to that of bootstrapping in statistics [39], where the properties of an estimator are estimated through multiple sampling from an approximated distribution. Usually a random subset (with replacements) of the original data set is taken, on which the estimator is devised. Multiple repetitions of this process (assuming the number of samples available in the first place is large enough) permit to have a mean value for the estimator. Please refer to [39] for a full reference on the matter of boostrapping.

In the same fashion as for the cross-validation, one can consider the possibilty of *cross-test*, in case the data set **x** is again small. Doing so is discussed in the following.

4.2 PRACTICAL NOTES ON DATA PROCESSING FOR MODEL BUILDING

These are practical notes giving one way to pre-process the data for building a model.

Here are compiled a few notes on taking care of the data before building a model and searching for an optimal set of hyper-parameters. They are mentioned to explain the procedure through which data has been handled, for the proposed publications, in this dissertation.

Take a random permutation of the original data.

- It is a good idea to make a random permutation of the data set (inputs and outputs in the same way) before starting anything. The original data set might have been sorted for some reason and the following division of the data set would lead to learning and validating on non-homogeneous data. In the steganalysis framework, for example, if all natural images depicting trees are in the beginning and sky shots in the end, we encounter the risk of having a model too specialized in one type of images, which will behave incorrectly on other types.

Normalize the data using the learning normalization factors.

- One also wants to normalize the data, for some models (zero mean and unit standard deviation, for example). By normalizing, it is meant for each of the columns of $\mathbf{x}$, the features, separately. A simple illustration is for a model based on the Euclidean distance (for example a k-Nearest Neighbors, see 4.3): if one of the features is valued in a large range compared to the other ones, the absence of normalization will distort the Euclidean distance and make the other features meaningless (since they will play a too small part in the distance evaluation). Normalization should also be done properly: for example, it is unfair to normalize the whole $\mathbf{x}$ and then divide it into learning and testing, since the normalization will have made use of data that is not supposed to be available (the test data), to compute the normalization factors (i.e., mean and standard deviation).

Divide the data carefully into learning, validation and test (see Fig. 12).

- As mentioned before, in the best case, we want to have a large learning set $\mathbf{x}_l$, a large validation set $\mathbf{x}_v$ and a large test set $\mathbf{x}_t$. With most available data sets, it is just not possible to have all three, and we can revert to using cross-validation and cross-test. A typical methodology is thus to divide the original data set $\mathbf{x}$ (randomly permuted) into $\mathbf{x}_{lv}$ and $\mathbf{x}_t$ (respectively of cardinals $N_{lv} = |\mathbf{x}_{lv}|$ and $N_t = |\mathbf{x}_t|$), with proportions to be decided. The $\mathbf{x}_{lv}$ set is then divided into k folds, and the cross-validation is performed (with normalization factors using only the learning part). The model is finally tested on the remaining test set $\mathbf{x}_t$ which has been also normalized using the learning normalization factors. Eventually, another random permutation of $\mathbf{x}$ is devised, and the whole process is repeated, to have a more reliable estimate of the test error. The main advantage of performing cross-test as described is that it enables to have a good estimation of the *generalization* error of the built model, meaning the error one would obtain were he to use this specific model on data unseen until now.

Figure 12 illustrates the previously described approach to processing data for a proper model training, validation and testing for the case of a 3-fold cross-validation process. The cross-test is depicted by the loop on the h parameter.

Now that the building of a model (for the supervised class of learning problems) has been detailed, let us review some of the most used machine learning models, in order to explicit some notations and important concepts in machine learning.

4.3 SOME MODEL CLASSES FOR MACHINE LEARNING

In the following are presented six different model classes for machine learning. This review is very limited, but mainly wishes to present the algorithms used for comparisons and benchmarks in some of the publications of this dissertation (see publications A and E, for example). A whole class of models is not presented in this chapter but in the next, the Random Projections based models, since they are the foundation of the proposed new model, the OP-ELM (chapter 5).

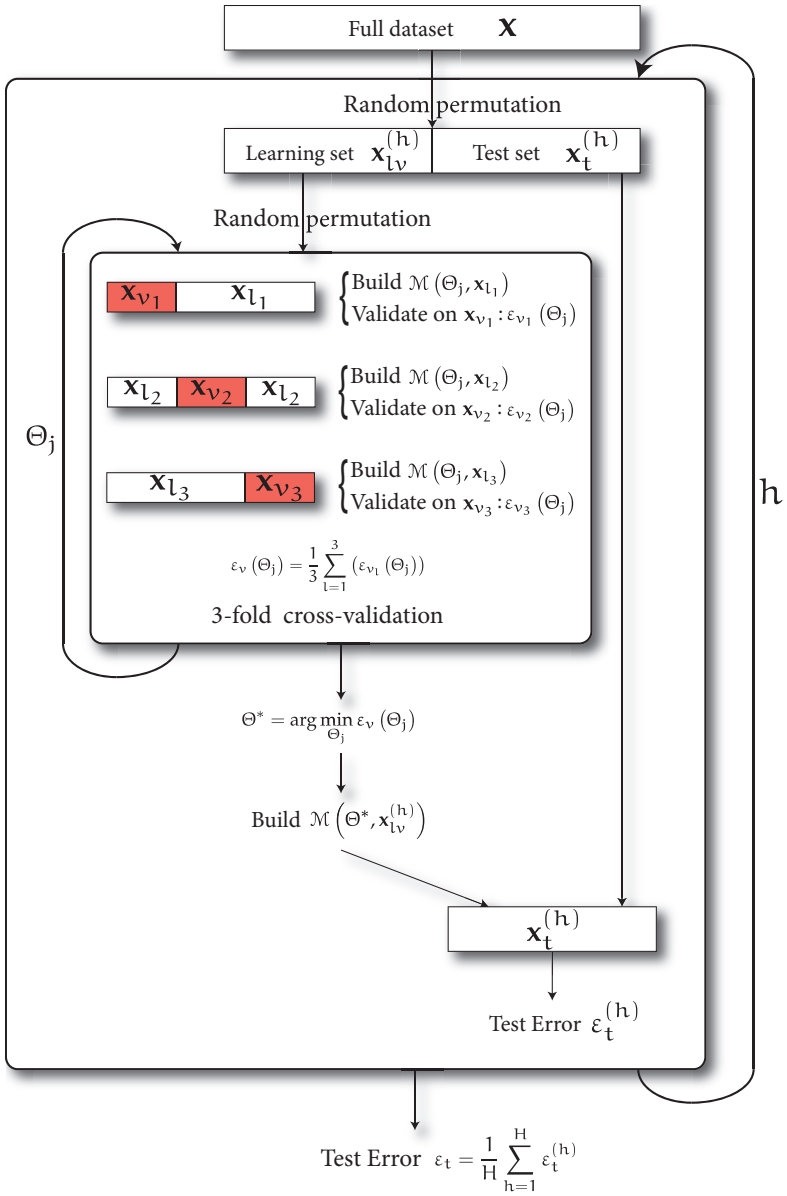

Figure 12: Proposed data processing scheme: a random permutation of the original data set $\mathbf{x}$ is first divided into two parts $\mathbf{x}_{lv}^{(h)}$ for learning and validation and $\mathbf{x}_{t}^{(h)}$ for the test. The set $\mathbf{x}_{lv}^{(h)}$ is then divided according to the k-fold (here $k = 3$) cross-validation approach and the model $\mathcal{M}$ is first trained with parameters Θ_j on $\mathbf{x}_{l_1}$ (the (h) notation is dropped within the cross-validation for simplicity). The trained model is validated on the validation set $\mathbf{x}_{v_1}$ to obtain the validation error $\varepsilon_{v_1}(\Theta_j)$. This process is repeated 3 times overall, for $(\mathbf{x}_{l_1}, \mathbf{x}_{v_1})$,$(\mathbf{x}_{l_2}, \mathbf{x}_{v_2})$ and $(\mathbf{x}_{l_3}, \mathbf{x}_{v_3})$. And also repeated for different values of the hyper-parameters Θ_j. The best set of parameters $\Theta^* = \arg\min_{\Theta_j} \varepsilon_v(\Theta_j)$ is then used to build the final model. It is used on the test data $\mathbf{x}_{t}^{(h)}$ to determine the error $\varepsilon_t^{(h)}$. This procedure is repeated $h = H$ times to obtain the final cross-test value $\varepsilon_t = \frac{1}{H}\sum_{h=1}^{H}\varepsilon_t^{(h)}$.

4.3.1 *Linear discrimination and regression*

The concept of linear regression/discrimination in machine learning is possibly the simplest of all since it uses a linear model to regress/ classify the data. Let us start by linear regression, and then present two models widely used in steganalysis, the Linear Discriminant Analysis and the Support Vector Machines.

Linear regression

Linear regression is very simple, but efficient on problems with a linear component.

In the regression learning problem case, the linear regression aims at finding a matrix $\mathbf{w}$ such that, for the input $\mathbf{x}$

$$\hat{\mathbf{y}} = [\mathbf{1}\,\mathbf{x}] \cdot \mathbf{w}, \tag{4.9}$$

where $[\mathbf{1}\,\mathbf{x}]$ denotes the matrix composed by the concatenation of a column of ones in the first colum and $\mathbf{x}$, to introduce a constant term (bias); the Equation 4.9 is usually solved under the constraint of minimizing the Squared Error between $\hat{\mathbf{y}}$ and the real output $\mathbf{y}$

$$\varepsilon = \|[\mathbf{1}\,\mathbf{x}] \cdot \mathbf{w} - \mathbf{y}\|_2^2, \tag{4.10}$$

where $\|\cdot\|_2$ is the Euclidean norm. This solution, named *Least Squares regression solution* (also called *Ordinary Least Squares solution* (OLS)) is given by

$$\mathbf{w} = \left([\mathbf{1}\,\mathbf{x}]^\mathsf{T} \cdot [\mathbf{1}\,\mathbf{x}]\right)^{-1} [\mathbf{1}\,\mathbf{x}]^\mathsf{T} \cdot \mathbf{y}. \tag{4.11}$$

The classical regression can also use a regularization parameter for ill-conditioned cases.

In the case where the matrix $\mathbf{x}$ is ill-conditioned, the Ridge regression (or Tikhonov regularization [126, 15]) can be used, which introduces a regularization term in the Squared Error (training error)

$$\varepsilon_{\mathtt{Tikh}} = \|[\mathbf{1}\,\mathbf{x}] \cdot \mathbf{w} - \mathbf{y}\|_2^2 + \|\mathbf{T} \cdot \mathbf{w}\|_2^2. \tag{4.12}$$

Usually, $\mathbf{T}$ is chosen to be $\alpha\mathbf{I}$, with $\mathbf{I}$ the identity matrix and α a scalar. From Equation 4.12 it is clear that the larger is the regularization term, the larger the training error will be, which reduces the risks of over-fitting.

This approach, although very simple and basic, usually deems decent results, even compared with more elaborate models [103].

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is widely used in the steganalysis field ([48, 87, 3]) for its good capacity at discriminating between two classes and its rather low computational requirements.

In LDA, the data is projected onto a hyperplane $\mathbf{w}$ (to determine) and a separating discriminant hyperplane is then found.

LDA makes the assumption that the probability density functions $P(\mathbf{x}|y = c_1)$ and $P(\mathbf{x}|y = c_2)$ for the two classes c_1 and c_2 (we consider only a binary classification problem) are normally distributed such that

$$\begin{aligned} P(\mathbf{x}|y = c_1) &\sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \\ P(\mathbf{x}|y = c_2) &\sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2), \end{aligned} \tag{4.13}$$

with $\boldsymbol{\mu}_j$ the mean and $\boldsymbol{\Sigma}_j$ the covariance for each of the classes (computed from $\mathbf{x}$). With the additional assumption that both covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are identical (*homoscedasticity* assumption) and full rank, we have that the maximum separation between the two classes occurs for

LDA uses probabilistic assumptions on the distribution of the classes.

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1), \tag{4.14}$$

where $\mathbf{w}$ is the normal to the discriminant hyperplane.

In practice, $\mathbf{x}$ is first projected onto $\mathbf{w}$ and then the threshold T such that $\mathbf{x} \cdot \mathbf{w} < T$ — defining the position of the hyperplane orthogonal to $\mathbf{w}$, and hence the position of the actual discriminant — is optimized. The following Support Vector Machine uses a similar idea of discriminating hyperplane but uses kernels to make the data more separable, with no particular assumption on the probability density function of the classes.

Support Vector Machine

The Support Vector Machine was originally proposed for the classification problem in [19] (the earlier version of [129] does not use kernels, which are contributing largely to the efficiency of the SVM), and was later extended to regression problems [37].

In the following, we denote $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T$ with $\mathbf{x}_j \in \mathbb{R}^M$ the inputs and similarly for $\mathbf{y} = (y_1, \ldots, y_N)^T$, $y_j \in \mathbb{R}$, the outputs.

The goal of SVM for classification (considering we use a kernel function $\mathcal{K}$) is to separate the two classes c_1 and c_2 in the so-called *feature space* defined by the kernel $\mathcal{K}$, using a hyperplane $\mathbf{w}$. The use of a kernel is meant to make the data $\mathbf{x}$ more separable in the feature space (induced by the kernel) than it is in the original space. In this feature space, one tries to find the optimal hyperplane $\mathbf{w}$ to separate the two classes, which comes down to solving the optimization problem

SVM looks for the best separating hyperplane in the induced kernel space.

$$\begin{aligned} &\min_{\mathbf{w},b,\xi} \ \|\mathbf{w}\|_2^2 + C\sum_{j=1}^{N} \xi_j^2 \\ &\text{s.t.} \begin{cases} y_j\left(\mathbf{w}^T \cdot \psi(\mathbf{x}_j) + b\right) \geqslant 1 - \xi_j, \ j \in [\![1, N]\!] \\ \xi_j \geqslant 0, j \in [\![1, N]\!] \end{cases} \end{aligned} \tag{4.15}$$

with $\mathcal{K}(\mathbf{x}_j, \mathbf{x}_k) = \psi(\mathbf{x}_j)^T \psi(\mathbf{x}_k)$, C the complexity of the model and ξ_j the *slack* variables which allow to solve the problem even if the two classes are not fully separable in the induced kernel space. This specific case of the SVM is called *soft margin SVM classifier* (due to the slack variables ξ_j) and the formulation in Equation 4.15 assumes that the two classes are coded in $\mathbf{y}$ as

$+1$ and -1. The parameter C controls the trade-off between the classification error and the complexity of the model.

There are very efficient ways to solve this optimization problem, for example the Sequential Minimization Optimization (SMO) algorithm [43], but the fact remains that for the classification problem considered here, there are already at least two hyper-parameters to select in this model: the hyper-parameters for the kernel function $\mathcal{K}$ (usually a Gaussian one, with only the width of the Gaussian σ to determine) and C. Using SVM for regression adds an additional hyper-parameter to optimize, making the computational load rather important for the learning of this model.

LS-SVM has a simpler formulation than the standard soft-margin SVM.

The Least-Squares Support Vector Machine [122] (LS-SVM) approach makes the optimization problem a little simpler as Equation 4.15 is replaced by a similar minimization problem, with an equality constraint

$$\begin{aligned} &\min_{\mathbf{w},b,e} \ \|\mathbf{w}\|_2^2 + \gamma \sum_{j=1}^{N} e_j^2 \\ &\text{s.t. } y_j \left(\mathbf{w}^T \psi(\mathbf{x}_j) + b\right) = 1 - e_j,\ j \in [\![1, N]\!] \end{aligned} \tag{4.16}$$

which can be solved in a computationally more efficient way than the previous one in Equation 4.15, in a Least Square sense. The e_j are error variables which allow for some misclassification as for the slack variables ξ_j of the SVM.

Thanks to the use of a kernel, the discriminant hyperplane of the SVM model is very efficient for many problems (see publication A, section III.B for a comparison of SVM with other models on various classification and regression data sets), but the computational time for the learning and validation usually suffers from the complexity and number of hyper-parameters of it. Some of the proposed recent improvements both in the SMO algorithm [75] and for approximating the hyperplane [22] could make the optimization of the parameters faster.

4.3.2 *Artificial Neural Networks*

ANNs are built to work similarly to biological neural networks.

The development of Artificial Neural Networks (ANN) was inspired by biological systems [63]. In this dissertation, we will limit ourselves to the case of Single-Layer Feedforward Neural Network (SLFN) for simplicity. Since ANNs can have a rather complicated structure (more than one hidden layer, recurrence, feedback, ...), with many hyper-parameters to optimize, they are often used in various applicative domains, such as air quality management [125] and finance [135], for example.

A notable point is that the SLFN (one hidden layer, feedforward strucutre) is an *universal function approximator*, meaning that given an error $\epsilon > 0$, there exists a set of hyper-parameters Θ for the SLFN such that

$$\forall \mathbf{x}_j \in \mathbf{x}, \left|\text{SLFN}\left(\Theta, \mathbf{x}_j\right) - f\left(\mathbf{x}_j\right)\right| < \epsilon, \tag{4.17}$$

with f being the function to approximate (continuous) [66, 63].

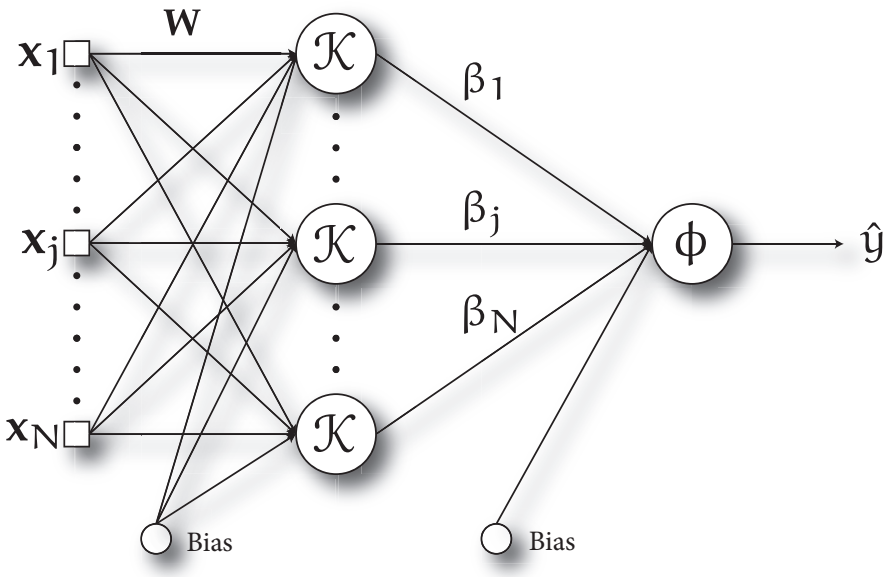

Figure 13: Classical structure of a Single Hidden Layer Feedforward Neural Network.

Figure 13 illustrates the case of a SLFN, with the *input layer* being the $\mathbf{x}_j$, the *hidden layer* containing a non-linear function $\mathcal{K}$, and the *output layer* calculated through a function ϕ, which we will consider as linear.

We consider the specific case of the SLFN here.

Overall, the estimated output $\hat{y}_v$ for sample $\mathbf{x}_v$ in a SLFN with L neurons is computed as

$$\hat{y}_v = \sum_{j=1}^{L} \boldsymbol{\beta}_j \mathcal{K}\left(\mathbf{w}_j \mathbf{x}_v + b_j\right), \tag{4.18}$$

with b_j being the biases and provided that the output layer is composed of a linear function ϕ (typical for regression and classification problems [123]). A typical choice for the $\mathcal{K}$ function is that of hyperbolic tangent tanh,

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{4.19}$$

The parameters of this model are numerous: the output weights $\boldsymbol{\beta} = \left(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{N_l}\right)^T$, the biases $\mathbf{b} = \left(b_1, \ldots, b_{N_l}\right)^T$ and the input weights $\mathbf{W} = \left(\mathbf{w}_1, \ldots, \mathbf{w}_{N_l}\right)^T$. The optimization problem resulting from the search of the parameters can be solved in various ways of which the most notable ones are probably back-propagation [63], Levenberg-Marquardt [14] or the scale conjugate gradient approach [93].

SLFN (and ANN in general) have many parameters to determine.

Given that the non-linear function $\mathcal{K}$ is determined, this class of model has only one hyper-parameter: the number of neurons L to use.

A notable specific architecture of ANN is the Radial Basis Function Network (RBFN) [105], which is a SLFN with a Radial Basis Function (RBF) as the activation non-linear function $\mathcal{K}$ (Radial Basis Functions are a class of functions such that $\mathcal{K}(\mathbf{x}) = \mathcal{K}(\|\mathbf{x}\|)$, with $\|\cdot\|$ a norm). This specific structure

can be trained in two parts : an unsupervised part, to determine the input weights $\mathbf{W}$ and a linear supervised part, where the output weights $\boldsymbol{\beta}$ are devised (no bias is considered in this model). Some heuristics and fast algorithms can be used to determine all the parameters of the model efficiently [12].

Again, one drawback to the efficiency of such models is the computational time required to search for the optimal parameters (hoping that the searching algorithm has not hit a local minima).

4.3.3 k-*Nearest Neighbors*

The k-Nearest Neighbors (KNN) is one of the simplest possible models, based on ranking of distances between samples.

In the case of binary classification, it consists in finding the set of k nearest neighbors (k is odd)

$$\mathbf{x}_{kNN}(\mathbf{x}_v) = (\mathbf{x}_{NN1}(\mathbf{x}_v), \ldots, \mathbf{x}_{NNk}(\mathbf{x}_v))^T, \tag{4.20}$$

for the point $\mathbf{x}_v$; that is, for the first neighbor $\mathbf{x}_{NN1}$,

$$\mathbf{x}_{NN1} = \min_{\mathbf{x}_j \in \mathbf{x}} d(\mathbf{x}_v, \mathbf{x}_j), \tag{4.21}$$

where $d(\cdot,\cdot)$ is a distance (typically the Euclidean one). The following nearest neighbors are found similarly. The estimated class for $\mathbf{x}_v$ is then obtained by a majority vote over the set of its k nearest neighbors $\mathbf{x}_{kNN}(\mathbf{x}_v)$.

k-NN is a distance-based technique, with ranking of the points.

The approach is similar for the regression problem, except that an average of the values of the nearest neighbors is used to predict the output $\hat{y}_v$,

$$\hat{y}_v = \frac{1}{k}\sum_{j=1}^{k} y_{NNj}, \tag{4.22}$$

with y_{NNj} the outputs corresponding to the nearest neighbors $\mathbf{x}_{NNj}$.

An advantage of the KNN is its simplicity: it only has one parameter to optimize, being the number of neighbors k which is clearly bounded. Also, although it relies on distances (and hence, can be subject to the problems arising in high-dimensional spaces), it only uses them for *ranking* purposes, which makes it a fast and reasonably efficient model in such spaces [13].

LVQ uses prototypes (codebooks) to cluster and sort data.

The *Learning Vector Quantization* model (LVQ) [82] lies somewhere between the k-NN approach and the ANN one. It uses so-called *codebook vectors* (which are prototypes of a certain class or cluster of data) which are updated according to their matching the class of the nearest neighbors (for a classification problem): if the nearest neighbor of a codebook has the right class, then it is likely that this codebook is in the proper "cluster" of points for this class and should be driven toward the "center" of the cluster. If not, it should be driven outward of the current cluster. Various update rules and extensions have been proposed for the original LVQ algorithm, of which the *Generalized LVQ* (GLVQ) [112], the *Relevance LVQ* (RLVQ) [17] and the one combining both improvements, the *Generalized Relevance LVQ* (GRLVQ) [60, 61] are possibly

the most used. A review of the state of the art can be found in [60], along with references to most of the recent developments on LVQ.

SOM also relies on prototypes and updates the array nodes according to the data.

Also, the *Self-Organizing Maps* (SOM) [81, 82] for regression problems [85] are quite similar to the Vector Quantization (VQ) concept (original unsupervised version of the LVQ), in the sense that the nodes of the array of the SOM are updated in a similar fashion to that of the codebook vectors of the VQ. The SOM provides, thanks to this nodes array structure, a two-dimensional map of much higher dimensional data. Please refer to [82] for a reference on the SOM.

Finally, a variant of the k-NN using a weighted sum in Equation 4.22 is proposed in [135, 136]: the OP-KNN. The approach is similar to that of the Optimally-Pruned Extreme Learning Machine (OP-ELM) presented in the next chapter 5, section 5.3, with the use of k-NN "neurons".

4.3.4 *Gaussian Processes*

The underlying idea in Gaussian Processes (GP) [108] is to make the assumption that the output $\mathbf{y}$ can be modeled by a multivariate Gaussian distribution (N-variate, in this case). Therefore, since "Gaussian processes generate data located throughout some domain such that any finite subset of the range follows a multivariate Gaussian distribution" (from [108]), it becomes possible to make $\mathbf{y}$ the result of a GP. In a rather general formulation, the output $\mathbf{y}$ is considered to be the result of a function f applied on the input $\mathbf{x}$ plus some zero-mean noise (as in Equation 4.2)

$$\mathbf{y} = f(\mathbf{x}) + \mathcal{N}\left(0, \sigma^2\mathbf{I}\right), \tag{4.23}$$

where $\mathcal{N}\left(0, \sigma^2\mathbf{I}\right)$ denotes the zero-mean noise and f the underlying process in data $\mathbf{x}$. Now in the case of predicting the value y_ν for a point $\mathbf{x}_\nu$ of $\mathbf{x}$, the covariance matrix $\mathbf{K}$ is first computed from the covariance function $\mathcal{K}\left(\mathbf{x}_j, \mathbf{x}_k\right)$ defined here as (classically a Radial Basis Function (RBF))

$$\mathcal{K}\left(\mathbf{x}_j, \mathbf{x}_k\right) = \sigma_0^2 \exp\left(-\frac{1}{2\lambda}\left|\mathbf{x}_j - \mathbf{x}_k\right|^2\right). \tag{4.24}$$

Note that for this type of covariance function, there are two hyper-parameters to optimize, λ and σ_0, which can be optimized using Bayesian inference — maximization of a marginal likelihood — if the required computations are feasible (requires matrix inversions), or by cross-validation.

A probabilistic framework is also used for the Gaussian Processes.

The joint marginal likelihood is then

$$P(\mathbf{y}, y_\nu) = \mathcal{N}\left(0, \mathbf{K}_{N_l+\nu} + \sigma^2\mathbf{I}\right), \tag{4.25}$$

with

$$\mathbf{K}_{N_l+\nu} = \begin{bmatrix} \mathbf{K}_{N_l} & \mathbf{K}_{\nu N_l}^T \\ \mathbf{K}_{\nu N_l} & \mathbf{K}_\nu \end{bmatrix}, \tag{4.26}$$

and denoting

$$\mathbf{K}_{N_l} = \begin{bmatrix} \mathcal{K}(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \mathcal{K}(\mathbf{x}_1, \mathbf{x}_{N_l}) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\mathbf{x}_{N_l}, \mathbf{x}_1) & \cdots & \mathcal{K}(\mathbf{x}_{N_l}, \mathbf{x}_{N_l}) \end{bmatrix}, \tag{4.27}$$

$\mathbf{K}_{\nu N_l} = \left[\mathcal{K}(\mathbf{x}_\nu, \mathbf{x}_1), \ldots, \mathcal{K}(\mathbf{x}_\nu, \mathbf{x}_{N_l})\right]$ and $\mathbf{K}_\nu = [\mathcal{K}(\mathbf{x}_\nu, \mathbf{x}_\nu)]$. From which we can get the predicted output value $\hat{y}_\nu$ as

$$\hat{y}_\nu = P(y_\nu|\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_\nu, \boldsymbol{\Sigma}_\nu), \tag{4.28}$$

The prediction is obtained from a Gaussian distribution with specific parameters.

with

$$\begin{aligned} \boldsymbol{\mu}_\nu &= \mathbf{K}_{\nu N_l}\left[\mathbf{K}_{N_l} + \sigma^2\mathbf{I}\right]^{-1}\mathbf{y} \\ \boldsymbol{\Sigma}_\nu &= \mathbf{K}_\nu - \mathbf{K}_{\nu N_l}\left[\mathbf{K}_{N_l} + \sigma^2\mathbf{I}\right]^{-1}\mathbf{K}_{\nu N_l}^T + \sigma^2\mathbf{I} \end{aligned}. \tag{4.29}$$

As can be seen from Equation 4.29 this requires matrix inversions, which can be very costly when working with large number of samples. Overall, as the experiments in publication A (section III.B) illustrate, the GP model is very efficient (often the best performances for the benchmark in A, Table IV) although costly, computationally speaking.

4.3.5 *A global drawback*

As we have noted in the description of these models, there are many hyper-parameters (and parameters) to optimize (putting aside the LDA and the linear regression). For models such as the ANN, where the overall direct computations of the model are simple, the amount of parameters (internal and hyper-) makes the learning and validation phase about as long as that of models with less parameters but larger direct computational times (the GP, for example).

The problem of most methods is the computational time: many parameters (and hyper-parameters) or costly training.

The linear regression and LDA are outliers in this reasoning, for they have virtually no hyper-parameters (the threshold for the LDA can be found quickly and does not need to compute again the inverse of the covariance matrix), given a relevant set of variables. Their learning and validation phase are thus very fast, compared to other models, but in the end, they do not give the best possible performance on non-linear problems.

In the end, there seems to be a tradeoff in choosing the model, between the computational difficulty of it and the number of parameters and hyper-parameters to optimize. One has to decide beforehand whether the model should be trained very quickly — and have suboptimal performances in case of non-linearity in the data — or if the training time does not matter as long as the performances are here.

4.4 CONCLUSION

The *Machine Learning* field aims at *learning* from the data it is being presented. In this dissertation, we consider the more specific case of *supervised learning*,

in which the model is presented with the input data and an output — obtained from a specific phenomenon — from which it can learn a relationship between input and output, and hopefully predict the future behavior of the underlying phenomenon.

The matter of training, validating and testing a model properly is a non-trivial issue, as seen in this chapter. Obtaining the optimal set of *hyper-parameters* for a chosen class of model requires many iterations, on a data set pre-processed in order to avoid the problem of *over-fitting*, for example (see Figure 12 for the proposed example of data processing and model training/testing).

This procedure, while having the advantage of giving reliable estimates for the *validation* and *test error* of a model on a data set, usually requires an important computational time, especially for models having many *hyper-parameters*.

In the following chapter 5, we present a class of models based on random projections which have the advantage of being very fast and simple to train, while keeping performances similar to that of state of the art algorithms. We claim that this class of models offers a very good (if not the best) ratio between performances and computational time and illustrate it throughout a set of experiments.

5 THE OPTIMALLY-PRUNED EXTREME LEARNING MACHINE

In this chapter, we propose a new machine learning model (from publications A and B) which intends to have a high *performance/ speed ratio*. We first motivate more precisely this requirement of speed for machine learning models, and then present the concept of random projections, which is used in the proposed Optimally-Pruned Extreme Learning Machine (OP-ELM). We finally detail the different steps of the OP-ELM model methodology and provide a speed improvement originally described in publication B.

5.1 A NEED FOR SPEED (AND EFFICIENCY)

As will be illustrated in more details in the next chapter 6, a large number of features requires a large number of samples: in theory, the number of samples grows exponentially with the number of features. If we consider the current typical amount of features (in the order of hundreds) used for steganalysis, the theoretical requirements concerning the number of samples are simply unachievable. The best that can be done is to use as many samples as possible, and estimate the reliability of the results, statistically.

Data sets can be large and models too slow for them.

Taking the examples provided by Guang-Bin Huang on [67], the training time (with a fixed set of hyper-parameters) for a data set consisting of 54 features and $100,000$ samples (the Forest Type prediction, a classification problem) is of 694 minutes for a SVM — using the LIBSVM implementation [25]. In the case of steganalysis, if using the DCT feature set from [99], we have 194 features, which would probably lead to much larger computational times, for the SVM (assuming the amount of samples is sufficient *vis a vis* the dimensionality).

The bottom line is that models with multiple hyper-parameters or large computational times are not practical when dealing with very large data sets. Optimizing the parameters for some models is sometimes barely feasible within days, let alone train many of them, for cross-validation or feature selection for example. For the example of the SVM on the Forest data, running a search for the hyper-parameters on a 20×20 grid (each hyper-parameter is tested on 20 different values) with such computational time is impossible.

The concept of feature selection and its importance are detailed more widely in section 6.1, but in most cases, performing it requires multiple trainings of the models, on different subsets of the whole feature set, to evaluate the performance for a specific choice of features. This, combined with the cross-validation and cross-test approaches makes the use of some models totally impractical: it would take many weeks to perform feature selection (whatever the scheme) on a set of $N = 10,000$ samples with $M = 194$ features using a classical SVM (even in classification, with only two hyper-parameters).

In some cases, one could consider that "infinite" computational power is available, and that the time required for the training of a model is an

irrelevant issue. For the example of a government agency willing to perform qualitative steganalysis, this can be a reasonable assumption: if the quality of the model and the set of features have already been asserted, there is no need to perform feature selection or cross-validation, and the only task remaining is training the model with the right parameters (which might already be known). In such a case, the learning time has indeed no relevance; once the model is trained, it can be used *ad libitum* to detect steganography, without any costs in computational time (or negligible one, the test phase always being very fast).

The infinite computational power assumption is problematic.

One detail to remember, though, is that the determination of the set of features (which will also be referred to as *variables* in the following) and of the model class and parameters has to be done, at some point. While this concern is eventually of minor importance to the government agency, it still is a major issue to the researcher performing this search.

In the end, the computational time (related to the complexity of the model) is a very important characteristic of a model, and should be taken into consideration when performing classification or regression.

In the following, we first present a class of models related to *random projections*, which offer a very good ratio performance/ speed, and then the Optimally-Pruned Extreme Learning Machine, an improvement of a classical method — the Extreme Learning Machine — which provides additional robustness.

5.2 EXISTING RECENT RANDOM PROJECTION BASED MODELS

One reason for the concern about the speed of the models is that of the number of features, as mentioned before. An obvious solution to this problem is to reduce the dimensionality of the space before training the model. Although the idea is simple, the realization is not: optimizing the whole projection matrix $\mathbf{P}$ such that the projected data $\mathbf{x}_P = \mathbf{P} \cdot \mathbf{x}$ gives an optimal result for the considered model $\mathcal{M}$ is computationally *very* expensive.

The J-L theorem states that a lower dimensional space can be found without losing too much information.

Random projections are a fast alternative to this. For the matter of projecting to a lower dimensional space, Johnson and Lindenstrauss in [72] have shown that for a set of N points in M-dimensional space (considering the distance measure as an Euclidean norm), there exists a linear transformation of the data toward a M_f-dimensional space, with

$$M_f \geqslant O\left(\epsilon^{-2} \log N\right), \tag{5.1}$$

which preserves the distances to a $1 + \epsilon$ factor. The topology of the data might also be preserved through this projection, such that the relevant information is not "lost". In [2], Achlioptas extends this result and proposes an algorithm to devise a very simple projection matrix that preserves the distances to the same factor than the J-L theorem mentions. This unfortunately comes at the expense of a probability on the distance conservation.

More recently, Fradkin in [44] showed that simple random projections obtained by one the methods proposed by Achlioptas in [2], deem decent results on data sets obtained from the UCI Machine Learning repository [8], yet behind the ones obtained using a simple Principal Component Analysis

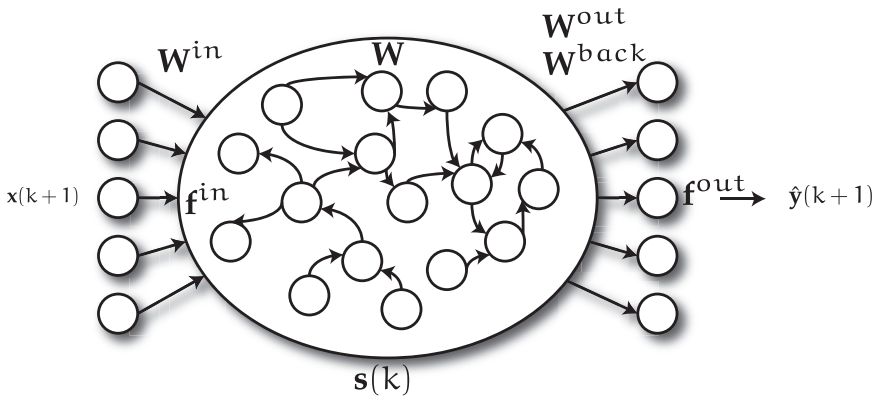

Figure 14: Illustration of the Reservoir Computing concept: a snapshot of the "pool" of interconnected neurons (in the middle) is taken (noted $\mathbf{s}(k)$) at step k and the output layer $\mathbf{W}^{out}$ is devised from the state $\mathbf{s}(k)$.

(PCA) [96]. The article concludes by stating that random projections are an interesting possibility if one is especially interested in obtaining results in a short time frame, but remains a suboptimal approach.

In this dissertation, we focus mostly on two frameworks based on random projections: the Reservoir Computing (RC) and the Extreme Learning Machine (ELM) [92]. For a wider presentation of random projection based models, one can refer to [130].

5.2.1 *Reservoir Computing*

The terminology *Reservoir Computing* (RC) — and the underlying framework [131] — is proposed to unify a panel of models such as Liquid-State Machines (LSM) [88] or Echo State Networks (ESN) [70, 71, 21], to name only these. The concept behind the RC approach (which is similar to that of the ELM presented below) is to randomly initialize a neural network (with recurrences and feedback, here), and only "train" the output weights, that is, determine them directly from the state of the network.

The training of the model is therefore very fast, since all the internal weights are initialized randomly and do not need to be determined.

Reservoir Computing aims at time-varying processes.

Reservoir Computing has been specifically meant for time-varying processes and data. In this sense, let us denote by $\mathbf{x}(k)$ the input data $\mathbf{x}$ at time step k and by $\mathbf{y}(k)$ the output data $\mathbf{y}$ at step k. The RC model uses a multi-layer (no longer a SLFN) ANN with recurrence in the hidden layers, and a possible feedback from the output. The inner layers of the ANN, interconnected, are typically referred to as a *pool* of neurons.

Figure 14 depicts the global structure of the most general case of Reservoir. Suppose we have the input data $\mathbf{x}(k) = (\mathbf{x}_1(k), \ldots, \mathbf{x}_N(k))^T$ and outputs $\mathbf{y}(k) = (y_1(k), \ldots, y_N(k))^T$ for time step k, the estimated output value $\hat{\mathbf{y}}(k+1)$ for the next step input $\mathbf{x}(k+1)$ is then obtained by taking an image $\mathbf{s}(k)$ of the pool of neurons, weighting it by the randomly initialized matrix $\mathbf{W}$, added

to the feedback from $\mathbf{y}(k)$ and the current weighted input $\mathbf{f}^{in}\left(\mathbf{W}^{in}\cdot\mathbf{x}(k+1)\right)$ as

$$\begin{aligned}\hat{\mathbf{y}}(k+1) = \quad & \mathbf{f}^{out}\left(\mathbf{W}^{out}\cdot\left[\mathbf{x}(k+1),\mathbf{f}^{in}\left(\mathbf{W}^{in}\cdot\mathbf{x}(k+1)\right.\right.\right.\\ & \left.\left.\left.+\mathbf{W}\cdot\mathbf{s}(k)+\mathbf{W}^{back}\cdot\mathbf{y}(k)\right),\mathbf{y}(k)\right]\right),\end{aligned} \tag{5.2}$$

with $\mathbf{W}^{out}$,$\mathbf{W}^{in}$, $\mathbf{W}$ and $\mathbf{W}^{back}$ the output weight matrix, input weight matrix, internal weight matrix and back-projection of output to internal network weight matrix respectively (random), $\mathbf{f}^{in}$ and $\mathbf{f}^{out}$ the internal network activation function (usually sigmoid) and readout function respectively and finally $\mathbf{s}$ denoting the internal network state.

It uses ANN with recurrence and feedback to model the output.

With certain restrictions and specificities on some weight matrices and functions, one can again find from this global definition, the ESN and LSM models, for example. One can refer to [131] for more details on these models.

As it stands out from the definition of the Reservoir Computing approach, this solution is especially meant for time-variant problems, and although it can be applied to classical regression and classification problem, the following Extreme Learning Machine is more straightforward to use in such a setup.

5.2.2 *ELM based*

The idea for the Extreme Learning Machine (ELM) by Guang-Bin Huang [69] is in substance similar to that of the Reservoir Computing: since the optimization of an ANN is very time-consuming, initialize the internal weights of the network randomly and only "train" the output layer of the network.

ELM is similar to RC: random initialization and output layer determination.

The main difference in the ELM concept is that the ANN is a SLFN, that is, there is no recurrence and only one hidden layer in the ANN. The structure is thus the same as that presented in section 4.3.2, on Figure 13.

Using the same notations, we have the inputs $\mathbf{x} = (\mathbf{x}_1,\ldots,\mathbf{x}_N)^T$, the outputs $\mathbf{y} = (y_1,\ldots,y_N)^T$, the output layer $\boldsymbol{\beta} = (\beta_1,\ldots,\beta_N)^T$, a linear output function ϕ, the input weight matrix $\mathbf{W}$ and the non-linear activation function $\mathcal{K}$; the estimated output $\hat{y}_\nu$ is then obtained for point $\mathbf{x}_\nu$ as

$$\hat{y}_\nu = \sum_{j=1}^{N}\beta_j\mathcal{K}\left(\mathbf{w}_j\cdot\mathbf{x}_\nu+b_j\right). \tag{5.3}$$

In the best possible case were $y_\nu = \hat{y}_\nu$ (the SLFN makes a perfect approximation) for all possible $\mathbf{x}_\nu \in \mathbf{x}$, we can express the Equation 5.3 as the matrix product

$$\mathbf{y} = \mathbf{H}\cdot\boldsymbol{\beta}, \tag{5.4}$$

with

$$\mathbf{H} = \begin{pmatrix} \mathcal{K}(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & \mathcal{K}(\mathbf{w}_N \cdot \mathbf{x}_1 + b_N) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & \mathcal{K}(\mathbf{w}_N \cdot \mathbf{x}_N + b_N) \end{pmatrix}. \tag{5.5}$$

Using these notations, Huang in [69] demonstrates the following theorem, stating that given an activation function $\mathcal{K}$ infinitely differentiable, it is possible to find the output weights β of the SLFN with random input weights $\mathbf{W}$ such that it approximates as well as possible ($\varepsilon > 0$) the output $\mathbf{y}$.

Theorem. *Given any $\varepsilon > 0$ and an activation function $\mathcal{K} : \mathbb{R} \mapsto \mathbb{R}$ infinitely differentiable in any interval, there exists $n < N$ such that for N distinct samples $(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^M, y_i \in \mathbb{R}$, for any weights $\mathbf{w}_i \in \mathbb{R}^M$ and biases $b_i \in \mathbb{R}$, $\left\| \mathbf{H}_{[N \times n]} \beta_{[n \times 1]} - \mathbf{y}_{[N \times 1]} \right\| < \varepsilon$.*

The solution to Equation 5.4 in terms of Least Squares can then be obtained by a Moore-Penrose pseudo-inverse [107].

ELM is also an universal approximator.

It was later proved in [68] that the ELM, as the SLFN, is an *universal approximator of functions*.

Recently, improvements to the original ELM have been proposed, such as the Error Minimized Extreme Learning Machine (EM-ELM) [41]; it proposes to add random neurons (one-by-one or group-by-group) to the original ELM model once the training has been performed. The authors prove that this EM-ELM actually converges. One of the most interesting points of this development is in the fact that the computational time is minimal while adding new random neurons to the ELM: the whole pseudo-inverse matrix does not need to be recomputed, but weights only have to be updated using fast update rules derived in the paper. It becomes not so important to have a good (sufficient) number of neurons in the first training phase, since they can anyway be added to the ELM without long computations afterwards.

The proposed Optimally-Pruned Extreme Learning Machine (OP-ELM, publications A and B) in the following section 5.3 uses a different approach than that of the EM-ELM: instead of adding neurons to the original model until the performances are satisfying enough, the idea is to take a large number of them in the first place, and prune out the most irrelevant ones, using a selection criterion.

5.3 OP-ELM

In this section, we present shortly the Optimally-Pruned Extreme Learning Machine (OP-ELM) model. For more details and especially experiments and benchmarks of this new method against state of the art ones (on publicly available data sets from UCI repository [8]), please refer to the publications included in this dissertation, A and B.

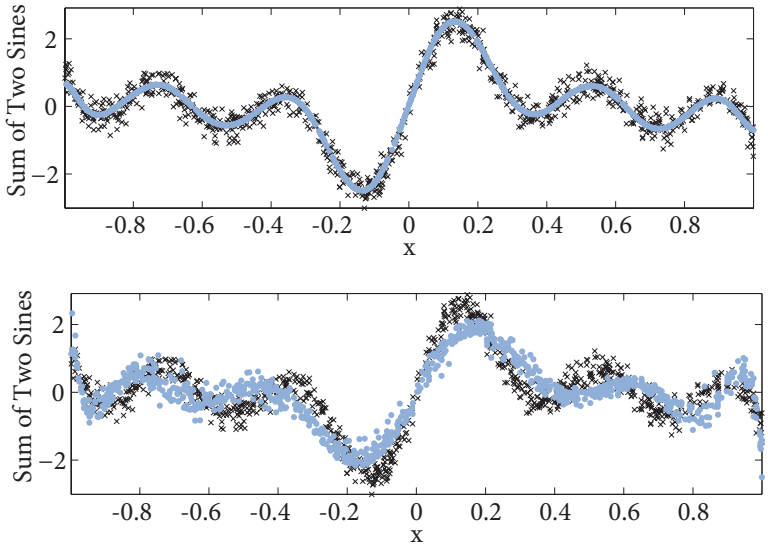

Figure 15: Example of the perturbation of the ELM model by the use of an additional irrelevant variable: ELM model is depicted in light dots, and the data itself in dark crosses.

5.3.1 *Some possible limitations of the ELM*

The elaboration of this model started as the realization that the original ELM model was sometimes behaving poorly on data sets containing irrelevant variables: the simple addition of a feature containing pure Gaussian noise was enough to perturbate the model and yield highly suboptimal performances.

ELM can be sensitive to irrelevant data.

A toy example from publication A (Figure 2) illustrates this clearly on a one-dimensional example. Figure 15 is extracted from the original publication.

The top plot on Figure 15 illustrates an ELM model trained on a sum of sines, without any other variable. The fit is good. When adding a noise variable to the data (not plotted for clarity), the ELM model on the bottom plot becomes pertubated, although the variable does not contain any information.

It appears that the ELM needs a step of feature selection, in order to remove such irrelevant variables. This can be performed by pruning the irrelevant neurons of the hidden layer.

5.3.2 *A methodology around ELM: OP-ELM*

The OP-ELM can be described as a methodology surrounding ELM. It uses three main steps which can be summarized as (see Figure 16):

The OP-ELM is a methodology around ELM to address its weaknesses.

1. Building a SLFN with a large number of neurons, using the ELM approach (i.e., randomized weight matrix);
2. Ranking the neurons by the use of the Multi-Response Sparse Regression algorithm, which provides an exact ranking of the neurons in this case;

Figure 16: The three steps of the OP-ELM methodology: construction of the SLFN using ELM; ranking of best neurons using MRSR; use of LOO criterion to decide how many neurons are kept.

3. Use a Leave-One-Out error estimation to choose how many of the hidden neurons should remain in the final model.

Using MRSR to rank neurons and LOO to decide which ones to keep.

The Multi-Response Sparse regression (MRSR) [117] is a computationally efficient extension of the famous Least Angle Regression (LARS) [40] to the multi-output case. The original LARS algorithm is a greedy variable selection scheme which selects variables one by one (according to a Least Squares criteria) and has the advantage of giving the best possible ranking of variables, in the specific case where the problem considered is linear. This is the case in the OP-ELM architecture, since the output is a linear combination of the hidden layer neurons. Hence, it provides the best possible ranking of the hidden neurons and makes the last step of thresholding easier.

In the third step, the performance of the SLFN with an increasing number of neurons (previously ranked) is devised, using a LOO error criterion. Again, since the problem is linear in this part of the SLFN, it is possible to use a fast estimation of the LOO error thanks to a closed form formula [94, 18]. The final structure of the OP-ELM is then obtained.

It should be noted also that the OP-ELM proposes to use also linear kernels, in addition to the classical Gaussian and sigmoid ones proposed in the original ELM.

In the end, the OP-ELM can be seen as a projection into a high dimensional space (hidden layer of the SLFN), followed by a linear model (output layer of the SLFN) on which variable selection is performed, by the MRSR. Variable selection techniques are detailed more widely in section 6.1.

Some of the results presented in publication A (Tables II, III, IV and V, especially) are reproduced here: Tables 3 and 5 (left) give the average computational times performed by five state of the art machine learning algorithms while Tables 4 and 5 (right) give the Mean Square Error and accuracy obtained on regression and classification data sets, respectively. Results are the average of ten bootstrap rounds, as detailed in publication A (section III.B).

In addition to the results of publication A, the OP-ELM and the SVM are compared here over two classical steganalysis problems: a regression one over the estimation of the embedding rate (initially between 0 and 30%, in the $R^{(1)}$ sense, see 2.3.2) and a classification one. The images used for this experiment are the whole 10000 from the BOWS2 [10] database, with two thirds used for training and one third for testing, using the same bootstrapping approach as in publication A. Proportions of half stego and half cover have been respected for the classification problem. Features used for both problems are the extended DCT features from [99].

	Abalone	Elevators	Servo	Bank	Stocks	Boston	Steg.
SVM	6.6e+4	5.8e+2	1.3e+2	1.6e+3	2.3e+3	8.5e+2	1.7e+5
MLP	2.1e+3	3.5e+3	5.2e+2	2.7e+3	1.2e+3	8.2e+2	
GP	9.5e+2	6.5e+3	2.2	1.7e+3	4.1e+1	8.5	
OP-ELM	5.7	29.8	2.1e-1	8.03	1.54	7.0e-1	2.4e+2
ELM	4.0e-1	1.6	3.9e-2	4.7e-1	1.1e-1	7.4e-2	

Table 3: Computational times (in seconds) for all five algorithms on regression data sets. Results are the average of ten bootstraps repetitions.

	Abalone	Elevators	Servo	Bank	Stocks	Boston	Steg.
SVM	**4.5**	**6.2e-6**	**6.9e-1**	**2.7e-2**	**5.1e-1**	**3.4e+1**	**2.5e-4**
	2.7e-1	6.8e-7	3.3e-1	8.0e-4	9.0e-2	3.1e+1	1.0e-5
MLP	**4.6**	**2.6e-6**	**2.2e-1**	**9.1e-4**	**8.8e-1**	**2.2e+1**	
	5.8e-1	9.0e-8	8.1e-2	4.2e-5	2.1e-1	8.8	
GP	**4.5**	**2.0e-6**	**4.8e-1**	**8.7e-4**	**4.4e-1**	**1.1e+1**	
	2.4e-1	5.0e-8	3.5e-1	5.1e-5	5.0e-2	3.5	
OP-ELM	**4.9**	**2.0e-6**	**8.0e-1**	**1.1e-3**	**9.8e-1**	**1.9e+1**	**1.6e-4**
	6.6e-1	5.4e-8	3.3e-1	1.0e-6	1.1e-1	2.9	1.4e-5
ELM	**8.3**	**2.2e-6**	**7.1**	**6.7e-3**	**3.4e+1**	**1.2e+2**	
	7.5e-1	7.0e-8	P5.5	7.0e-4	9.35	2.1e+1	

Table 4: Average Mean Square Error in bold (standard deviation in plain) for all five algorithms for regression data sets. Results are the average of ten bootstraps repetitions.

	Iris	Wine	Stegano
SVM	**95.4**	**95.8**	**89.7**
	1.9	2.9	0.95
MLP	**94.8**	**96.0**	
	3.8	2.4	
GP	**95.6**	**96.1**	
	2.3	2.1	
OP-ELM	**95.0**	**90.7**	**90.1**
	2.1	4.9	1.03
ELM	**72.2**	**81.8**	
	1.01	6.2	

	Iris	Wine	Stegano
SVM	2.3e+2	3.8e+2	8.9e+5
MLP	7.6e+2	1.2e+3	
GP	7.6e-1	1.9	
OP-ELM	7.4e-2	4.4e-1	2.6e+2
ELM	2.4e-2	2.7e-2	

Table 5: Accuracy (in percent) in boldface (standard deviation in plain) for all five algorithms for classification data sets (left) and associated computational times (in seconds, right). Results are the average of ten bootstraps repetitions.

It can be seen that although the ELM is about one order of magnitude faster than the OP-ELM, its performances can be unstable and are almost always outperformed by the OP-ELM (both in terms of standard deviation of the results and accuracy/Mean square error). Compared to the other machine learning algorithms used here, the OP-ELM is a very good compromise between speed and accuracy: it remains in the range of performance of the other state of the art methods such as GP or SVM, but is several orders of magnitude faster than them.

One potential issue — realized recently — with the use of the PRESS Leave-One-Out criterion lies in the matrix computations performed which require matrix inversions. If the LOO error is not performed properly because of numerical instabilities (some data sets have shown such instabilities, even if minor), the following pruning by the MRSR is most likely suboptimal. An improvement of the OP-ELM would for example lie in the use of another criterion than the LOO for selecting the model complexity, or to solve the potential numerical problems of the LOO.

The following section introduces a different complexity selection criterion than the Leave-One-Out, for increased speed of the model training.

5.3.3 *A possibly faster version: HQ criterion*

In the OP-ELM, the model complexity is decided by the results from the Leave-One-Out criterion: if the LOO error starts to increase while adding more neurons (in the order for which they have been ranked by the MRSR), the remaining neurons are dropped and only the ones kept until then are retained for the final model architecture.

While the LOO can be expressed in a closed form for this specific case, it remains that the formula requires matrix inversions (see publication B (section 2) for more details), which are still costly.

In B, we propose to replace the LOO criterion of the OP-ELM by an information criterion, the Hannan-Quinn (HQ) one [62].

The HQ criterion can increase the computational speed.

The choice of the Hannan-Quinn (HQ) criterion over some more classical information criteria such as the Bayesian Information Criterion (BIC) [113] or Akaike's Information Criterion (AIC) [4] was mostly done experimentally (such experiments are not present in the original publication B).

Over several data sets from the UCI Machine Learning Repository [8] it was deemed that the penalty term $2k \log\log N$ (with k the number of parameters of the model and N the number of samples) grows much more slowly, thanks to the double log, than its counterpart for the BIC and AIC, as in Equation 5.6, below.

$$\begin{aligned} HQ &= N \times \log(\varepsilon_{MSE}) + 2k \times \log\log N \\ BIC &= N \times \log(\varepsilon_{MSE}) + k \times \log N \\ AIC &= N \times \log(N \times \varepsilon_{MSE}) + 2k \end{aligned} \tag{5.6}$$

From the benchmarking experiments conducted in publication B (section 4), this choice enables to divide the computational time (compared to the LOO-based OP-ELM) by four to five folds on the tested data sets (regression

problems, from UCI also [8]). The gain in speed for another model of a similar kind, the OP-KNN [136], is even greater (up to 24 fold).

It can be noted, though, that for the OP-ELM case the number of selected neurons is globally increased, when using this information criterion instead of the LOO one. While the LOO one has potential numerical instabilities issues, it still yields a more compact model in terms of neurons. The need for speed has to be evaluated before choosing one criterion or the other, if one cares about the number of neurons selected.

5.4 CONCLUSION

The double problem of large data set in steganalysis and important training time for the model is addressed in this chapter, by the use of a *random projection* based model, the *OP-ELM*. Using a *Single Layer Feedforward Network* (SLFN) with random weights in the first layer, and linear output layer, it is possible to obtain an universal function approximator (under some assumptions). This is known as the *Extreme Learning Machine* (ELM). The OP-ELM is based on this structure and adds a neuron selection step based on the *MRSR* ranking algorithm, which is used to rank the relevance of the neurons in the SLFN, according to a *Leave-One-Out* criterion.

We proposed finally the use of a different criterion for the neurons ranking: the Hannan-Quinn criterion, with the advantage of a three to four fold speed increase over the whole OP-ELM methodology.

The next two chapters present the results obtained for two specific steganalysis problems, using the presented OP-ELM for the methodologies and experiments.

Part III

USING MACHINE LEARNING FOR STEGANALYSIS PROBLEMS

6

A PRACTICAL APPROACH TO BENCHMARKING STEGANOGRAPHIC SCHEMES

Here we propose to address the methodologies and experiments from publications C, D and E, specifically. We start by motivating the need for *feature selection*, based on some well-established facts about the *Curse of Dimensionality*, followed by a refinement of these problems in the steganalysis framework. The three different classes of features selection schemes — filtering, wrapper and embedded methods — are shortly described and the global methodology presented through publications C, D and E is described in two major steps. Please note that a majority of the results is kept in the publications, and only an introduction to these results and to some of the conclusions from the publications, are given here. Hence, the reader is refered to C, D and E for more details.

6.1 WHY IS FEATURE SELECTION SO IMPORTANT ?

We have presented in the previous chapter 4, in section 5.2 the possibility of using random projections to reduce the dimensionality of the input space. As stated, there is currently no algorithm capable of reducing the dimensionality in a very short time frame, for non-linear problems, without losing some information about the data in the process. The optimization of the projection matrix $\mathbf{P}$ regarding the validation criterion is a highly time-consuming process, when the problem has a large number of variables.

Feature selection can improve performances, reduce dimensionality and increase interpretability.

A different approach to that of the projection is the *feature selection* (which can be considered as a special case of projection). Denoting the data matrix $\mathbf{x}$ by columns instead of the usual rows notation, $\mathbf{x} = (\mathbf{x}_1^T, \dots, \mathbf{x}_M^T)$ with $\mathbf{x}_j^T \in \mathbb{R}^N$, a feature selection process finds a subset FS_{sub} of the original full set of features $FS = \{1, \dots, M\}$ such that $|FS_{sub}| < |FS|$ and that the error criterion is satisfied.

The search for a subset of features can have three different goals:

1. Improve the performances: obtain a subset FS_{sub} of features of FS such that the performances are better using FS_{sub} than FS;
2. Reduce the dimensionality while keeping similar performances: obtain a subset FS_{sub} of features of FS such that the performances using FS_{sub} are within a certain range of the ones using FS;
3. Interpretability: in the case where the features have a physical — or interpretable — meaning, find a ranking of the features which highlights their importance regarding the problem at stake. The order in which the features are ranked gives hints about the structure of the underlying problem.

The only main difference between the first two goals is about the number of features to finally select: the first approach will be more aggressive in terms of

performances at the expense of a subset possibly still of high dimensionality; the second approach is more oriented toward the dimensionality reduction while performances are only a constraint. The third one can be combined with the first two goals, and used as an additional step once the primary performance or dimensionality goal has been achieved.

Using the same notations as in chapter 4, a typical requirement on the error criterion for the second approach is that the difference between the error

$$\varepsilon_{FS_{sub}} = \mathbf{f}_{risk}\left(\mathcal{M}\left(\Theta, \mathbf{x}\left(FS_{sub}\right)\right), \mathbf{y}\right), \tag{6.1}$$

Criterion can be devised depending on the goal of feature selection.

of the model using the reduced set of features FS_{sub} and the one using the original set of features ε_{FS} (defined in the same way), is less than a certain threshold T

$$\varepsilon_{FS_{sub}} - \varepsilon_{FS} < T. \tag{6.2}$$

This criterion is merely an example and is restrictive to the case of variable subset selection, as detailed in 6.1.3.

The criterion for the second approach is more oriented toward performances and can be expressed as find a subset FS_{sub} of FS such that

$$\varepsilon_{FS_{sub}} < \varepsilon_{FS}. \tag{6.3}$$

With these notations, let us describe some of the problems that arise due to the high-dimensionality of the space (large number of features), in general first, and for the specific case of the steganalysis setup afterwards.

6.1.1 *Issues in high-dimensional spaces*

Although the set of problems caused by high-dimensional spaces is usually designated by the terminology *Curse of Dimensionality*, the original meaning intended by Bellman in [11] was about the exponential increase of the number of function evaluations with the dimensionality: optimizing a function of M binary variables with an exhaustive search over the input data requires 2^M function evaluations.

In the following, we will review three typical issues related to the high dimensionality of a problem. For a more thorough review of the Curse of dimensionality and more generally high dimensional data analysis, one can refer to [45] for example.

A "need" for samples

In high dimensional space, there are never enough samples.

The *need for samples* and the *empty space phenomenon* are two sides of the same problem, related to the lack of samples to fill sufficiently the space.

Consider a M-dimensional space (M variables) with a Cartesian grid of step ϵ defined in it. If one wants to have at least a sample in each "sub-cube" of side ϵ of the whole grid, the amount of samples required is of the order of $O\left((1/\epsilon)^M\right)$, which grows exponentially fast with M. Therefore, in a $M = 10$ dimensional space, with a step $\epsilon = 1/10$, one already needs 10^{10} samples to

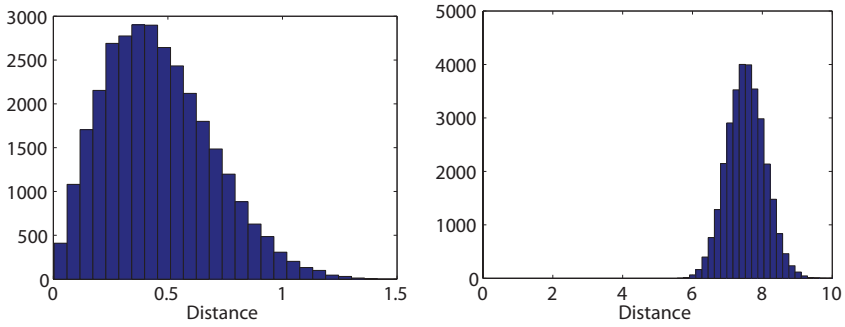

Figure 17: Illustration of the concentration of distances effect for a set of uniformly distributed samples: histogram of the pairwise distances between uniformly drawn samples, in dimension 2 (left) and 100 (right).

fill correctly the whole space, which is impossible to handle at the moment (both in terms of storage and processing).

The empty space phenomenon describes the same problem, from the opposite point of view: suppose we already have N samples in the M-dimensional space. Using the same grid to define a hypercube that spans the whole space, there will be a certain amount of samples in each sub-cube. With the increase of M (and keeping the same N), most of the sub-cubes will end up empty.

Not enough samples means a learning space not filled and thus a model which will extrapolate.

One might wonder, though, why the space need be filled with samples uniformly, as in the cases of the grid described. If a model $\mathcal{M}$ is trained on samples which do not span correctly the whole space in which it is supposed to be able to provide estimations, it might extrapolate for test points that lie outside of its "learning space". The correct behavior for a model is to interpolate, on new data, and extrapolation might be hazardous, regarding the quality of the estimation.

In practice, as shown in publications D (section 4.1) and E (section 4.2.1), the requirements on a sufficient number of samples (regarding statistically relevant results) are very much below the theoretical expectations, fortunately.

The concentration of distances

As seen in the previous part of this dissertation, some models are based on distances between samples, the k Nearest Neighbors, for example.

A problem in such spaces is that distances *concentrate*: the range of possible distances is not fully spanned anymore, and most of the samples are at large distances from each other. Figure 17 illustrates this situation for a set of points drawn from a uniform distribution, in a two-dimensional space on the left and 100-dimensional space on the right. For the two-dimensional case, distances exist in the whole possible range $\left[0, \sqrt{2}\right]$, while in dimension 100, only a very small part of the histogram is filled, mostly with large distances.

Distances tend to be all similar and large.

The bottomline of these two problems (empty space and distance concentration) is that there are never enough points in high-dimensional spaces, and they all tend to be far away from each other, rendering the relevance of distance-based models questionable [13].

Complexity of models

We have seen that models can suffer from irrelevant (or not relevant enough) variables, as in the case of the ELM, for example. Although it is often claimed that the SVM is not so sensitive to high-dimensional spaces and irrelevant variables, it is shown practically in [134] that on the contrary, SVM as most other models benefits from a most relevant set of variables.

The model complexity is always related to the dimensionality (directly or not).

Once truly irrelevant variables have been pruned out, going further into feature selection becomes a matter of performance or computational speed.

Indeed, all machine learning have a complexity either related to the dimensionality M or to the number of samples N (sometimes both), linear in the best cases, and most of the time at least quadratic with one of them. Therefore, lowering the number of variables helps in reducing the amount of samples required to fill the space and thus the computational time for the model training (and validation).

6.1.2 *More specifically: for steganalysis problems*

In the steganalysis framework, the need for samples can be adressed rather easily (compared to some specific fields where acquiring an additional sample has important costs), even though the storage of the images can become a problem for a very large base. Also, the computational and data processing time issues — since one has to extract the features from each image — are not negligible and both are in favour of dimensionality reduction.

The possibility of interpreting the variables in steganalysis motivates the feature selection.

More specific to steganalysis is the interpretability of the variables selected: if a certain variable is retained in the dimensionality reduction process, then it must relate specifically to the output; for the case of qualitative steganalysis for example, it means that this variable enables to have a more reliable detection of the stego image over the cover one. Which implies that the stego algorithm alters the cover image in the meaning related to this variable.

This leaves room for interpretation and possibly securing the stego algorithm, if it is known. If not known, it makes possible a sort of "reverse-engineering" on the stego algorithm, highlighting some of its weaknesses and therefore revealing partially its functioning.

Also, steganalysis is about performances. Whether it is in regression or classification, the goal is usually to perform as best as possible. In this sense, and even with the best possible set of features, an insufficient amount of samples might cripple the model and lead to suboptimal performances; or on the contrary, surprisingly (unreliable) good performances if the model is not validated and tested as described for example in 4.2: the variance of the results might be very large and make the results statistically insignificant. Publication E (see e.g., Figure 5) addresses this issue practically.

6.1.3 *Performing feature selection*

On the general level, it is possible to identify three main classes of feature selection: wrapper, filter and embedded methods. In the following, we give a few example of feature selection schemes belonging to each class, with an emphasis on wrapper methods, since it is the class used in the publications

included in this dissertation. A more detailed review of feature selection techniques for machine learning can be found for example in [59].

Filtering

Filtering methods can be seen as a pre-processing step on the data $\mathbf{x}$. In this class, no model is considered, but a scoring function $SF\left(\mathbf{x}_j^T, \mathbf{y}\right)$ is typically used: the score computed between the variable j of the data set $\mathbf{x}$ and the output $\mathbf{y}$ permits to decide whether this variable j should be retained. Thus, variables are considered separately and one at a time, in this method. One of the main advantages over wrapper and embedded schemes is the low computational time it requires (for M variables, only M evaluations of the scoring function SF have to be computed). Examples of scoring functions include Pearson's correlation coefficient (used for example in [102]) and mutual information, among others [59]. While Pearson's coefficient is easy to compute, it only reveals linear dependencies between the considered variable and the output. Mutual information suffers from the difficulty we have to estimate it practically.

Filtering methods use each variable separately in relation to the output.

The following wrapper and embedded schemes do not consider each variable separately regarding the output, but by subsets or sequential adding/removing.

Wrapper and embedded methods

These classes of feature selection techniques require the use of a model. Instead of the previous scoring function, the model is here used to verify the relevance of the selected subset of variables. One advantage of these methods lies in the fact that various combinations of variables can be considered, regarding the output, and not only one, as in the filtering approach.

Embedded methods are a bit specific: selection is inside the model.

The embedded methods are using a process of feature selection *inside* the machine learning model itself. Decision trees are an instance of this type of methods [20].

The wrapper methods are popular since they allow the use of any model as the decision criterion on whether a variable should be kept or not: in this scheme, the machine learning model acts as a black box. Let us cite two *greedy* methods as wrappers: the *forward selection* and the *backward elimination*. The adjective greedy comes from the fact that once a variable as been added/removed, it is never considered again. The forward and backward methods are actually variable ranking methods: the selection of the subset remains a decision of the user in the end.

Wrapper methods such as the forward are often used for they are easy to implement and can use any model.

The forward selection starts with an empty set of selected variables S and a full set of remaining variables R. Each of the variables in R is added in turn to S, and performance is evaluated for each set S containing *one* variable. The variable deeming the best results is put in S definitely and removed from R. The algorithm iterates $M(M-1)/2$ times (with M the number of variables), considering sets of the first selected variable and one of the remaining in R, and finishes once R is empty. The algorithm is put in a more formal algorithmic way in publication E (section 3.3.1), for example.

The backward elimination works in the same way, except that the initial set of selected variables is full and the remaining one is empty. The algorithm

processes this time by eliminating the variable contributing the less to the performance, until none is left.

At the end, one obtains a *ranking* of the variables, which enables to select a sufficient amount of variables regarding the performances of the model.

The forward selection is used in the following description of a practical benchmarking for stego algorithms, to select the steganalysis features that are most relevant.

6.2 PRACTICAL BENCHMARKING OF STEGO ALGORITHMS

Chronologically, publication C is the first one to propose the use of feature selection for steganalysis problems. The set of features used is the original 23 DCT features set of Fridrich [46], and the paper shows through a methodology using the forward selection scheme, that it is possible for the stego algorithms Outguess, F5 and Steghide, to reduce the set of 23 features to a subset of 13 (different for each stego algorithm) while retaining similar performances. This early work gave the basis for the methodology that is to be presented quickly in the following, and published in E.

A possible practical benchmark for steganography: find how many images should be used and perform feature selection.

One problem which arose while experimenting was that of the relation between the number of samples (images here) and the reliability of the results. We realized when using a larger feature set, that results were no longer statistically stable, with large deviations on different permutations of the data set.

Publication D deals mainly with this problem, when using the extended version of the DCT feature set, containing 193 features. Using a similar methodology to that of C, the variance of the results is estimated when the number of samples varies.

Finally, a global critique on the methodology used until then was on the use of a different machine learning model for the variable selection (a k-NN) and for the final evaluation of the performances (a SVM). It was argued that variables selected using one machine learning method might not be the most appropriate for another model, and hence, that the selection of variables was possibly sub-optimal for a SVM.

It should be noted that in all three publications, the terminology "confidence interval" is used to designate the interval based on the standard deviation of the results around the computed mean value.

Even though the forward selection approach has a rather limited number of iterations (compared to an exhaustive search, for example) it remains computationally intractable to perform it on a large data set using the SVM. In this regard, the OP-ELM was proposed and used, in E, to obtain a global practical benchmark for stego algorithms. The following presents shortly the two main parts of this benchmark approach, and the reader is refered to the original publications C, D and E for the full set of results and a wider analysis.

6.2.1 *Determining a sufficient number of points*

As described at more length in publication E, the determination of a sufficient number of points is using a bootstrap approach [39] to obtain statistically

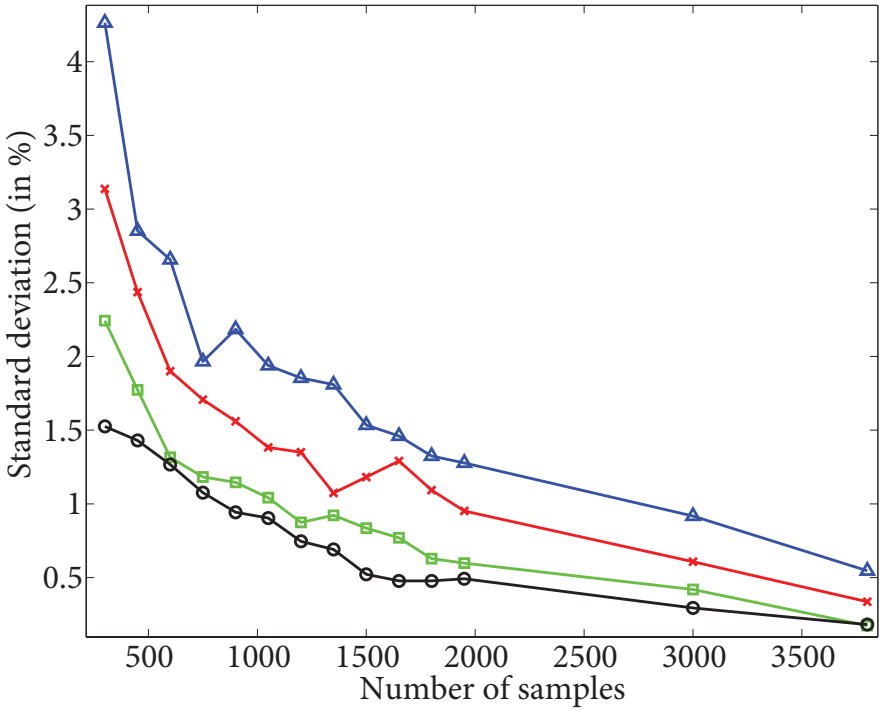

Figure 18: Standard deviation in percentage of the average classification result (relative variation) versus the number of images used, for the four embedding rates for the Outguess algorithm: black circles (◯) for 20%, green squares (□) for 15%, red crosses (×) for 10%, and blue triangles (△) for 5%. Please that the embedding rates are measured here using the $R^{(2)}$ definition from section 2.3.2.

relevant results for varying sizes of the subset drawn randomly from the original data set. The 100 runs of the bootstrap over large data sets are feasible, computationally speaking, with the OP-ELM as the machine learning model. Again, the reader is refered to publication E for more details on the methodology.

First find how many images are required for the task...

As a mere illustration of the effect of the size of the data set over the relevance of the results, let us take the example of the Outguess stego algorithm, for the following. Publication E gives results for five other stego algorithms: F5, MM3, JPHS, MBSteg and Steghide.

Figure 18 plots the standard deviation of the results (in percentage of the average accuracy) versus the number of samples used (images), for the OP-ELM model on a boostrap with 100 repetitions. Note that the embedding rates are computed as in the $R^{(2)}$ definition from section 2.3.2. Two main points can be made:

1. the larger is the embedding rate, the lower is the standard deviation of the results: a large embedding rate yields more statistically stable results;

2. the more images are used to train the model and perform the steganalysis, the more relevant are the results (statistically speaking).

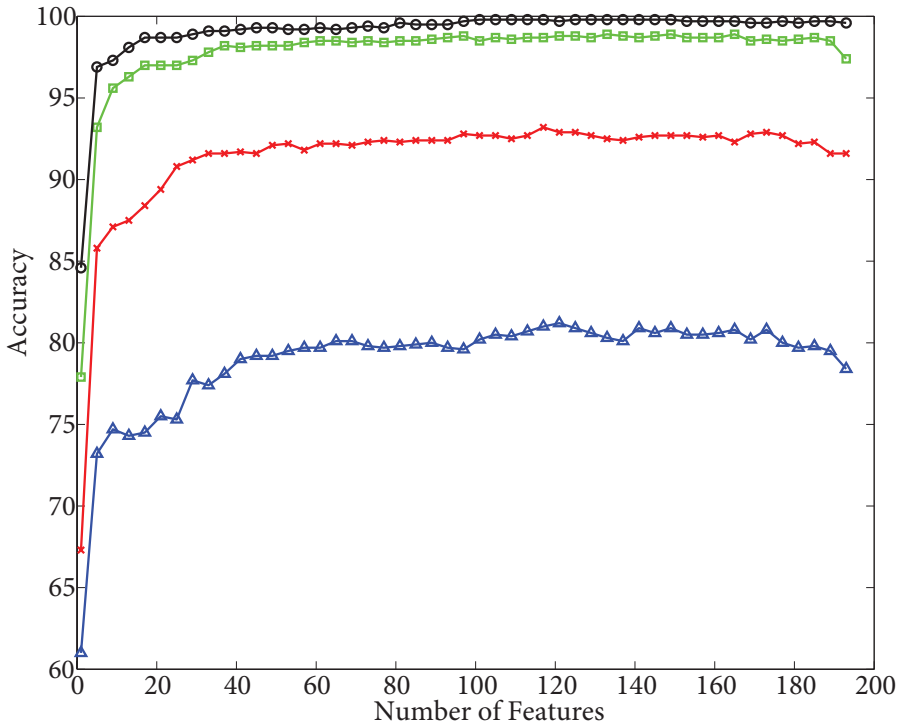

Figure 19: Accuracy versus number of features used, for the four embedding rates for the Outguess algorithms: black circles (◯) for 20%, green squares (□) for 15%, red crosses (×) for 10%, and blue triangles (△) for 5%. Not all points are plotted for clarity.

The plots for all six stego algorithms behave in a similar way to that for the Outguess one.

In publication E, a sufficient number of images is chosen based on a threshold arbitrarily set on the relative standard deviation of the results. The 1% figure was deemed sufficient for the rest of the methodology, but could be taken to be any other non-zero value, if the steganalysis system is reliable. This is basically a user-defined value, depending on the desired maximum variance of the results.

The outcome of this step is a number of images sufficient to obtain statistically relevant results, number which is used in the rest of the experiments regarding the feature selection.

6.2.2 *Determining a sufficient number of features*

And then select the features and interpret the selection.

The second step of the methodology is to determine a sufficient number of features for each stego algorithm. Again, let us overview the results for the Outguess algorithm.

In Figure 19, it can be seen that for this specific stego algorithm, only a fraction (less than 50 seems enough) of the full 193 DCT feature set is sufficient to obtain a good accuracy, similar to the best possible result, within the noise of the results. This means that the dimensionality of the data set can be reduced by at least a factor four, for the Outguess.

The decision over the number of features to keep is again a user-defined tradeoff between accuracy and desired number of features. To be precise, it could be defined by taking the numerical derivative of the accuracy plot (versus number of features), and a threshold (user-defined) on this derivative would give the number of features beyond which the accuracy increase is not important enough anymore.

This conclusion for the Outguess cannot be generalized to the other stego algorithms, as noted in publication E: even though the shape of the plot is similar, the size of the feature set required can be very different (and vary largely between the embedding rates, interestingly).

Also, the selected features are different, for each stego algorithm, which enables the previously mentioned analysis on the possible weaknesses of the algorithm.

We refer the reader to the publication for more details on this study, along with a list of the selected variables and their importance.

6.3 CONCLUSION

Feature selection (or *variable selection*) is an important concept in machine learning. Even in the hypothetical case where infinite amount of samples, infinite computational power and perfect models would exist, feature selection would not be useless. Not only does it provide lower dimensional spaces making the task easier in terms of model and amount of samples required, but it also gives interpretability to the selected features. This reveals additional information about the problem and the phenomenon underlying in the data. In the specific case of steganalysis, and given that the features used can be interpreted in a practical sense, the selected features give practical insight over the stego algorithm used and some of its weaknesses (preservation of the AC histograms, for example).

Throughout publications C, D and E, we have developed a methodology to address the problems of high-dimensional spaces in the steganalysis framework: first, making sure that the amount of samples to perform any experiment is sufficient regarding the number of features used (dimensionality of the space); second, determining a sufficient number of features for a particular problem, enabling a deeper analysis of the selected features for a possible reverse-engineering of the stego scheme used.

The problem of the sufficient number of images regarding the accuracy of the results is especially important for low embedding rates and for some algorithms; for example MBSteg with a 5% embedding rate yields up to 7% of relative standard deviation of the average result. Too frequently, the amount of samples sufficient for a proper model training is overlooked and the results presented tend to have high variance, and are hence unreliable.

Again, the presentation of these steps is described shortly here to introduce the publications and the results. More details are available from the publications in this dissertation (C, D and E for this specific chapter).

While this chapter is mostly focused on classification problems (qualitative steganalysis), the following proposes to address the quantitative steganalysis case, with a novel practical approach on the matter.

7

A NOVEL APPROACH TO QUANTITATIVE STEGANALYSIS AND IMAGE RELIABILITY ESTIMATION

This chapter aims at presenting the recent novel results from publication F about the quantitative steganalysis problem. The methodology and approach presented in publication F are the first steps in this direction for quantitative steganalysis and image reliability estimation, presented for one stego algorithm (nsF5). As for the previous chapter, the concepts from publication F are described here, along with some results and possible clarifications. The reader is invited to refer to publication F for the full details and experiments on this new approach to quantitative steganalysis.

7.1 RE-EMBEDDING CONCEPT FOR QUANTITATIVE STEGANALYSIS

In the following — and in publication F — we place ourselves in a specific case of quantitative steganalysis, with certain assumptions:

Assumptions for the re-embedding concept.

- we know the stego algorithm used by the sender (this can be identified by blind steganalysis, for example);
- we have a model $\mathcal{M}$ that can predict accurately the embedding rate (we use the $R^{(1)}$ definition,here; see section 2.3.2);
- the message embedded by the sender does not span the whole capacity of the image. This is a reasonable assumption if the sender does not want to be too easily caught.

In this setup of quantitative steganalysis, we want to know how large is the message (quantitative steganalysis setup) that has been embedded by the sender in a suspicious image i. The direct use of a feature set — for example the full calibrated DCT one, see section 3.3.4 — on the image i with the model $\mathcal{M}$ should give proper results in quantitative steganalysis, as in [103] with an OLS for the model. We try to improve the quality of the results, and also find additional information — the number of embedding changes and the number of original non-zero AC coefficients — by interpolation using the re-embedding concept.

Re-embedding adds an additional message in a suspicious image.

In order to both improve the estimation of the original embedding rate and give a confidence interval of this estimation, we use a re-embedding technique. The idea is here depicted for one single image i (experiments in publication F are performed on 700 images from the BOWS2 database [10]), for simplicity of notations. The 193 calibrated DCT features from [99] are used.

Suppose we have intercepted the image i_o coming from a suspicious sender and we want to estimate the number of embedding changes E_o made to that image (in relation with the definition of the embedding rate $R^{(1)}$). For this, we make in a copy i_j of image i_o, and perform E_j embedding changes on it (E_j being uniformly drawn in a certain range).

This process of re-embedding is repeated N times, giving a set of images $\{i_j, 1 \leqslant j \leqslant N\}$, containing the original embedding changes and the additional ones.

We approximate the final embedding rate R_j as in Eq. 7.1.

We propose to approximate the embedding rate R_j for the re-embedded images i_j (in the same sense as for the definition of $R^{(1)}$) as

$$R_j = \frac{E_o + E_j}{A_o} = R_o + \frac{1}{A_o} E_j, \tag{7.1}$$

where A_o is the number of non-zero AC coefficients in the original image (before a message was embedded and it became i_o). This approximation is shown to be sensible in publication F, under the assumptions mentioned previously — careful steganographer, i.e. embedding rate rather small. The idea is then to have many E_j to be able to estimate the constant term R_o and the first order coefficient $\frac{1}{A_o}$ from a set of equations as Equation 7.1.

Indeed, using the model $\mathcal{M}$ which is supposed to be able to estimate embedding rates of the $R^{(1)}$ form, we can estimate the R_j from the i_j and obtain the $\hat{R}_j$ such that

$$\hat{R}_j = R_o + \frac{1}{A_o} E_j + \varepsilon_j, \tag{7.2}$$

with ε_j the error made by $\mathcal{M}$ for image i_j. The approximation made in Equation 7.1 regarding the total embedding rate is investigated in publication F (see e.g. Figure 3) and proved to be reasonable for low $E_o + E_j$.

The approximation is reasonable for low $E_o + E_j$, see publication F.

By using this approach, we aim at obtaining a better estimation of the original embedding rate, as well as a confidence interval on that value indicating how reliable is the estimation.

7.2 EMBEDDING RATE AND CONFIDENCE INTERVAL ESTIMATION

Using a set of N equations (each equation coming from one re-embedding), the linear system

$$R_o + \frac{1}{A_o} \mathbf{E} = \hat{\mathbf{R}} \tag{7.3}$$

is obtained, in which $\mathbf{E}$ is the vector of all the E_j and $\hat{\mathbf{R}}$ is the vector of all the R_j. As described at more length in publication F, the constant coefficient found from the solution of the system (solved in a Least-Squares sense) is the estimation of the original embedding rate used by the sender and the first order coefficient is the estimation of the inverse of the number of non-zero AC coefficients. From these two, it is possible to obtain the estimation of the number of original embedding changes E_o.

Multiple repetitions of the re-embedding allow for the embedding rate estimation by linear regression.

In addition, it is possible to compute a confidence interval on the estimated values for R_o and $\frac{1}{A_o}$ (the value A_o permits the calculation of the original number of embedding changes, from R_o). In publication F, it is done using the Matlab® function `regress`, which uses a Student's t score, as described in [36].

Figure 20 illustrates the idea (figure from publication F) for the image set used in the publication. The value at the ordinate for abscissa zero gives

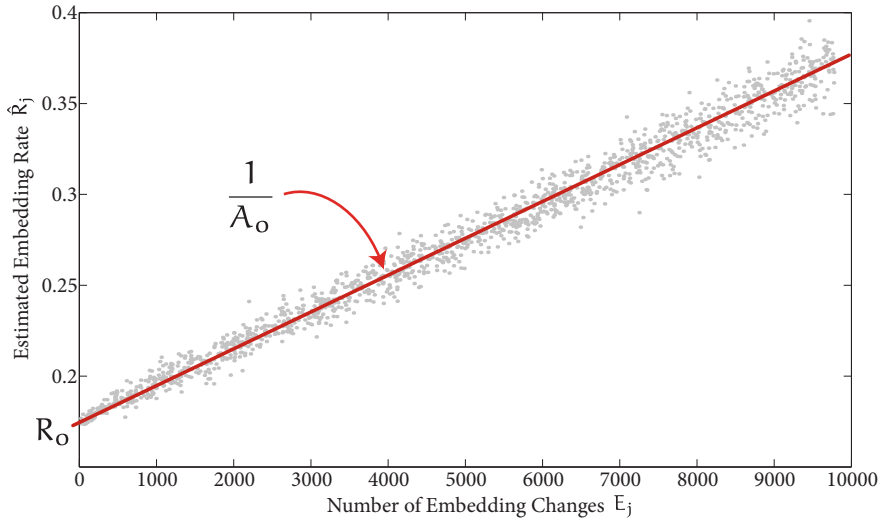

Figure 20: Estimated embedding rate $\hat{R}_j$ versus number of re-embedding changes E_j. The slope of the linear regression gives the first order term $\frac{1}{A_o}$ while the ordinate when $E_j \longrightarrow 0$ gives the constant term R_o.

the estimate of the original embedding rate R_o, while the slope of the linear regression gives $\frac{1}{A_o}$. The confidence interval on both values is also computed for that specific image i_o.

The width of this confidence interval is then used to obtain an estimation of the image difficulty and thus of the reliability of the estimate.

7.3 INNER IMAGE DIFFICULTY/ RELIABILITY ESTIMATION

An original work of Böhme and Ker on quantitative steganalysis [16] analyzed a quantity similar to that of the inner image difficulty discussed here and in publication F, the *within-image* error. By defining a two error model of the estimation of the payload size, the authors analyze the within-image error which takes into account the errors caused by the possible dependencies between the cover image and the message embedded in it. We propose a similar idea to measure the inner difficulty of the image and show that it can be estimated for a stego image thanks to the re-embedding approach.

7.3.1 *A possible measure of the difficulty*

Defining the inner difficulty of an image is problematic: the inner difficulty should be an universal value in the sense that it should not depend on the model used or the set of features. In the best scenario, one wants a number explaining the inner difficulty for any steganalysis setup. This value should summarize how often the image considered is misclassified for the qualitative steganalysis problem for example, or be related to the error (Mean Square Error, for example) that steganalysis systems make in the framework of quantitative steganalysis. Hence, this number (or possibly a set of numbers, for both quantitative and qualitative steganalysis) reflects

how "dangerous" an image potentially is, for the false positive or false negative rates (respectively the mean square error on the embedding rate) tend to deviate from the average behavior of the steganalysis system. In this prospect, it is sensible to use a measure based on the standard deviation of the error performed for that image, compared to the average situation for other images.

We define a measure of the image difficulty, given the genuine image.

In this analysis, we use the 193 DCT calibrated feature set and propose to estimate this inner difficulty D by the variation of the predictions for a given original embedding rate E_o when the embedding stego-key, or the embedded message varies.

To measure it, we propose to use the real original image (before a message was embedded in it) i, which is obviously not available in a real case. This "theoretical" analysis is merely to demonstrate that the estimated confidence interval on R_o can be used as a measure of the inner difficulty.

We propose to make L embeddings to L copies of i, with varying number of embedding changes E_j^O, leading to an embedding rate R_j^O, $1 \leqslant j \leqslant L$ (in publication F, $L = 100$). The model $\mathcal{M}$ is used to obtain an estimate $\hat{R}_j^O$ of the embedding rate R_j^O.

The measure D uses the steganalysis error for first embeddings with different stego-keys/ payloads.

The inner difficulty D for image i can then be estimated over the set of L realizations by

$$D(i) = \mathrm{std}\left(\hat{R}_j^O - R_j^O\right). \tag{7.4}$$

This measure is used in the following to illustrate the relevance of the width of the confidence interval on the estimated value of R_o as an estimator of the inner image difficulty. We do not claim that this measure is absolute and universal for the inner difficulty of an image, but only that the value $D(i)$ should react positively when the image is difficult for the stego algorithm and feature set used.

Later work focused on the estimation of this inner image difficulty should compare the behavior of this proposed measure when the feature set, the stego algorithm and the embedding rates range, vary. Most likely, some more elaborate measure could generalize this early concept to a wider range of stego algorithms and steganalysis systems.

Before we present some of the results from publication F, let us first review a specific test which could be called "conality test".

7.3.2 *A "conality" test*

The results concerning the estimation of the inner difficulty D of an image revealed a "cone-shaped" distribution (see results section 3 of publication F), with a non-uniform repartition of the samples.

This test permits to check that the data follows a "cone" distribution.

We want to quantify how this distribution of points grows on average, and also how the variance of the distribution grows: for a data set which lies inside a cone, as on Figure 21, the ordinate y of the points grows on average with the abscissa $\mathbf{x}$, and so does the variance. This toy example is to illustrate how this test is based on locality functions.

In order to measure and plot the evolution of this variance and mean of the distribution, we use a k-NN approach:

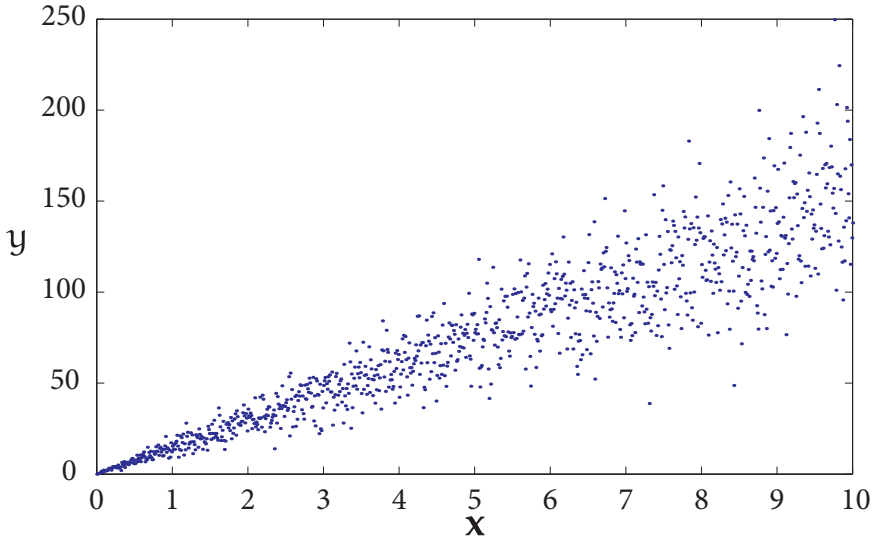

Figure 21: Plot of the original toy data with a distribution shaped as a cone.

1. For each point in the distribution, find the k nearest neighbors (pairwise distance is measured only using the abscissa **x** of the points);
2. Compute for each point the mean and variance over the k nearest neighbors (mean and variance computed using the ordinate y).

The result for the toy data of Figure 21 is depicted on Figure 22. The red central points are the means over the 30 nearest neighbors for each point, and the black surrounding points are the means plus/ minus twice the standard deviation (again, computed over the 30 nearest neighbors).

The distribution of points grows in mean and variance, as for a "flat cone".

It can be seen that the mean is constantly growing with the abscissa, and that the width of the cone (the "variance for each point") is also increasing. This characterizes the "conality" of the data: the ordinate value y increases on average and the spreading of the points grows with it.

7.3.3 *Inner image difficulty estimation*

Using the width of the confidence interval obtained on $\hat{R}_o$, we first check against the difficulty D value. Figure 23 (from publication F) represents all the 700 images used, each with $N = 1500$ repetitions.

We estimate the inner image difficulty D by the width of the confidence interval.

Although the distribution of the points in the plot is less obvious than for the toy example previously mentioned, the "cone" shape is still present. We propose to highlight this shape by using the conality test presented, and find that the "mean for each point" — for its 30 nearest neighbors — grows in a linear fashion (see F for the plots). The variance behaves similarly.

Overall this correlation between the width of the confidence interval for the estimated $\hat{R}_o$ and the proposed inner image difficulty measure D proves that the re-embedding approach for this estimation is justified.

As stated in F, it is noteworthy that the variance also increases with respect to the width of the confidence interval. This basically means that the larger

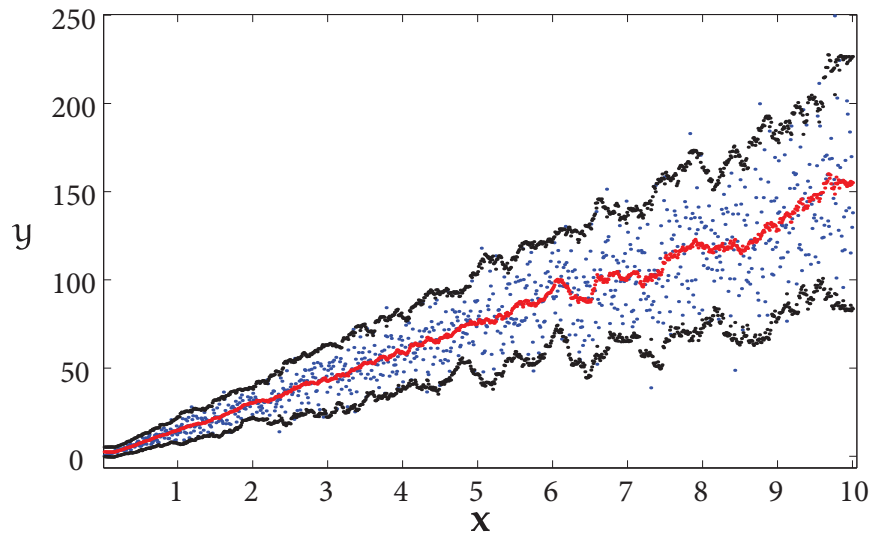

Figure 22: Results of the "conality" test: central red dots depict the growing mean over the 30 nearest neighbors and surrounding black dots the mean ± 2 times the standard deviation (also over the 30 nearest neighbors).

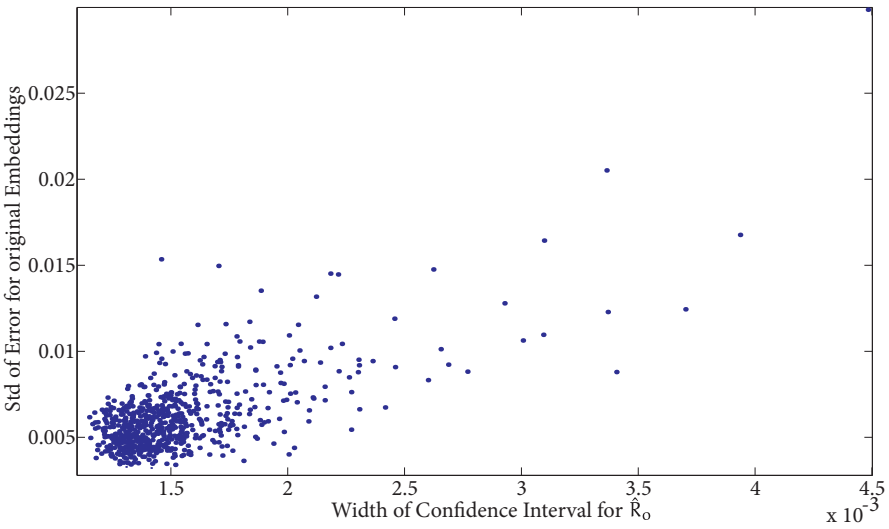

Figure 23: D (measure of inner image difficulty) versus width of confidence interval for $\hat{R}_o$. The distribution of points is not uniform and shaped like a cone.

is the obtained confidence interval, the harder it is to obtain an accurate estimation of the difficulty.

7.4 CONCLUSION

We have reviewed a methodology using the concept of *re-embedding* for quantitative steganalysis. Using this concept, we derive a simple estimation of the original embedding rate used by the sender, R_o along with a confidence interval on it. We further verify that the width of this confidence interval can be used to measure the inner image difficulty, by comparing to quantity measured with the use of the original genuine image.

While the results from publication F are rather new and novel, they possibly need deeper research and tests on a larger range of images and stego algorithms, since the concept is only tested on one stego method here, nsF5. We would like to point out on a problem which is eluded in publication F — since it does not belong to the presented concept — the amount of data to process. Originally, the experiments were carried out on about 2000 images from the BOWS2 set [10], and not 700. Handling of the 1500 repetitions over 2000 images became a problem, in terms of memory and calculations, and we reverted to 700 images for the moment. Fortunately, the use of the OP-ELM made the computations tractable even on such large data set. In the future, we would like to validate these results on a larger set of images — for example to obtain a more uniform distribution of points on Figure 23 — and test the methodology for other stego algorithms to verify if the behavior is similar or if the difficulty of the steganalysis task (using this same set of features) influences the results on the confidence interval and inner image difficulty estimation.

8 SUMMARY AND CONCLUSIONS

In this dissertation, we have addressed two different sides of the global steganalysis problem: the classical steganalysis one (using classification) and the quantitative steganalysis one (regression).

For the classical "qualitative" steganalysis, we have proposed a two-step methodology which originates from publications C and D. The first step of this methodology attempts at estimating the relevance of the results — in a statistical significance way — and helps devising a sufficient number of samples required for the experiments. By measuring the variance of the results over multiple bootstrap iterations, for different amounts of samples, it is possible to find a threshold above which the results of a specific model become statistically significant (i.e. the variance of the results is small enough).

The second step aims at performing feature selection on the set of features used for the steganalysis task. Since these sets tend to be large (in the magnitude of hundreds of features), the associated data is becoming more challenging to process and interpret. By reducing the dimensionality using Forward feature selection, the computational time required for the model to be trained is notably decreased (several orders of magnitude): for some of the tested stego algorithms, a reduction of the feature set by a factor of approximately 10 yields similar results for all the tested embedding rates (in the example of Outguess). In the last part of this step, the selection of features allows for interpretation and the analysis of the ranking of features by the Forward gives information on what reacts the most vividly to the embedding of a message. This can give precious information on the potential weaknesses of the stego algorithm studied, and eventually reveal parts of its functioning.

In order to conduct this methodology on a larger scale than in publications C and D, a fast and efficient machine learning model, which could be kept throughout the whole methodology, was needed. Indeed, in order to reduce the variations in the methodology, it is better to use the same model for determining the sufficient number of samples and then select the features. The OP-ELM (publication A) is proposed in this spirit and uses the original ELM [69] to which is added a neuron pruning strategy. A Single Layer Feed forward Neural Network is built using random projections in the first layer (weights randomly initialized), following the ELM original structure. A large number of neurons is used in the first place and the irrelevant ones are finally pruned using a neuron-ranking algorithm, the MRSR [117], with a Leave-One-Out decision criterion.

This model achieves state of the art performances, while having computational times reduced by orders of magnitude (compared, for example, with a SVM). We claim that it is among the best performance/ speed ratio.

In publication B, the Leave-One-Out decision criterion of the OP-ELM is proposed to be replaced by an information criterion, the Hannan-Quinn one. This has the effect of reducing by three to four folds (for the datasets tested) the computational time of the OP-ELM, while retaining similar performances.

The OP-ELM is used in the methodology presented in publication E, which enables to perform the estimation of a sufficient number of samples and of a reduced feature set for a large database of images (the BOWS2 challenge base [10]) and for six popular stego algorithms.

Quantitative steganalysis is then addressed, with the aim of estimating a confidence interval for the estimation of the original embedding rate (related to the message size embedded by the sender). A novel approach is used in this sense: Re-embedding. The idea is to embed a new message of known size in the suspicious image (which might already contain one). By performing this operation many times (on different copies of the suspicious image each time) for varying sizes of the newly embedded message, we propose to estimate the original embedding rate used by the sender. This is done by a simple linear regression. A confidence interval is also devised by this method, for the estimated original embedding rate.

This approach gives better results (for the one stego algorithm tested) than a "standard" quantitative steganalysis using directly the feature set on the suspicious image.

In addition, we propose to estimate the inner image difficulty (in the sense that it is behaving in an "unusual way" for steganalysis tasks), with the width of the estimated confidence interval. By measuring the inner difficulty of the image using the genuine version of it, we show that the width of the confidence interval is correlated with it.

In conclusion, the machine learning setup in the steganalysis framework is not a usual one for two main reasons. First, new data to train and test the model used is easy and rather costless to acquire: taking new pictures of outdoor or family scenes and using them with stego algorithms does not yield heavy processing costs as for other domains where acquiring a new sample uses complicated equipment and costs large amounts of money. Second, a goal in steganalysis (and steganography) is to have as good as possible a model of the image considered; we have seen that this can be approximated by the use of features, which are nowadays numerous, to always model better the image and its characteristics. The machine learning problem becomes high-dimensional, and the data grows exponentially larger with the new features devised, leading to a problems harder to solve, in terms of machine learning models. There is hence a need in steganalysis, for fast and efficient machine learning models, and as importantly methodologies to obtain reliable and relevant results.

If the work achieved was to be pursued, there are several directions to explore, to possibly improve the results and performances but also to widen the view and approach.

First of, the work on the quantitative steganalysis and inner image difficulty estimation is recent and it could be extended in the future to more stego algorithms and a larger database of images, to proof this approach on other similar problems.

Second, the OP-ELM might be improved, in terms of "stability" — since the OP-ELM uses random projections and is not deterministic. Indeed, the only hyper-parameter of the OP-ELM, the number of neurons, can be crucial for some problems, and its determination is mostly heuristic. Hence, an inappropriate choice made by the user on this hyper-parameter can lead to unstable results — which are spotted quickly in validation. The idea of

the OP-ELM (pruning irrelevant neurons) and of the EM-ELM (adding new neurons to an existing structure) could be combined, in a possibly slightly slower method, but providing more stability in terms of the results obtained.

Third, the mentioned reverse-engineering infered from the selected features in the classical steganalysis methodology should be put to use. By identifying the most sensitive features for a specific scheme, it must be possible to improve its original scheme and perform a better steganography. Although we mention which features are selected for which stego algorithm, this study has never gone any further. The author believes that even if such improvements will not render a very insecure stego system suddenly secure, it should improve it nonetheless; which might in turn raise a new interest for it in the steganalysis community.

Finally, as Andrew Ker stated it in a talk given at Telecom ParisTech school, the field of steganography has still some important, although barely explored, areas such as the Batch Steganography and Pooled Steganalysis problem [76]. The matter of hiding a message in *multiple* images is very different from hiding into just one — the approach is probably more statistical and less "hands-on" than with typical JPEG stego algorithms — and has not yet received an important attention from the researchers of the community. This is rather surprising though, since the most realistic case of a "smart and evil" steganographer would very likely involve the use of multiple images...

BIBLIOGRAPHY

[1] *The RigVeda*. est. 1700-1100 BC. (Cited on page v.)

[2] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003. ISSN 0022-0000. doi: 10.1016/S0022-0000(03)00025-4. (Cited on page 64.)

[3] Sos S. Agaian and Hong Cai. Color wavelet based universal blind steganalysis. In *IEEE International Workshop on Spectral Methods and Multirate Signal Processing*, volume 7, pages 3710–3713, Vienna, Austria, 2004. (Cited on page 54.)

[4] Hirotsugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974. (Cited on page 71.)

[5] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001. ISSN 1532-4435. doi: 10.1162/15324430152733133. (Cited on page 47.)

[6] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2004. (Cited on pages 43 and 47.)

[7] Ross J. Anderson. *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996. (Cited on page 10.)

[8] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007. URL http://www.ics.uci.edu/mlearn/. (Cited on pages 64, 67, 71, and 72.)

[9] Ismail Avcibas, Nasir Memon, and Bulent Sankur. Steganalysis using image quality metrics. *Image Processing, IEEE Transactions on*, 12(2):221–229, feb 2003. ISSN 1057-7149. doi: 10.1109/TIP.2002.807363. (Cited on page 28.)

[10] Patrick Bas and Teddy Furon. BOWS2 Challenge: Break Our Watermarking Scheme: http://bows2.gipsa-lab.inpg.fr/, ECRYPT European Network of Excellence. (Cited on pages 69, 85, 91, and 94.)

[11] Richard E. Bellman. *Adaptive Control Processes: a Guided Tour*. Princeton University Press, Princeton, NJ, 1961. (Cited on page 76.)

[12] Nabil Benoudjit, Cédric Archambeau, Amaury Lendasse, John A. Lee, and Michel Verleysen. Width optimization of the gaussian kernels in radial basis function networks. In M. Verleysen, editor, *ESANN 2002, European Symposium on Artificial Neural Networks, Bruges (Belgium)*, pages 425–432. d-side publ. (Evere, Belgium), April 2002. (Cited on page 58.)

[13] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Database Theory — ICDT'99*, volume 1540/1999 of *Lecture Notes in Computer Science*, pages 217–235. Springer-Verlag, 1999. (Cited on pages 58 and 77.)

[14] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1996. (Cited on page 57.)

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007. (Cited on page 54.)

[16] Rainer Böhme and Andrew D. Ker. A two-factor error model for quantitative steganalysis. In Edward J. Delp III and Ping Wah Wong, editors, *Proceedings of SPIE*, volume 6072, page 607206. SPIE, 2006. doi: 10.1117/12.643701. (Cited on page 87.)

[17] Thorsten Bojer, Barbara Hammer, Daniel Schunk, and Katharina Tluk von Toschanowitz. Relevance determination in learning vector quantization. In Michel Verleysen, editor, *Proceedings of European Symposium on Artificial Neural Networks (ESANN'01)*, pages 271–276, Brussels, Belgium, 2001. D-facto publications. (Cited on page 58.)

[18] Gianluca Bontempi, Mauro Birattari, and Hugues Bersini. Recursive lazy learning for modeling and control. In *European Conference on Machine Learning*, pages 292–303, 1998. (Cited on page 69.)

[19] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi: 10.1145/130385.130401. (Cited on page 55.)

[20] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. (Cited on page 79.)

[21] Michael Buehner and Peter Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820–824, 2006. (Cited on page 65.)

[22] Chris J.C. Burges and Bernhard Schölkopf. Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems 9*, pages 375–381. MIT Press, 1997. (Cited on page 56.)

[23] Christian Cachin. An information-theoretic model for steganography. In *Information Hiding*, volume 1525/1998 of *Lecture Notes in Computer Science*, pages 306–318. Springer Berlin / Heidelberg, January 1998. (Cited on pages 23 and 24.)

[24] Rajarathnam Chandramouli and Nasir D. Memon. A distribution detection framework for watermark analysis. In *Proc. of the ACM Multimedia Workshop on Multimedia and Security*, pages 123–126, New York, NY, USA, 2000. (Cited on page 28.)

[25] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. (Cited on page 63.)

[26] Brian Chen and Gregory W. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Trans. on Information Theory*, 47:1423–1443, 1999. (Cited on page 22.)

[27] Pedro Comesaña and Fernando Pérez-González. On the capacity of stegosystems. In *MM&Sec '07: Proceedings of the 9th workshop on Multimedia & security*, pages 15–24, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-857-2. doi: 10.1145/1288869.1288873. (Cited on page 12.)

[28] JPEG Comittee. URL http://www.jpeg.com. (Cited on page 16.)

[29] Pierre Comon. Independent component analysis - a new concept? *Signal Processing*, 36:287–314, 1994. (Cited on page 45.)

[30] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalke. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2008. (Cited on pages xxi, 10, 28, 31, and 32.)

[31] Ron Crandall. Some notes on steganography. Posted on Steganography Mailing List, 1998. URL http://os.inf.tu-dresden.de/~westfeld/crandall.pdf. (Cited on pages 14 and 20.)

[32] Nedeljko Cvejic, Anja Keskinarkaus, and Tapio Seppänen. Audio watermarking using m-sequences and temporal masking. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 227–230, New Paltz, New York, October 21-24 2001. (Cited on page 9.)

[33] Luc P. Devroye and Terry J. Wagner. Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions on Information Theory*, 25(2):202–207, 1979. (Cited on page 51.)

[34] Luc P. Devroye and Terry J. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Trans. on Information Theory*, 25(5):601–604, 1979. (Cited on page 51.)

[35] Dariush Divsalar, Hui Jin, and Robert J. McEliece. Coding theorems for "turbo-like" codes. In *36th Allerton Conference on Communications, Control and Computing*, pages 201–210, 1998. (Cited on page 22.)

[36] Norman R. Draper and Harry Smith. *Applied Regression Analysis, 3rd edition*. Wiley-Interscience, 1998. (Cited on page 86.)

[37] Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alexander Smola, and Vladimir N. Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems 9, NIPS*, pages 155–161. The MIT Press, 1996. (Cited on page 55.)

[38] Sorina Dumitrescu and Xiaolin Wu. A new framework of lsb steganalysis of digital media. *IEEE Transactions on Signal Processing*, 53: 3936–3947, October 2005. doi: 10.1109/TSP.2005.855078. (Cited on page 33.)

[39] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Boostrap*. Chapman and Hall/CRC, 1994. (Cited on pages 51 and 80.)

[40] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. In *Annals of Statistics*, volume 32, pages 407–499. 2004. (Cited on page 69.)

[41] Guorui Feng, Guang-Bin Huang, Qingping Lin, and Robert Gay. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*, 20 (8):1352–1357, 2009. ISSN 1045-9227. doi: 10.1109/TNN.2009.2024147. (Cited on page 67.)

[42] Tomas Filler, Andrew D. Ker, and Jessica Fridrich. The square root law of steganographic capacity for markov covers. In Edward J. Delp III, Jana Dittmann, Nasir D. Memon, and Ping Wah Wong, editors, *Media Forensics and Security*, volume 7254, pages 801–811. SPIE, 2009. doi: 10.1117/12.805911. (Cited on page 12.)

[43] Gary William Flake and Steve Lawrence. Efficient svm regression training with smo. *Machine Learning*, 46(1-3):271–290, 2002. ISSN 0885-6125. doi: 10.1023/A:1012474916001. (Cited on page 56.)

[44] Dmitriy Fradkin and David Madigan. Experiments with random projections for machine learning. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–522, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956812. (Cited on page 64.)

[45] Damien François. *High-dimensional Data Analysis: From Optimal Metrics to Feature Selection*. VDM Verlag, 2008. (Cited on page 76.)

[46] Jessica Fridrich. Feature-based steganalysis for jpeg images and its implications for future design of steganographic schemes. In *Information Hiding*, volume 3200 of *Lecture Notes in Computer Science*, pages 67–81, 2004. (Cited on pages xxi, xxiii, 28, 34, 35, 36, and 80.)

[47] Jessica Fridrich. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, December 2009. (Cited on page 39.)

[48] Jessica Fridrich and Miroslav Goljan. Practical steganalysis-state of the art. In *Proc. SPIE Photonics West*, volume 4675, pages 1–13, California, USA, January 2002. (Cited on pages 28, 33, and 54.)

[49] Jessica Fridrich, Miroslav Goljan, and Rui Du. Reliable detection of lsb steganography in grayscale and color images. In *Proc. of the ACM Workshop on Multimedia and Security*, pages 27–30, Ottawa, Canada, October 2001. (Cited on pages 13 and 17.)

[50] Jessica Fridrich, Miroslav Goljan, Dorin Hogea, and David Soukal. Quantitative steganalysis of digital images: Estimating the secret message length. *ACM Multimedia Systems Journal, Special issue on Multimedia Security*, 9(3):288–302, 2003. (Cited on page 28.)

[51] Jessica Fridrich, Miroslav Goljan, and David Soukal. Searching for the stego key. In *Proceedings of SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 70–82, San Jose, CA, USA, 2004. (Cited on pages 23 and 24.)

[52] Jessica Fridrich, Miroslav Goljan, Petr Lisonek, and David Soukal. Writing on wet paper. *Signal Processing, IEEE Transactions on*, 53(10): 3923–3935, October 2005. ISSN 1053-587X. doi: 10.1109/TSP.2005.855393. (Cited on page 15.)

[53] Jessica Fridrich, Miroslav Goljan, David Soukal, and Taras Holotyak. Forensic steganalysis: Determining the stego key in spatial domain steganography. In *Proceedings of SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 631–642, San Jose, CA, USA, January 16–20 2005. (Cited on page 28.)

[54] Jessica Fridrich, Miroslav Goljan, and David Soukal. Wet paper codes with improved embedding efficiency. *Information Forensics and Security, IEEE Transactions on*, 1(1):102–110, March 2006. ISSN 1556-6013. doi: 10.1109/TIFS.2005.863487. (Cited on page 15.)

[55] Jessica Fridrich, Petr Lisoněk, and David Soukal. On steganographic embedding efficiency. In *Information Hiding 2006*, volume 4437/2007 of *Lecture Notes in Computer Science*, pages 282–296, 2007. (Cited on page 15.)

[56] Jessica Fridrich, Tomáš Pevný, and Jan Kodovský. Statistically undetectable jpeg steganography: dead ends challenges, and opportunities. In *MMSec'07: Proceedings of the 9th workshop on Multimedia & security*, pages 3–14, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-857-2. doi: 10.1145/1288869.1288872. (Cited on pages 20 and 21.)

[57] S. I. Gel'fand and Mark S. Pinsker. Coding for channel with random parameters. *Problems of Control Theory*, 9(1):19–31, 1980. (Cited on page 12.)

[58] Neil A. Gershenfeld and Andreas S. Weigend. The future of time series: Learning and understanding. Working Papers 93-08-053, Santa Fe Institute, August 1993. URL `http://ideas.repec.org/p/wop/safiwp/93-08-053.html`. (Cited on page 48.)

[59] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003. (Cited on page 79.)

[60] Barbara Hammer and Thomas Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002. ISSN 0893-6080. doi: 10.1016/S0893-6080(02)00079-5. (Cited on pages 58 and 59.)

[61] Barbara Hammer, Marc Strickert, and Thomas Villmann. On the generalization ability of grlvq networks. *Neural Processing Letters*, 21(2): 109–120, April 2005. (Cited on page 58.)

[62] Edward J. Hannan and Barry G. Quinn. The determination of the order of an autoregression. *Journal of the Royal Statistical Society, B*, 41: 190–195, 1979. (Cited on pages 2 and 71.)

[63] Simon Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 2nd edition edition, July 1998. ISBN 0132733501. (Cited on pages 56 and 57.)

[64] Herodotus. *The Histories*. Herodotus, 440 BC. (Cited on page 10.)

[65] Stefan Hetzl and Petra Mutzel. A graph-theoretic approach to steganography. In Dittmann J., Katzenbeisser S., and Uhl A., editors, *CMS 2005*, Lecture Notes in Computer Science 3677, pages 119–128. Springer-Verlag, 2005. (Cited on page 19.)

[66] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. ISSN 0893-6080. doi: 10.1016/0893-6080(91)90009-T. (Cited on page 56.)

[67] Guang-Bin Huang. Elm benchmarking, 2010. URL `http://www3.ntu.edu.sg/home/egbhuang/ELM_Benchmarking.htm`. (Cited on page 63.)

[68] Guang-Bin Huang, Chen Lei, and Chee-Kheong Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE transactions on neural networks*, 17(4): 879–892, 2006. (Cited on page 67.)

[69] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70:489–501, 2006. (Cited on pages 2, 3, 66, 67, and 93.)

[70] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks, gmd report 148. Technical report, German National Research Institute for Computer Science, 2001. (Cited on page 65.)

[71] Herbert Jaeger. Adaptive nonlinear system identification with echo state networks. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 593–600. MIT Press, Cambridge, MA, 2003. (Cited on page 65.)

[72] William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In American Mathematical Society, editor, *Conference in modern analysis and probability*, volume 26, pages 189–206, New Haven, Connecticut, 1982. (Cited on page 64.)

[73] Christian Jutten and Jeanny Herault. Blind separation of sources, part 1: an adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1–10, 1991. ISSN 0165-1684. doi: 10.1016/0165-1684(91)90079-X. (Cited on page 45.)

[74] Michael Kearns and Dana Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11(6): 1427–1453, 1999. ISSN 0899-7667. doi: 10.1162/089976699300016304. (Cited on page 51.)

[75] S. Sathiya Keerthi, Shirish K. Shevade, Chiranjib Bhattacharyya, and K. R. Krishna Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649, 2001. ISSN 0899-7667. doi: 10.1162/089976601300014493. (Cited on page 56.)

[76] Andrew D. Ker. Batch steganography and pooled steganalysis. In *Information Hiding*, volume 4437/2007 of *Lecture Notes in Computer Science*, pages 265–281. Springer-Verlag, September 2007. (Cited on pages 15, 22, and 95.)

[77] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, 9:5–38, January 1883. (Cited on page 24.)

[78] Anja Keskinarkaus, Anu Pramila, Tapio Seppänen, and Jaakko Sauvola. Wavelet domain print-scan and jpeg resilient data hiding method. In *Digital Watermarking*, volume 4283/2006 of *Lecture Notes in Computer Science*, pages 82–95, November 2006. (Cited on page 9.)

[79] Younhee Kim, Zoran Duric, and Dana Richards. Modified matrix encoding technique for minimal distortion steganography. In *Information Hiding 2007*, volume 4437/2007, pages 314–327, 2007. (Cited on page 21.)

[80] Jan Kodovský, Tomáš Pevný, and Jessica Fridrich. Modern steganalysis can detect yass. In *Proceedings of SPIE, Electronic Imaging, Media Forensics and Security XII*, pages 02–01–02–11, San Jose, CA, USA, January 17—21 2010. (Cited on pages 28 and 37.)

[81] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. pages 509–521, 1988. (Cited on page 59.)

[82] Teuvo Kohonen. *Self-Organizing Maps, 3rd edition*. Springer, 2001. (Cited on pages 58 and 59.)

[83] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Phys. Rev. E*, 69(6):66–138, June 2004. doi: 10.1103/PhysRevE.69.066138. (Cited on page 25.)

[84] Allan Latham. Jphide&seek, August 1999. URL `http://linux01.gwdg.de/~alatham/stego.html`. (Cited on page 21.)

[85] Amaury Lendasse, John A. Lee, Vincent Wertz, and Michel Verleysen. Forecasting electricity consumption using nonlinear projection and self-organizing maps. *Neurocomputing*, 48(1-4):299–311, October 2002. (Cited on page 59.)

[86] Bin Li, Yun Q. Shi, and Jiwu Huang. Steganalysis of yass. In *MM&Sec '08: Proceedings of the 10th ACM workshop on Multimedia and security*, pages 139–148, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-058-6. doi: 10.1145/1411328.1411354. (Cited on page 38.)

[87] Qingzhong Liu, Andrew H. Sung, and Bernardete M. Ribeiro. Statistical correlations and machine learning for steganalysis. In *Adaptive and Natural Computing Algorithms*, pages 437–440, Coimbra, Portugal, 2005. (Cited on page 54.)

[88] Wolfgang Maass, Robert A. Legenstein, and Henry Markram. A new approach towards vision suggested by biologically realistic neural microcircuit models. In *BMCV '02: Proceedings of the Second International Workshop on Biologically Motivated Computer Vision*, pages 282–293, London, UK, 2002. Springer-Verlag. ISBN 3-540-00174-3. (Cited on page 65.)

[89] Benjamin M. Marlin. *Missing Data Problems in Machine Learning*. PhD thesis, University of Toronto, 2008. (Cited on page 44.)

[90] Paul Merlin, Antti Sorjamaa, Bertrand Maillet, and Amaury Lendasse. X-SOM and l-SOM: A double classification approach for missing value imputation. *Neurocomputing*, to appear. (Cited on page 44.)

[91] Michael K. Meyerhoff. A timetable for talking. *Pediatrics for Parents*, June 2007. (Cited on page 43.)

[92] Yoan Miche, Benjamin Schrauwen, and Amaury Lendasse. Machine learning techniques based on random projections. In Michel Verleysen, editor, *ESANN2010: 18th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 295–302, Bruges, Belgium, April 28–30 2010. d-side Publications. (Cited on page 65.)

[93] Martin F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993. (Cited on page 57.)

[94] Raymond H. Myers. *Classical and Modern Regression with Applications, 2nd edition*. Duxbury, Pacific Grove, CA, USA, 1990. (Cited on pages 50 and 69.)

[95] Mark Noto. Mp3stego: Hiding text in mp3 files. September 2001. URL `http://www.tulane.edu/~park/courses/ElectronicMusicHistory/papers/Mp3Stego-DataHidingInMP3.pdf`. (Cited on page 11.)

[96] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901. (Cited on page 65.)

[97] Tomáš Pevný. *Kernel Methods in Steganalysis*. PhD thesis, Binghamton University, SUNY, May 2008. (Cited on page 25.)

[98] Tomáš Pevný and Jessica Fridrich. Towards multi-class blind steganalyzer for jpeg images. In *Digital Watermarking*, volume 3710/2005 of *Lecture Notes in Computer Science*, pages 39–53. Springer Berlin / Heidelberg, 2005. (Cited on page 28.)

[99] Tomáš Pevný and Jessica Fridrich. Merging markov and dct features for multi-class jpeg steganalysis. In Edward J. Delp III and Ping Wah Wong, editors, *Proc. SPIE Electronic Imaging, Photonics West*, volume 6505, pages 03–04. SPIE, January 2007. (Cited on pages 28, 36, 37, 38, 63, 69, and 85.)

[100] Tomáš Pevný and Jessica Fridrich. Novelty detection in blind steganalysis. In *Proceedings of the 10th workshop on Multimedia & security*, pages 167–176, Oxford, UK, UK, September 22–23 2008. ACM, New York, USA. (Cited on page 28.)

[101] Tomáš Pevný and Jessica Fridrich. Benchmarking for steganography. In K. Solanki, editor, *Information Hiding, 10th International Workshop*, Lecture Notes in Computer Science, pages 251–267, Santa Barbara, CA, May 19–21, 2008. Springer-Verlag, New York. (Cited on page 25.)

[102] Tomáš Pevný, Patrick Bas, and Jessica Fridrich. Steganalysis by subtractive pixel adjacency matrix. In *Proceedings of the 11th workshop on Multimedia & security, Princeton, NJ, USA, September 7–8*, pages 75–84. ACM, New York, USA, 2009. (Cited on pages 38, 39, and 79.)

[103] Tomáš Pevný, Jessica Fridrich, and Andrew D. Ker. From blind to quantitative steganalysis. In Edward J. Delp III, Jana Dittmann, Nasir D. Memon, and Ping Wah Wong, editors, *Proc. SPIE, Electronic Imaging, Media Forensics and Security XI*, volume 7254, pages oC1–oC14, San Jose, CA, USA, January 18–22 2009. SPIE. (Cited on pages 28, 54, and 85.)

[104] John C. Platt, Nello Cristianini, and John Shawe-taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, pages 547–553. MIT Press, 2000. (Cited on page 47.)

[105] Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1989. (Cited on page 57.)

[106] Niels Provos. Defending against statistical steganalysis. In *10th USENIX Security Symposium*, pages 323–335, 13-17 April 2001. (Cited on page 18.)

[107] C. Radhakrishna Rao and Sujit Kumar Mitra. *Generalized Inverse of Matrices and Its Applications*. John Wiley & Sons Inc, January 1972. (Cited on page 67.)

[108] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. (Cited on page 59.)

[109] Jim Reeds. Solved: the ciphers in book iii of trithemius' steganographia. *Cryptologia*, 22(4):291–318, October 1998. (Cited on page 9.)

[110] William H. Rogers and Terry J. Wagner. A finite sample distribution-free performance bound for local discrimination rules. *The Annals of Statistics*, 6(3):506–514, May 1978. (Cited on page 51.)

[111] Phil Sallee. Model-based steganography. In *Digital Watermarking*, volume 2939/2004 of *Lecture Notes in Computer Science*, pages 154–167. Springer Berlin / Heidelberg, 2004. (Cited on pages 14 and 20.)

[112] Atsushi Sato and Keiji Yamada. Generalized learning vector quantization. In *Advances in Neural Information Processing Systems*, volume 8, pages 423–429. The MIT Press, 1996. (Cited on page 58.)

[113] Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978. (Cited on page 71.)

[114] Toby Sharp. An implementation of key-based digital signal steganography. In *IHW '01: Proceedings of the 4th International Workshop on Information Hiding*, pages 13–26, London, UK, 2001. Springer-Verlag. ISBN 3-540-42733-3. (Cited on page 17.)

[115] John Shaver. Implemenation of steganography via mp3 and rsa. Plenary Talk in BOISECRYPT'09, November 2009. (Cited on page 11.)

[116] Yun Q. Shi, Chunhua Chen, and Wen Chen. A markov process based approach to effective attacking jpeg steganography. In *Information Hiding*, volume 4437/2007 of *Lecture Notes in Computer Science*, pages 249–264, September 2007. (Cited on page 37.)

[117] Timo Similä and Jarkko Tikka. Multiresponse sparse regression with application to multidimensional scaling. In *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005*, volume 3697/2005, pages 97–102. Springer, Warsaw, Poland, September 11-15 2005. (Cited on pages 2, 3, 69, and 93.)

[118] Harald Slatky. *Algorithms for direction specific Processing of Sound Signals - the Realisation of a binaural Cocktail-Party-Processor-System*. PhD thesis, Department of Electrical Engineering, Ruhr-University Bochum, Germany, 1992. (Cited on page 45.)

[119] Kaushal Solanki, Anindya Sarkar, and B. S. Manjunath. Yass: Yet another steganographic scheme that resists blind steganalysis. In *Information Hiding*, volume 4567/2007 of *Lecture Notes in Computer Science*, pages 16–31. Springer Berlin / Heidelberg, 2007. (Cited on pages 21 and 22.)

[120] Antti Sorjamaa, Paul Merlin, Bertrand Maillet, and Amaury Lendasse. A non-linear approach for completing missing values in temporal databases. *European Journal of Economic and Social Systems*, 22(1):99–117, November 2009. doi: 10.3166/EJESS.22.99-117. (Cited on page 44.)

[121] Antti Sorjamaa, Amaury Lendasse, Yves Cornet, and Eric Deleersnijder. An improved methodology for filling missing values in spatiotemporal climate data set. *Computational Geosciences*, 14:55–64, January 2010. doi: 10.1007/s10596-009-9132-3. (Cited on page 44.)

[122] Johan A.K. Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. Least squares support vector machines. In *World Scientific*, Singapore, 2002. (Cited on page 56.)

[123] Robert Tibshirani, Trevor Hastie, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2003. (Cited on page 57.)

[124] Johannes Trithemius. *Steganographia: Ars per occultam Scripturam animi sui voluntatem absentibus aperiendi certu*. Johannes Trithemius, 1500. (Cited on page 9.)

[125] Stefania Tronci, Francesco Corona, Massiomiliano Grosso, Roberto Calento, and Francesca Murena. Comparing neural networks and regression models for air quality management. In *Proceedings of EANN 2005 International Conference on Engineering Applications of Neural Networks, Lille (France)*, pages 93–100, August 24-26 2005. (Cited on page 56.)

[126] Andrey Nikolayevich Tychonoff. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics*, 4:1035—-1038, 1963. (Cited on page 54.)

[127] A.J. Umbarkar, A.P. Joshi, A.A. Jadhav, and A.R. Buchade. Wave steganography approach by modified lsb. In *Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on*, pages 862–865, December 16-18 2009. doi: 10.1109/ICETET.2009.230. (Cited on page 11.)

[128] Derek Upham. Jsteg. URL http://zooid.org/~paul/crypto/jsteg/. (Cited on page 21.)

[129] Vladimir N. Vapnik and Alexander Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24(6): 774–780, 1963. (Cited on page 55.)

[130] Santosh S. Vempala. *The Random Projection Method*. American Mathematical Society, February 2005. (Cited on page 65.)

[131] David Verstraeten, Benjamin Schrauwen, Michel D'Haene, and Dirk Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, Jan 2007. doi: 10.1016/j.neunet.2007.04.003. (Cited on pages 65 and 66.)

[132] Andreas Westfeld. F5-a steganographic algorithm. In *Information Hiding: 4th International Workshop*, volume 2137, pages 289–302, 25-27 Avril 2001. (Cited on pages 15, 19, and 21.)

[133] Andreas Westfeld and Andreas Pfitzmann. Attacks on steganographic systems. In *IH '99: Proceedings of the Third International Workshop on Information Hiding*, pages 61–76, London, UK, 2000. Springer-Verlag. ISBN 3-540-67182-X. (Cited on pages xxi, 17, 29, and 30.)

[134] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir N. Vapnik. Feature selection for svms. In *Advances in Neural Information Processing Systems 13*, pages 668–674. MIT Press, 2000. (Cited on page 78.)

[135] Qi Yu, Antti Sorjamaa, Yoan Miche, and Eric Séverin. A methodology for time series prediction in finance. In Amaury Lendasse, editor, *ESTSP, European Symposium on Time Series Prediction*, pages 285–293, Porvoo, Finland, September 17-19 2008. Multiprint Oy / Otamedia , Espoo, Finland. (Cited on pages 56 and 59.)

[136] Qi Yu, Yoan Miche, Antti Sorjamaa, Alberto Guillén, Amaury Lendasse, and Eric Séverin. OP-KNN: Method and applications. *Advances in Artificial Neural Systems*, 2010(597373):6 pages, February 2010. doi: 10.1155/2010/597373. (Cited on pages 2, 59, and 72.)

Part IV

PUBLICATIONS

PUBLICATION A

TITLE:
OP-ELM: Optimally-Pruned Extreme Learning Machine

AUTHORS:
Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten and Amaury Lendasse

PUBLISHED IN:
IEEE Transactions on Neural Networks, January 2010,
Number 1, pp. 158–162 , Volume 21

DOI:
http://dx.doi.org/10.1109/TNN.2009.2036259

Brief Papers

OP-ELM: Optimally Pruned Extreme Learning Machine

Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten, and Amaury Lendasse

***Abstract*—In this brief, the optimally pruned extreme learning machine (OP-ELM) methodology is presented. It is based on the original extreme learning machine (ELM) algorithm with additional steps to make it more robust and generic. The whole methodology is presented in detail and then applied to several regression and classification problems. Results for both computational time and accuracy (mean square error) are compared to the original ELM and to three other widely used methodologies: multilayer perceptron (MLP), support vector machine (SVM), and Gaussian process (GP). As the experiments for both regression and classification illustrate, the proposed OP-ELM methodology performs several orders of magnitude faster than the other algorithms used in this brief, except the original ELM. Despite the simplicity and fast performance, the OP-ELM is still able to maintain an accuracy that is comparable to the performance of the SVM. A toolbox for the OP-ELM is publicly available online.**

***Index Terms*—Classification, extreme learning machine (ELM), least angle regression (LARS), optimally pruned extreme learning machine (OP-ELM), regression, variable selection.**

I. Introduction

Since the data can be collected automatically from various and numerous sources, the global amount of information tends to grow rapidly in many fields of science. Although these data most likely improve the precision and details about the considered phenomena, they are also raising many new challenges. Storing of large data sets can get difficult, while actual processing of it can only be automated and by using very fast algorithms. "Manual" analysis is clearly impossible and the computational complexity of the used methodologies have to be kept as low as possible to be able to process even more data.

Among the most famous algorithms used for data processing through machine learning techniques lie feedforward neural networks [1]. While multilayer feedforward neural networks have been proven to be universal approximators [2], they tend not to be widely used when processing important data sets. Hence, linear models are often preferred for industrial applications, because they are much faster to build compared to the computational complexity required for a neural network, or most nonlinear models in general.

Manuscript received December 05, 2008; accepted October 29, 2009. First published December 08, 2009; current version published January 04, 2010. This work was supported in part by the Academy of Finland Centre of Excellence, by the Adaptive Informatics Research Centre, and by the Finnish Funding Agency for Technology and Innovation under the NoTeS project.

Y. Miche is with the Gipsa-Lab, INPG/CNRS, Grenoble 38402, France and also with the Department of Information and Computer Science, Helsinki University of Technology, Espoo 02015, Finland (e-mail: yoan.miche@tkk.fi).

A. Sorjamaa, O. Simula, and A. Lendasse are with the Department of Information and Computer Science, Helsinki University of Technology, Espoo 02015, Finland (e-mail: antti.sorjamaa@tkk.fi; olli.simula@tkk.fi; lendasse@tkk.fi).

P. Bas and C. Jutten are with the Gipsa-Lab, INPG/CNRS, Grenoble, France (e-mail: patrick.bas@inpg.fr; christian.jutten@inpg.fr).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2036259

The slow building of these neural networks comes from a few facts that remain inherent to the various existing training algorithms. Usually many parameters are required for a proper selection of the model structure and afterwards, the training. Moreover, these parameters are selected and tuned via slow algorithms and the whole model structure and training has to be repeated many times to make sure the model is fitting the data sufficiently well.

Recently, in [3], Huang *et al.* proposed an original algorithm called extreme learning machine (ELM). This method makes the selection of the weights of the hidden neurons very fast in the case of single-layer feedforward neural network (SLFN). Hence, the overall computational time for model structure selection and actual training of the model is often reduced even by hundreds, compared to some classical methods [2], [4]–[6]. Furthermore, the algorithm remains rather simple, which makes its implementation easy.

It is believed though that the ELM algorithm can have some issues when encountering irrelevant or correlated data. For this reason, a methodology named optimally pruned extreme learning machine (OP-ELM), based on the original ELM algorithm, is proposed in this brief. The OP-ELM extends the original ELM algorithm and wraps this extended algorithm within a methodology using a pruning of the neurons, leading to a more robust overall algorithm. Pruning of neurons in a network built using ELM has been proposed recently by Rong *et al.* in [7], for classification purposes, and using statistical tests as a measure of relevance of the neurons regarding the output. The OP-ELM presented here applies to both classification and regression problems and uses a leave-one-out (LOO) criterion for the selection of an appropriate number of neurons.

In the next section, the actual OP-ELM and the whole wrapping methodology are presented, along with the original ELM. Section III presents the data sets used for the experiments as well as results concerning computational speed and accuracy for the OP-ELM, ELM, multilayer perceptron network (MLP), Gaussian process (GP), and support vector machines (SVMs).

II. The Methodology

The OP-ELM methodology is based on the original ELM algorithm from which it borrows the original SLFN construction. In the following, the main concepts and theory of the ELM algorithm are shortly reviewed, with an example on the possible problems encountered by the ELM on data sets with irrelevant variables.

The OP-ELM algorithm is introduced as a more robust methodology regarding irrelevant variables situation. The steps of the algorithm are detailed and the network pruning algorithm, multiresponse sparse regression (MRSR), is described, along with the validation method LOO.

There is a Matlab toolbox available online for performing the OP-ELM methodology [8], along with a detailed user's manual.[1] A version of the toolbox translated to C language is coming soon.

A. ELM and OP-ELM

1) Extreme Learning Machine: The ELM algorithm was originally proposed by Huang *et al.* in [3] and it makes use of the SLFN. The main concept behind the ELM lies in the random initialization of the SLFN weights and biases. Then, using Theorem 1 and under the conditions of the theorem, the input weights and biases do not need to be adjusted

[1]Available at: http://www.cis.hut.fi/projects/tsp/index.php?page=OPELM

and it is possible to calculate implicitly the hidden-layer output matrix and hence the output weights. The network is obtained with very few steps and very low computational cost.

Consider a set of M distinct samples $(\mathbf{x}_i, \mathbf{y}_i)$ with $\mathbf{x}_i \in \mathbb{R}^{d_1}$ and $\mathbf{y}_i \in \mathbb{R}^{d_2}$; then, an SLFN with N hidden neurons is modeled as the following sum:

$$\sum_{i=1}^{N} \boldsymbol{\beta}_i f(\mathbf{w}_i \mathbf{x}_j + b_i), \qquad 1 \le j \le M \tag{1}$$

with f being the activation function, $\mathbf{w}_i$ the input weights, b_i the biases, and $\boldsymbol{\beta}_i$ the output weights.

In the case where the SLFN perfectly approximates the data, the errors between the estimated outputs $\hat{\mathbf{y}}_i$ and the actual outputs $\mathbf{y}_i$ are zero and the relation is

$$\sum_{i=1}^{N} \boldsymbol{\beta}_i f(\mathbf{w}_i \mathbf{x}_j + b_i) = \mathbf{y}_j, \qquad 1 \le j \le M \tag{2}$$

which writes compactly as $\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}$, with

$$\mathbf{H} = \begin{pmatrix} f(\mathbf{w}_1 \mathbf{x}_1 + b_1) & \cdots & f(\mathbf{w}_N \mathbf{x}_1 + b_N) \\ \vdots & \ddots & \vdots \\ f(\mathbf{w}_1 \mathbf{x}_M + b_1) & \cdots & f(\mathbf{w}_N \mathbf{x}_M + b_N) \end{pmatrix} \tag{3}$$

and $\boldsymbol{\beta} = (\beta_1^T \cdots \beta_N^T)^T$ and $\mathbf{Y} = (\mathbf{y}_1^T \cdots \mathbf{y}_M^T)^T$.

With these notations, Theorem 1 is proposed in [3], which is the pillar of the ELM idea. The theorem states that with randomly initialized input weights and biases for the SLFN, and under the condition that the activation function is infinitely differentiable, then the hidden-layer output matrix can be determined and will provide an approximation of the target values as good as wished (nonzero).

Theorem 1: Given any $\varepsilon > 0$ and an activation function $f : \mathbb{R} \mapsto \mathbb{R}$ infinitely differentiable in any interval, there exists $n < M$ such that for M distinct samples $(\mathbf{x}_i, \mathbf{y}_i)$, $\mathbf{x}_i \in \mathbb{R}^{d_1}$, $\mathbf{y}_i \in \mathbb{R}^{d_2}$, for any $\mathbf{w}_i \in \mathbb{R}^{d_1}$ and $b_i \in \mathbb{R}$, $||\mathbf{H}_{[M\times n]}\beta_{[n\times d_2]} - \mathbf{Y}_{[M\times d_2]}|| < \varepsilon$.

The way to calculate the output weights β from the knowledge of the hidden-layer output matrix $\mathbf{H}$ and target values is proposed with the use of a Moore–Penrose generalized inverse of the matrix $\mathbf{H}$, denoted as $\mathbf{H}^\dagger$ [9]. Overall, the ELM algorithm is summarized as follows.

Algorithm 1: ELM

Given a training set $(\mathbf{x}_i, \mathbf{y}_i)$, $\mathbf{x}_i \in \mathbb{R}^{d_1}$, $\mathbf{y}_i \in \mathbb{R}^{d_2}$, an activation function $f : \mathbb{R} \mapsto \mathbb{R}$, and the number of hidden nodes N:

1: Randomly assign input weights $\mathbf{w}_i$ and biases $b_i, 1 \le i \le N$;

2: Calculate the hidden-layer output matrix $\mathbf{H}$;

3: Calculate output weights matrix $\beta = \mathbf{H}^\dagger \mathbf{Y}$.

The proposed solution to the equation $\mathbf{H}\beta = \mathbf{Y}$ in the ELM algorithm, as $\beta = \mathbf{H}^\dagger \mathbf{Y}$ has three main properties making it an appealing solution.

1) It is one of the least squares solutions of the mentioned equation, hence the minimum training error can be reached with this solution.
2) It is the solution with the smallest norm among the least squares solutions.
3) The smallest norm solution among the least squares solutions is unique and it is $\beta = \mathbf{H}^\dagger \mathbf{Y}$.

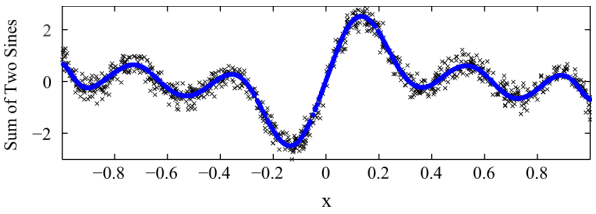

Fig. 1. Example of a training result using ELM, on a sum of two sines. Dots represent the ELM model fitting the data points (crosses).

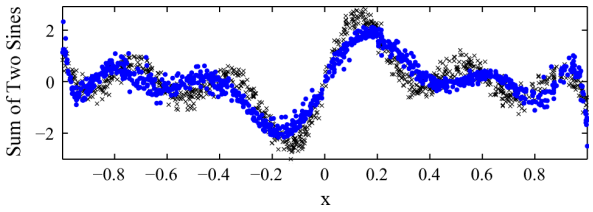

Fig. 2. Example using the same sum of sine as in Fig. 1 and an additional noisy variable (not represented here) for training. The obtained ELM model is much more spread and approximate, due to the irrelevant variable included.

Theoretical proofs and a more thorough presentation of the ELM algorithm are detailed in the original paper [3]. In Huang *et al.*'s later work, it has been proved that the ELM is able to perform universal function approximation [10].

2) The Problem of ELM With Irrelevant Variables: As already mentioned, the ELM models tend to have problems when irrelevant or correlated variables are present in the training data set. As an illustration of this, a toy example with two cases, without and with an irrelevant variable, are tested and compared.

Fig. 1 shows the ELM model obtained by training on the sum of sines example. In this case, the ELM model fits very well to the training data, with no apparent perturbation or distortion.

In Fig. 2, an additional variable containing a pure Gaussian noise, totally unrelated to the actual data, is also used as an input. The additional noise variable is not shown in the figure. The ELM model on top of the data is much more spread and approximate than in the previous case. Overall, the global fitting of the ELM model to the actual data is not as good as before.

For this reason, it is proposed in the OP-ELM methodology, to perform a pruning of the irrelevant variables, via pruning of the related neurons of the SLFN built by the ELM.

3) Optimally Pruned ELM: The OP-ELM is made of three main steps summarized in Fig. 3.

The very first step of the OP-ELM methodology is the actual construction of the SLFN using the original ELM algorithm with a lot of neurons.

Second and third steps are presented in more details in Sections II-A4 and II-A5 and are meant for an effective pruning of the possibly unuseful neurons of the SLFN: MRSR algorithm enables to obtain a ranking of the neurons according to their usefulness, while the actual pruning is performed using the results of the LOO validation.

The OP-ELM algorithm uses a combination of three different types of kernels, for robustness and more generality, where the original ELM proposed to use only sigmoid kernels. The used types are linear, sigmoid, and Gaussian kernels. Having the linear kernels included in the network helps when the problem is linear or nearly linear.

The Gaussian kernels have their centers taken randomly from the data points, similarly as in [11], and widths randomly drawn between

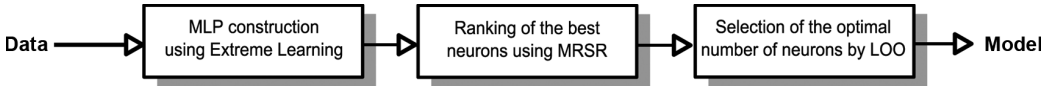

Fig. 3. Three steps of the OP-ELM algorithm.

percentile 20% and percentile 80% of the distance distribution of the input space, as suggested in [12].

The sigmoid weights are drawn randomly from a uniform distribution in the interval $[-5, 5]$ in order to cover the whole zero mean and unit variance data range.

The OP-ELM methodology can also handle multiple-output—multiple-class problems in both regression and classification using multiple inputs.

4) Multiresponse Sparse Regression: In order to get rid of the unuseful neurons of the hidden layer, the MRSR, proposed by Similä and Tikka [13], is used.

The main idea of the algorithm is as follows. Denote by $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_m]$ the $n \times m$ regressor matrix. MRSR adds each column of the regressor matrix one by one to the model $\hat{\mathbf{Y}}^k = \mathbf{X}\mathbf{W}^k$, where $\hat{\mathbf{Y}}^k = [\hat{\mathbf{y}}_1^k \dots \hat{\mathbf{y}}_p^k]$ is the target approximation of the model. The $\mathbf{W}^k$ weight matrix has k nonzero rows at kth step of the MRSR. With each new step, a new nonzero row and a new column of the regressor matrix are added to the model.

More specific details of the MRSR algorithm can be found from the original paper [13].

It can be noted that the MRSR is mainly an extension of the least angle regression (LARS) algorithm [14] and hence, it is actually a variable ranking technique, rather than a selection one. An important detail shared by the MRSR and the LARS is that the ranking obtained is exact, if the problem is linear. In fact, this is the case with the OP-ELM, since the neural network built in the previous step is linear between the hidden layer and the output. Therefore, the MRSR provides an exact ranking of the neurons for our problem. Because of the exact ranking provided by the MRSR, it is used to rank the kernels of the model. The target is the actual output $\mathbf{y}_i$, while the "variables" considered by the MRSR are the outputs of the kernels $\mathbf{h}_i = \mathrm{Ker}(\mathbf{x}_i^T)$, the columns of $\mathbf{H}$.

5) Leave-One-Out: Since the MRSR only provides a ranking of the kernels, the decision over the actual best number of neurons for the model is taken using an LOO validation method.

One problem with the LOO error is that it can be very time consuming, if the data set has a high number of samples. Fortunately, the PREdiction Sum of Squares (PRESS) statistics provide a direct and exact formula for the calculation of the LOO error for linear models (see [15] and [16] for details of this formula and its implementations)

$$\varepsilon^{\mathrm{PRESS}} = \frac{y_i - \mathbf{h}_i \mathbf{b}_i}{1 - \mathbf{h}_i \mathbf{P} \mathbf{h}_i^T} \tag{4}$$

where $\mathbf{P}$ is defined as $\mathbf{P} = (\mathbf{H}^T\mathbf{H})^{-1}$ and $\mathbf{H}$ is the hidden-layer output matrix.

The final decision over the appropriate number of neurons for the model can then be taken by evaluating the LOO error versus the number of neurons used. Here, the neurons are already ranked by the MRSR.

In order to give an overview of the usefulness of the ranking step performed by the MRSR algorithm, the final model structure selection for the OP-ELM model using the Ailerons data set (see Section III) is shown in Fig. 4.

It can be seen from Fig. 4 that the OP-ELM benefits greatly from the MRSR ranking step. The convergence is faster, because the LOO error gets to the minimum faster when the MRSR is used than when it is not. Also, the number of neurons is far fewer in the LOO error

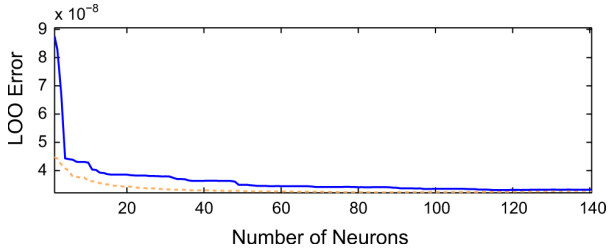

Fig. 4. Comparison of LOO error with and without the MRSR ranking. The solid line represents the LOO error without and the dashed line with the MRSR ranking.

TABLE I
INFORMATION ABOUT THE SELECTED DATA SETS. NUMBER OF VARIABLES AND NUMBER OF SAMPLES FOR BOTH TRAINING AND TESTING, TWO THIRDS OF THE WHOLE SET FOR TRAINING AND ONE THIRD FOR TEST. FOR CLASSIFICATION PROBLEMS, THE VARIABLES COLUMN ALSO INCLUDES THE NUMBER OF CLASSES IN THE DATA SET

		Samples	
Regression	# of Variables	Train	Test
Abalone	8	2784	1393
Ailerons	5	4752	2377
Elevators	6	6344	3173
Computer	12	5461	2731
Auto price	15	106	53
CPU	6	139	70
Servo	4	111	56
Breast Cancer	32	129	65
Bank	8	2999	1500
Stocks	9	633	317
Boston	13	337	169
Classification			
Iris	4/3	100	50
Wisconsin Breast Cancer	30/2	379	190
Pima Indians Diabetes	8/2	512	256
Wine	13/3	118	60

minimum point when using the MRSR ranking, thus leading to more sparse network with the same performance.

In the end, an SLFN possibly using a mix of linear, sigmoid, and Gaussian kernels is obtained, with a highly reduced number of neurons, all within a small computational time.

III. EXPERIMENTS

In the following, five methodologies are compared using several regression and classification tasks. The compared methods are GP, SVM, MLP, the original ELM, and the proposed OP-ELM.

A. Data Sets

Fifteen different data sets have been chosen for the experiments, 11 for regression and four for classification problems. The data sets are collected from the University of California at Irvine (UCI) Machine Learning Repository [17] and they have been chosen by the overall heterogeneity in terms of number of samples, variables, and classes for classification problems.

Table I summarizes the different attributes for the 15 data sets. All data sets have been preprocessed in the same way. Ten different random permutations of the whole data set are taken without replacement, and two thirds are used to create the training set and the remaining third is

TABLE II
COMPUTATIONAL TIMES (IN SECONDS) FOR ALL FIVE METHODOLOGIES ON THE REGRESSION DATA SETS. ALGORITHMS HAVE BEEN SORTED BY COMPUTATIONAL TIME. "AUTO P." STANDS FOR AUTO PRICE DATA SET AND "BREAST C." FOR BREAST CANCER DATA SET

	Abalone	Ailerons	Elevators	Computer	Auto P.	CPU	Servo	Breast C.	Bank	Stocks	Boston
SVM	6.6e+4	4.2e+2	5.8e+2	3.2e+5	2.6e+2	3.2e+2	1.3e+2	3.2e+2	1.6e+3	2.3e+3	8.5e+2
MLP	2.1e+3	3.5e+3	3.5e+3	8.2e+3	7.3e+2	5.8e+2	5.2e+2	8.0e+2	2.7e+3	1.2e+3	8.2e+2
GP	9.5e+2	2.9e+3	6.5e+3	6.3e+3	2.9	3.2	2.2	8.8	1.7e+3	4.1e+1	8.5
OPELM	5.7	16.8	29.8	26.2	2.7e-1	2.0e-1	2.1e-1	4.2e-1	8.03	1.54	7.0e-1
ELM	4.0e-1	9.0e-1	1.6	1.2	3.8e-2	4.2e-2	3.9e-2	4.8e-2	4.7e-1	1.1e-1	7.4e-2

used for the test set. Then, the training set is normalized, zero mean, and unit variance, and the test set is also normalized using the same mean and variance used for the training set. Because the test set is normalized using the same normalization parameters as for the training, it is most likely not exactly zero mean and unit variance.

It should also be noted that the proportions of the classes, for the classifications cases, have been kept balanced: each class is represented in an equal proportion, in both training and test sets. This is important in order to have relevant test results.

B. Experiments

Experiments have been conducted using the online versions of the methodologies, unaltered. All experiments have been run on the same x86_64 Linux machine with at least 4 GB of memory (no swapping for any of the experiments) and 2+ GHz processor. It should be noted that even though some methodologies are using parallelization of the tasks, the computational times are reported considering single-threaded execution on one single core, for the sake of comparisons.

The hyperparameters for the SVM and the MLP are selected using a tenfold cross validation.

The SVM is performed using the SVM toolbox [6] with the default settings for the hyperparameters and the grid search: the grid is logarithmic between 2^{-2} and 2^{10} for each hyperparameter; nu-SVC has been used for classification and epsilon-SVR for regression, with radial basis function kernel. The original grid search has been replaced by a parallelization process, which distributes parts of the grid over different machines.

The MLP [4] is performed using a neural network toolbox, which is part of the Matlab© software from the MathWorks, Inc. (Natick, MA). The training of the MLP is performed using the Levenberg–Marquardt backpropagation.

In order to decrease the possibility of local minima with the MLP, the training is repeated ten times for each fold and the best network according to the training error is selected for validation. For example, in order to validate the MLP network using 12 hidden neurons, we have to train a total of 100 MLP networks with 12 hidden neurons to evaluate the validation error. This procedure is done for each number of hidden nodes from 1 to 20 and the appropriate number according to the validation MSE is selected.

The GP is performed using a GPML toolbox for Matlab from Rasmussen and Williams [5]. The GP is performed using the default settings taken from the examples of usage of the toolbox.

Finally, the OP-ELM was used with all possible kernels, linear, sigmoid, and Gaussian, using a maximum number of 100 neurons.

1) Computational Times: Computational times are first reviewed for all five methodologies. Tables II and III give the computational times for training and test steps (sum of both), for each methodology. It can be noted that for all five methodologies, the computational times for the test steps are negligible compared to the training times; this is especially clear for large training times, like the SVM or MLP ones.

According to Tables II and III, the ELM is the fastest algorithm by several orders of magnitude compared, for example, to the SVM. This is in line with the claims of the ELM authors. The proposed OP-ELM is between one and three orders of magnitude slower than the original ELM, but still much faster than the rest of the compared methods in all data sets.

TABLE III
COMPUTATIONAL TIMES (IN SECONDS) COMPARED FOR ALL FIVE METHODOLOGIES FOR CLASSIFICATION DATA SETS. "WISC. B.C." FOR WISCONSIN BREAST CANCER DATA SET AND "PIMA I.D." FOR PIMA INDIANS DIABETES DATA SET

	Iris	Wisc. B.C.	Pima I.D.	Wine
SVM	2.3e+2	2.9e+3	3.3e+3	3.8e+2
MLP	7.6e+2	1.7e+3	4.1e+2	1.2e+3
GP	7.6e-1	6.1	5.8	1.9
OPELM	7.4e-2	1.1	9.6e-1	4.4e-1
ELM	2.4e-2	4.3e-2	4.8e-2	2.7e-2

However, the ranking of the SVM, MLP, and GP regarding the computational times is not exactly the same in all data sets, but in every case they are clearly slower than the ELM and OP-ELM.

The main reason why the OP-ELM has been designed in the first place is to add more robustness to the very simple and fast ELM algorithm. Experimental results for this robustness are presented in Section III-B2 through test results.

2) Test Errors: Because the validation results, while providing a good measure of the model fit to the data, do not measure the actual interpolation properties of the model, only the test results for the five models are presented in Tables IV and V.

According to the test results, the SVM is very reliable on average. Meanwhile, as mentioned earlier, the ELM can have good results with respect to its computational speed, but also it can have very high mean square errors (MSEs) on some test sets, for example, in Auto price and central processing unit (CPU) data sets.

In this regard, the OP-ELM manages to keep a good MSE, when comparing to other algorithms, and even rather close to the performance of the SVM (and of the GP) on many data sets used in the experiments. This comforts the earlier claims that the OP-ELM keeps a part of the speed of the ELM and, therefore, is much faster than most common algorithms, while remaining robust and accurate and providing good interpolation models.

Finally, in order to give an overview of the pruning result for the OP-ELM, Table VI lists the selected neurons for two data sets, one for regression and one for classification, namely, Ailerons and Iris.

One can see that the total number of kept neurons is fairly stable, and so is the number of linear neurons. It is interesting to note that the amount of neurons for each type is more stable for classification data sets than for regression one. On average, the situation depicted here is globally similar for other data sets.

Whether the stability of the number of neurons is a consequence of the size of the data set or the type of the problem, warrants further investigation.

IV. CONCLUSION

In this brief, the OP-ELM methodology has been detailed through the presentation of the three steps: the plain original ELM as the first step to build the SLFN, followed by a ranking of the neurons by the MRSR algorithm, and finally, the selection of the neurons that will remain in the final model through LOO validation.

TABLE IV
MEAN SQUARE ERROR RESULTS IN BOLDFACE (AND STANDARD DEVIATIONS IN REGULAR) FOR ALL FIVE METHODOLOGIES FOR THE REGRESSION DATA SETS. "AUTO P." STANDS FOR AUTO PRICE DATA SET AND "BREAST C." FOR BREAST CANCER DATA SET

	Abalone	Ailerons	Elevators	Computer	Auto P.	CPU	Servo	Breast C.	Bank	Stocks	Boston
SVM	**4.5**	**1.3e-7**	**6.2e-6**	**1.2e+2**	**2.8e+7**	**6.5e+3**	**6.9e-1**	**1.2e+3**	**2.7e-2**	**5.1e-1**	**3.4e+1**
	2.7e-1	2.6e-8	6.8e-7	8.1e+1	8.4e+7	5.1e+3	3.3e-1	7.2e-1	8.0e-4	9.0e-2	3.1e+1
OPELM	**4.9**	**2.8e-7**	**2.0e-6**	**3.1e+1**	**9.5e+7**	**5.3e+3**	**8.0e-1**	**1.4e+3**	**1.1e-3**	**9.8e-1**	**1.9e+1**
	6.6e-1	1.5e-9	5.4e-8	7.4	4.0e+6	5.2e+3	3.3e-1	3.6e+2	1.0e-6	1.1e-1	2.9
ELM	**8.3**	**3.3e-8**	**2.2e-6**	**4.9e+2**	**7.9e+9**	**4.7e+4**	**7.1**	**7.7e+3**	**6.7e-3**	**3.4e+1**	**1.2e+2**
	7.5e-1	2.5e-9	7.0e-8	6.2e+1	7.2e+9	2.5e+4	5.5	2.0e+3	7e-4	9.35	2.1e+1
GP	**4.5**	**2.7e-8**	**2.0e-6**	**7.7**	**2.0e+7**	**6.7e+3**	**4.8e-1**	**1.3e+3**	**8.7e-4**	**4.4e-1**	**1.1e+1**
	2.4e-1	1.9e-9	5.0e-8	2.9e-1	1.0e+7	6.6e+3	3.5e-1	1.9e+2	5.1e-5	5.0e-2	3.5
MLP	**4.6**	**2.7e-7**	**2.6e-6**	**9.8**	**2.2e+7**	**1.4e+4**	**2.2e-1**	**1.5e+3**	**9.1e-4**	**8.8e-1**	**2.2e+1**
	5.8e-1	4.4e-9	9.0e-8	1.1	9.8e+6	1.8e+4	8.1e-2	4.4e+2	4.2e-5	2.1e-1	8.8

TABLE V
CORRECT CLASSIFICATION RATES IN BOLDFACE (AND STANDARD DEVIATIONS IN REGULAR) FOR ALL FIVE METHODOLOGIES FOR CLASSIFICATION DATA SETS. "WISC. B.C." FOR WISCONSIN BREAST CANCER DATA SET AND "PIMA I.D." FOR PIMA INDIANS DIABETES DATA SET

	Iris	Wisconsin B.C.	Pima I.D.	Wine
SVM	**95.4**	**91.6**	**72.7**	**95.83**
	1.9	1.7	1.5	2.9
OPELM	**95.0**	**95.6**	**74.9**	**90.7**
	2.1	1.3	2.4	4.9
ELM	**72.2**	**95.6**	**72.2**	**81.8**
	1.01	1.2	1.9	6.2
GP	**95.6**	**97.3**	**76.3**	**96.1**
	2.3	0.9	1.8	2.1
MLP	**94.8**	**95.6**	**75.2**	**96.0**
	3.8	1.9	1.9	2.4

TABLE VI
DETAILS OF NUMBERS OF SELECTED NEURONS IN OP-ELM FOR THE DELTA AILERONS AND IRIS DATA SETS. "L" STANDS FOR LINEAR NEURONS, "S" FOR SIGMOID ONES, AND "G" FOR GAUSSIAN ONES

	Ailerons				Iris			
Run #	L	S	G	Total	L	S	G	Total
1	5	50	25	80	2	16	6	24
2	5	50	30	85	3	17	4	24
3	5	49	21	75	2	16	6	24
4	5	50	45	100	2	8	4	14
5	5	50	40	95	2	13	4	19
6	4	43	13	60	2	4	3	9
7	5	48	17	70	2	7	5	14
8	4	36	10	50	2	5	2	9
9	5	50	45	100	2	10	2	14
10	3	27	5	35	2	13	4	19

By the use of these steps, the speed and accuracy of the OP-ELM methodology has been demonstrated, through experiments using 12 different data sets for both regression and classification problems, all very different in terms of number of samples, variables, and outputs. The OP-ELM achieves roughly the same level of accuracy than the other well-known methods such as SVM, MLP, or GP. Even though the original ELM is much faster than the OP-ELM based on it, the accuracy of the ELM can be problematic in many cases, while the OP-ELM remains robust to all tested data sets.

The main goal in this brief was **not** to show that the OP-ELM is either the best in terms of MSE or the computational time. The main goal is to prove that it is a very good compromise between the speed of the ELM and the accuracy and robustness of much slower and complicated methods. Indeed, very accurate results, close to the SVM accuracy, can be obtained in a very small computational time. This makes the OP-ELM a valuable tool for the applications in need for a small response time with a good accuracy.

REFERENCES

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[3] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, Dec. 2006.

[4] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[5] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[6] C. C. Chang and C. J. Lin, LIBSVM: A Library for Support Vector Machines, 2001 [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm

[7] H. jun Rong, Y.-S. Ong, A.-W. Tan, and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing*, vol. 72, no. 1–3, pp. 359–366, 2008.

[8] A. Lendasse, A. Sorjamaa, and Y. Miche, OP-ELM Toolbox, 2008 [Online]. Available: http://www.cis.hut.fi/projects/tsp/index.php?page=OPELM

[9] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and Its Applications*. New York: Wiley, 1972.

[10] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[11] T. Poggio and F. Girosi, *A Theory of Networks for Approximation and Learning*. Cambridge, MA: MIT Press, 1989, vol. 1140.

[12] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. Cambridge, MA: MIT Press, Dec. 2001, 0262194759.

[13] T. Similä and J. Tikka, "Multiresponse sparse regression with application to multidimensional scaling," in *Proc. Int. Conf. Artif. Neural Netw.*, 2005, vol. 3697/2005, pp. 97–102.

[14] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.

[15] R. Myers, *Classical and Modern Regression With Applications*, 2nd ed. Pacific Grove, CA: Duxbury, 1990.

[16] G. Bontempi, M. Birattari, and H. Bersini, "Recursive lazy learning for modeling and control," in *Proc. Eur. Conf. Mach. Learn.*, 1998, pp. 292–303.

[17] A. Asuncion and D. Newman, UCI Machine Learning Repository, Univ. California Irvine, Irvine, CA, 2007 [Online]. Available: http://archive.ics.uci.edu/ml/

PUBLICATION B

B

TITLE:
A Faster Model Selection Criterion for OP-ELM and OP-KNN: Hannan-Quinn Criterion

AUTHORS:
Yoan Miche and Amaury Lendasse

PUBLISHED IN:
ESANN'09: European Symposium on Artificial Neural Networks, April 2009, Michel Verleysen ed., published by d-side publications.
pp. 177–182

URL:
http://www.cis.hut.fi/projects/tsp/Publications/Publication119.pdf

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

A faster model selection criterion for OP-ELM and OP-KNN: Hannan-Quinn criterion

Yoan Miche[1,2] and Amaury Lendasse[1]

1- Helsinki University of Technology - ICS Lab.
Konemiehentie 2, 02015 TKK - Finland

2- INPG Grenoble - Gipsa-Lab, UMR 5216
961 rue de la Houille Blanche, Domaine Universitaire, 38402 GRENOBLE - France

Abstract. The Optimally Pruned Extreme Learning Machine (OP-ELM) and Optimally Pruned K-Nearest Neighbors (OP-KNN) algorithms use the a similar methodology based on random initialization (OP-ELM) or KNN initialization (OP-KNN) of a Feedforward Neural Network followed by ranking of the neurons; ranking is used to determine the best combination to retain. This is achieved by Leave-One-Out (LOO) cross-validation. In this article is proposed to use the Hannan-Quinn (HQ) Criterion as a model selection criterion, instead of LOO. It proved to be efficient and as good as the LOO one for both OP-ELM and OP-KNN, while decreasing computations by factors of four to five for OP-ELM and up to 24 for OP-KNN.

1 Introduction

Since data can be collected automatically from various and numerous sources, the global amount of information grows rapidly in most fields of science. Although this data most likely improves precision and details, it also raises many new challenges such as storage and processing. Among the most famous algorithms used for data processing through machine learning techniques, lies Feedforward neural networks. While multilayer feedforward neural networks have been proved to be universal approximators [1], they tend not to be used widely when processing important datasets because of the computational time it takes to actually train and build them: many parameters are required for a proper selection of the model structure and afterwards, the training.

In order to make model training and selection of single hidden layer feedforward neural networks faster, OP-ELM [2] (based on ELM [3]) and OP-KNN [4] have been proposed recently. In this paper, is proposed a different model structure selection criterion (inside the OP-ELM/KNN algorithm) to replace the previously used Leave-One-Out; it is just as efficient and faster for large datasets. The next section presents the OP-ELM/KNN shortly, while section 3 details the Hannan-Quinn criterion used for complexity selection. Experiments and results using this improved methodology are presented in section 4.

2 OP-ELM and OP-KNN

The Otimally Pruned Extreme Learning Machine [2] (OP-ELM, based on original ELM [3]) and Optimally Pruned KNN [4] (OP-KNN) are based on similar

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

first steps which Figure 1 summarizes.

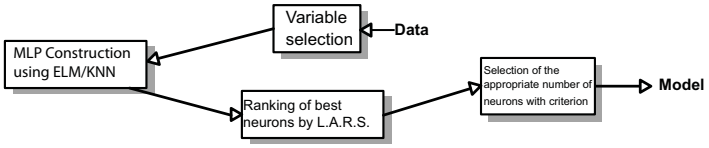

Fig. 1: OP-ELM/KNN methodology: first steps are similar. Last step of selection of neurons is performed using a criterion: Leave-One-Out (LOO) in the original algorithms.

A priori variable selection is first performed on the data. Then, the MultiLayer Perceptron (MLP), which is actually a single hidden layer feedforward network, is initialized, either by ELM (for OP-ELM) or by KNN (for OP-KNN). In the OP-ELM case, by a random initialization of the weights and biases of the MLP, while for OP-KNN, deterministic initialization using K-NN is used.

Neurons are then ranked using a MRSR [5] technique, which main idea is: Denote by $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ the $N \times M$ matrix of inputs, the MRSR adds each column of the regressor matrix one by one to the model $\hat{\mathbf{Y}}^k = \mathbf{X}\mathbf{W}^k$ where $\hat{\mathbf{Y}}^k = [\hat{\mathbf{y}}_1^k, \ldots, \hat{\mathbf{y}}_p^k]$ is the target approximation of the model. The $\mathbf{W}^k$ weight matrix has k nonzero rows at k-th step of the MRSR. With each new step a new nonzero row and a new column of the regressor matrix is added to the model. An important fact is that the obtained ranking by MRSR is exact in the case of a linear problem, as here since the neural network is linear between the hidden layer and output.

Finally, a criterion is used to decide which number of neurons will be retained (ranked neurons, so only the best ones are kept). This criterion is a Leave-One-Out, for the original OP-ELM/KNN.

The Leave-One-out (LOO) is usually a costly way of estimating a model's fit to the data, since it requires to go through all points of the data separately to estimate the model's output for each. In order to keep the OP-ELM/KNN fast, the PRESS Statistics [6] formula is used in order to compute this validation error, as in Eq. 1.

$$\epsilon_i^{\text{PRESS}} = \frac{y_i - \mathbf{x}_i \mathbf{b}}{1 - \mathbf{x}_i \mathbf{P} \mathbf{x}_i'}, \tag{1}$$

where $\mathbf{P} = (\mathbf{X}'\mathbf{X})^{-1}$, and $\mathbf{b}$ are the output weights of the MLP. While this formula makes it possible to evaluate the LOO error with simple matrix calculations, it still requires a matrix inversion and various matrix products, which can still be long. The goal of this paper is to present another criterion for the complexity selection (by the selection of the number of neurons to keep) which is much faster for it does not requires these matrix operations.

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

3 The Hannan-Quinn criterion

In order to perform complexity selection (by selecting the neurons to retain in the OP-ELM/KNN model), the classical LOO was used in the original versions of the two algorithms OP-ELM/KNN.

There are many possible criteria for complexity selection in machine learning. Typical examples are Akaike's information criterion (AIC) [7] or the Bayesian Information Criterion (BIC) [8]. Their expression is based on the residual sum of squares (Res) of the considered model (first term of the criterion) plus a penalty term (second term). Differences between criteria mostly occur on the penalty term. AIC penalizes only with the number of parameters p of the model (so that not too many free parameters are used to obtain a good fit by the model), Eq. 3; BIC takes into account the number of samples N used for the model training, in Eq. 2.

$$BIC = N \times \log\left(\frac{Res}{N}\right) + p \times \log N \tag{2}$$

$$AIC = N \times \left(\log\left(\frac{2\pi Res}{N}\right) + 1\right) + 2 \times p \tag{3}$$

The AIC is known to have consistency problems: while minimizing AIC, it is not guaranteed that complexity selection will converge toward an optima if the number of samples goes to infinity [9]. The main problem using such criteria is in trying to balance underfitting and overfitting knowing that convergence might never be achieved. One solution is through the penalty term, for example, by having a $\log N$ term in the penalty (with N the number of samples), which the BIC has. Unfortunately, for the experiments conducted in this paper, the BIC criterion did not give proper complexity selection (most likely due to the too fast increase of the penalty term with the number of samples).

The Hannan-Quinn Information Criterion [10] is close to these other criteria, as can be seen from the expressions of the AIC and BIC below (Eq. 2 and Eq. 3). The idea behind the design of this criterion is to provide a consistent criterion (regarding for example AIC which is not consistent in its standard definition) in which the second term (the penalty) $2 \times p \times \log\log N$ grows but at a very slow rate, regarding the number of samples.

$$HQ = N \times \log\left(\frac{Res}{N}\right) + 2 \times p \times \log\log N \tag{4}$$

From Figure 2, it can be seen that both criteria have very similar convergence regarding the number of neurons used for building the model. In this particular case, the HQ criterion is consistent since it enables a stable convergence. Hence, and from the following experiments, it can be considered that the HQ criterion is as good as the original LOO.

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

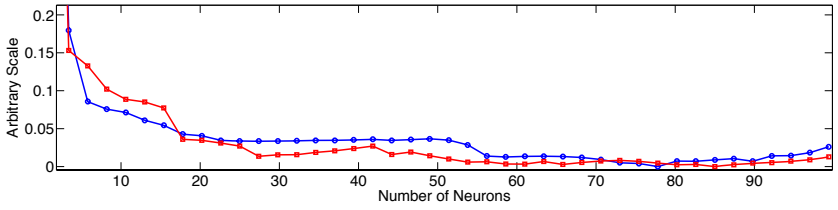

Fig. 2: Plot of the criterion value for both LOO and HQ versions of the OP-ELM (for the Bank dataset from UCI [11]): red squares for HQ and blue circles for LOO. Plots have been scaled to fit on same scale (HQ and LOO criteria have very different values). Convergence is very similar.

4 Experiments and results

Experiments for testing the effect of the HQ criterion on both OP-ELM and OP-KNN, have been conducted using seven different data sets from UCI machine learning repository [11]. The choice of these datasets has been made so that their variety in terms of number of samples and variables, covers usual "real life" datasets. Table 1 summarizes the characteristics of theses datasets.

		Samples	
Regression	# of Variables	Train	Test
Ailerons (D.A.)	5	4752	2377
Elevators (D.E.)	6	6344	3173
Auto price (A.P.)	15	106	53
Servo	4	111	56
Breast Cancer (B.C.)	32	129	65
Bank	8	2999	1500
Stocks	9	633	317

Table 1: Selected datasets: Number of variables and number of samples for training and testing (two thirds and one third respectively).

The datasets have been divided in two parts: training and test sets. Two thirds of the whole dataset for training and the remaining third for testing.

The original OP-ELM/KNN and their HQ modified versions have both been tested on these datasets, and results for test mean square errors and computational times are presented in Tables 2 and 3. It can be seen from Table 2 that the HQ version of the algorithms perform just as good, on average, as the original LOO-based version (or even slightly better). Results are within close range with SVM values (from LS-SVM [12]).

Computational times are highly reduced, when using the HQ criterion instead of the LOO, as expected (Table 3). A factor of four to five between the computational times, can be observed for OP-ELM, and up to 24 for the OP-

	A.P.	Bank	B.C.	D.A.	D.E.	Servo	Stocks
SVM	3.8E+06	2.2E-03	8.9E+02	2.6E-08	2.8E-06	4.2E-01	2.2E-01
OP-ELM	4.5E+06	1.1E-03	6.7E+02	2.7E-08	1.9E-06	5.8E-1	6.1E-1
OP-ELM-HQ	1.4E+06	1.1E-03	9.2E+02	2.6E-08	1.9E-06	5.7E-1	5.8E-1
# Neur. (HQ)	20 (50)	98 (98)	8 (4)	55 (95)	36 (26)	39 (100)	99 (99)
OP-KNN	2.7E+06	1.3E-03	1.1E+03	3.4E-08	2.5E-06	4.0E-01	4.8E-01
OP-KNN-HQ	3.1E+06	1.3E-03	1.1E+03	3.4E-08	2.4E-06	3.8E-01	4.8E-01
# Neur. (HQ)	46 (100)	45 (15)	2 (6)	23 (20)	17 (17)	59 (59)	11 (11)

Table 2: Test Mean Square errors comparisons for OP-ELM/KNN and HQ criterion version. Number of neurons for each are given: standard version in plain and HQ in parenthesis. 100 neurons used for OP-ELM and maximum 100-nearest neighbours for OP-KNN. SVM values for reference (using LS-SVM [12]).

KNN. It can also be seen that this difference is mostly noticeable when using large datasets (Ailerons, Elevators, Bank, here). While the difference is smaller for smaller datasets, it remains important enough to be considered when the OP-ELM/KNN is used many times, for variable selection with a Forward-Backward algorithm, for example. In these case, the small difference in computational time makes a clear difference on the many iterations.

	A.P.	Bank	B.C.	D.A.	D.E.	Servo	Stocks
SVM	492	6.5E+05	645	8.7E+04	7.7E+05	863	2188
OP-ELM	0.14	5.37	0.29	18.99	21.89	0.42	1.33
OP-ELM-HQ	0.13	1.88	0.24	4.59	5.42	0.40	0.84
Ratio	1.08	2.86	1.21	4.14	4.04	1.05	1.58
OP-KNN	1.6	17.7	0.09	43.16	67.04	0.08	0.91
OP-KNN-HQ	0.09	1.02	0.06	1.78	2.47	0.05	0.19
Ratio	17.78	17.35	1.5	24.25	27.14	1.6	4.79

Table 3: Computational times (seconds) when using OP-ELM/KNN and the HQ criterion version. Ratios between standard and HQ versions are given. Improvement with HQ criterion is visible when working with large datasets (matrix products for classical PRESS LOO need to be consequent for the HQ based version to take advantage). SVM values for reference (using LS-SVM [12]).

5 Conclusion

This paper presents a modification of the original OP-ELM/KNN algorithms in the place of the model structure selection criterion. The classical Leave-One-Out criterion is replaced by the Hannan-Quinn (HQ) criterion, which performances match the ones of the LOO (or perform actually slightly better, on average). The main advantage of this other criterion over the LOO one is to decrease the

computational times by a factor of four to five for OP-ELM and up to 24 for OP-KNN, for the conducted experiments. It seems very likely that for much larger datasets than the ones used for the experiments, the gain in computational time could be even higher.

References

[1] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[2] Y. Miche, P. Bas, C. Jutten, O. Simula, and A. Lendasse. A methodology for building regression models using extreme learning machine: OP-ELM. In *ESANN 2008, European Symposium on Artificial Neural Networks, Bruges, Belgium*, April 23-25 2008.

[3] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1–3):489–501, December 2006.

[4] Q. Yu, A. Sorjamaa, Y. Miche, A. Lendasse, A. Guillén, E. Séverin, and F. Mateo. Optimal pruned k-nearest neighbors: OP-KNN - application to financial modeling. In *HIS 2008, 8th International Conference on Hybrid Intelligent Systems*, September 10-12 2008.

[5] T. Similä and J. Tikka. Multiresponse sparse regression with application to multidimensional scaling. In *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005*, volume 3697/2005, pages 97–102. 2005.

[6] R.H. Myers. *Classical and Modern Regression with Applications, 2nd edition*. Duxbury, Pacific Grove, CA, USA, 1990.

[7] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974.

[8] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[9] R. J. Bhansali and D. Y. Downham. Some properties of the order of an autoregressive model selected by a generalization of akaike's epf criterion. *Biometrika*, 64(3):547–551, 1977.

[10] E. J. Hannan and B. G. Quinn. The determination of the order of an autoregression. *Journal of the Royal Statistical Society, B*, 41:190–195, 1979.

[11] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[12] Suykens J.A.K., Van Gestel T., De Brabanter J., B. De Moor B., and Vandewalle J. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

C

PUBLICATION C

TITLE:
A Feature Selection Methodology for Steganalysis

AUTHORS:
Yoan Miche, Benoit Roue, Patrick Bas and Amaury Lendasse

PUBLISHED IN:
MRCS'06, International Workshop on Multimedia Content Representation, Classification and Security, Istanbul (Turkey), B. Gunsel, A. K. Jain, A. M. Tekalp and B. Sankur eds., Lecture Notes in Computer Science, Springer-Verlag, 2006.
Volume 4105, pp. 49–56

DOI:
http://dx.doi.org/10.1007/11848035_9

A Feature Selection Methodology for Steganalysis

Yoan Miche[1], Benoit Roue[2], Amaury Lendasse[1], and Patrick Bas[1,2]

[1] Laboratory of Computer and Information Science
Helsinki University of Technology
P.O. Box 5400 FI-02015 Hut Finland
[2] Laboratoire des Images et des Signaux de Grenoble
961 rue de la Houille Blanche Domaine universitaire
B.P. 46 38402 Saint Martin d'Hères cedex France

Abstract. This paper presents a methodology to select features before training a classifier based on Support Vector Machines (SVM). In this study 23 features presented in [1] are analysed. A feature ranking is performed using a fast classifier called K-Nearest-Neighbours combined with a forward selection. The result of the feature selection is afterward tested on SVM to select the optimal number of features. This method is tested with the Outguess steganographic software and 14 features are selected while keeping the same classification performances. Results confirm that the selected features are efficient for a wide variety of embedding rates. The same methodology is also applied for Steghide and F5 to see if feature selection is possible on these schemes.

1 Introduction

The goal of steganographic analysis, also called steganalysis, is to bring out drawbacks of steganographic schemes by proving that an hidden information is embedded in a content. A lot of steganographic techniques have been developed in the past years, they can be divided in two classes: *ad hoc* schemes (schemes that are devoted to a specific steganographic scheme) [1,2,3] and schemes that are generic and that use classifiers to differentiate original and stego images[4,5]. The last ones work in two steps, generic feature vectors (high pass components, prediction of error...) are extracted and then a classifier is trained to separate stego images from original images. Classifier based schemes have been more studied recently, and lead to efficient steganalysis. Thus we focus on this class in this paper.

1.1 Advantages of Feature Selection for Steganalysis

Performing feature selection in the context of steganalysis offers several advantages.

- it enables to have a more rational approach for classifier-based steganalysis: feature selection prunes features that are meaningless for the classifier;

B. Gunsel et al. (Eds.): MRCS 2006, LNCS 4105, pp. 49–56, 2006.

- feature selection may also be used to improve the classification performance of a classifier (in [6] it is shown that the addition of meaningless features decreases the performance of a SVM-based classifier);
- another advantage of performing feature selection while training a classifier is that the selected features can help to point out the features that are sensitive to a given steganographic scheme and consequently to bring a highlight on its weaknesses.
- The last advantage of performing feature selection is the reduction of complexity for both generating the features and training the classifier. If we select a set of N features from a set of M, the training time will be divided by M/N (this is due to the linear complexity of classifiers regarding the dimension). The same complexity reduction can also be obtained for feature generation if we assume that the complexity to generate each feature is equivalent.

2 Fridrich's Features

The features used in this study were proposed by Fridrich *et al* [1]. All features are computed in the same way: a vector functional F is applied to the *stego JPEG image* J_1 and to the virtual *clean JPEG image* J_2 obtained by cropping J_1 with a translation of 4×4 pixels. The feature is finally computed taking the $L1$ of the difference of the two functionals :

$$f = ||F(J_1) - F(J_2)||_{L1}. \tag{1}$$

The functionals used in this paper are described in the Table 1.

Table 1. List of the 23 used features

Functional/Feature name	Functional **F**
Global histogram	$\mathbf{H}/\|\|\mathbf{H}\|\|$
Individual histogram for 5 DCT Modes	$\mathbf{h}^{21}/\|\|\mathbf{h}^{21}\|\|$,$\mathbf{h}^{12}/\|\|\mathbf{h}^{12}\|\|$,$\mathbf{h}^{13}/\|\|\mathbf{h}^{13}\|\|$, $\mathbf{h}^{22}/\|\|\mathbf{h}^{22}\|\|$,$\mathbf{h}^{31}/\|\|\mathbf{h}^{31}\|\|$
Dual histogram for 11 DCT values $(-5,\ldots,5)$	$\mathbf{g}^{-5}/\|\|\mathbf{g}^{-5}\|\|$,$\mathbf{g}^{-4}/\|\|\mathbf{g}^{-4}\|\|$,$\mathbf{g}^{-3}/\|\|\mathbf{g}^{-3}\|\|$,$\mathbf{g}^{-2}/\|\|\mathbf{g}^{-2}\|\|$,$\mathbf{g}^{-1}/\|\|\mathbf{g}^{-1}\|\|$, $\mathbf{g}^{0}/\|\|\mathbf{g}^{0}\|\|$,$\mathbf{g}^{1}/\|\|\mathbf{g}^{1}\|\|$,$\mathbf{g}^{2}/\|\|\mathbf{g}^{2}\|\|$,$\mathbf{g}^{3}/\|\|\mathbf{g}^{3}\|\|$,$\mathbf{g}^{4}/\|\|\mathbf{g}^{4}\|\|$,$\mathbf{g}^{5}/\|\|\mathbf{g}^{5}\|\|$
Variation	$\mathbf{V}$
$L1$ and $L2$ blockiness	$\mathbf{B}_1$, $\mathbf{B}_2$
Co-occurrence	N_{00}, N_{01}, N_{11}

3 Classifiers for Steganalysis

This section presents two classifiers that differ in term of complexity and a method to estimate the mean and variance of the classification accuracy obtained by any classifier.

- K-Nearest Neighbours: the K-NN classifiers use an algorithm based on a majority vote: using a norm (usually Euclidean), the K nearest points from the

one to classify are determined. The classification is then based on the class that belongs to the most numerous closest points, as shown on the figure (Fig 1). The choice of the K value is dependent on the data, and the best value is found using using a leave-one-out cross-validation procedure [7]. Note that if K-NN classifiers are usually less accurate than SVM classifiers, nevertheless, the computational time for training a K-NN is around 10 times smaller than for training a SVM.

- Support Vector Machines: SVM classification uses supervised learning systems to map in a non-linear way the features space into a higher dimensional feature space [8]. A hyper-plane can then be found in this high-dimensional space, which is at the maximum distance from the nearest data points of the two classes so that points to be classified can benefit from this optimal separation.

- Bootstrapping for noise estimation: the bootstrap algorithm enables to have a confidence interval for the performances [7]. A random mix with repetitions of the test set is created, and then used with the SVM model computed before with a fixed train set. This process is repeated R times and thus gives by averaging a correct noise estimation when N is large enough.

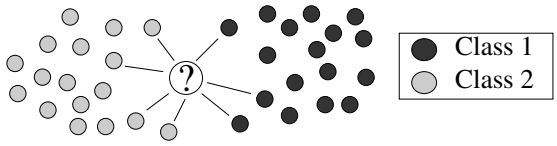

Fig. 1. Illustration of the K-NN algorithm. Here, $K = 7$: The Euclidean distance between the new point (?) and the 7 nearest neighbours is depicted by a line. In this case we have the majority for the light grey (4 nearest neighbours): the new point is said to be of class 2.

4 Feature Selection Methods

This section presents two different feature selection methods.

- Exhaustive search: in this case, we use a full scan of all possible features combinations and keep the one giving the best result. If you consider N features, the computational time to perform the exhaustive search equals the time to train/test one classifier multiplied by $2^N - 1$. Consequently this method can only be used with fast classification algorithms.

- The "forward" selection algorithm: The forward approach proposes a suboptimal but efficient way to incrementally select the best features [9]. The following steps illustrate this algorithm:

1. try the $\alpha_{i,i\in[\![1,N]\!]}$ features one by one;
2. keep the feature α_{i1} with the best results;
3. try all couples with α_{i1} and one feature among the remaining $N - 1$;
4. keep the couple $(\alpha_{i1}, \alpha_{i2})$ giving the best results;

5. try all triplets with $(\alpha_{i1}, \alpha_{i2})$ and one feature among the remaining $N-2$;
6. ... iterate until none remains.

The result is an array containing the N the features ranked by minimum error. The computational time is equal to $N \times (N+1)/2$ multiplied by the time spent to train/test one classifier.

4.1 Applying Feature Selection to SVMs

Using the forward algorithm directly on SVM is too time-consuming. Consequently we propose to perform the feature selection for SVMs in three steps depicted on Figure 2.

1. Forward using K-NN: in this step, we use the explained forward algorithm with a K-NN classification method to rank features vectors. Since the K-NN is fast enough, it is possible to run this step in a reasonable time.
2. SVM and Bootstrapping: using the ranked features list found by the K-NN forward algorithm, we run 23 SVMs using the 23 different feature vectors, and a bootstrap on the test set, with approximately 5000 iterations.
3. Features selection: in the end, the curve from the bootstrap data shows that within the noise estimation, we can reduce the number of features, based on the fact that the addition of some features degrades the classification result. Within the noise range, the first $L < N$ selected features present the best compromise for a same classification performance.

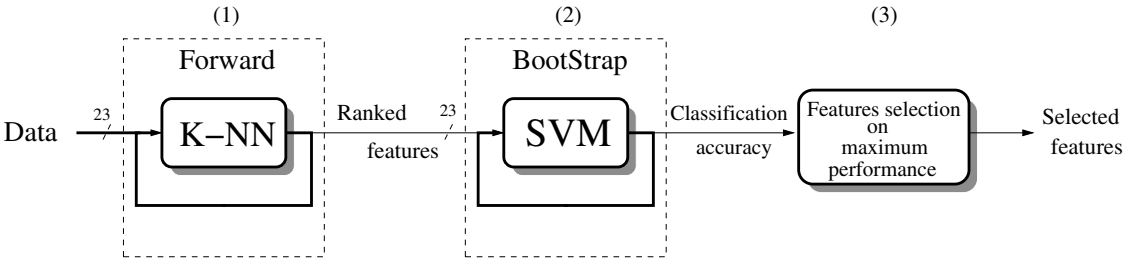

Fig. 2. Feature selection steps: features are first ranked by importance by the K-NN forward algorithm (1), SVMs give then improvement and an accuracy estimation thanks to a bootstrap (2). Features are in the end taken from the best SVM result (3).

5 Experimental Results

The experiments have been performed using a set of 5075 images from 5 different digital cameras (all over 4 megapixels). A mix of these images has then been made, and half of them have been watermarked using Outguess 0.2 [10], with and embedding rate of 10% of non zero quantised DCT coefficients. Each image has been scaled and cropped to 512×512, converted in grey levels and compressed using a JPEG quality factor of 80%. The extracted features from the 5075 images have then been divided in a training (1500 samples) and test set (3575 samples). The SVM library used is the libSVMtl [11].

5.1 Accuracy of KNN with Feature Selection

We present here (Fig 3) the classification accuracy of the forward algorithm using the K-NN method. In our case, the decision on whether to keep or leave out a feature has been made only on the results of the leave-one-out (i.e. using only the training set). As one can see from the curves, it finds the best set of features with only 6 of them (Leave-one-out classification rate around 0.705). Adding more features only results here in a degradation of the classification result.

But tryouts using only those 6 features have proven that it is not the best solution for SVM. Consequently, we choose to use this step of the process only to obtain a ranking of the features.

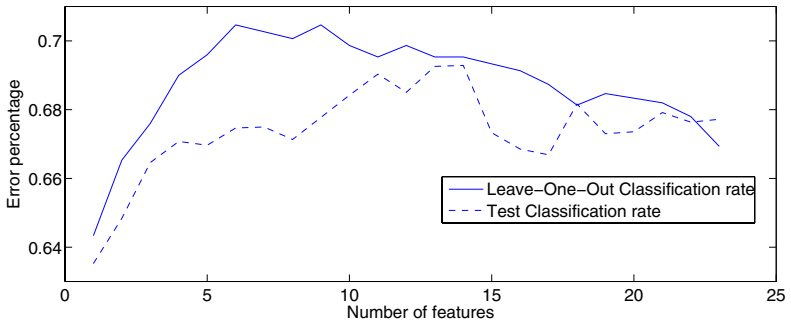

Fig. 3. The K-NN accuracy using the forward algorithm

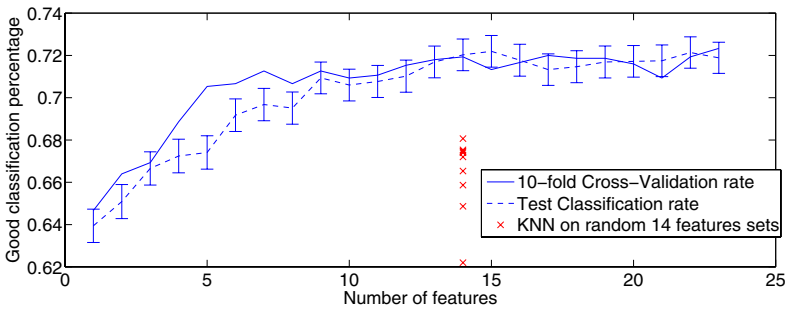

Fig. 4. The SVM accuracy using the result of the K-NN forward. The vertical segments show the noise estimation obtained using the bootstrap technique. Crosses present the results of K-NN on 10 sets of 14 features randomly selected.

5.2 Accuracy of SVM with Feature Selection

Since the 6 forward K-NN selected features are not enough, this process step uses all features, but according to the ranking order given by the forward K-NN. The SVM is thus used (RBF-type kernel), with the same training and test sets. As mentioned before, we use here a bootstrap technique to have a more robust result and an estimation of the noise. As it can be seen (cf Figure 4), the best accuracy is obtained using 14 features, achieving 72% of correct classification (10-fold cross-validation). In this case, the test error curve stays close to the 10-fold one. For comparison purposes we have also plotted the performance of the

K-NN on sets of 14 features taken randomly from the original ones. As illustrated on figure 3, it never achieves more than 68% in correct classification (training). This proves that the selected features using the forward technique are relevant enough.

5.3 Selected Features

Table 2 presents the set of features that have been selected. For sake of simplicity the cardinal part for each feature has been skipped. Table 3 presents the final results from the explained method. It can be seen that the selected 14 features set is giving better results (within the noise estimation) than with all 23 features. Note that even-though the result is always superior using only 14 features, the noise is still to take into account (Fig 4).

Table 2. List of the selected features done by the forward algorithm using K-NN. Feature are ordered according to the forward algorithm.

N_{11}	$\mathbf{g}^{-1}$	$\mathbf{g}^{-2}$	$\mathbf{g}^{-3}$	$\mathbf{g}^{1}$	$\mathbf{H}$	$\mathbf{g}^{4}$	$\mathbf{g}^{0}$	$\mathbf{h}^{21}$	$\mathbf{g}^{-4}$	N_{01}	$\mathbf{B}_2$	$\mathbf{h}^{13}$	$\mathbf{h}^{12}$

Table 3. The test error (in plain) and 10-fold cross-validation error (bracketed) for 14 and 23 features at different embedding rates

Embedding rate	14 features	23 features
10%	72.0% (71.9%)	71.9% (72.3%)
25%	88.0% (92.9%)	87.2% (93.1%)
50%	97.8% (99.3%)	97.0% (99.2%)
75%	99.2% (99.7%)	98.0% (99.8%)

5.4 Weaknesses of Outguess

Feature selection enables to link the nature of the selected features with Outguess v0.2, the steganographic software that has been used [10] and then to outline its weaknesses. We recall that Outguess embeds information by modifying the least significant bits of the quantised DCT coefficients of a JPEG coded image. In order to prevent easy detection, the algorithm does not embed information into coefficients equal to 0 and 1. Outguess also preserves the global histogram of the DCT coefficients between the original and stego image by correcting statistical deviations.

The selected features presented in Table 2 present strong links with the way the embedding scheme performs:

- The feature N_{11} is the first feature selected by the forward algorithm and describes the difference between co-occurrence values for coefficients equal to 1 or -1 on neighbouring blocks. This feature seems to react mainly to the flipping between coefficients -1 and -2 during the embedding. Note also that coefficients -2 and 2 are, after 0 and 1, the most probable DCT coefficients in a given image.

- The second and third selected features are $\mathbf{g}^{-1}$ and $\mathbf{g}^{-2}$. They represent the dual histogram of coefficients respectively equal to -1 and -2 with respect to their coordinates. Once again, these features concern the same coefficients than previously but only on the first order (histogram).

- We can notice that nearly half of features related to the dual histogram have been selected. Due to symmetry one might think that features $\mathbf{g}^{-5}$, $\mathbf{g}^{-4}$, $\mathbf{g}^{-3}$, $\mathbf{g}^{-2}$ carry respectively the same information than $\mathbf{g}^{5}$, $\mathbf{g}^{4}$, $\mathbf{g}^{3}$, $\mathbf{g}^{2}$, consequently it is not surprising that only one in each set has been chosen (with the exception of $\mathbf{g}^{-4}$ and $\mathbf{g}^{4}$).

- Note that it can seem first curious that features $\mathbf{g}^{0}$ and $\mathbf{g}^{1}$ have been selected as meaningful features for the classifier because they are not modified by the embedding algorithm. However, these features can have been affected on the stego and cropped image: coefficients equal to 2 or 3 on the stego image can be reduced to 1 or 2 on the cropped image. Another reason can be that feature $\mathbf{g}^{1}$ can be selected in association with feature $\mathbf{g}^{-1}$ because it has a different behaviour for watermarked images but a similar behaviour for original images.

5.5 Obtained Results for Other Steganographic Schemes

This feature selection method has also been tested for two other popular steganographic schemes called F5 and Steghide. Our test confirms that it is also possible to use K-NN-based feature selection on Steghide and to select 13 features which provide similar performances. The list of the 13 selected features is given on table 4 and the performances for different embedding rates is given on table 5. However, we have noticed that for the F5 algorithm performing feature selection is not efficient if the ratio of selected features is below 80%. Forward feature selection for F5 selects still 15 features and backward feature selection selects 22 features. The high number of selected features means that nearly each of the initial feature for F5 is significant for the detection process. Such a consideration is not surprising because F5 is the most undetectable of the three analysed steganographic schemes.

Table 4. List of the 13 selected features done by the forward algorithm using K-NN for Steghide. Features are ordered according to the forward algorithm.

N_{00}	$\mathbf{g}^{2}$	$\mathbf{h}^{22}$	$\mathbf{H}$	$\mathbf{g}^{5}$	N_{01}	$\mathbf{g}^{-2}$	$\mathbf{g}^{-1}$	$\mathbf{h}^{13}$	$\mathbf{g}^{-5}$	$\mathbf{g}^{1}$	$\mathbf{g}^{5}$	$\mathbf{V}$

Table 5. The test error (in plain) and 10-fold cross-validation error (bracketed) for 13 and 23 features at different embedding rates for Steghide algorithm

Embedding rate	13 features	23 features
10%	67.28% (69.39%)	68.73% (68.79%)
25%	75.21% (77.90%)	77.81% (81.03%)
50%	91.66% (90.77%)	93.25% (93.79%)
75%	97.84% (97.93%)	98.37% (98.88%)

6 Conclusions and Future Works

This paper proposes a methodology to select meaningful features for a given steganographic scheme. Such a selection enables both to increase the knowledge on the weakness of a steganographic algorithm and to reduce its complexity while keeping the classification performances. Our future works will consist in combining input selection techniques with feature scaling in order to increase the performance of the classifiers.

References

1. J.Fridrich. (In: 6th Information Hiding Workshop, LNCS, vol. 3200)
2. S.Dumitrescu, X.Wu, Z.Wang: Detection of LSB steganography via sample pair analysis. In: IEEE transactions on Signal Processing. (2003) 1995–2007
3. B.Roue, P.Bas, J-M.Chassery: Improving lsb steganalysis using marginal and joint probabilistic distributions. In: Multimedia and Security Workshop, Magdeburg (2004)
4. S.Lyu, H.Farid: Detecting hidden message using higher-order statistics and support vector machine. In: 5^{th} International Workshop on Information Hiding, Netherlands (2002)
5. T.Pevny, J.Fridrich: Toward multi-class blind steganalyser for jpeg images. In: International Workshop on Digital Watermarking, LNCS vol. 3710. (2005) 39–53
6. Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., Vapnik, V.: Feature selection for SVMs. In Leen, T.K., Dietterich, T.G., Tresp, V., eds.: NIPS, MIT Press (2000) 668–674
7. Efron, B., Tibshirani, R.: An Introduction to the Bootstrap. Chapman and Hall, London (1993)
8. Zhang, T.: An introduction to support vector machines and other kernel-based learning methods. AI Magazine (2001) 103–104
9. Rossi, F., Lendasse, A., François, D., Wertz, V., Verleysen, M.: Mutual information for the selection of relevant variables in spectrometric nonlinear modelling. Chemometrics and Intelligent Laboratory Systems, vol 80 (2006) 215–226
10. Provos, N.: Defending against statistical steganalysis. In USENIX, ed.: Proceedings of the Tenth USENIX Security Symposium, August 13–17, 2001, Washington, DC, USA, USENIX (2001)
11. Ronneberger, O.: Libsvmtl extensions to libsvm. http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/ (2004)

PUBLICATION D

TITLE:
Advantages of Using Feature Selection Techniques on Steganalysis Schemes

AUTHORS:
Yoan Miche, Patrick Bas, Amaury Lendasse, Christian Jutten and Olli Simula

PUBLISHED IN:
IWANN'07: International Work-Conference on Artificial Neural Networks, San Sebastian, Spain, June 2007, Francisco Sandoval et al. eds., Lecture Notes in Computer Science, Springer Berlin / Heidelberg.
Volume 4507/2007, pp. 606–613

DOI:
http://dx.doi.org/10.1007/978-3-540-73007-1_73

Advantages of Using Feature Selection Techniques on Steganalysis Schemes

Yoan Miche[1,2], Patrick Bas[1,2], Amaury Lendasse[1],
Christian Jutten[2], and Olli Simula[1]

[1] Helsinki University of Technology - Laboratory of Computer and Information Science
P.O. Box 5400, FI-02015 HUT, Finland
[2] INPG - Laboratoire des Images et des Signaux,
INPG, 46 avenue Félix Viallet, 38031 Grenoble cedex, France

Abstract. Steganalysis consists in classifying documents as steganographied or genuine. This paper presents a methodology for steganalysis based on a set of 193 features with two main goals: determine a sufficient number of images for effective training of a classifier in the obtained high-dimensional space, and use feature selection to select most relevant features for the desired classification. Dimensionality reduction is performed using a forward selection and reduces the original 193 features set by a factor of 13, with overall same performance.

1 Introduction

Steganography has been known and used for a very long time, as a way to exchange information in an unnoticeable manner between parties, by embedding it in another, apparently innocuous, document. For example, during the 80's, Margaret Thatcher decided to have each word processor of the government's administration members changed with an unique word spacing for each, giving a sort of "invisible signature" [7] to documents. This was done to prevent the continuation of sensitive government information leaks.

Nowadays steganographic techniques are also used on digital contents. The online newspaper, Wired News, reported in one of its articles [4] on steganography that several steganographic contents have been found on websites with very large image database such as eBay.

Most of the time research about steganography is not as much to hide information, but more to detect that there is hidden information. This "reverse" part of the steganography is called steganalysis and is specifically aimed at making the difference between genuine documents, and steganographied – called stego – ones. Consequently, steganalysis can be seen as a classification problem where the goal is to build a classifier able to distinguish these two sorts of documents.

During the steganographic process, a message is embedded in an image so that it is as undetectable as possible. Basically, it uses several heuristics in order to guarantee that the statistics of the stego content are as close as possible to the statistics of the original one. Afterwards, steganalysis techniques classically

F. Sandoval et al. (Eds.): IWANN 2007, LNCS 4507, pp. 606–613, 2007.

use features extracted from the analysed image and an appropriately trained classifier to decide whether the image is genuine or not.

In this paper the Outguess algorithm proposed by Niels Provos in [9] is analysed. This algorithm, according to its author, is supposed to resist especially well to statistical attacks by work on Least Significant Bits of quantized DCT coefficients of JPEG compressed images. In practice, it is often used as a reference steganographic algorithm for performance comparison, eventhough it is highly detectable, as shown for example in [6,14].

In this work, the 193 image features proposed by Pevni and Fridrich in [12] have been used. Theses features consider statistics of JPEG compressed images such as histograms of DCT coefficients for different frequencies, histograms of DCT coefficients for different values, global histograms, blockiness measures and co-occurrence measures. They are an extension of an original set of 23 features [6].

Fridrich proposes afterwards to train a classifier according to the extracted features. Consequently a set of 193 features for each image of the database is obtained, giving an especially high dimensionality space for classifiers to work on. Earlier research about these high dimensionality spaces has shown that a lot of issues come out when the number of features is as high as the one we use.

2 Drawbacks of Performing Steganalysis in High-Dimensional Spaces with a Constrained Data-Set

The common term "curse of dimensionality" [2] refers to a wide range of problems related to a high number of features. Some details are given below about three inherent issues that occur in the framework of steganalysis, namely the need for data points (images), the increase of complexity and the lack of interpretability.

The need for data points: In the general case, in order for any tool to be able to analyze and find an underlying structure within the data, the number of needed points is growing exponentially with the dimension. Indeed, consider a d-dimensional unit side hypercube, the number of points needed to fill the Cartesian grid of step ϵ inside of it, is growing as $O((1/\epsilon)^d)$. Thus, using a common grid of step $1/10$ and a dimension of 10, it requires 10^{10} points to fill the grid. In practice, steganalysis work makes often use of at least 10 to 20 dimensions, implying a "needed" number of images impossible to achieve. As a consequence, the feature space may be not correctly filled with data points, which can give wrong models when building classifiers, having to extrapolate for the missing images: thus, an estimation of the required minimum number of images has to be obtained, in order to have a reliable training of the selected model.

Increase of complexity: Computational time is another main reason. Nearest neighbours methods are usually implemented with a $O(d)$ dimension relationship, as for SVMs. Clearly, reducing the dimensionality by a significant order of magnitude gives much more achievable computational times. As a consequence, one can use more images and lower this "missing images" effect. Meanwhile, the

finally chosen number of images should still remain within a reasonable range defined by these computational times. The best compromise between the number of images and the future reliability of our model has to be found.

Lack of interpretability: Eventhough the nearest neighbours classifiers keep good performance in high dimensions [3,8], other obvious problems of high dimensionality motivate the idea of feature selection. The interpretability is an important one: high performance can indeed be reached using the whole 193 features set for classification. Meanwhile, if looking for the weaknesses and reasons why these features react vividly to a specific algorithm, it seems rather impossible on this important set. Reducing the required number of features to a small amount through feature selection enables to understand better why a steganographic model is weak on these particular details, highlighted by the selected features. Indeed, steganalysis can tend to make the whole process aimed only at performance without possible interpretations, while having knowledge about the selected features gives interesting insights on the steganographic algorithm.

3 Methodology and Techniques Used

3.1 Classifiers and Appropriate Number of Images

The "appropriate" number of images (or at least the minimum required for the dimensionality of our data) should be determined. For this matter, a KNN classifier is used with a Monte-Carlo technique [11]. This enables to estimate the noise and give a confidence interval for our results. We randomly draw (without repetitions) a subset of the whole data set, and use the obtained classifier on it.

For our experiments, two different types of classifiers have mainly been used: the first one, KNN, for its overall good performance even in high dimensional spaces, but most of all, because it is computationally very fast. SVM was also chosen because it is among the classifiers giving the best results. Major drawback is of course the computational time. KNN classifier is a supervised distance-based classifier, proposed by Devijver and Kittler in [1], usually using the euclidean metric. It is based on a majority vote among the k nearest neighbours classes to assign the class of the new considered point. The SVM has been created by Vapnik [10] in 1963 and then improved more recently (1992,1995) by Boser, Guyon and Vapnik [5]. The original idea was to separate data using a hyperplane: this was a linear classifier. The extension of this method adds a non-linear part by the use of kernel functions.

3.2 Feature Selection Technique: Forward Selection

The forward selection algorithm is a greedy algorithm proposed in [13]; the algorithm selects one by one the dimensions, trying to find the one that combines best with the already selected ones. Even if its capacity to isolate efficient features is obvious, the forward technique has some drawbacks: in the case where two features would have a high dependency and be "unefficient" when alone but

very good when put together, forward might not take these into account soon enough in the selection process. Nevertheless, the feature selection using forward has been showing very good results and seems to perform well on our feature set; this is presented in the next section.

3.3 Our Methodology

The three main points of the proposed methodology are detailed in the following. Fig. 1 illustrates the process. First is seeked a possibly good candidate for the

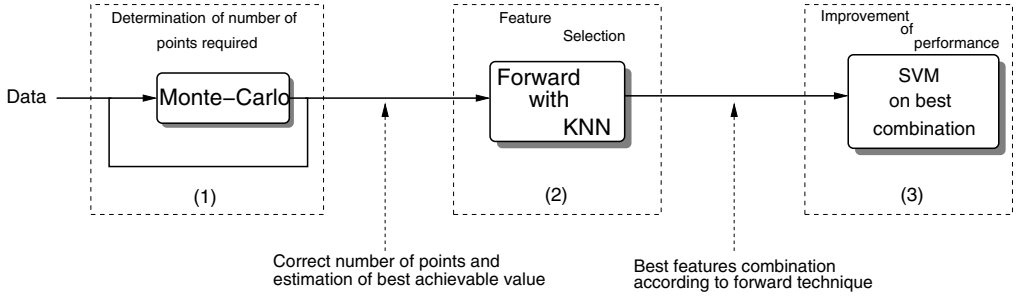

Fig. 1. Schematic view of the proposed methodology: (1) An appropriate number of data points to work with is determined using a Monte-Carlo method for statistical stability; (2) The forward selection is performed using a KNN classifier; (3) A good feature set is selected and performance is improved using SVM.

number of images to use for training with the prepared database. Using a Monte-Carlo method on low numbers of images with both SVM and KNN, averaged plots are obtained. From it, a correct idea of a sufficient number of images for the later study can be obtained, as shown in the following experiments.

Since KNN is the fastest classifier between the two presented, it is used for the next step with forward technique. This produces a ranking of the features showing how much each new feature contributes to the correct classification rate. The best features combination is selected. A SVM is finally used on this combination, to improve the performance and obtain the final best classification rate achieved.

4 Experiments, Results and Analysis

Our image base was constituted of 13 000 images of natural scenes, coming from 5 different digital cameras. Images are then all reduced to a size of 800×600 (multiples of 8) to avoid some possible block effects and artifacts due to JPEG recompression on another grid. At the same time, they are changed from their original colorspace to grayscale colorspace (256 gray levels).

A cropping operation to 512×512 follows, since our implementation of the extractor of Fridrich's 193 features works on 512×512 image blocks (powers of 2). In the end, the whole set of images is separated into two equal parts: one is

kept genuine while the other one is steganographied with the Outguess algorithm at an embedding rate of 25%.

This choice of half steganographied and half genuine can be discussed as it does not reflects a real world situation. Meanwhile, the whole steganalysis process presented is designed to be used on one image at a time, determining whether it is genuine or not. Furthermore, this choice has been done to be able to compare performances with the steganalysis community current research.

For classification and test purposes, the training set has been made with at most 8000 images. Test set is composed of the remaining, that is 5000 images. The 193 features proposed by Fridrich are used as in [12].

4.1 Determination of Sufficient Number of Images

Presented first is the result of the evaluation of a sufficient number of images, as explained in the methodology. The Monte-Carlo is used on randomly taken subsets of 200 up to 2000 images with 10 iterations. Each model built – using KNN and SVM – is also evaluated on the test set of 5000 images.

A single point is evaluated with a randomly chosen set of 4000 images, since computational time becomes very important with such number of images. In practice, on Fig. 2 presenting the results of this study, the two cross-validation results (SVM and KNN) should not be strictly compared since they do not use the same number of images to validate the model: SVM uses a 10-fold cross-validation, while a Leave-One-Out (LOO) method is used for KNN. Test results are, on the other hand, comparable.

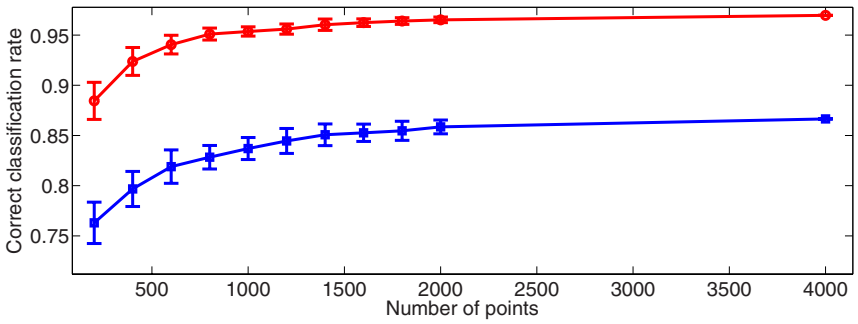

Fig. 2. Correct classification rate for SVM (circles, top curve) and KNN (squares, bottom curve) with associated variance

One can really see on these plots that an apparently sufficient number of images is over 2000, since the classification rate seems to increase exponentially slowly over this value. For the experiments, a bigger set of 4000 images has been chosen.

4.2 Forward Selection and Optimisation by SVM

Here, the results of the forward selection are presented shortly. As can be seen from Fig. 3, the whole process of forward selection is not fully achieved – for

computational time reasons – since we do not go over 21 features. Meanwhile, as will be more detailled in the analysis part of these results, good performance is already performed before this value. Since the goal is to have the smallest possible feature set, while keeping average same performances, the forward selection could be stopped at this point.

Test and 10-fold cross-validation remain in a much thinner interval than for our only-KNN tryouts. Moreover, the performance gain with SVM is significative as expected and reaches up to 2% in the frame of these plots.

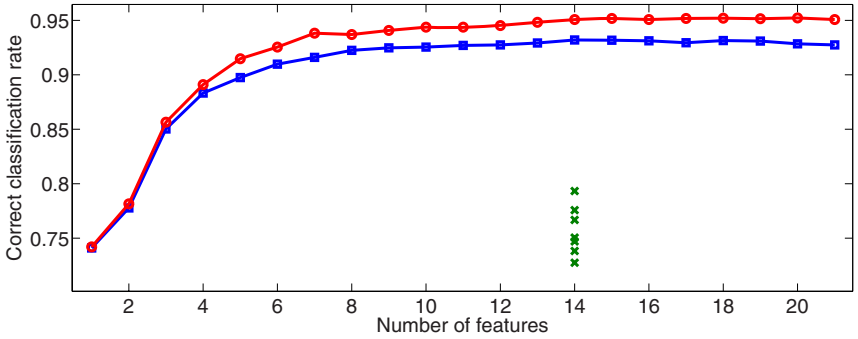

Fig. 3. Plot of the correct classification rate for SVM: 10-fold cross-validation (circles, top curve) and test (squares, bottom curve). Crosses are for performance for random sets of 14 features with a KNN classifier.

Table 1 presents the main values obtained using 193 features set. Our feature selection gives interesting results on this set. Indeed, using as few as 14 features, we are less than 1.9% behind the value obtained with all features for 10-fold cross-validation. Test values are following the exact same pattern.

Table 1. Results of the different classifiers for cross-validation and test

	LOO / 10-fold	Test	Comp. time
KNN 193	86.65%	85.89%	4.5min
KNN 193→14	93.20%	89.02%	60s
SVM 193	96.92%	96.76%	49h
SVM 193→14	95.08%	94.86%	4.5h

4.3 Analysis

From machine learning point of view, a major achievement was obtained: reducing the dimensionality by more than 13 and keeping roughly the same performance, in the variance interval. This result is interesting for different reasons: First, the computational time is drastically reduced, since the classifiers complexity relationships to dimensionality are linear. Second, because computational time is decreased by 11, it allows future new analyses and experiments previously not possible.

From steganalysis point of view, the obtained results are of course behind the actual best values, obtained for the Outguess algorithm in [12]. Nevertheless, the two advantages coming out of these results – namely the decrease of computational time and the gain in interpretability – can counterbalance this opinion. In the end, this set of features describes in a more precise way the functionning and problems of the Outguess algorithm. Taking these into account might help improve the steganographic scheme and make it less detectable: the first (and thus most "efficient") features selected by the forward algorithm show that the Outguess algorithm is especially weak when the analysis is made on features using -1 and -2 DCT coefficients, leading to already more than 90% of correct classification with the SVM classifier.

5 Conclusions and Future Work

This paper has presented a new methodology for dimensionality reduction by feature selection in the framework of steganalysis.

The issues of dimensionality have been adressed and the first step of our methodology proves that the theoretically required number of images for correct training is far from being needed. By the use of a Monte-Carlo technique on up to 4000 images, it has been shown that such numbers of images are sufficient for stable results. A set of 193 features extracted from all images serves the classification process, preceded by the dimensionality reduction step. This part of our methodology is achieved using a forward selection with a KNN classifier. It enables to reduce the number of required features to 14, while keeping roughly the same classification results. Computational time is thus greatly improved, divided by about 11. Further analysis becomes again possible with this low number of features: conclusions and precisions about the steganographic scheme can be infered from the obtained feature set. The last step using SVM for improvement over the previous KNN results achieves high classification results for so small a feature set, proving that many features among the full 193 set might not be relevant enough to be kept for classification purposes.

A comparison between the obtained reduced sets of features for various steganographic algorithms might reveal some common sensitive features. An analysis of these common points could help design a more generic steganalysis method using a "low" number of features.

Acknowledgements

Part of the work of Yoan Miche, Patrick Bas and Christian Jutten is supported by the French national funding under the project RIAM Estivale (ANR-05-RIAM-O1903). Part of the work of Yoan Miche, Olli Simula and Amaury Lendasse is supported by the project of New Information Processing Principles, 44886, of the Academy of Finland.

References

1. Devijver, P.A., Kittler, J.: Pattern recognition: a statistical approach. Prentice Hall, New York (1982)
2. Bellman, R.: Adaptive control processes: a guided tour. Princeton University Press, Princeton (1961)
3. François, D.: High-dimensional data analysis: optimal metrics and feature selection. PhD thesis, Université catholique de Louvain (September 2006)
4. McCullagh, D.: Secret messages come in .wavs. Online Newspaper: Wired News(February 2001) http://www.wired.com/news/politics/0,1283,41861,00.html
5. Boser, B. E., Guyon, I. M., Vapnik, V. N.: A training algorithm for optimal margin classifiers. In: Fifth Annual Workshop on Computational Learning Theory, pp. 144–152 (27-29 Juillet 1992)
6. Fridrich, J.: Feature-based steganalysis for jpeg images and its implications for future design of steganographic schemes. In: Information Hiding: 6th International Workshop, May 23-25, LNCS, vol. 3200, pp. 67–81. Springer, Heidelberg (2004)
7. Maxemchuck, J.M.: Electronic document distribution. AT and T Technical Journal 73(5), 73–80 (1994)
8. Verleysen, M., François, D.: The curse of dimensionality in data mining and time series prediction. In: IWANN'05 : 8th International Work-Conference on Artificial Neural Network, Lecture Notes in Computer Science, vol. 3512, pp. 758–770 (8-10 Juin, 2005)
9. Provos, N.: Defending against statistical steganalysis. In: 10th USENIX Security Symposium, pp. 323–335 (April 13-17, 2001)
10. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience, New York (1998)
11. Christian, P.R., Casella, G.: Monte Carlo statistical methods. Springer, Heidelberg (1999) ISBN:038798707X.
12. Pevny, T., Fridrich, J.: Merging markov and dct features for multi-class jpeg steganalysis. In: IS and T/SPIE EI 2007, Lecture Notes in Computer Science, vol. 6505, January 29th - February 1st (2007)
13. Whitney, A.W.: A direct method of nonparametric measurement selection. IEEE Transactions on Computers C-20, 1100–1103 (1971)
14. Miche, Y., Roue, B., Lendasse, A., Bas, P.: A feature selection methodology for steganalysis. In: Gunsel, B., Jain, A.K., Tekalp, A.M., Sankur, B. (eds.) MRCS 2006. LNCS, vol. 4105, pp. 49–56. Springer, Heidelberg (2006)

PUBLICATION E

TITLE:
Reliable Steganalysis Using a Minimum Set of Samples and Features

AUTHORS:
Yoan Miche, Patrick Bas, Amaury Lendasse, Christian Jutten and Olli Simula

PUBLISHED IN:
EURASIP Journal on Information Security, March 2009. Hindawi Publishing Corporation.
Volume 2009, Article ID 901381, pp. 1–13

DOI:
http://dx.doi.org/10.1155/2009/901381

Hindawi Publishing Corporation
EURASIP Journal on Information Security
Volume 2009, Article ID 901381, 13 pages
doi:10.1155/2009/901381

Research Article

Reliable Steganalysis Using a Minimum Set of Samples and Features

Yoan Miche,[1,2] Patrick Bas,[2] Amaury Lendasse,[1] Christian Jutten (EURASIP Member),[2] and Olli Simula[1]

[1] *Laboratory of Information and Computer Science, Helsinki University of Technology, P.O. Box 5400, FI-02015 HUT, Finland*
[2] *GIPSA-Lab, 961 rue de la Houille Blanche, BP 46, F-38402 Grenoble Cedex, France*

Correspondence should be addressed to Yoan Miche, ymiche@cc.hut.fi

Received 1 August 2008; Revised 14 November 2008; Accepted 13 March 2009

Recommended by Miroslav Goljan

This paper proposes to determine a sufficient number of images for reliable classification and to use feature selection to select most relevant features for achieving reliable steganalysis. First dimensionality issues in the context of classification are outlined, and the impact of the different parameters of a steganalysis scheme (the number of samples, the number of features, the steganography method, and the embedding rate) is studied. On one hand, it is shown that, using Bootstrap simulations, the standard deviation of the classification results can be very important if too small training sets are used; moreover a minimum of 5000 images is needed in order to perform reliable steganalysis. On the other hand, we show how the feature selection process using the OP-ELM classifier enables both to reduce the dimensionality of the data and to highlight weaknesses and advantages of the six most popular steganographic algorithms.

1. Introduction

Steganography has been known and used for a very long time, as a way to exchange information in an unnoticeable manner between parties, by embedding it in another, apparently innocuous, document.

Nowadays steganographic techniques are mostly used on digital content. The online newspaper Wired News reported in one of its articles [1] on steganography that several steganographic contents have been found on web sites with very large image database such as eBay. Provos and Honeyman [2] have somewhat refuted these facts by analyzing and classifying two million images from eBay and one million from USENet network and not finding any steganographic content embedded in these images. This could be due to many reasons, such as very low payloads, making the steganographic images less detectable to steganalysis and hence more secure.

In practice the concept of security for steganography is difficult to define, but Cachin in [3] mentions a theoretic way to do so, based on the Kullback-Leibler divergence. A stego process is thus defined as ϵ-secure if the Kullback-Leibler divergence δ between the probability density functions of the cover document p_{cover} and those of this very same content embedding a message p_{stego} (i.e., stego) is less than ϵ:

$$\delta\left(p_{\text{cover}}, p_{\text{stego}}\right) \leq \epsilon. \tag{1}$$

The process is called *secure* if $\epsilon = 0$, and in this case the steganography is perfect, creating no statistical differences by the embedding of the message. Steganalysis would then be impossible.

Fortunately, such high performance for a steganographic algorithm is hardly achievable when the payload (the embedded information) is of nonnegligible size; also, several schemes have weaknesses.

One way of measuring the payload is the *embedding rate*, defined as follows.

Let A be a steganographic algorithm, and let C be a cover medium. A, by its design, claims that it can embed at most T_{Max} information bits within C; T_{Max} is called the *capacity* of the medium and highly depends on the steganographic (stego) algorithm as well as the cover medium itself. The

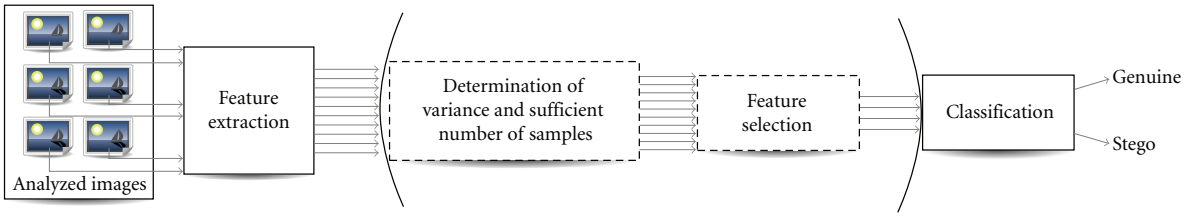

Figure 1: Overview of the typical global processing for an analyzed image: features are first extracted from the image and then processed through a classifier to decide whether the image is cover or stego. In the proposed processing is added an extra step aimed at reducing the features number and having an additional interpretability of the steganalysis results, by doing a feature selection.

embedding rate T is then defined as the part of T_{Max} used by the information to embed.

For T_i bits to embed in the cover medium, the embedding rate is then $T = T_i/T_{\text{Max}}$, usually expressed as percentage. There are other ways to measure the payload and the relationship between the amount of information embedded and the cover medium, such as the number of *bits per nonzero coefficient*. Meanwhile, the embedding rate has the advantage of taking into account the stego algorithm properties and is not directly based on the cover medium properties—since it uses the stego algorithm estimation of the maximum capacity. Hence the embedding rate has been chosen for this analysis of stego schemes.

This paper is focused onto feature-based steganalysis. Such steganalysis typically uses a certain amount of images for training a classifier: features are extracted from the images and fed to a binary classifier (usually Support Vector Machines) for training. The output of this classifier is "stego" (modified using a steganographic algorithm) or "cover" (genuine). This process is illustrated on Figure 1 for the part without parenthesis.

The emphasis in this paper is more specifically on the issues related to the increasing number of features, which are linked to the universal steganalyzers. Indeed, the very first examples of LSB-based steganalysis made use of less than ten features, with an adapted and specific methodology for each stego algorithm. The idea of "universal steganalyzers" then became popular. In 1999, Westfeld proposes a χ^2-based method, on the LSB of DCT coefficients [4]. Five years after, Fridrich in [5] uses a set of 23 features obtained by normalizations of a much larger set, whilst Lyu and Farid already proposed in 2002 a set of 72 features [6]. Some feature sets [7] also have variable size depending on the DCT block sizes. Since then, an increasing number of research works use supervised learning-based classifiers in very high-dimensional spaces. The recent work of Shi et al. [8] is an example of an efficient result by using 324 features based on JPEG blocks differences modeled by Markov processes.

These new feature sets usually do achieve better and better performance in terms of detection rate and enable to detect most stego algorithm for most embedding rates. Meanwhile, there are some side-effects to this growing number of features. It has been shown, for example, in [9] that the feature space dimensionality in which the considered classifier is trained can have a significant impact on its performances: a too small amount of images regarding dimensionality (the number of features) might lead to an improper training of the classifier and thus to results with a possibly high statistical variance.

In this paper is addressed the idea of a practical way of comparing steganalysis schemes in terms of performance reliability. Ker proposed [10] such comparison by focusing on the pdf of one output of the classifier. Here are studied multiple parameters that can influence this performance:

(1) the number of images used during the training of the classifier: how to determine a sufficient number of images for an efficient and reliable classification (meaning that final results have acceptable variance)?

(2) the number of features used: what are the sufficient and most relevant features for the actual classification problem?

(3) the steganographic method: is there an important influence of the stego algorithm on the general methodology?

(4) the embedding rate used: does the embedding rate used for the steganography modify the variance of the results and the retained best features (by feature selection)?

It can also be noted that images of higher sizes would lead to a smaller secure steganographic embedding rate (following a root-square law), but this phenomenon has already been studied by Filler et al. [11].

The next section details some of the problems related to the number of features used (dimensionality issues) and commonly encountered in steganalysis: (1) the empty space and the distance concentration phenomena, (2) the large variance of the results obtained by the classifier whenever the number of images used for training is not sufficient regarding the number of features, and finally, (3) the lack of interpretability of the results because of the high number of features. In order to address these issues, the methodology sketched on Figure 1 is used and more thoroughly detailed: a sufficient number of images regarding the number of features is first established so that the classifier's training is "reliable" in terms of variance of its results; then, using feature selection the interpretability of the results is improved.

The methodology is finally tested in Section 4 with six different stego algorithms, each using four different embedding rates. Results are finally interpreted thanks to the most relevant selected features for each stego algorithm.

A quantitative study of selected features combinations is then provided.

2. Dimensionality Issues and Methodology

The common term "curse of dimensionality" [12] refers to a wide range of problems related to a high number of features. Some of these dimensionality problems are considered in the following, in relation with the number of images and features.

2.1. Issues Related to the Number of Images

2.1.1. The Need for Data Samples. In order to illustrate this problem in a low-dimensional case, one can consider four samples in a two-dimensional space (corresponding to four images out of which two features have been extracted); the underlying structure leading to the distribution of these four samples seems impossible to infer, and so is the creation of a model for it. Any model claiming it can properly explain the distribution of these samples will behave erratically (because it will extrapolate) when a new sample is introduced. On the contrary, with hundreds to thousands of samples it becomes possible to see clusters and relationships between dimensions.

More generally, in order for any tool to be able to analyze and find a structure within the data, the number of needed samples is growing exponentially with the dimensionality. Indeed, consider a d-dimensional unit side hypercube; the number of samples needed to fill the Cartesian grid of step ϵ inside of it is growing as $O((1/\epsilon)^d)$. Thus using a common grid of step 1/10 in dimension 10, it requires 10^{10} samples to fill the grid.

Fortunately, for a model to be built over some high-dimensional data, that data does not have to fill the whole space in the sense of the Cartesian grid. The required space to fill highly depends on the density to be estimated.

In practice, most data sets in steganalysis use at least 10 to 20 dimensions, implying a "needed" number of samples impossible to achieve: storing and processing such number of images is currently impossible. As a consequence, the feature space is not filled with enough data samples to estimate the density with reliable accuracy, which can give wrong or high variance models while building classifiers, having to extrapolate for the missing samples: obtained results can have rather high confidence interval and hence be statistically irrelevant. A claim of performance improvement of 2% using a specific classifier/steganalyzer/steganographic scheme with a variance of 2% is rather meaningless.

2.1.2. The Increasing Variance of the Results. The construction of a proper and reliable model for steganalysis is also related to the variance of the results it obtains. Only experimental results are provided to support this claim: with a low number of images regarding the number of features (e.g., a few hundreds of images for 200 features), the variance of the classifier's results can be very important (i.e., the variance of the detection probability).

When the number of images increases, this variance decreases toward low enough values for feature-based steganalysis and performances comparisons. These claims are verified in the next section with the experiments.

2.1.3. Proposed Solution to the Lack of Images. Overall, these two problems lead to the same conclusion: the number of images has to be important regarding dimensionality. Theory states that this number is exponential with the number of features, which is impossible to reach for feature-based steganalysis. Hence, the first step of the proposed methodology is to find a "sufficient" number of images for the number of features used, according to a criterion on the variance of the results.

A Bootstrap [13] is proposed for that task: the number of images used for the training of the classifier is increased, and for each different number of images, the variance of the results of the classifier is assessed. Once the variance of the classifier is below a certain threshold, a sufficient number of images have been found (regarding the classifier and the feature set used).

2.2. Issues Related to the Number of Features

2.2.1. The Empty Space Phenomenon. This phenomenon that was first introduced by Scott and Thompson [14] can be explained with the following example: draw samples from a normal distribution (zero mean and unit variance) in dimension d, and consider the probability to have a sample at distance r from the mean of the distribution (zero). It is given by the probability density function:

$$f(r,d) = \frac{r^{d-1}}{2^{d/2-1}} \cdot \frac{e^{-r^2/2}}{\Gamma(d/2)} \tag{2}$$

having its maximum at $r = \sqrt{d-1}$. Thus, when dimension increases, samples are getting farther from the mean of the distribution. A direct consequence of this is that, for the previously mentioned hypercube in dimension d, the "center" of it will tend to be empty, since samples are getting concentrated in the borders and corners of the cube.

Therefore, whatever model is used in such a feature space will be trained on scattered samples which are not filling the feature space at all. The model will then not be proper for any sample falling in an area of the space where the classifier had no information about during the training. It will have to extrapolate its behavior for these empty areas and will have unstable performances.

2.2.2. Lack of Interpretability for Possible "Reverse Engineering". The interpretability (and its applications) is an important motivation for feature selection and dimensionality reduction: high performances can indeed be reached using the whole 193 features set used in this paper for classification. Meanwhile, if we are looking for the weaknesses and reasons why these features react vividly to a specific algorithm, it seems rather impossible on this important set.

Reducing the required number of features to a small amount through feature selection enables to understand

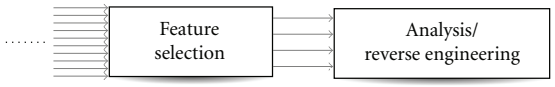

Figure 2: Scheme of the possible reverse engineering on an unknown stego algorithm, by using feature selection for identification of the specific weaknesses.

better why a steganographic model is weak on these particular details, highlighted by the selected features. Such analysis is performed in Section 4.3 for all six steganographic algorithms.

Through the analysis of these selected features, one can consider a "reverse engineering" of the stego algorithm as illustrated on Figure 2. By the identification of the most relevant features, the main characteristics of the embedding method can be inferred, and the steganographic algorithm can be identified if known, or simply understood.

2.2.3. Proposed Solution to the High Number of Features. These two issues motivate the feature selection process: if one can reduce the number of features (and hence the dimensionality), the empty space phenomena will have a reduced impact on the classifier used. Also, the set of features obtained by the feature selection process will give insights on the stego scheme and its possible weaknesses.

For this matter, a classical feature selection technique has been used as the second step of the proposed methodology.

The following methodology is different from the one presented previously in [15, 16]. Indeed, in this article, the goal is set toward statistically reliable results. Also, feature selection has the advantage of reducing the dimensionality of the data (the number of features), making the classifier's training much easier. The interpretation of the selected features is also an important advantage (compared to having only the classifier's performance) in that it gives insights on the weaknesses of the stego algorithm.

3. Methodology for Benchmarking of Steganographic Schemes

Addressed Problems. The number of data points to be used for building a model and classification is clearly an issue, and in the practical case, how many points are needed in order to obtain accurate results—meaning results with small standard deviation.

Reduction of complexity is another main addressed concern in this framework. Then for the selected number of points to be used for classification and also the initial dimensionality given by the features set, two main steps remain.

(i) Choosing the feature selection technique. Since analysis and computation can hardly be done on the whole set of features, the technique used to reduce the dimensionality has to be selected.

(ii) Building a classifier. This implies choosing it, selecting its parameters, training, and validating the chosen model.

The following paragraphs presents the solutions for these two major issues, leading to a methodology combining them, presented on Figure 3.

3.1. Presentation of the Classifier Used: OP-ELM. The Optimally-Pruned Extreme Learning Machine (OP-ELM [17, 18]) is a classifier based on the original Extreme Learning Machine (ELM) of Huang et al. [19] (available at: http://www.cis.hut.fi/projects/tsp/index.php?page=OPELM). This classifier makes use of single hidden layer feedforward neural networks (SLFNs) for which the weights and biases are randomly initialized. The goal of the ELM is to reduce the length of the learning process for the neural network, usually very long (e.g., if using classical back-propagation algorithms). The two main theorems on which ELM is based will not be discussed here but can be found in [19]. Figure 4 illustrates the typical structure of an SLFN (simplified to a few neurons in here).

Supposing the neural network is approximating the output $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$ perfectly, we would have

$$\sum_{i=1}^{M} \boldsymbol{\beta}_i f\left(\mathbf{w}_i \mathbf{x}_j + b_i\right) = \mathbf{y}_j, \quad j \in [\![1, N]\!], \tag{3}$$

where N is the number of inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ (number of images in our case), and M is the number of neurons in the hidden layer. In the case of steganalysis as performed in this article, $\mathbf{x}_i$ denotes the feature vector corresponding to image i, while $\mathbf{y}_i$ is the corresponding class of the image (i.e., stego or cover).

As said, the novelty introduced by the ELM is to initialize the weights $\mathbf{W}$ and biases $\mathbf{B}$ randomly. OP-ELM, in comparison to ELM, brings a greater robustness to data with possibly dependent/correlated features. Also, the use of other functions f (activation functions of the neural network) makes it possible to use OP-ELM for the case where linear components have an important contribution in the classifier's model, for example.

The validation step of this classifier is performed using classical Leave-One-Out cross-validation, much more precise than a k-fold cross-validation and hence not requiring any test step [13]. It has been shown on many experiments [17, 18] that the OP-ELM classifier has results very close to the ones of a Support Vector Machine (SVM) while having computational times much smaller (usually from 10 to 100 times).

3.2. Determination of a Sufficient Number of Images. A proper number of images, regarding the number of features, has to be determined. Since theoretical values for that number are not reachable, a sufficient number regarding a low enough value of the variance of the results is taken instead (standard deviation will be used instead of variance, in the following).

The OP-ELM classifier is hence used along with a Bootstrap algorithm [13] over 100 repetitions; a subset of the

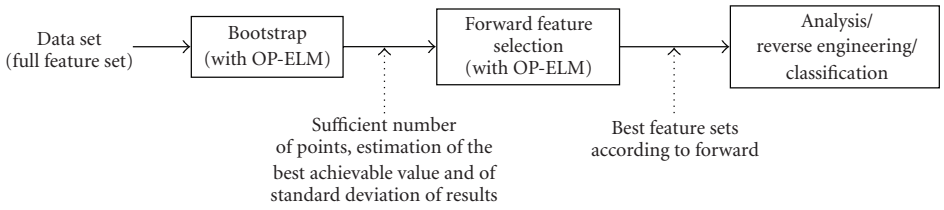

Figure 3: Schematic view of the proposed methodology. (1) An appropriate number of data samples to work with are determined using a Bootstrap method for statistical stability. (2) The Forward selection is performed using an OP-ELM classifier to find a good features set, from which follows a possible interpretation of the features or the typical classification for steganalysis.

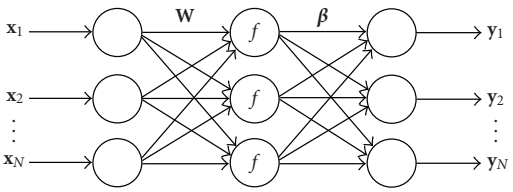

Figure 4: Structure of a classical Single Layer Feedforward Neural Network (SLFN). The input values (the data) $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ are weighted by the $\mathbf{W}$ coefficients. A possible bias $\mathbf{B}$ (not on the figure) can be added to the weighted inputs $\mathbf{w}_i\mathbf{x}_i$. An activation function f taking this weighted inputs (plus bias) as input is finally weighted by output coefficients $\boldsymbol{\beta}$ to obtain the output $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$.

R = $\{x^i, i \in [\![1, d]\!]\}$
S = $\emptyset$
while R $\neq \emptyset$ **do**
  for $x^j \in$ **R do**
    Evaluate performance with **S** $\cup\, x^j$
  end for
  Set **S** = **S** $\cup\, \{x^k\}$, **R** = **R** $- x^k$ with x^k the dimension giving the best result in the loop
end while

Algorithm 1: Forward.

complete data set (10000 images, 193 features) is randomly drawn (with possible repetitions). The classifier is trained with that specific subset. This process is repeated 100 times (100 random drawings of the subset) to obtain a statistically reliable estimation of the standard deviation of the results. The size of the subset drawn from the complete data set is then increased, and the 100 iterations are repeated for this new subset size.

The criterion to stop this process is a threshold on the value of the standard deviation of the results. Once the standard deviation of the results gets lower than 1%, it is decided that the subset size S, is sufficient. S is then used for the rest of the experiments as a sufficient number of images regarding the number of features in the feature set.

3.3. Dimensionality Reduction: Feature Selection by Forward with OP-ELM. Given the sufficient number of images for reliable training of the classifier, S, feature selection can be performed. The second step of the methodology, a Forward algorithm with OP-ELM (Figure 3), is used.

3.3.1. The Forward Algorithm. The forward selection algorithm is a greedy algorithm [20]; it selects one by one the dimensions, trying to find the one that combines best with the already selected ones. The algorithm is detailed in Algorithm 1 (with x^i denoting the ith dimension of the data set).

Algorithm 1 requires $d(d-1)/2$ instances to terminate (to be compared to the $2^d - 1$ instances for an exhaustive search), which might reach the computational limits, depending on the number of dimensions and time to evaluate the performance with one set. With the OP-ELM as a classifier, computational times for the Forward selection are not much of an issue.

Even if its capacity to isolate efficient features is clear, the Forward technique has some drawbacks. First, if two features present good results when they are put together but poor results if only one of them is selected, Forward might not take these into account in the selection process.

Table 1: Performances for OP-ELM LOO for the best features set along with the size of the reduced feature set (number). Performances using the reduced set are within the 1% range of standard deviation of the best results. The size of the set has been determined to be the smallest possible one giving this performance.

	5%	Number	10%	Number
F5	73.3	46	83.9	38
JPHS	90.7	41	92.1	21
MBSteg	63.3	57	70.9	93
MM3	78.00	81	86.2	49
OutGuess	81.2	65	93.2	49
Steghide	82.3	149	91.2	89
	15%	Number	20%	Number
F5	90.5	33	96.3	15
JPHS	93.7	41	97.3	25
MBSteg	83.5	73	88.5	69
MM3	86.6	57	86.6	73
OutGuess	98.8	33	100.0	29
Steghide	96.4	73	99	73

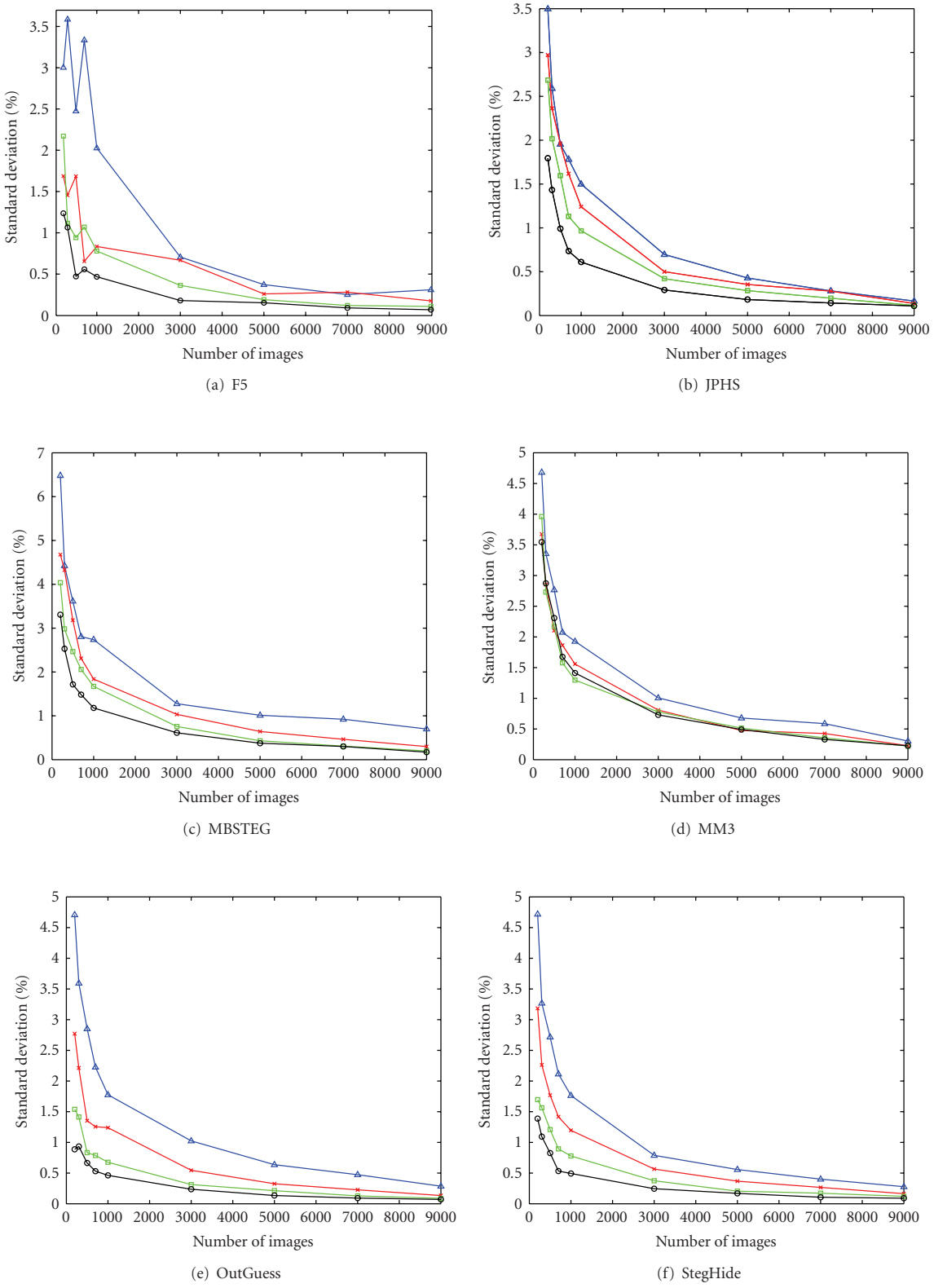

Figure 5: Standard deviation in percentage of the average classification result versus the number of images, for all six steganographic algorithms, for the four embedding rates: black circles (○) for 20%, green squares (□) for 15%, red crosses (×) for 10%, and blue triangles (△) for 5%. These estimations have been performed with the Bootstrap runs (100 iterations). Plots do not have the same scale, vertically.

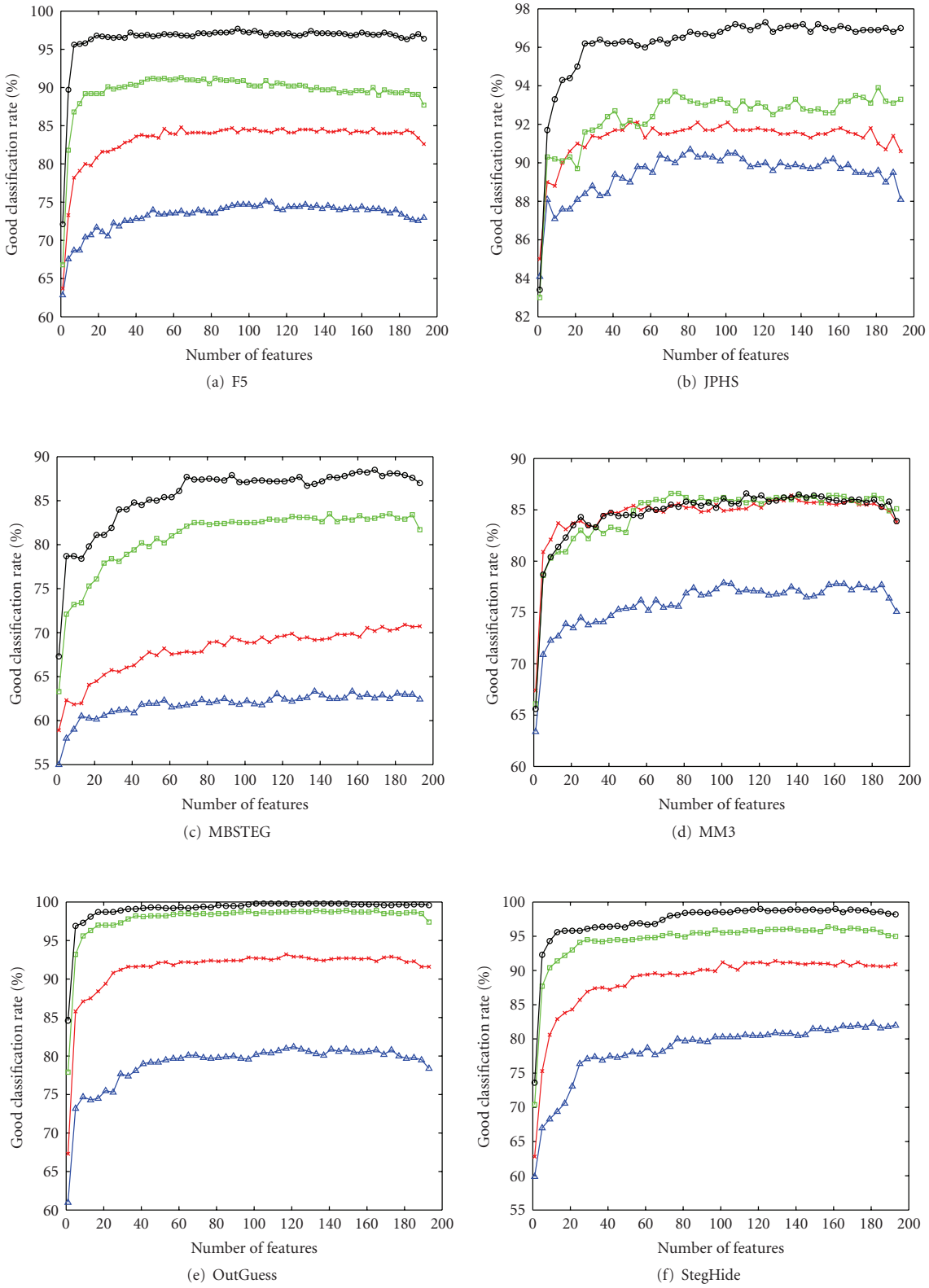

FIGURE 6: Performance in detection for all six stego algorithms versus the number of features, for the four embedding rates: black circles (○) for 20%, green squares (□) for 15%, red crosses (×) for 10%, and blue triangles (△) for 5%. Features are ranked using the Forward selection algorithm. These plots are the result of a single run of the Forward algorithm. Plots do not have the same scale, vertically.

Table 2: The 23 features previously detailed.

Functional/Feature	Functional **F**
Global histogram	$\mathbf{H}/\|\mathbf{H}\|$
Individual histogram for 5 DCT Modes	$\mathbf{h}^{21}/\|\mathbf{h}^{21}\|, \mathbf{h}^{12}/\|\mathbf{h}^{12}\|, \mathbf{h}^{13}/\|\mathbf{h}^{13}\|,$ $\mathbf{h}^{22}/\|\mathbf{h}^{22}\|, \mathbf{h}^{31}/\|\mathbf{h}^{31}\|$
Dual histogram for 11 DCT values	$\mathbf{g}^{-5}/\|\mathbf{g}^{-5}\|, \mathbf{g}^{-4}/\|\mathbf{g}^{-4}\|, \ldots,$ $\mathbf{g}^{4}/\|\mathbf{g}^{4}\|, \mathbf{g}^{5}/\|\mathbf{g}^{5}\|$
Variation	$\mathbf{V}$
$L1$ and $L2$ blockiness	$\mathbf{B}_1, \mathbf{B}_2$
Cooccurrence	N_{00}, N_{01}, N_{11}

Second, it does not allow to "go back" in the process, meaning that if performances are decreasing along the selection process, and that the addition of another feature makes performances increase again, combinations of previously selected features with this last one are not possible anymore.

The Forward selection is probably not the best possible feature selection technique, and recent contribution to these techniques such as Sequential Floating Forward Selection (SFFS) [21] and improvements of it [22] has shown that the number of computations required for feature selection can be reduced drastically. Nevertheless, the feature selection using Forward has been showing very good results and seems to perform well on the feature set used in this paper. It is not used here in the goal of obtaining the best possible combination of features but more to reduce the dimensionality and obtain some meaning out of the selected features. Improvements of this methodology could make use of such more efficient techniques of feature selection.

3.4. General Methodology. To summarize the general methodology on Figure 3 uses first a Bootstrap with 100 iterations on varying subsets sizes, to obtain a sufficient subset size and statistically reliable classifiers' results regarding the number of features used. With this number of images feature selection is performed using a Forward selection algorithm; this enables to highlight possible weaknesses of the stego algorithm.

This methodology has been applied to six popular stego algorithms for testing. Experiments and results as well as a discussion on the analysis of the selected features are given in the next section.

4. Experiments and Results

4.1. Experiments Setup

4.1.1. Steganographic Algorithms Used. Six different steganographic algorithms have been used: F5 [23], Model-Based (MBSteg) [24], MMx [25] (in these experiments, MM3 has been used), JP Hide and Seek [26], OutGuess [27], and StegHide [28]; all of them with four different embedding rates: 5%, 10%, 15%, and 20%.

4.1.2. Generation of Image Database. The image base was constituted of 10 000 images from the BOWS2 Challenge [29] database (hosted by Westfeld [30]). These images are 512×512 PGM greyscale (also available in color).

The steganographic algorithms and the proposed methodology for dimensionality reduction and steganalysis are only performed on these 512×512 images. It can also be noted that depending on image complexity, as studied in [31], local discrepancies might be observed (a classically trained steganalyzer might have troubles for such images), but on a large enough base of images, this behavior will not be visible.

4.1.3. Extraction of the Features. In the end, the whole set of images is separated in two equal parts: one is kept as untouched cover while the other one is stego with the six steganographic algorithms at four different embedding rates: 5%, 10%, 15%, and 20%. Fridrich's 193 DCT features [32] have been used for the steganalysis.

4.2. Results. Results are presented following the methodology steps. A discussion over the selected features and the possible interpretability of it are developed afterward. In the following, the term "detection rate" stands for the performance of the classifier on a scale from 0% to 100% of classification rate. It is a measure of the performance instead of a measure of error.

4.2.1. Determination of Sufficient Number of Samples. Presented first is the result of the evaluation of a sufficient number of images, as explained in the previous methodology, in Figure 5. The Bootstrap (100 rounds) is used on randomly taken subsets of 200 up to 9000 images out of the whole 10 000 from the BOWS2 challenge.

It can be seen on Figure 5 that the standard deviation behaves as expected when increasing the number of images for the cases of JPHS, MBSteg, MMx, OutGuess, and StegHide: its value decreases and tends to be below 1% of the best performance when the number of images is 5000 (even if for MBSteg with embedding rate of 5% it is a bit above 1%). This sufficient number of samples is kept as the reference and sufficient number. Another important point is that with very low number of images (100 in these cases), the standard deviation is between 1% and about 6.5% of the average classifier's performance; meaning that results computed with small number of images have at most a $\pm 6.5\%$ confidence interval. While the plots decrease very quickly when increasing the number of images, values of the standard deviation remain very high until 2000 images; these results have to take into account the embedding rate, which tends to make the standard deviation higher as it decreases.

Indeed, while differences between 15% and 20% embedding rates are not very important on the four previously mentioned stego algorithms, there is a gap between the 5%–10% plots and the 20% ones. This is expected when looking at the performances of the steganalysis process: low embedding rates tend to be harder to detect, leading to a range of possible performances wider than with high embedding rates. Figure 6 illustrates this idea on all six cases (F5, JPHS, MMx, MBSteg, StegHide, and OutGuess).

The final "sufficient" number of samples retained for the second step of the methodology—the feature selection—is 5000, for two reasons: first, the computational times are acceptable for the following computations (feature selection step with training of classifier for each step); second, the standard deviation is small enough to consider that the final classification results are given with at most 1% of standard deviation (in the case of MBSteg at 5% of embedding rate).

4.2.2. Forward Feature Selection. Features have first been ranked, using the Forward feature selection algorithm, and detection rates are plotted with increasing number of features (using the ranking provided by the Forward selection) on Figure 6.

The six analyzed stego algorithms give rather different results.

(i) F5 reaches very quickly the maximum performance for all embedding rates: only few features contribute to the overall detection rate.

(ii) JPHS reaches a plateau in performance (within the standard deviation of 1%) for all embedding rates with 41 features and remains around that performance.

(iii) OutGuess has this same plateau at 25 features, and performances are not increasing anymore above that number of features (still within the standard deviation of the results).

(iv) StegHide can be considered to have reached the maximum result (within the standard deviation interval) at 60 features.

(v) In the MM3 case, performances for embedding rates 10%, 15%, and 20% are very similar as are selected features. Performances stable at 40 features. The difference for the 5% case is most likely due to matrix embedding which makes detection harder when the payload is small.

(vi) Performances for MBSteg are stable using 70 features for embedding rates 15% and 20%. Only 30 are enough for embedding rate 5%. The case of embedding rate 10% has the classifier's performances increasing with the addition of features.

Interestingly, the features retained by the Forward selection for each embedding rate differ slightly within one steganographic algorithm. Details about the features ranked as first by the Forward algorithm are discussed afterward.

4.3. Discussion. First, the global performances, when using the reduced and sufficient feature sets mentioned in the results section above, are assessed. Note that feature selection for performing reverse engineering of a steganographic algorithm is theoretically efficient only if the features are carrying different information (if two features represent the same information, the feature selection will select only one of them).

4.3.1. Reduced Features Sets. Based on the ranking of the features obtained by the Forward algorithm, it has been decided that once performances were within 1% of the best performance obtained (among all Forward tryouts for all different sets of features), the number of features obtained was retained as a "sufficient" feature set. Performances using reduced feature sets (proper to each algorithm and embedding rate) are first compared in Table 1. It can be seen that, globally, the required size of the set of features for remaining within 1% of the best performance decreases.

It should be noted that since the aim of the feature selection is to reduce as much as possible the feature set while keeping overall same performance, it is expected that within the standard deviation interval the performance with the lowest possible number of features is behind the "maximum" one.

It remains possible, for the studied algorithms, as Figure 6 shows, to find a higher number of features for which the performance is closer or equal to the maximum one—even though this is very disputable, considering the maximal 1% standard deviation interval when using 5000 images. But this is not the goal of the feature selection step of the methodology.

4.4. Feature Sets Analysis for Reverse Engineering. Common feature sets have been selected according to the following rule: take the first common ten features (in the order ranked by the Forward algorithm) to each feature set obtained for each embedding rate (within one algorithm). It is hoped that through this selection the obtained features will be generic regarding the embedding rate.

We recall first the meaning of the different features used in this steganalysis scheme. Notations for the feature set used [32] are given for the original 23 features set, in Table 2.

This set of 23 features is expanded up to a set of 193, by removing the L_1 norm used previously and keeping all the values of the matrices and vectors. This results in the following 193 features set.

(i) A global histogram of 11 dimensions $\mathbf{H}(i)$, $i = [\![-5, 5]\!]$.

(ii) 5 low frequency DCT histograms each of 11 dimensions $\mathbf{h}^{21}(i) \cdots \mathbf{h}^{31}(i)$, $i = [\![-5, 5]\!]$.

(iii) 11 dual histograms each of 9 dimensions $\mathbf{g}^{-5}(i) \cdots \mathbf{g}^{5}(i)$, $i = [\![1, 9]\!]$.

(iv) Variation between blocks, of dimension 1 $\mathbf{V}$.

(v) 2 blockinesses of dimension 1 $\mathbf{B}_1$, $\mathbf{B}_2$.

(vi) Cooccurrence matrix of dimensions 5×5 $\mathbf{C}_{i,j}$, $i = [\![-2, 2]\!]$, $j = [\![-2, 2]\!]$.

The following is a discussion on the selected features for each steganographic algorithm.

Tables of selected feature sets are presented in Tables 3–8, with an analysis for each algorithm. Fridrich's DCT features are not the only ones having a possible physical interpretation. They have been chosen here because it is believed that most of the features can be interpreted. The

Table 3: Common feature set for F5 with average rank for each feature.

$\mathbf{B}_1$	$\mathbf{C}_{-1,-1}$	$\mathbf{C}_{-2,0}$	$\mathbf{H}(0)$	$\mathbf{B}_2$
(4)	(8)	(12)	(13)	(19)
$\mathbf{V}$	$\mathbf{g}^0(1)$	$\mathbf{h}^{22}(-3)$	$\mathbf{h}^{12}(3)$	$\mathbf{h}^{13}(-3)$
(21)	(22)	(26)	(31)	(55)

Table 4: Common feature set for MM3 with average rank for each feature.

$\mathbf{C}_{-1,1}$	$\mathbf{C}_{-2,0}$	$\mathbf{h}^{13}(-1)$	$\mathbf{H}(-1)$	$\mathbf{h}^{21}(-3)$
(1)	(3)	(7)	(22)	(22)
$\mathbf{g}^{-5}(1)$	$\mathbf{C}_{1,0}$	$\mathbf{h}^{22}(-3)$	$\mathbf{h}^{31}(-2)$	$\mathbf{H}(-3)$
(35)	(40)	(41)	(42)	(49)

Table 5: Common feature set for JPHS with average rank for each feature.

$\mathbf{g}^0(4)$	$\mathbf{B}_1$	$\mathbf{h}^{21}(-3)$	$\mathbf{h}^{21}(-1)$	$\mathbf{H}(-5)$
(1)	(25)	(26)	(30)	(30)
$\mathbf{h}^{13}(3)$	$\mathbf{h}^{31}(3)$	$\mathbf{g}^0(5)$	$\mathbf{g}^3(7)$	$\mathbf{g}^{-3}(1)$
(34)	(52)	(61)	(61)	(65)

Table 6: Common feature set for MBSteg with average rank for each feature.

$\mathbf{C}_{2,-2}$	$\mathbf{C}_{2,2}$	$\mathbf{C}_{-2,2}$	$\mathbf{C}_{2,0}$	$\mathbf{g}^{-2}(3)$
(4)	(6)	(10)	(10)	(24)
$\mathbf{g}^0(4)$	$\mathbf{H}(-2)$	$\mathbf{C}_{-1,-2}$	$\mathbf{H}(1)$	$\mathbf{H}(2)$
(27)	(31)	(32)	(36)	(50)

Table 7: Common feature set for OutGuess with average rank for each feature.

$\mathbf{C}_{-2,0}$	$\mathbf{H}(-2)$	$\mathbf{C}_{-2,-2}$	$\mathbf{C}_{0,-2}$	$\mathbf{h}^{13}(0)$
(3)	(3)	(7)	(8)	(12)
$\mathbf{H}(-3)$	$\mathbf{C}_{-1,0}$	$\mathbf{h}^{22}(1)$	$\mathbf{H}(0)$	$\mathbf{g}^{-4}(8)$
(14)	(23)	(31)	(41)	(45)

Table 8: Common feature set for StegHide with average rank for each feature.

$\mathbf{C}_{2,0}$	$\mathbf{C}_{-2,2}$	$\mathbf{C}_{-2,0}$	$\mathbf{B}_1$	$\mathbf{B}_2$
(6)	(22)	(22)	(25)	(27)
$\mathbf{g}^1(1)$	$\mathbf{h}^{13}(5)$	$\mathbf{h}^{21}(-3)$	$\mathbf{C}_{-2,-1}$	$\mathbf{g}^{-1}(4)$
(28)	(45)	(46)	(47)	(54)

proposed short analysis of the weaknesses of stego algorithms is using this interpretation.

4.4.1. F5I . F5 (Table 3) is rather sensitive to both blockiness detections and, interestingly, is the only of the six tested algorithms to be sensitive to the variation $\mathbf{V}$. As for other algorithms, cooccurrence coefficients are triggered.

4.4.2. MM3. MM3 (Table 4) tends to be sensitive to global histogram features as well as DCT histograms, which are not preserved.

4.4.3. JPHS. JPHS (Table 5) seems not to preserve the DCT coefficients histograms. Also the dual histograms react vividly for center values and extremes ones (-3 and 3).

4.4.4. MBSteg. The features used (Table 6) include global histograms with values 1, -2, and 2, which happens only because of the calibration in the feature extraction process. MBSteg preserves the coefficients' histograms but does not take into account a possible calibration. Hence, the unpreserved histograms are due to the calibration process in the feature extraction. Information leaks through the calibration process. Also cooccurrence values are used, which is a sign that MBSteg does not preserve low and high frequencies.

4.4.5. OutGuess. Cooccurrence values are mostly used (values -2, -1) in the feature set for OutGuess (Table 7) and a clear weak point. The calibration process has also been of importance since the global histograms of extreme values -3 and -2 have been taken into account.

4.4.6. StegHide. For StegHide (Table 8), blockiness and cooccurrence values are mostly used, again for low and high frequencies.

From a general point of view, it can be seen that most of the analyzed algorithms are sensitive to statistics of lowpass-calibrated DCT coefficients, represented by features $\mathbf{h}^{13}$ and $\mathbf{h}^{21}$. This is not surprising since these coefficients contain a large part of the information of a natural image; their associated densities are likely to be modified by the embedding process.

5. Conclusions

This paper has presented a methodology for the estimation of a sufficient number of images for a specific feature set using the standard deviation of the detection rate obtained by the classifier as a criterion (a Bootstrap technique is used for that purpose); the general methodology presented can nonetheless be extended and applied to other feature sets. The second step of the methodology aims at reducing the dimensionality of the data set, by selecting the most relevant features, according to a Forward selection algorithm; along with the positive effects of a lower dimensionality, analysis of the selected features is possible and gives insights on the steganographic algorithm studied.

Three conclusions can be drawn from the methodology and experiments in this paper.

(i) Results on standard deviation for almost all studied steganographic algorithms have proved that the feature-based steganalysis is reliable and accurate only if a sufficient number of images is used for the actual training of the classifier used. Indeed, from most of the results obtained concerning standard deviation values (and therefore statistical stability of the results), it is rather irrelevant to possibly increase detection performance by 2% while working with a standard deviation for these same results of 2%.

Table 9: The 40 first features ranked by the Forward algorithm for the F5 algorithm at 5% embedding rate.

$h^{13}(0)$	$H(0)$	B_1	V	$C_{0,0}$	$g^0(2)$	$h^{31}(-1)$	$C_{2,1}$	$g^0(7)$	$C_{2,-1}$
$g^{-2}(7)$	$g^{-3}(4)$	B_2	$h^{12}(-5)$	$g^{-1}(9)$	$g^4(5)$	$g^5(3)$	$g^{-4}(5)$	$g^{-4}(9)$	$g^3(5)$
$h^{31}(3)$	$h^{13}(1)$	$g^{-1}(6)$	$g^{-2}(1)$	$h^{13}(2)$	$h^{12}(5)$	$g^3(6)$	$C_{1,-2}$	$h^{13}(-5)$	$h^{22}(5)$
$g^{-4}(1)$	$g^4(9)$	$C_{2,-2}$	$g^{-3}(6)$	$g^{-5}(9)$	$h^{12}(3)$	$h^{31}(0)$	$h^{21}(-4)$	$g^2(9)$	$g^0(9)$

Table 10: The 40 first features ranked by the Forward algorithm for the JPHS algorithm at 5% embedding rate.

$g^0(4)$	$h^{22}(0)$	$C_{1,0}$	B_1	$H(1)$	$h^{21}(0)$	$g^1(4)$	$g^0(8)$	$g^{-2}(9)$	$g^{-2}(5)$
$g^4(5)$	$g^0(5)$	$g^1(9)$	$g^{-1}(2)$	B_2	$g^2(8)$	$C_{0,0}$	$h^{31}(5)$	$g^0(9)$	$h^{22}(1)$
$g^{-2}(2)$	$g^{-1}(7)$	$g^{-3}(8)$	$g^0(1)$	$h^{31}(-3)$	$h^{21}(-1)$	$h^{22}(-1)$	$g^{-4}(6)$	$C_{-1,-2}$	$g^5(7)$
$h^{12}(-5)$	$g^{-5}(8)$	$h^{21}(2)$	$g^0(7)$	$h^{12}(-2)$	$h^{22}(-4)$	$h^{31}(0)$	$C_{0,2}$	$H(2)$	$g^5(5)$

Table 11: The 40 first features ranked by the Forward algorithm for the MBSteg algorithm at 5% embedding rate.

$g^{-2}(1)$	$H(2)$	$g^{-4}(7)$	$h^{13}(1)$	$h^{22}(1)$	$C_{2,-2}$	$C_{-1,-1}$	$h^{31}(1)$	$g^4(7)$	$g^{-2}(4)$
$h^{21}(0)$	$h^{31}(-4)$	$h^{21}(-4)$	$C_{0,2}$	$C_{1,2}$	$h^{31}(-1)$	$H(0)$	$h^{21}(3)$	$g^{-5}(6)$	$h^{22}(-3)$
$h^{13}(-1)$	$C_{2,0}$	$C_{1,2}$	$g^5(6)$	$C_{-2,-1}$	$g^{-3}(6)$	$g^5(4)$	$g^{-2}(7)$	$g^{-1}(7)$	$g^{-4}(8)$
$h^{22}(-1)$	$g^2(1)$	$g^0(8)$	$h^{22}(-5)$	$H(-2)$	$h^{12}(-4)$	$g^5(5)$	$h^{12}(-2)$	$g^2(4)$	$h^{21}(-3)$

Table 12: The 40 first features ranked by the Forward algorithm for the MM3 algorithm at 5% embedding rate.

$C_{-1,-1}$	$h^{13}(-1)$	$C_{0,-2}$	$C_{1,1}$	$g^0(9)$	$C_{2,0}$	$h^{21}(-1)$	$h^{13}(1)$	$g^{-3}(2)$	$C_{1,0}$
$H(-2)$	$g^4(4)$	$g^2(2)$	$C_{-2,0}$	$C_{0,-1}$	$C_{-1,-2}$	$g^{-2}(3)$	$h^{22}(-3)$	$g^2(3)$	$h^{13}(3)$
$h^{31}(-1)$	$g^{-1}(9)$	$g^{-2}(8)$	$g^0(7)$	$h^{21}(-5)$	$h^{21}(3)$	$C_{-1,1}$	$g^{-1}(3)$	$g^5(3)$	$h^{31}(1)$
$g^0(3)$	B_1	$C_{-2,1}$	B_2	$g^{-4}(6)$	$C_{0,2}$	$H(-1)$	$g^2(5)$	$h^{13}(0)$	$g^2(7)$

Table 13: The 40 first features ranked by the Forward algorithm for the Outguess algorithm at 5% embedding rate.

$h^{13}(0)$	$C_{0,-1}$	$C_{-2,0}$	$H(-2)$	B_1	$C_{0,-2}$	$g^0(7)$	$h^{31}(-3)$	$C_{-2,-1}$	$g^0(2)$
B_2	$H(-1)$	$g^{-2}(2)$	$h^{13}(-1)$	$h^{22}(-1)$	$h^{22}(0)$	$h^{12}(-3)$	$g^{-2}(5)$	$g^1(8)$	$h^{21}(-2)$
$g^{-2}(9)$	$g^1(1)$	$H(5)$	$H(4)$	$g^2(1)$	$g^0(1)$	$g^{-3}(5)$	$g^0(9)$	$g^{-3}(8)$	$g^{-3}(3)$
$g^{-5}(4)$	$g^{-5}(5)$	$C_{-2,-2}$	$g^{-1}(6)$	$g^{-2}(6)$	$g^4(3)$	$C_{-1,-1}$	$C_{-1,0}$	$g^{-2}(7)$	$C_{-1,1}$

Table 14: The 40 first features ranked by the Forward algorithm for the Steghide algorithm at 5% embedding rate.

$C_{0,-1}$	$g^0(2)$	$C_{0,2}$	$C_{2,-2}$	B_1	B_2	$C_{1,1}$	$C_{0,-2}$	$C_{-2,2}$	$h^{13}(-1)$
$g^{-5}(3)$	$h^{21}(-3)$	$C_{0,1}$	$h^{13}(0)$	$C_{1,-1}$	$h^{31}(-1)$	$g^{-3}(3)$	$g^3(6)$	$h^{31}(-2)$	$g^1(3)$
$h^{22}(1)$	$C_{-2,-2}$	$g^{-4}(4)$	$h^{13}(1)$	$C_{-2,0}$	$g^1(4)$	$C_{2,1}$	$H(-1)$	$C_{2,2}$	$h^{22}(5)$
$g^2(5)$	$C_{-1,-1}$	$g^1(9)$	$C_{2,0}$	$g^2(7)$	$g^{-1}(1)$	$h^{31}(5)$	$H(-2)$	$h^{21}(1)$	$g^{-2}(9)$

(ii) Through the second step of the methodology, the required number of features for steganalysis can be decreased. This with three main advantages: (a) performances remain the same if the reduced feature set is properly constructed; (b) the selected features from the reduced set are relevant and meaningful (the selected set can possibly vary, according to the feature selection technique used) and make reverse-engineering possible; (c) the weaknesses of the stego algorithm also appear from the selection; this can lead, for example, to improvements of the stego algorithm.

(iii) The analysis on the reduced common feature sets obtained between embedding rates of the same stego algorithm shows that the algorithms are sensitive to roughly the same features, as a basis. Meanwhile, when embedding rates get as low as 5%, or for very efficient algorithms, some very specific features appear.

Hence, the first step of the methodology is a requirement for not only any new stego algorithm but also new feature sets/steganalyzers, willing to present its performances: a sufficient number of images for the stego algorithm and the

steganalyzer used to test it have to be assessed in order to have stable results (i.e., with a small enough standard deviation of its results to make the comparison with current state of the art techniques meaningful).

Also, from the second step of the methodology, the most relevant features can be obtained and make possible a further analysis of the stego algorithm considered, additionally to the detection rate obtained by the steganalyzer.

Appendix

Features Ranked by the Forward Algorithm

In appendix are given the first 40 features obtained by the Forward ranking for each stego algorithm with 5% embedding rate (Tables 9–14). Only one embedding rate result is given for space reasons. 5% embedding rate results have been chosen since they tend to be different (in terms of ranked features by the Forward algorithm) from the other embedding rates and also because 5% embedding rate is a difficult challenge in terms of steganalysis; these features are meaningful for this kind of difficult steganalysis with these six algorithms.

Acknowledgments

The authors would like to thank Jan Kodovsky and Jessica Fridrich for their implementation of the DCT Feature Extraction software. Also many thanks to Tomás Pevný for his helpful comments and suggestions on this article. The work in this paper was supported (in part) by the French national funding under the project RIAM Estivale (ANR-05-RIAM-O1903), ANR projects Nebbiano (ANR-06-SETI-009), and TSAR French Project (ANR SSIA 2006-2008).

References

[1] D. McCullagh, "Secret Messages Come in .Wavs. Wired News," February 2001, http://www.wired.com/news/politics/0,1283,41861,00.html.

[2] N. Provos and P. Honeyman, "Detecting steganographic content on the internet," in *Proceedings of the Network and Distributed System Security Symposium (NDSS '02)*, San Diego, Calif, USA, February 2002.

[3] C. Cachin, "An information-theoretic model for steganography," in *Proceedings of the 2nd International Workshop on Information Hiding (IH '98)*, vol. 1525 of *Lecture Notes in Computer Science*, pp. 306–318, Portland, Ore, USA, April 1998.

[4] A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," in *Proceedings of the 3rd International Workshop on Information Hiding (IH '99)*, vol. 1768 of *Lecture Notes in Computer Science*, pp. 61–76, Springer, Dresden, Germany, September-October 2000.

[5] J. Fridrich, "Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes," in *Proceedings of the 6th International Workshop on Information Hiding (IH '04)*, vol. 3200 of *Lecture Notes in Computer Science*, pp. 67–81, Toronto, Canada, May 2004.

[6] S. Lyu and H. Farid, "Detecting hidden messages using higher-order statistics and support vector machines," in *Proceedings of the 5th International Workshop on Information Hiding (IH '02)*, vol. 2578 of *Lecture Notes in Computer Science*, pp. 340–354, Noordwijkerhout, The Netherlands, October 2003.

[7] S. S. Agaian and H. Cai, "New multilevel dct, feature vectors, and universal blind steganalysis," in *Security, Steganography, and Watermarking of Multimedia Contents VII*, vol. 5681 of *Proceedings of SPIE*, pp. 653–663, San Jose, Calif, USA, January 2005.

[8] Y. Q. Shi, C. Chen, and W. Chen, "A Markov process based approach to effective attacking JPEG steganography," in *Proceedings of the 8th International Workshop on Information Hiding (IH '06)*, vol. 4437 of *Lecture Notes in Computer Science*, pp. 249–264, Alexandria, Va, USA, July 2007.

[9] D. François, *High-dimensional data analysis: optimal metrics and feature selection*, Ph.D. thesis, Université Catholique de Louvain, Louvain, Belgium, September 2006.

[10] A. D. Ker, "The ultimate steganalysis benchmark?" in *Proceedings of the 9th Multimedia and Security Workshop (MM/Sec '07)*, pp. 141–148, Dallas, Tex, USA, September 2007.

[11] T. Filler, A. D. Ker, and J. Fridrich, "The square root law of steganographic capacity for Markov covers," in *Media Forensics and Security*, E. J. Delp III, J. Dittmann, N. D. Memon, and P. W. Wong, Eds., vol. 7254 of *Proceedings of SPIE*, pp. 1–11, San Jose, Calif, USA, January 2009.

[12] R. Bellman, *Adaptive Control Processes: A Guided Tour*, Princeton University Press, Princeton, NJ, USA, 1961.

[13] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall/CRC, Londres, Argentina, 1994.

[14] D. W. Scott and J. R. Thompson, "Probability density estimation in higher dimensions," in *Computer Science and Statistics: Proceedings of the 15th Symposium on the Interface*, S. R. Douglas, Ed., pp. 173–179, North-Holland, Houston, Tex, USA, March 1983.

[15] Y. Miche, P. Bas, A. Lendasse, C. Jutten, and O. Simula, "Extracting relevant features of steganographic schemes by feature selection techniques," in *Proceedings of the 3rd Wavilla Challenge (Wacha '07)*, pp. 1–15, St. Malo, France, June 2007.

[16] Y. Miche, B. Roue, A. Lendasse, and P. Bas, "A feature selection methodology for steganalysis," in *Proceedings of the International Workshop on Multimedia Content Representation, Classification and Security (MRCS '06)*, vol. 4105 of *Lecture Notes in Computer Science*, pp. 49–56, Istanbul, Turkey, September 2006.

[17] Y. Miche, P. Bas, C. Jutten, O. Simula, and A. Lendasse, "A methodology for building regression models using extreme learning machine: OP-ELM," in *Proceedings of the 16th European Symposium on Artificial Neural Networks (ESANN '08)*, pp. 1–6, Bruges, Belgium, April 2008.

[18] A. Sorjamaa, Y. Miche, R. Weiss, and A. Lendasse, "Long-term prediction of time series using NNE-based projection and OP-ELM," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '08)*, pp. 2674–2680, Hong Kong, June 2008.

[19] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[20] F. Rossi, A. Lendasse, D. François, V. Wertz, and M. Verleysen, "Mutual information for the selection of relevant variables in spectrometric nonlinear modelling," *Chemometrics and Intelligent Laboratory Systems*, vol. 80, no. 2, pp. 215–226, 2006.

[21] D. Ververidis and C. Kotropoulos, "Fast and accurate sequential floating forward feature selection with the Bayes classifier applied to speech emotion recognition," *Signal Processing*, vol. 88, no. 12, pp. 2956–2970, 2008.

[22] D. Ververidis and C. Kotropoulos, "Fast sequential floating forward selection applied to emotional speech features estimated on des and susas data collections," in *Proceeding of the 14th European Signal Processing Conference (EUSIPCO '06)*, EURASIP, Ed., pp. 1–5, Florence, Italy, September 2006.

[23] A. Westfeld, "F5—a steganographic algorithm," in *Proceedings of the 4th International Workshop on Information Hiding (IH '01)*, vol. 2137 of *Lecture Notes in Computer Science*, pp. 289–302, Pittsburgh, Pa, USA, April 2001.

[24] P. Sallee, "Model-based steganography," in *Proceedings of the 2nd International Workshop Digital Watermarking (IWDW '03)*, vol. 2939 of *Lecture Notes in Computer Science*, pp. 254–260, Seoul, Korea, October 2004.

[25] Y. Kim, Z. Duric, and D. Richards, "Modified matrix encoding technique for minimal distortion steganography," in *Proceedings of the 8th International Workshop on Information Hiding (IH '06)*, vol. 4437 of *Lecture Notes in Computer Science*, pp. 314–327, Alexandria, Va, USA, July 2007.

[26] A. Latham, "Jphide & seek," August 1999, http://linux01.gwdg.de/~alatham/stego.html.

[27] N. Provos, "Defending against statistical steganalysis," in *Proceedings of the 10th USENIX Security Symposium*, p. 24, Washington, DC, USA, August 2001.

[28] S. Hetzl and P. Mutzel, "A graph-theoretic approach to steganography," in *Proceedings of the 9th IFIP TC-6 TC-11 International Conference on Communications and Multimedia Security (CMS '05)*, vol. 3677 of *Lecture Notes in Computer Science*, pp. 119–128, Springer, Salzburg, Austria, September 2005.

[29] "Watermarking Virtual Laboratory (Wavila) of the European Network of Excellence ECRYPT," The 2nd bows contest (break our watermarking system), 2007.

[30] A. Westfeld, "Reproducible signal processing (bows2 challenge image database, public)".

[31] Q. Liu, A. H. Sung, B. Ribeiro, M. Wei, Z. Chen, and J. Xu, "Image complexity and feature mining for steganalysis of least significant bit matching steganography," *Information Sciences*, vol. 178, no. 1, pp. 21–36, 2008.

[32] T. Pevny and J. Fridrich, "Merging Markov and DCT features for multi-class JPEG steganalysis," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505 of *Proceedings of SPIE*, pp. 1–13, San Jose, Calif, USA, January 2007.

PUBLICATION F

TITLE:
Using Multiple Re-embeddings for Quantitative Steganalysis and Image Reliability Estimation

AUTHORS:
Yoan Miche, Patrick Bas and Amaury Lendasse

PUBLISHED IN:
TKK Reports in Information and Computer Science, June 2010, Espoo.
Number TKK-ICS-R34.

ISBN:
978-952-60-3249-8 (Print)
ISBN:
978-952-60-3250-4 (Online)
URL:
http://lib.tkk.fi/Reports/2010/isbn9789526032504

TKK Reports in Information and Computer Science
Espoo 2010 TKK-ICS-R34

USING MULTIPLE RE-EMBEDDINGS FOR QUANTITATIVE STEGANALYSIS AND IMAGE RELIABILITY ESTIMATION

Yoan Miche, Patrick Bas and Amaury Lendasse

Aalto University
School of Science
and Technology

TKK Reports in Information and Computer Science
Espoo 2010 TKK-ICS-R34

USING MULTIPLE RE-EMBEDDINGS FOR QUANTITATIVE STEGANALYSIS AND IMAGE RELIABILITY ESTIMATION

Yoan Miche, Patrick Bas and Amaury Lendasse

Aalto University School of Science and Technology
Faculty of Information and Natural Sciences
Department of Information and Computer Science

Aalto-yliopiston teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Tietojenkäsittelytieteen laitos

Distribution:
Aalto University School of Science and Technology
Faculty of Information and Natural Sciences
Department of Information and Computer Science
PO Box 15400
FI-00076 AALTO
FINLAND
URL: http://ics.tkk.fi
Tel. +358 9 470 01
Fax +358 9 470 23369
E-mail: series@ics.tkk.fi

ISBN 978-952-60-3249-8 (Print)
ISBN 978-952-60-3250-4 (Online)
ISSN 1797-5034 (Print)
ISSN 1797-5042 (Online)
URL: http://lib.tkk.fi/Reports/2010/isbn9789526032504.pdf

TKK ICS
Espoo 2010

ABSTRACT: The quantitative steganalysis problem aims at estimating the amount of payload embedded inside a document. In this paper, JPEG images are considered, and by the use of a re-embedding based methodology, it is possible to estimate the number of original embedding changes performed on the image by a stego source and to slightly improve the estimation regarding classical quantitative steganalysis methods. The major advance of this methodology is that it also enables to obtain a confidence interval on this estimated payload. This confidence interval then permits to evaluate the difficulty of an image, in terms of steganalysis by estimating the reliability of the output. The regression technique comes from the OP-ELM and the reliability is estimated using linear approximation. The methodology is applied with publicly available stego algorithms, regression model and large databases of images. The methodology is generic and can be used for any quantitative steganalysis problem of this class.

KEYWORDS: Steganography, Steganalysis, OP-ELM, Quantitative Steganalysis, Re-embedding, Inner Image Difficulty

CONTENTS

1 INTRODUCTION

The classical goal of steganalysis is to detect whether a document (considered to be images, here) has been tampered with or not. While this detection is important, one can wish to obtain more information about the actual payload present in the image. This problem is addressed by quantitative steganalysis: it estimates the embedded payload, usually by estimating directly the number of embedding changes that have been made to the image in the first place. An initial approach to this has been proposed in [6, 15]. Such a problem has been addressed recently for example by the use of classical blind steganalysis features such as [5]: the knowledge of the stego algorithm is supposed to be given, following Kerckhoff's principles [9] — or inferred by some usual means of blind steganalysis [13, 14] for example —, and the problem of payload estimation comes down to a regression problem, with the output being the payload to predict and inputs being the blind steganalysis features. In a recent paper, this regression has been achieved through the use of Ordinary Least Squares (OLS) and Support Vector Regression (SVR) [15].

In such a setup, it is assumed that one can use the identified stego algorithm in order to train an OLS or SVR model, for example on a known dataset. Such a model can then be used on new unknown images (the intercepted images on a specific channel) to estimate a possible embedded payload.

Although this usually leads to a good estimation, it is interesting to also have a confidence interval on such estimation, which gives information on the quality of the estimation as well as the possible "difficulty" of the considered image (reliability), i.e. the reliability of the output.

This problem of image reliability is important for future steganography. Indeed, in the case where a specific image is known to be "difficult", a steganographer will prefer using it, knowing that it is more likely to be misclassified or have a payload estimation that is unreliable. In [16], the authors propose to estimate the embedding capacity of the image beforehand, in order to embed the payload into the possibly most appropriate images. Such an approach, combined with reliability estimation can lead to more secure steganography. For example, the estimation of the difficulty of the image could be a starting point to perform batch steganography by embedding a payload function of the difficulty of the image.

This idea of image difficulty was first related to the error in steganalysis in the work of Böhme [2]. In this paper, the authors define a two-error model for the quantitative steganalysis setup, with a *within-image* error and a *between-image* one. The between-image error relates to the possible inaccurate assumptions made on the cover image and is thus related to images as a whole.

The within-image error is highly related to the concept of difficulty used in this paper and attempts to take into account the errors caused by the possible dependencies between a cover image and the message embedded in it.

In the original paper, the authors illustrate through the use of numerous types of steganalysis on a LSB replacement steganography scheme that the between-image error and the within-image error are quite different in nature: the between-image error follows rather closely that of a Student's t distribu-

tion, while the within-image error is similar to a Gaussian one. It also seems that some of the steganalysis schemes tested by the authors are more prone to one type of error than the other.

The within-image error is related in [2] to a measure of the local variance of the image, introduced in the paper and computed over the original image. The concept of difficulty and the measure for it proposed in section 2 are tightly related to the within-image error and uses multiple repetitions of steganography with different messages on the same image. One main difference here is that a blind approach is used to determine it, i.e. it is assumed that the original image is not available and it is only possible to rely on the intercepted suspicious image.

In this paper, a methodology applicable to any stego algorithm is proposed in order to devise a confidence interval on the provided estimation of the original embedding rate, by using re-embeddings on the considered image. Using this methodology, it is possible to obtain:

- A better estimate of the original embedding rate used on an intercepted suspicious image which is tantamount to the number of embedding changes or the initial number of non-zero AC coefficients;
- An estimate of the original number of non-zero AC coefficients of the genuine image (and hence, from the embedding rate and this, the number of embedding changes);
- An estimated confidence interval on the embedding rate and on the number of non-zero AC coefficients;
- Using the confidence interval, a measure of the "difficulty" of the image.

Follows a description of the methodology, in section 2, and a set of experiments on the BOWS2 [1] and BOSS [12] images sets (respectively 10000 and 9074 images) in section 3.

2 METHODOLOGY

The following methodology is described for a single image, for the sake of simplicity of notations.

In the following, the embedding rate is defined as the ratio R between the number of embedding changes E and the number of non-zero AC coefficients A: $R = \frac{E}{A}$.

Assume that we have intercepted an image I_o coming from a suspicious source, as in Figure 2, with a payload embedded P_o, which will be in the following assimilated to the number of embedding changes E_o performed on I_o.

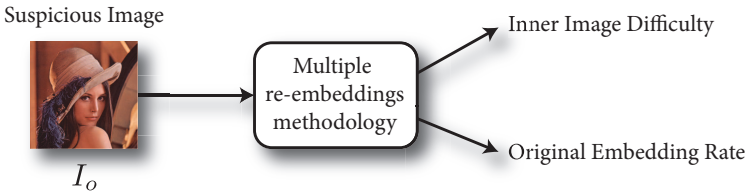

Figure 1: Suspicious image I_o with unknown payload P_o, assimilated to the number of embedding changes made in the image E_o, by a stego algorithm S. The proposed methodology gives an estimate of E_o and of the inner image difficulty.

According to Kerckhoffs' principle [9], the stego algorithm S can be considered known; if not, it can be devised by the means of blind steganalysis, using multi-class classifiers [5], for example.

A model $\mathcal{M}$ that estimates the embedding rates R is first trained on a given training set for which the embedding rates are known. This model is supposed to be available in the following.

2.1 Re-embedding concept

In this paper we propose to use the re-embedding idea to embed again some information inside the considered image I_o. The rationale here is to assume that the reliability of the estimation of the initial embedding rate is function of the reliability after multiple re-embeddings. Multiple such re-embedding with different sizes provide images with a larger embedding rate, of which a part is known. The global idea of the re-embedding and its use in this paper is illustrated in Figure 2.

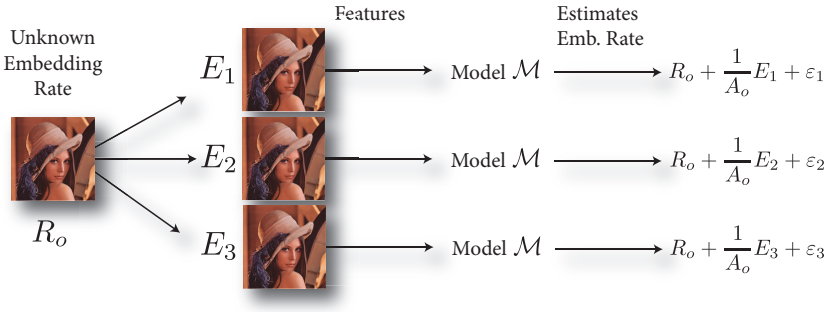

Figure 2: The Re-embedding concept: the original image I_o supposedly having a payload with embedding rate R_o is duplicated N times ($N = 3$ here) and payloads with number of embedding changes E_i are embedded in it. Features are extracted from each duplicate image (with additional embedding changes) and the previously built model $\mathcal{M}$ is used on these features to devise the final embedding rate $\hat{R}_i$.

Consider the intercepted image I_o; the idea is to make a known amount E_i of new embedding changes to I_o. This process is repeated N times $\{E_i, 1 \leq i \leq N\}$ on the image I_o, in order to obtain a set of images $\{I_i, 1 \leq i \leq N\}$ for each of which E_i re-embedding changes are performed.

After this re-embedding procedure, the actual embedding rate for image I_i is approximated as

$$R_i = \frac{E_o + E_i}{A_o} = R_o + \frac{1}{A_o} E_i, \tag{1}$$

with E_o and A_o the number of embedding changes and the number of non-zero AC coefficients in the considered image I_o, respectively (the sender of the suspicious image I_o has caused E_o embedding changes). It is assumed in this context that the number of non-zero AC coefficients A might vary due to an embedding. Some stego algorithms attempt to not modify this quantity, though.

In order to illustrate that Eq. 1 is a good approximation for low E_o and E_i, let us introduce two additional notations: the total number of pixels in the image I, $N_{\text{pix}}(I)$ and the *real* total number of embedding changes E_i^{tot}, measured between the original "clean" image I and the image I_i for which re-embedding with E_i embedding changes has been performed.

If the stego algorithm S is assumed to modify directly LSBs of pixels for each embedding change to perform (no matrix encoding, for example), it is possible to estimate the probability P_{pix} of a pixel to be modified by both the first embedding (by the sender) and the re-embedding. Using these notations, it is straightforward,

$$P_{\text{pix}} = \frac{E_o}{N_{\text{pix}}(I)} \times \frac{E_i}{N_{\text{pix}}(I)}. \tag{2}$$

Figure 3 illustrates the validity of the approximation made by Eq. 1, for small $E_o + E_i$ (the experiment uses the nsF5 algorithm [17, 7] and Fridrich's extended DCT calibrated features [5]). Note that the plot of $E_o + E_i - P_{\text{pix}}(E_o + E_i)$ would be barely distinguishable from that of $E_o + E_i$ here, due to $P_{\text{pix}} \ll 1$. This is the case when the assumptions on E_o and the range of E_i made in this paper are met: "low" E_o (compared to N_{pix}) and a controlled small range for E_i. In the event of a careless steganographer (E_o exceptionally large) for example, this result might not hold as well as here.

In addition, the absolute error made by the approximation of Eq. 1 versus the number of re-embedding changes E_i is depicted on Figure 4 for one image (the behavior is the same for all images used in this paper, for both stego algorithms). Consequently, the larger E_i, the more probable it is that some "overlap" happens, between the initial embedding changes E_o and the re-embeddings E_i, which is expected from Eq. 2.

The rationale in this paper is that the sender is not careless about the embedding rate used and that the number of re-embedding changes E_i are controlled in a certain range. With these assumptions, Eq. 1 is a reasonable approximation.

Then, in the very same way as that of the quantitative steganalysis, it is possible to obtain an estimation of the R_i, using a previously trained regression model $\mathcal{M}$. Denoting $\mathbf{X}_i = (x_i^1, \ldots, x_i^d)$ the d-dimensional feature vector extracted for image I_i, one gets the predicted embedding rate $\hat{R}_i = \mathcal{M}(\mathbf{X}_i)$.

From Eq. 1 comes

$$\hat{R}_i = R_o + \frac{1}{A_o} E_i + \varepsilon_i, \tag{3}$$

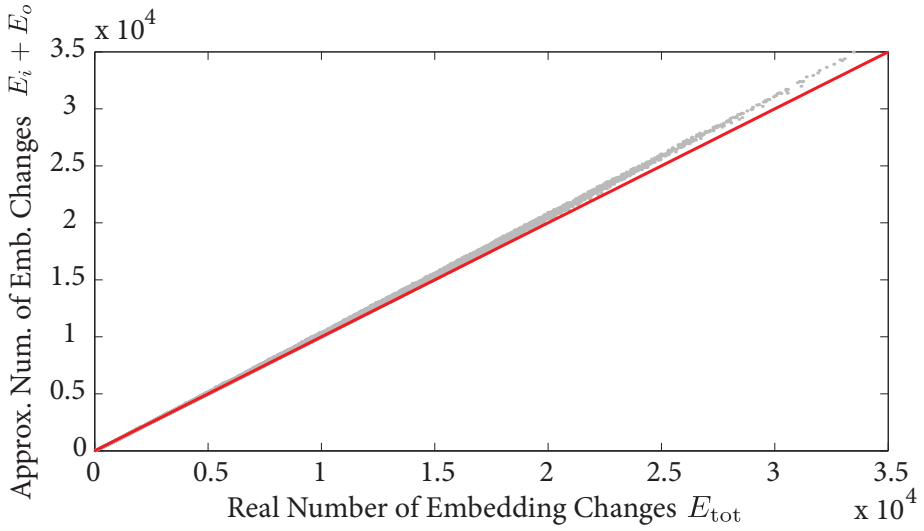

Figure 3: Approximated number of total embedding changes by Eq. 1, $E_o + E_i$, versus the *real* total number of embedding changes E_{tot}. The solid line denotes the case where $(E_o + E_i) = E_{\text{tot}}$ exactly. The plot of $E_o + E_i - P_{\text{pix}} \times N_{\text{pix}}$ is not distinguishable from that of $E_o + E_i$ and is not depicted here. This experiment uses the nsF5 stego algorithm [17, 7]. $E_o = 4122$ for this graph.

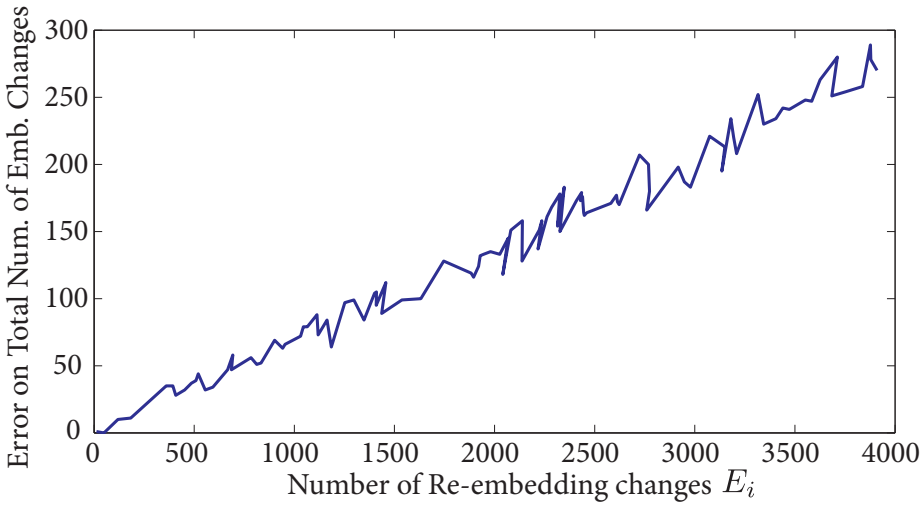

Figure 4: Error on the total number of embedding changes $(E_o + E_i) - (E_{\text{tot}})$ versus the number of re-embedding changes E_i. $E_o = 4122$ for this graph.

with ε_i the error made in the estimation of R_i. It is assumed in the following that the ε_i are independent from each other and from the E_i, for simplicity.

2.2 Confidence interval estimation

Since both quantities $\hat{R}_i$ and E_i are known, the confidence interval and the estimation of the original embedding rate $\hat{R}_o$ can then be obtained by solving the linear system

$$\frac{E_o}{A_o} + \frac{1}{A_o}\mathbf{E} = \hat{\mathbf{R}}, \tag{4}$$

with $\hat{\mathbf{R}} = (\hat{R}_1, \ldots, \hat{R}_N)^T$ the vector holding the estimations made by model $\mathcal{M}$ and $\mathbf{E} = (E_1, \ldots, E_N)^T$ the vector of the embedding changes performed.

This system is solved in a Least Squares sense, by minimizing $||\varepsilon||^2$, where $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_N)^T$, which comes down to the problem

$$\min_{\alpha,\beta} \left\| \alpha + \beta \cdot \mathbf{E} - \hat{\mathbf{R}} \right\|^2, \tag{5}$$

with $\alpha = \frac{E_o}{A_o}$ and $\beta = \frac{1}{A_o}$. This is solved by a classical pseudo-inverse formulation.

The constant term in the minimization problem is the original rate R_o for which we will obtain "an estimate" $\hat{R}_o$, along with a confidence interval on the value $\hat{R}_o$, denoted $\left[\hat{R}_o^{\text{INF}}, \hat{R}_o^{\text{SUP}}\right]$. This confidence interval is obtained using the Matlab® function `regress`, which uses a Student's t score, as described in [4]: $\hat{R}_o^{\text{INF}}$ is obtained by

$$\hat{R}_o^{\text{INF}} = \hat{R}_o - t_{\alpha/2,\nu}\hat{\sigma}\left(\hat{R}_o\right), \tag{6}$$

where $t_{\alpha/2,\nu}$ is the t score (inverse Student t cdf) with parameter $\alpha/2$ (for a $100(1-\alpha)\%$ confidence interval) with ν degrees of freedom (here $\nu = N-2$), and $\hat{\sigma}(\hat{R}_o)$ is the estimated standard deviation of $\hat{R}_o$. The upper bound $\hat{R}_o^{\text{SUP}}$ is computed similarly, and the confidence interval for the first order term also (please refer to [4] for the derivations). One can also obtain the number of non-zero AC coefficients A_o when solving the system, and hence recover the original number of embedding changes E_o.

This is illustrated on two sets of images in the experiments section 3.

2.3 Estimation of the inner image difficulty

The inner difficulty of the image can be represented as the variation of the predictions for a given original embedding rate E_o when the embedding key, or the embedded message fluctuates (similarly to [2]). Note that this variation is solely due to the characteristics of the cover image. Consequently our rationale is to measure the image difficulty as the standard deviation of the error performed for various embeddings on this image (no re-embeddings).

That is, for a genuine image I, L different embeddings are performed with different number of embedding changes $\{E_i^{\mathrm{O}}, 1 \leq i \leq L\}$. The error $\varepsilon_{\mathrm{i}}^O$ between the estimated value of the embedding rate $\hat{R}_i^{\mathrm{O}}$ (by model $\mathcal{M}$) and the true value R_i^{O} is then defined as $\varepsilon_{\mathrm{i}}^O = R_i^{\mathrm{O}} - \hat{R}_i^{\mathrm{O}}$.

The standard deviation of this quantity over the L different realizations is the proposed measure of the inner image difficulty D for image I:

$$D_I = \mathrm{std}\left(\varepsilon^O\right), \tag{7}$$

with $\varepsilon^O = \left(\varepsilon_1^O, \ldots, \varepsilon_L^O\right)^T$.

In order to show that the estimated confidence interval gives information on the inner image difficulty, through the re-embeddings, the quantity D_I inherent to each image I, is compared to the width of the estimated confidence interval for $\hat{R}_o$.

A dependence between the two proves the width of the estimated confidence interval can be used as an indicator of the image difficulty measured by D_I.

Note that the calculation of D_I for an image requires the use of the genuine image, which is not accessible in practice. In the following, these L embeddings on the cover image are referred to as "original embeddings".

The following section presents results for this methodology with publicly available algorithms and images.

3 RESULTS

3.1 Technical difficulties

It should be noted that the following experiments are non trivial in terms of amounts of data to process and store. Indeed, let us assume, as follows in the experimental setup, $L = 200$ repetitions for the image difficulty estimation and $N = 500$ repetitions for the re-embeddings, with an image database of 10000 images such as that of BOWS2 [1]. For the estimation of the image difficulty, this amounts to $10000 \times L = 2 \times 10^6$ image files to process (that is, store and extract features from), while the re-embedding methodology proposed leads to $10000 \times N = 5 \times 10^6$ images to process. Given that the images have an average size of about 50 kB, such an experiment amounts to 350GB of data, not counting the size of the extracted features from each of the images (in the range of 50GB for this experiment only). Needless to say, the data containing the features on which is performed the proposed methodology is challenging to handle.

3.2 Experimental setup

For the following experiments, the 10000 images from the BOWS2 database have been used [1], as well as the 9074 from the BOSS [12] contest, with $L = 200$ repetitions for the estimation of the image difficulty D_I and $N = 500$ repetitions for the re-embeddings. Experiments are carried out on two different large image databases in order to assess the importance of the images on the results. From the following, it is clear that the choice of the

database of images does not really matter (as long as it contains an important number of images). The choice of the stego algorithm leads to more variation in the results, though, as shown below.

For each image, initial embedding rates (supposed to be the embedding rate in the intercepted suspicious image) uniformly selected between 0 and 30% are used.

Re-embeddings follow the same range of rates, leading to final embedding rates R_i between 0 and about 50% for the I_i. Two different stego algorithms are used in the experiments: nsF5 [17, 7] and steghide [8]. The outcome of the methodology varies slightly between the two algorithms tested, although only in the range of the errors estimated. The global behavior remains the same.

In this paper, the model $\mathcal{M}$ used for the regression is an OP-ELM [11] (the toolbox from `http://www.cis.hut.fi/projects/eiml` was used), which is a feedforward neural network using random projections. It has the advantage of performing very well (with similar performances to state of the art Machine Learning techniques such as Support Vector Machines) while keeping a rather low computational time. The OP-ELM optimizes the Mean Square Error. Default parameters (Linear, Sigmoid and Gaussian kernels, 300 maximum number of kernels) have been used for the experiments.

The OP-ELM model $\mathcal{M}$ is used on the 274 DCT-based features extracted from image I_o [5] augmented by the number of non-zero DCT coefficients of the image I_o.

3.3 Estimation of the original embedding rate $\hat{R}_o$

First, Figure 5 illustrates the solution of Eq. 5 for one image only (the behavior is the same for all images for both stego algorithms and image databases): by solving the linear system in a Least Squares sense, the values of $\beta = \frac{1}{A_o}$ (the slope) and $\alpha = \frac{E_o}{A_o}$ (estimated embedding rate for $E_i \longrightarrow 0$) are devised. Here, all $N = 500$ values obtained for each re-embedding are plotted.

In order to show that the minimization problem is correctly solved for the whole range of embedding rates and for all images, Figure 6 represents the estimated value of the original embedding rate $\hat{R}_o$ versus the real value R_o. The actual Normalized Mean Square Error (NMSE) for the 10000 images in this figure (BOWS2, Steghide) using the re-embeddings is 0.0586, while using the same model $\mathcal{M}$ directly on each image (classical quantitative steganalysis, no re-embedding) leads to a 0.0676 NMSE in this case. Performances are similar for nsF5, and over the BOSS images set as well.

Hence, using this methodology on the 10000 images yields on average an improvement of 13% of the NMSE for quantitative steganalysis.

It can be noted that the OP-ELM already performs very well [10] and the nsF5 and Steghide stego problems are "easy enough", hence the difficulty to improve “radically” the performances obtained in the first place. While an improvement of 13% is not very important, it is sufficient to illustrate that this methodology yields on average at least as good results as the classical quantitative steganalysis with no re-embeddings.

To investigate the influence of the number of re-embeddings N, a variable number of re-embeddings has been used to establish Figure 7. It illustrates

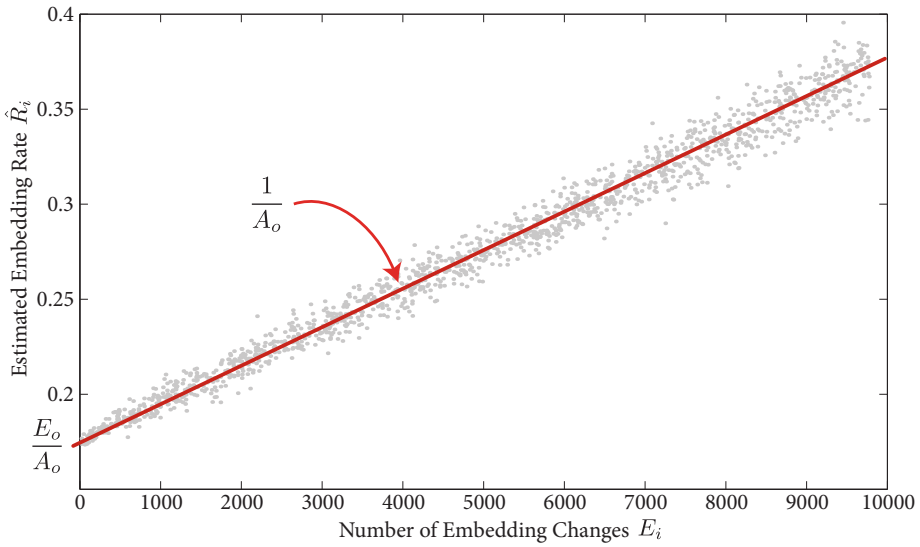

Figure 5: Plot of the estimated embedding rate $\hat{R}_i$ versus the number of embedding changes E_i, for one image, for nsF5 (behavior is identical for Steghide). From Eq. 5, the slope gives the $\beta = \frac{1}{A_o}$ term while the value for $E_i \longrightarrow 0$ gives the $\alpha = \frac{E_o}{A_o}$ term.

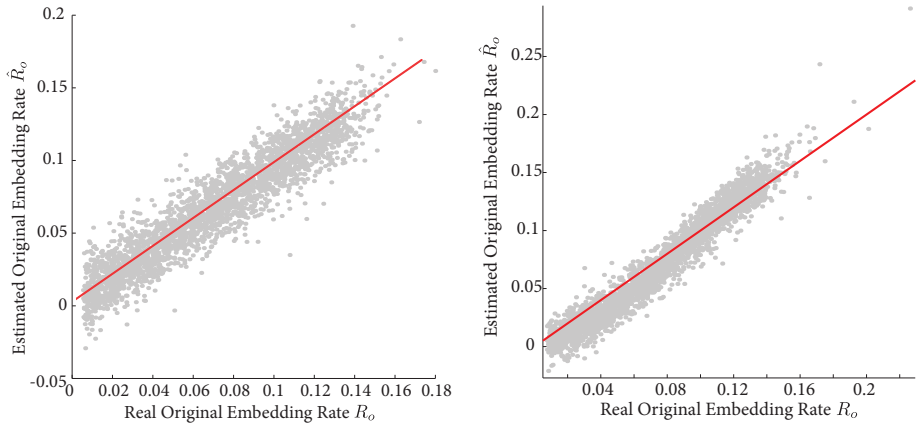

Figure 6: Plot of the estimated original embedding Rate $\hat{R}_o$ through the re-embeddings versus the original R_o, for all 10000 images of BOWS2. Left is for nsF5 and right for Steghide.

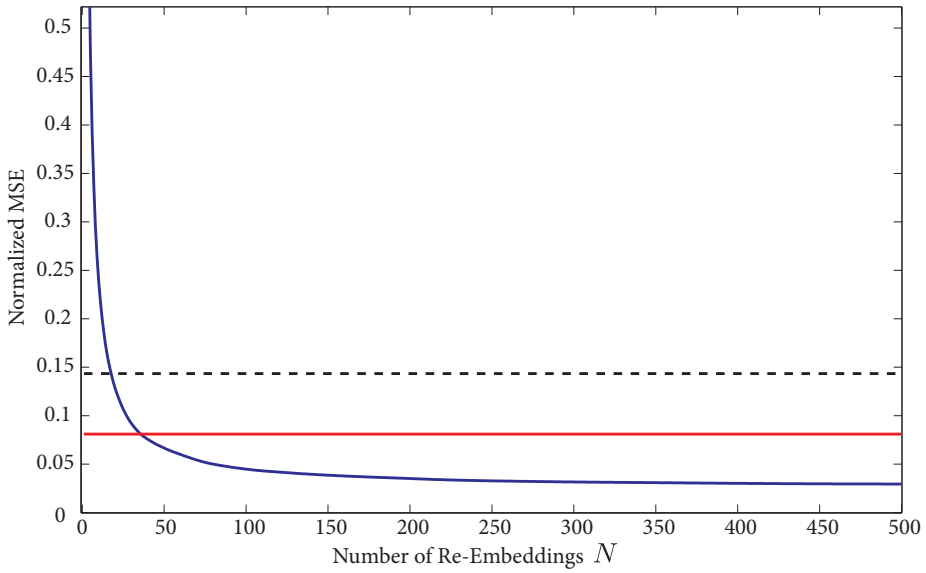

Figure 7: Plot of the Normalized Mean Square Error (NMSE) made on $\hat{R}_o$ versus the number of re-embeddings performed. The solid straight line gives the NMSE using the OP-ELM for classical quantitative steganalysis (no re-embedding), and the straight dashed line the NMSE for an OLS model. This is for nsF5 on a BOSS image.

the evolution of the NMSE using the re-embedding approach, with a varying number of re-embeddings N. It is interesting to note that the error decreases dramatically with the number of re-embeddings N and stabilizes once a certain amount of re-embeddings is used. In most cases (i.e. on most images), the re-embedding approach yields better results than the OLS and OP-ELM used without re-embeddings.

As Figure 8 illustrates, though, it happens for some images that the number of re-embeddings N is insufficient to beat the classical OLS or OP-ELM approach. Globally, the $N = 500$ is in more than 95% of the cases (for both nsF5 and Steghide for BOSS and BOWS2 sets) sufficient to perform better than the OLS or OP-ELM in the classical way.

In fact, once there are enough samples (equations) in the system to solve Eq. 4, new re-embeddings (and hence, new equations in the system) do not provide sufficient additional information for the regression problem. Hence the rather small improvement when the number of re-embeddings N is beyond a certain threshold.

3.4 On the use of the width of the confidence interval

The confidence interval for the experiments has been set to 95% [3], and calculated using the Matlab® `regress` function [4].

Following results make use of the width of the confidence interval on the estimation of $\hat{R}_o$. The goal of this experiment is to establish a dependence between the estimated confidence interval $\left[\hat{R}_o^{INF}, \hat{R}_o^{\text{SUP}}\right]$ for the embedding rate $\hat{R}_o$ and the inner difficulty D_I of the image I considered, in the first

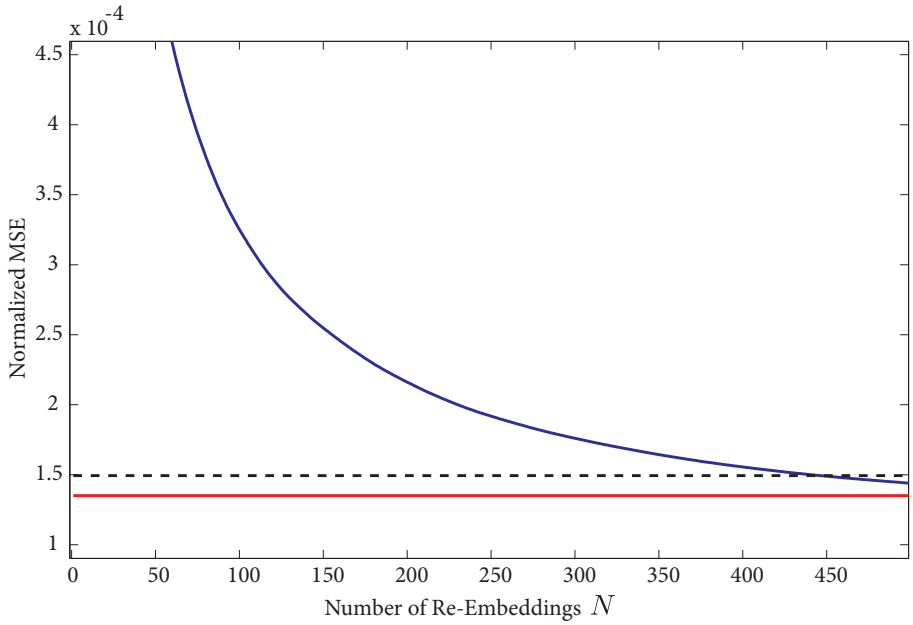

Figure 8: Plot of the Normalized Mean Square Error (NMSE) made on $\hat{R}_o$ versus the number of re-embeddings performed. The solid straight line gives the NMSE using the OP-ELM for classical quantitative steganalysis (no re-embedding), and the straight dashed line the NMSE for an OLS model. This is for Steghide on a BOWS2 image.

place.

Figure 9 is a graph of the standard deviation of the error made on the "original embeddings" $D_I = \text{std}\left(\varepsilon^O\right)$ versus $\hat{R}_o^{\text{SUP}} - \hat{R}_o^{\text{INF}}$ (again, results are so similar over BOSS and BOWS2 images that both plots are shown here).

There appears to be a dependence between the "difficulty" (as estimated by the original embeddings), and the width of the confidence interval estimated by the re-embedding approach. Indeed, one can say that the larger is the estimated confidence interval for $\hat{R}_o$, the larger the probability of the error and therefore the more probable the image is a difficult one.

The high correlation between the difficulty and the confidence interval is not very easy to notice on Figure 9 because of the non-uniform distribution

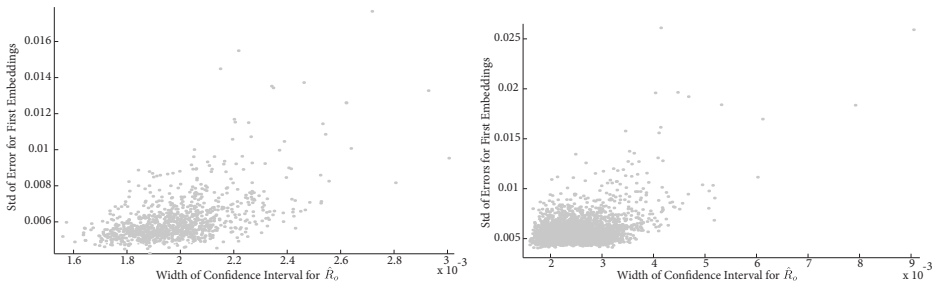

Figure 9: Plot of D_I, the standard deviation of the error made on the L "original embeddings" $D_I = \text{std}\left(\varepsilon^O\right)$ versus the width of the estimated confidence interval $\hat{R}_o^{\text{SUP}} - \hat{R}_o^{\text{INF}}$. Left is nsF5 on BOWS2 and right is Steghide on BOSS.

of the samples along the abscissa. In order to overcome this visualisation drawback, a local average using the 30 nearest neighbors regarding the x-coordinate is computed, the y coordinate being computed by the average of y-coordinates the corresponding points. The result is depicted on Figure 10 where the relation between the estimated confidence interval and the difficulty of the images is straightforward.

Figure 10 shows the evolution of this average versus the width of the estimated confidence interval. In fact, if one considers the cloud of points of Figure 9 as a "flat cone", Figure 10 plots the evolution of the center of the cone. It is then obvious that the larger the estimated confidence interval, the more difficult is the image to handle in steganalysis, in terms of the criterion D_I (inner difficulty).

Finally, Figure 11 shows the evolution of the variance of D_I for the 30 nearest neighbors for each image. The growth shows that the larger the confidence interval, the more difficult it is to have an accurate estimation of the difficulty. From Figures 10 and 11, we can conclude that the probability to get a large D_I is increasing with respect to the width of the calculated confidence interval.

4 CONCLUSION

In this paper, an approach based on multiple re-embeddings is used to estimate in terms of quantitative steganalysis, the original embedding rate (and the number of embedding changes) in an intercepted image. The proposed methodology makes it possible to obtain a reliable estimation of this embedding rate — with a small improvement in terms of accuracy —, along with a confidence interval on this value.

The estimated confidence interval in turn enables the steganalyzer to measure the inherent difficulty of the image (reliability estimation), in terms of classical quantitative steganalysis. Through the width of this confidence interval, it becomes possible to rank the images of a database in terms of their probability of difficulty for quantitative steganalysis, without possessing the genuine images nor having any information on their being stego or genuine.

The proposed methodology has the advantage of being usable for any stego algorithm (given the assumptions made in section 2) and any regression model. Future work will apply this methodology to more stego algorithms (MMX, JPHS, Outguess...) and other regression models, such as SVR. Also, an analysis of the error ε_i (in its relation to the embedding changes E_i and on the assumed independence between the ε_i) could lead to a better modelisation and a more accurate estimation of the embedding rate and hence of inner image difficulty.

ACKNOWLEDGMENTS

The authors would like to thank Tomáš Pevný for his advices on addressing the Quantitative Steganalysis problem.

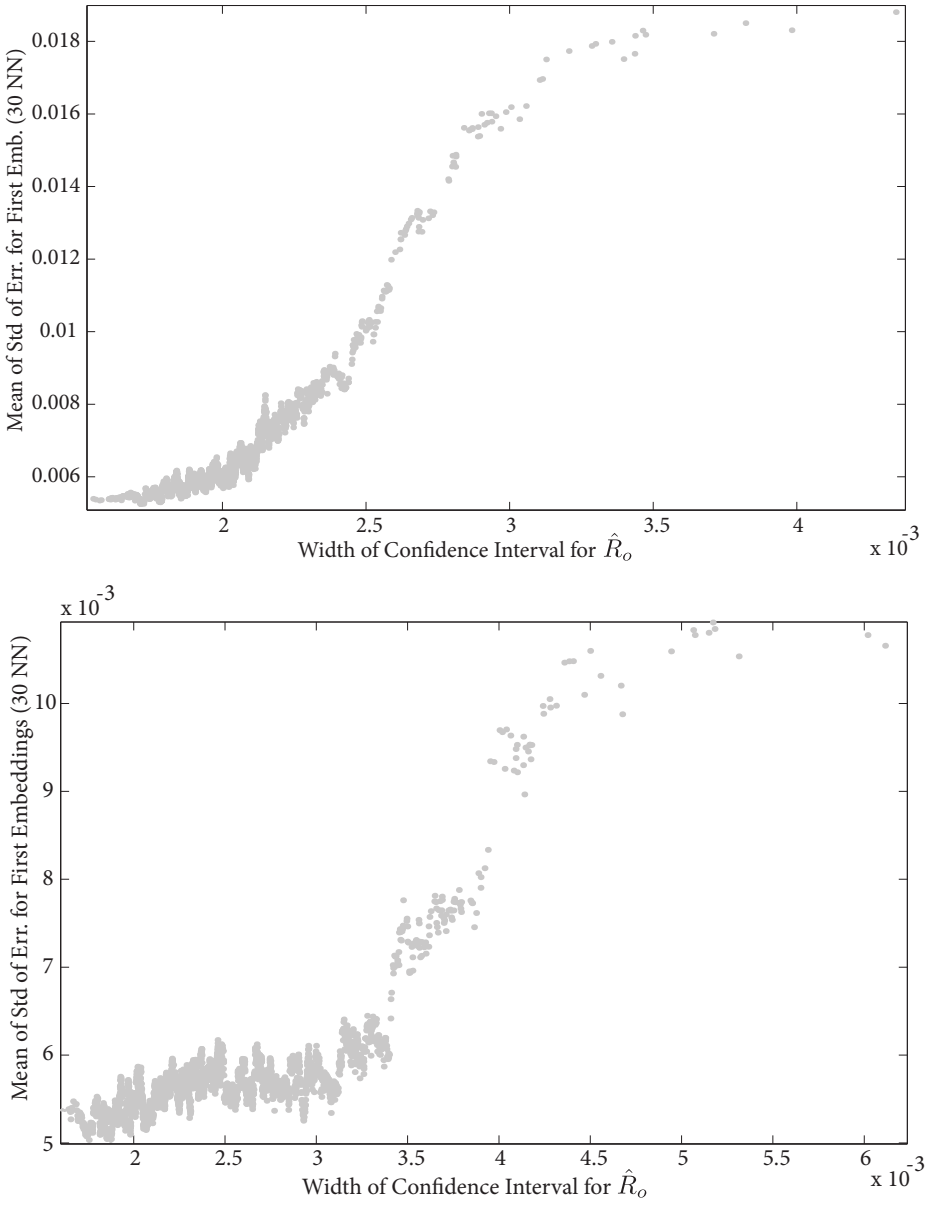

Figure 10: Plot of the mean of D_I for the 30 nearest neighbors (with respect to D_I) versus the width of the estimated confidence interval. Top is nsF5 on BOWS2 and bottom is Steghide on BOSS.

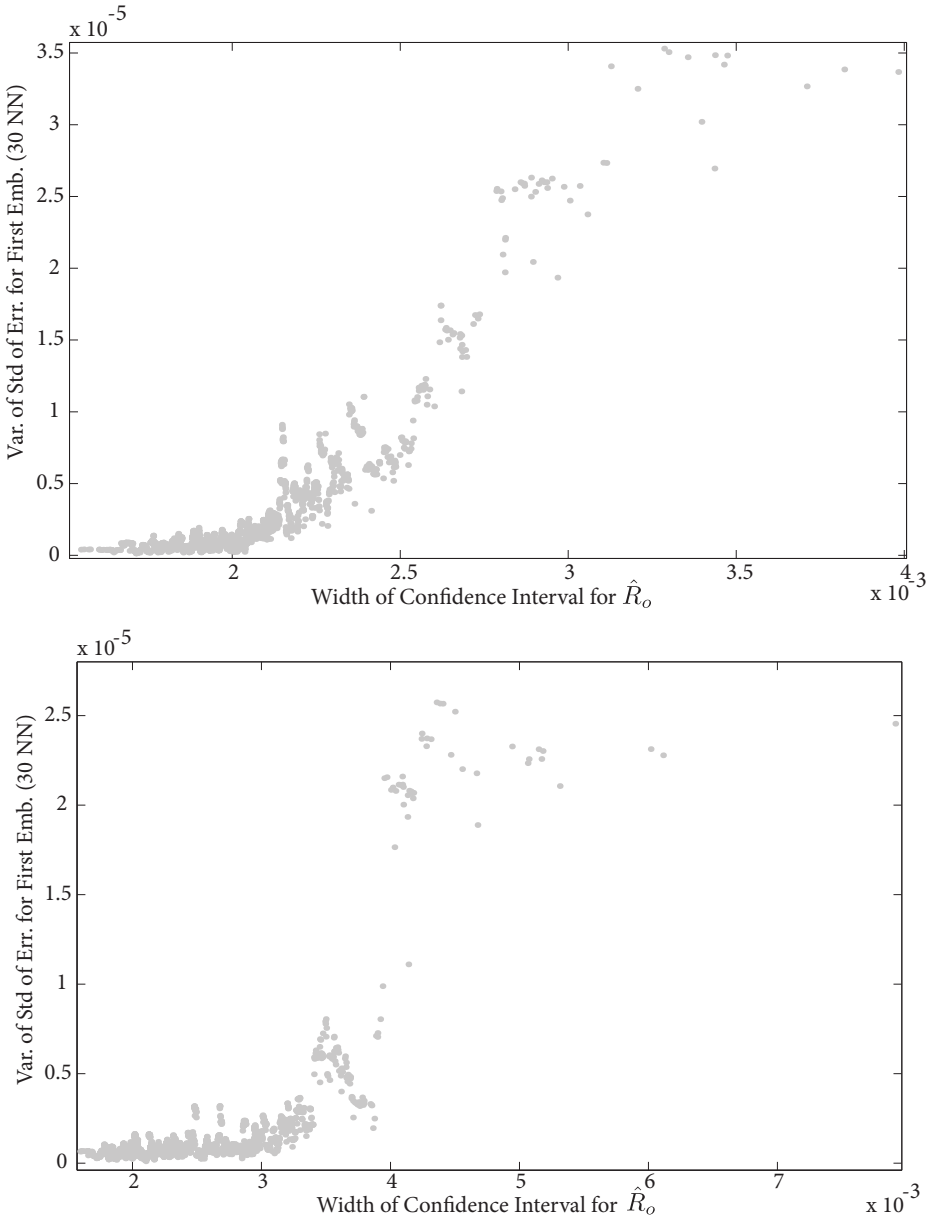

Figure 11: Plot of the variance of D_I for the 30 nearest neighbors (with respect to D_I) versus the width of the estimated confidence interval. Top is nsF5 on BOWS2 and bottom is Steghide on BOSS.

REFERENCES

[1] Patrick Bas and Teddy Furon. BOWS2 challenge: Break our watermarking scheme. ECRYPT European Network of Excellence, `http://bows2.gipsa-lab.inpg.fr/`.

[2] Rainer Böhme and Andrew D. Ker. A two-factor error model for quantitative steganalysis. In Edward J. Delp III and Ping Wah Wong, editors, *Proceedings of SPIE*, volume 6072, page 607206. SPIE, 2006.

[3] Samprit Chatterjee and Ali S. Hadi. Influential observations, high leverage points, and outliers in linear regression. *Statistical Science*, 1(3):379–393, 1986.

[4] Norman R. Draper and Harry Smith. *Applied Regression Analysis, 3rd edition*. Wiley-Interscience, 1998.

[5] Jessica Fridrich. Feature-based steganalysis for jpeg images and its implications for future design of steganographic schemes. In *Information Hiding: 6th International Workshop*, volume 3200 of *Lecture Notes in Computer Science*, pages 67–81, May 23-25 2004.

[6] Jessica Fridrich, Miroslav Goljan, Dorin Hogea, and David Soukal. Quantitative steganalysis of digital images: estimating the secret message length. *Multimedia systems*, 9(3):288–302, 2003.

[7] Jessica Fridrich, Tomáš Pevný, and Jan Kodovský. Statistically undetectable jpeg steganography: dead ends challenges, and opportunities. In *MM&Sec '07: Proceedings of the 9th workshop on Multimedia & security*, pages 3–14, New York, NY, USA, 2007. ACM.

[8] Stefan Hetzl and Petra Mutzel. A graph-theoretic approach to steganography. In Dittmann J., Katzenbeisser S., and Uhl A., editors, *CMS 2005*, Lecture Notes in Computer Science 3677, pages 119–128. Springer-Verlag, 2005.

[9] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, 9:5–38, January 1883.

[10] Yoan Miche, Patrick Bas, Amaury Lendasse, Christian Jutten, and Olli Simula. Reliable steganalysis using a minimum set of samples and features. *EURASIP Journal on Information Security*, 2009(1):1–13 (Article ID 901381), March 2009. http://www.hindawi.com/journals/is/2009/901381.html.

[11] Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten, and Aamaury Lendasse. OP-ELM: Optimally-pruned extreme learning machine. *IEEE Transactions on Neural Networks*, 21(1):158–162, January 2010.

[12] Tomáš Pevný, Tomáš Filler, and Patrick Bas. Boss challenge: Break our steganography system, 2010.

[13] Tomáš Pevný and Jessica Fridrich. Towards multi-class blind steganalyzer for jpeg images. In *International Workshop on Digital Watermarking 2005*, volume 3710 of *LNCS*, pages 39–53, 2005.

[14] Tomáš Pevný and Jessica Fridrich. Multiclass blind steganalysis for jpeg images. In *SPIE Electronic Imaging*, volume 6072, page 60720O, 16-19 January 2006.

[15] Tomáš Pevný, Jessica Fridrich, and Andrew D. Ker. From blind to quantitative steganalysis. In Edward J. Delp III, Jana Dittmann, Nasir D. Memon, and Ping Wah Wong, editors, *Media Forensics and Security*, volume 7254, page 72540C. SPIE, 2009.

[16] Hedieh Sajedi and Mansour Jamzad. Secure steganography based on embedding capacity. *International Journal of Information Security*, 8(6), December 2009.

[17] Andreas Westfeld. F5-a steganographic algorithm. In *IHW '01: Proceedings of the 4th International Workshop on Information Hiding*, pages 289–302, London, UK, 2001. Springer-Verlag.

TKK REPORTS IN INFORMATION AND COMPUTER SCIENCE

TKK-ICS-R24 Timo Honkela, Nina Janasik, Krista Lagus, Tiina Lindh-Knuutila, Mika Pantzar, Juha Raitio
Modeling communities of experts. December 2009.

TKK-ICS-R25 Jani Lampinen, Sami Liedes, Kari Kähkönen, Janne Kauttio, Keijo Heljanko
Interface Specification Methods for Software Components. December 2009.

TKK-ICS-R26 Kari Kähkönen
Automated Test Generation for Software Components. December 2009.

TKK-ICS-R27 Antti Ajanki, Mark Billinghurst, Melih Kandemir, Samuel Kaski, Markus Koskela, Mikko Kurimo, Jorma Laaksonen, Kai Puolamäki, Timo Tossavainen
Ubiquitous Contextual Information Access with Proactive Retrieval and Augmentation. December 2009.

TKK-ICS-R28 Juho Frits
Model Checking Embedded Control Software. March 2010.

TKK-ICS-R29 Miki Sirola, Jaakko Talonen, Jukka Parviainen, Golan Lampi
Decision Support with Data-Analysis Methods in a Nuclear Power Plant. March 2010.

TKK-ICS-R30 Teuvo Kohonen
Contextually Self-Organized Maps of Chinese Words. April 2010.

TKK-ICS-R31 Jefrey Lijffijt, Panagiotis Papapetrou, Niko Vuokko, Kai Puolamäki
The smallest set of constraints that explains the data: a randomization approach. May 2010.

TKK-ICS-R32 Tero Laitinen
Extending SAT Solver With Parity Constraints. June 2010.

TKK-ICS-R33 Antti Sorjamaa, Amaury Lendasse
Fast Missing Value Imputation using Ensemble of SOMs. June 2010.

ISBN 978-952-60-3249-8 (Print)
ISBN 978-952-60-3250-4 (Online)
ISSN 1797-5034 (Print)
ISSN 1797-5042 (Online)

COLOPHON

This thesis was printed by Multiprint Oy.

Final Version as of October 22, 2010 at 10:55.