

The file fragment classification problem: A combined neural
network and linear programming discriminant model approach

E.F. Wilgenbus

20264984

Dissertation submitted in partial fulfilment of the requirements for the
degree *Master of Science in Computer Sciences* at the Potchefstroom
campus of the North-West University

Supervisor: Prof. H.A. Kruger
Co-supervisor: Dr. J.V. du Toit
April 2013

For Peter & Gina

“Essentially, all models are wrong, but some are useful.”

Box and Draper, 1987, p. 424

Acknowledgement

My dissertation is dedicated to my parents, Peter and Gina. No amount of words is enough to thank a father who has generously ensured that his children will have a better life than he had. Likewise, no amount of words is enough to thank a mother who has sacrificed so much of herself in order to ensure her children will have the opportunities she never had. Thank you for giving me the gift of a good education. More than that, thank you for teaching me to believe in myself and to take pride in who I am. Because of the strength and courage you have cultivated in me, I am now able to boldly face the expectations of modern adulthood. All my successes are yours also.

My aunt, Pam, needs more than an acknowledgement — she deserves an award. Remaining patient with me for seven years is no small feat. I thank you for that.

I would like to thank my supervisors, Prof. Hennie Kruger and Dr. Tiny du Toit. I am grateful for your contributions and support.

Finally, though his “contributions” are often few and far between, my brother, Ulrich, holds a special place in my heart. Bolla, I present you with exhibit one: my MSc dissertation. I hope this proves, once and for all, that I am smarter than you. Do not worry about it — I still love you.

Erich

15 April 2013

This work forms part of research done at the North-West University within the Telkom Centre of Excellence research program funded by Telkom and THRIP.

Abstract

The increased use of digital media to store legal, as well as illegal data, has created the need for specialized tools that can monitor, control and even recover this data. An important task in computer forensics and security is to identify the true file type to which a computer file or computer file fragment belongs. File type identification is traditionally done by means of metadata, such as file extensions and file header and footer signatures. As a result, traditional metadata-based file object type identification techniques work well in cases where the required metadata is available and unaltered. However, traditional approaches are not reliable when the integrity of metadata is not guaranteed or metadata is unavailable. As an alternative, any pattern in the content of a file object can be used to determine the associated file type. This is called content-based file object type identification.

Supervised learning techniques can be used to infer a file object type classifier by exploiting some unique pattern that underlies a file type's common file structure. This study builds on existing literature regarding the use of supervised learning techniques for content-based file object type identification, and explores the combined use of multilayer perceptron neural network classifiers and linear programming-based discriminant classifiers as a solution to the multiple class file fragment type identification problem.

The purpose of this study was to investigate and compare the use of a single multilayer perceptron neural network classifier, a single linear programming-based discriminant classifier and a combined ensemble of these classifiers in the field of file type identification. The ability of each individual classifier and the ensemble of these classifiers to accurately predict the file type to which a file fragment belongs were tested empirically.

The study found that both a multilayer perceptron neural network and a linear programming-based discriminant classifier (used in a round robin) seemed to perform well in solving the multiple class file fragment type identification problem. The results of combining multilayer perceptron neural network classifiers and linear programming-based discriminant classifiers in an ensemble were not better than those of the single optimized classifiers.

Keywords: *file type identification, file fragment type identification, multilayer perceptron neural network, linear programming-based discriminant analysis, ensembles, classification*

Uittreksel

Die toenemende gebruik van digitale media om wettige data, sowel as onwettige data te berg skep die behoefte aan gespesialiseerde tegnieke om die data te monitor, te beheer en selfs te herwin. 'n Belangrike taak in rekenaar forensiese ondersoeke en rekenaarsekuriteit is die identifisering van 'n rekenaarlêer of lêerfragment se lêerformaat. Rekenaarlêerformaat-identifikasie word tradisioneel gedoen deur gebruik te maak van metadata soos lêernaamuitbreidings en die lêer se kop- en voetsegmenthandtekeninge. Hierdie tradisionele metadatagebaseerde lêerformaat-identifikasietegnieke werk goed in gevalle waar die vereiste metadata beskikbaar en onveranderd is. Wanneer die integriteit van die metadata nie gewaarborg is nie of die nodige metadata nie beskikbaar is nie, kan die tradisionele benaderings nie vertrou word nie. As 'n alternatief kan enige patroon in die inhoud van 'n lêervoorwerp gebruik word om die gepaardgaande lêerformaat vas te stel. Hierdie benadering staan bekend as inhoudgebaseerde lêervoorwerpformaat-identifikasie.

Gekontroleerde-leertegnieke kan gebruik word om 'n lêervoorwerpformaatklassifiseerder af te lei deur die unieke patroon wat 'n rekenaarlêerformaat onderlê te ontgin. Hierdie studie bou voort op bestaande literatuur met betrekking tot die gebruik van gekontroleerde-lêertegnieke vir inhoudgebaseerde lêervoorwerpformaat-identifikasie, en verken die gesamentlike gebruik van multilaag-perseptron neurale netwerkklassifiseerders en lineêre programmeringgebaseerde diskriminant-klassifiseerders as 'n oplossing vir die meervoudige lêerfragmentformaat-identifikasieprobleem.

Die doel van hierdie studie was om die gebruik van 'n enkele multilaag-perseptron neurale netwerk, 'n enkele lineêre programmeringgebaseerde diskriminantklassifiseerder, sowel as 'n gekombineerde ensemble van hierdie klassifiseerders in die gebied van die lêerformaat-identifikasie te bestudeer en te vergelyk. Die vermoë van die individuele klassifiseerders en die ensemble van klassifiseerders om die lêerformaat van 'n rekenaarlêerfragment akkuraat te voorspel, is empiries getoets.

Hierdie studie het bevind dat beide 'n multilaag-perseptron neurale netwerk en 'n lineêre programmeringgebaseerde diskriminantklassifiseerder (gebruik in 'n rondomtalie) goed presteer met betrekking tot die meervoudige klas lêerfragmentformaat-identifikasieprobleem. Die akkuraatheid waargeneem vir 'n kombinasie van multilaag-perseptron neurale netwerke en 'n lineêre programmeringgebaseerde diskriminantklassifiseerders in 'n ensemble was nie beter as dié van die individuele geoptimaliseerde klassifiseerders nie.

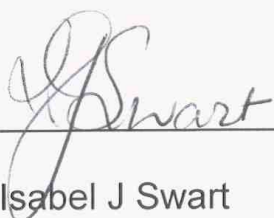
Sleutelwoorden: *rekenarlêerformaatidentifisering, rekenarlêerfragmentformaatidentifisering, lêerfragmentformaatidentifisering, multilaag-perseptron neurale netwerk, lineêre programmeringgebaseerde diskriminantklassifiseerder, ensembles, klassifikasie*

This serves to confirm that I, Isabella Johanna Swart, registered with and accredited by the South African Translators' Institute as translator, registration number 1001128, language edited the following dissertation.

The file fragment classification problem: A combined neural network and linear programming discriminant model approach

by

EF (Erich) Wilgenbus



Dr Isabel J Swart



Date

23 Poinsettia Close
Van der Stel Park
Dormehlsdrift
GEORGE
6529
Tel: (044) 873 0111
Cell: 082 718 4210
e-mail: isaswart@telkomsa.net

Contents

Contents	i
List of Figures	iv
List of Tables	v
List of Algorithms	vi
List of Selected Terms	vii
List of Acronyms	x
Chapter I – Background, introduction and problem statement	1
1.1 Machine learning: supervised vs. unsupervised	2
1.2 Problem statement	4
1.3 Research objectives	4
1.4 Research methodology	5
1.5 Dissertation outline	6
1.6 Chapter summary	7
Chapter II – Supervised learning for file object type identification	8
2.1 Supervised learning	8
2.1.1 The classification problem	10
2.2 The neural network classification model	12
2.2.1 Perceptrons and multilayer perceptrons	13
2.2.2 The back propagation algorithm	16
2.2.3 Learning and generalization	19
2.2.4 Advantages and disadvantages of neural networks	23
2.3 Linear discriminant classification model	24
2.3.1 Linear programming	25
2.3.2 Simplex algorithm	26
2.3.3 Linear programming-based discriminant classifiers	30
2.3.4 Advantages and disadvantage of LPDA	33
2.4 k -nearest neighbour algorithm	34

2.5	Ensemble techniques	35
2.5.1	Techniques for generating diverse classifiers	37
2.5.2	Techniques to combine the output of multiple classifiers	39
2.5.3	Advantages and disadvantages of ensembles	40
2.6	Chapter summary	41
Chapter III – Literature review of file object type identification		42
3.1	Instance under observation	42
3.2	The input pattern	43
3.3	The size of the classification problem	45
3.4	Exemplar set and file types	47
3.5	Model representation	48
3.5.1	k -nearest neighbour algorithm	49
3.5.2	Discriminant analysis	52
3.5.3	Neural networks	53
3.5.4	Support vector machines	54
3.5.5	Other approaches	55
3.6	Chapter summary	55
Chapter IV – Research Methodology and Experimental Design		56
4.1	Feature selection by mutual information	56
4.2	Hyper-parameter optimization	59
4.3	Cross-validation	61
4.4	Model evaluation measures	63
4.4.1	Confusion matrix	63
4.4.2	Precision and recall	64
4.4.3	Model accuracy	64
4.5	Model comparison	65
4.6	Experimental design	66
4.6.1	Data preparation: The data set	66
4.6.2	Data preparation: The input pattern	68
4.6.3	Data preparation: Feature selection	70
4.6.4	Data mining: The experiments	70
4.7	Chapter summary	72
Chapter V – Results and Discussion		73
5.1	A note on hyper-parameter optimization	73
5.2	Experiment 1: k -nearest neighbour	74
5.2.1	Hyper-parameter selection	75
5.2.2	Model evaluation	76
5.3	Experiment 2: Single MLP neural network classifier	77

5.3.1	Hyper-parameter selection	77
5.3.2	Model evaluation	78
5.4	Experiment 3: LP-based discriminant classifiers in round robin	79
5.4.1	Hyper-parameter selection	80
5.4.2	Model evaluation	81
5.5	Experiment 4: Ensembles	82
5.5.1	Experiment 4a: Ensemble of multiple MLP neural networks	82
5.5.2	Experiment 4b: Ensemble of multiple LP-based discriminant classifiers	82
5.5.3	Experiment 4c: Combining multiple MLP neural network and multiple LP-based discriminant classifiers	83
5.5.4	Model evaluation	83
5.6	Model comparison and conclusion	87
5.7	Chapter summary	92
Chapter VI – Summary and Conclusion		93
6.1	Research objectives	93
6.2	Research methodology	95
6.3	Research results	96
6.4	Research limitations	96
6.5	Future research	97
6.6	Chapter summary	98
Appendix A – Related research		99
Appendix B – The simplex algorithm: An example		113
Appendix C – “Greedy” mutual information-based feature ranking: An example		119
Appendix D – Example byte frequency histograms		122
Bibliography		129

List of Figures

Figure 1.1	KDDM process model	5
Figure 2.1	The classification problem	11
Figure 2.2	A biological neuron	12
Figure 2.3	The perceptron	14
Figure 2.4	The multilayer perceptron neural network	14
Figure 2.5	Fitting a polynomial function	20
Figure 2.6	Learning and generalization	21
Figure 2.7	Discriminant analysis	25
Figure 2.8	The k -nearest neighbour algorithm	35
Figure 2.9	Three fundamental reasons for ensemble learning	36
Figure 4.1	Grid vs. random search	60
Figure 4.2	Process followed to construct base the data set	68
Figure 4.3	The byte frequency histogram of a pdf file fragment.	69
Figure 4.4	Process followed to construct the data set used for feature ranking	71
Figure 5.1	Hyper-parameter search: k -nearest neighbours	75
Figure 5.2	Hyper-parameter search: LPDC	81
Figure 5.3	Empirical results: Accuracy of each ensemble	84
Figure 5.4	Empirical results: Profile of misclassified fragments in each ensembles	85
Figure 5.5	Empirical results: Ensemble diversity	86
Figure 5.6	Empirical results: Accuracy of each experiment	87
Figure 5.7	Empirical results: Accuracy of all experiments per file type	88
Figure 5.8	Empirical results: Average precision and recall per file type	89
Figure 6.1	KDDM process model	95

List of Tables

Table 2.1	Example code matrix used in error correction output coding	39
Table 3.1	Literature summary: File object under observation	43
Table 3.2	Literature summary: Attributes used to describe a file object	43
Table 3.3	Literature summary: Formulation of the classification problem	46
Table 3.4	Literature summary: File types	47
Table 3.5	Literature summary: Source of exemplar file object sources	48
Table 3.6	Literature summary: Model representations	48
Table 3.7	Literature summary: Instance selection	50
Table 3.8	Literature summary: Distance measures	50
Table 3.9	Literature summary: k -nearest neighbour results	51
Table 3.10	Literature summary: Discriminant analysis results	52
Table 3.11	Literature summary: Neural network results	53
Table 3.12	Literature summary: Support vector machine results	54
Table 4.1	Example of a confusion matrix	64
Table 4.2	Descriptive statistics: File set	67
Table 4.3	Descriptive statistics: File fragments set	68
Table 4.4	Feature ranking by mutual information.	71
Table 5.1	Empirical results: Confusion matrix for the 1-nearest neighbour algorithm	76
Table 5.2	Hyper-parameter search: MLP neural network classifier	78
Table 5.3	Empirical results: Confusion matrix for the MLP neural network classifier	79
Table 5.4	Empirical results: Confusion matrix for the composite LPDC	81
Table 5.5	Empirical results: Improvement over best individual classifier	87
Table 5.6	Empirical results: Two tailed t -test	90

List of Algorithms

Algorithm 2.1	Back propagation algorithm for batch learning	18
Algorithm 2.2	The simplex algorithm	30
Algorithm 4.1	Mutual information-based feature ranking	58
Algorithm 4.2	k -Fold cross validation for model <i>selection</i>	62
Algorithm 4.3	k -Fold cross validation for model <i>evaluation</i>	63
Algorithm 5.1	Linear programming-based discriminant classifiers in round robin . .	80

List of Selected Terms

classification function see classification problem.

classification problem the problem to find a function that can be used to assign a class label, from a predefined set of class labels, to an instance for which the class label is not known. Such a function is called a classification function or a classifier.

classifier see classification problem.

content-based file object type identification the file type to which a computer file object belongs is identified by looking for known patterns in the content of a file object. These patterns are learned using machine learning techniques.

data mining the activity in which machine learning techniques are applied to find patterns in the relationship between data elements. The data mining activity is one step in the knowledge discovery process.

empirical loss the average difference in the predicted dependent variable value and the actual dependent variable value for a given hypothesis function over the exemplar data set, where the difference is measured by some loss function. It is the estimate of the generalization loss..

ensemble a collection of hypothesis functions used together to make a collective decision.

file a sequence of bytes stored on digital media.

file fragment contiguous piece of data taken from a file.

file object collectively refers to files and file fragments.

file type the name that describes some common data structure found in files.

generalization loss the expected difference in the predicted dependent variable value and the actual dependent variable value for a given hypothesis function over all possible input-output pairs, where the difference is measured by some loss function.

hyper-parameters determine the region of the hypothesis space in which the induction algorithm operates (model complexity parameters), as well as fine-tune the induction algorithm used to search the hypothesis space (algorithmic parameters).

hyper-parameter optimization process in which different model specifications, as defined by the model complexity hyper-parameters, in the chosen family of model representations are considered. Also considers different algorithmic parameters. Implemented as an outer loop to the parameter search. Referred to as hyper-parameter search.

hypothesis function a function that maps predictor variables into a predicted variable, refers to regression functions as well as classification functions.

induction algorithm an algorithm that takes as input a set of exemplar instances and produces a hypothesis function that generalizes beyond these instances, also referred to as a learning algorithm or a training algorithm.

knowledge discovery and data mining process set of processing steps which seeks to gain insight into the relationship between data elements.

loss function some measure of the predicted dependent variable value and the actual dependent variable value.

machine learning the scientific discipline concerned with the design and development of algorithms that allow computers to infer knowledge from input data.

metadata refers to the descriptive information available for a given file, such as file extensions and file header and footer signatures.

metadata-based file object type identification the file type to which a computer file object belongs is identified by comparing the available metadata in the file object to an operating system managed metadata dictionary.

model accuracy a model evaluation measure defined as the number of correct predictions made by a model over the total number of instances in a data set.

model selection see hyper-parameter optimization.

parameter defines the best-fitting model with a given model representation.

parameter optimization performed by some induction algorithm to find those parameters, which optimize the evaluation criteria over the exemplar data set. Also referred to as parameter search or model induction.

supervised learning supervised learning is the task of fitting a function to a set of labelled data instances.

unsupervised learning the problem of trying to find hidden structures in a set of unlabelled data instances.

List of Acronyms

BFH byte frequency histogram.

KDDM knowledge discovery and data mining.

LP linear programming.

LPDA linear programming-based discriminant analysis.

LPDC linear programming-based discriminant classifier.

MLP multilayer perceptron.

BACKGROUND, INTRODUCTION AND PROBLEM STATEMENT

I

Today computer technology plays an increasingly important role in the processing, storage and transmission of data. As such, digital media has become the preferred data and information storage medium. Data is stored on digital media as a sequence of bytes in logical units called files (Tanenbaum, 2008). The data contained in a file is organized within some data structure. Applications that know the details with regard to the data structure used within a file, are able to read data from the file as well as write data to the file. The file type associated with a file is the name that describes such a common data structure. Examples of file types include `html`, `jpeg`, `xls` and `doc` file formats.

The increased use of digital media to store legal, as well as illegal data, has created the need for specialized tools that can monitor, control and even recover this data (Fetherston and Gollins, 2012; Pal and Memon, 2009). File object type identification is a recurrent task performed by security, network and digital forensic specialists that can benefit from such specialized tools. File object type identification, comprising of file type identification and file fragment type identification, involves identifying the file type to which a computer file object belongs. This is an important task because the file type associated with a file object draws a connection between the file object and the application that can effectively use such a file object. Furthermore, it is important to know the file type a file object belongs to in order to decide how to use, or not to use, this file object.

File type identification is traditionally done by means of metadata such as file extensions and file header and footer signatures (McDaniel, 2001). The file type to which a computer file belongs is identified by comparing the available metadata in the file to an operating system managed metadata dictionary. The metadata used for traditional file type identification can, however, be altered by malicious software or end users in an attempt to circumvent security and control measures. Traditional file type identification is impeded further as file header and footer metadata is not available in a growing number of file types.

File fragment type identification is done by assigning a file fragment to the file type associated with its parent file. The operating system keeps track of the file fragments that make up a specific file in a file system table (Tanenbaum, 2008). This file system table is queried to link a specific file fragment to its parent file. When the file system table is no longer available it is not possible to infer the file type a file fragment belongs to from the parent file. As a file fragment can originate from any location in the file, header and footer metadata are most likely not available to support file fragment type identification either.

As a result, traditional metadata-based file object type identification techniques work well in cases where the required metadata is available and unaltered. However, traditional approaches are not reliable when the integrity of metadata is not guaranteed or metadata

is unavailable. Security, network and digital forensic specialists need tools with which to determine a file object's true file type that do not rely on the availability and integrity of metadata. As an alternative, any patterns in the content of a file object can be used to determine the associated file type (McDaniel, 2001). This is called content-based file object type identification.

Content-based identification tools can be used to verify the integrity of available metadata, as well as assign a file type to a file object for which no metadata is available. Furthermore, content-based identification tools can help identify important data within memory or disk images (Conti *et al.*, 2010); help identify misrepresented data (Roussev and Garfinkel, 2009), as well as help guide the reconstruction of files (Pal and Memon, 2009; Garfinkel, 2007). Furthermore, content-based file fragment identification tools have applications in network services, such as routing, firewalls and intrusion detection (Moore and Zuev, 2005).

Existing research with regard to content-based file object type identification applies supervised machine learning techniques to find the unique patterns that underlie a file type's common file structure (Harris, 2007). It is the use of supervised machine learning techniques, more specifically the combined use of neural network classifiers and linear programming-based discriminant classifiers, applied to the domain of file object type identification that is of interest in this dissertation.

The purpose of this chapter is to guide the reader through the dissertation. A background to the research problem has already been given. Machine learning is defined in Section 1.1 before supervised learning is distinguished from unsupervised learning. The neural network classifier and linear programming-based discriminant classifier are briefly introduced as part of the discussion with regards to supervised learning. A problem statement and research objectives for this dissertation is formulated in Section 1.2 and Section 1.3 respectively. The research methodology employed is outlined in Section 1.4. The layout of the study, explaining the purpose of each chapter, is presented in Section 1.5.

1.1 Machine learning: supervised vs. unsupervised

Machine learning is the scientific discipline concerned with the design and development of algorithms that allow computers to infer knowledge from input data. The knowledge inferred is the novel and potentially useful relationships between data elements (Fayyad *et al.*, 1996). Within the scope of machine learning, supervised and unsupervised techniques can be distinguished (Maimon and Rakach, 2010; Kotsiantis *et al.*, 2006).

Unsupervised learning refers to the problem of trying to find hidden structures in a set of unlabelled data instances (Maimon and Rakach, 2010). That is, previously unlabelled data instances are labelled by finding groups of homogeneous instances within the data (Russel and Novig, 2010). Homogeneity is usually measured by some distance measure. For example, the k -means algorithm iteratively assigns instances to k clusters within the

feature space. The objective of the algorithm is to find a clustering in which the total distance between a cluster centroid and all the instances assigned to a cluster is minimized.

Conversely, supervised learning is the task of fitting a function to a set of labelled data instances (Kotsiantis *et al.*, 2006). The class labels can be continuous or discrete values. If the class label is continuous, then the supervised learning problem is called a regression problem. A function that maps a data instance into a continuous real valued class variable is called a regression function. If the class label is discrete, then the supervised learning problem is called a classification problem or pattern recognition problem (Fayyad *et al.*, 1996; Michie *et al.*, 1994). The objective of a classification problem is to find a function that can be used to assign a class label, from a predefined set of class labels, to an instances for which the class label is not known. Such a function is called a classification function or a classifier. For example, suppose a set of labelled instances that belong to two groups, say group A and group B, is given; then supervised learning can be used to derive a classifier that will classify some previously unlabelled instance as belonging to either group A or group B.

The file fragment identification problem is an example of this more general classification problem. Supervised learning techniques can therefore be used to infer a file object type classifier by exploiting some unique pattern that underlie a file type’s common file structure. The derived classifier can then be used to classify file objects into one of the predetermined file types. Neural network classifiers and linear discriminant classifiers are two supervised learning techniques that can be used in this regard.

A neural network constitutes a non-linear classification function (Zhang, 2010). The multilayer perceptron (MLP) is a neural network with a specific topology. Such a neural network is structured as three or more distinct layers: an input layer, one or more hidden layers, and an output layer. Each layer constitutes a set of nodes, with nodes from different layers being highly interconnected in a feed-forward manner. All nodes from the input layer are connected to all nodes in the first hidden layer, all nodes in the first hidden layer are connected to all nodes in the second hidden layer, and so forth, until all nodes in the last hidden layer is connected to all nodes in the output layer. The neural network maintains a set of connection weights used to adjust the input signal as it propagates through the network towards the output layer. During network training, the weights are adjusted to minimizes the extent of the misclassification in a exemplar data set.

Linear discriminant classifiers are defined by the discriminant boundary that optimally separates two groups of objects in a feature space. Linear programming models can be used to derive such a discriminant boundary (Lame and Moy, 1997). Linear programming is a technique that identifies a combination of variable values in a linear system of equations which optimize some linear objective function. In this case, linear programming is used to find the set of weights that defines the linear discriminant boundary which minimizes the extent of the misclassification in an exemplar data set (Olson and Shi, 2006).

Existing research with regards to content-based file object type identification focuses

on using a single classifier to assign file types to file objects. Alternatively, various individual classifiers can be combined in an ensemble with the hope that the collective decision made would be more accurate than the decisions made by the individual classifiers (Kuncheva, 2004). The usefulness of an ensemble of MLP neural network classifiers and linear programming-based discriminant classifiers in the domain of file fragment identification is investigated in this dissertation.

1.2 Problem statement

The file fragment type identification problem involves identifying the true file type to which a computer file fragment belongs. This study builds on existing literature regarding the use of supervised learning techniques for content-based file object type identification, and explores the combined use of MLP neural network classifiers and linear programming-based discriminant classifiers as a solution to the file fragment type identification problem. The ability of an ensemble of these classifiers to accurately predict the file type to which a file fragment belongs is tested empirically.

The research hypothesis of this dissertation is as follows: more accurate file fragment type identification can be achieved by using MLP neural network classifiers and linear programming-based discriminant classifiers in an ensemble instead of using these classifiers individually.

1.3 Research objectives

Though the capacity of supervised learning algorithms to solve the file type classification problem has been shown in other literature, using an ensemble of classifiers to solve the file fragment classification problem is somewhat unique. As such, the primary research objective of this project is to describe how MLP neural network classifiers and linear programming-based discriminant classifiers can be used in an ensemble to solve the file fragment type classification problem.

The secondary objectives, which will contribute towards achieving the primary objective, are as follows:

1. Give a brief overview of supervised learning as relevant to classification problems.
2. Describe the MLP neural network classifier and linear programming-based discriminant classifier.
3. Give an overview of literature relating to file type and file fragment type identification.

4. Outline how MLP neural network classifiers and linear programming-based discriminant classifiers can be used individually and in an ensemble to solve the file fragment type identification problem.
5. Empirically test the effectiveness of individual classifiers and ensembles in the domain of file fragment type identification.
6. Compare the classification accuracy of classifiers used in the existing literature to the classification accuracy of the proposed ensembles.

1.4 Research methodology

The activity in which machine learning techniques are applied to find patterns in the relationship between data elements is called data mining (Fayyad *et al.*, 1996). The data mining activity is one step in the knowledge discovery process (Kurgan and Musilek, 2006). This process seeks to gain insight into the relationship between data elements. In this dissertation the relationship between file fragments and their associated file types is of interest.

To provide a better understanding of a knowledge discovery endeavour a general process model is useful. Such a process model consists of a set of processing steps needed to complete a knowledge discovery and data mining (KDDM) project (Kurgan and Musilek, 2006). Various process models have been proposed. Kurgan and Musilek (2006) surveyed some popular KDDM process models and proposed the synthesized model depicted in Figure 1.1. The process model consists of six steps and several feedback loops.

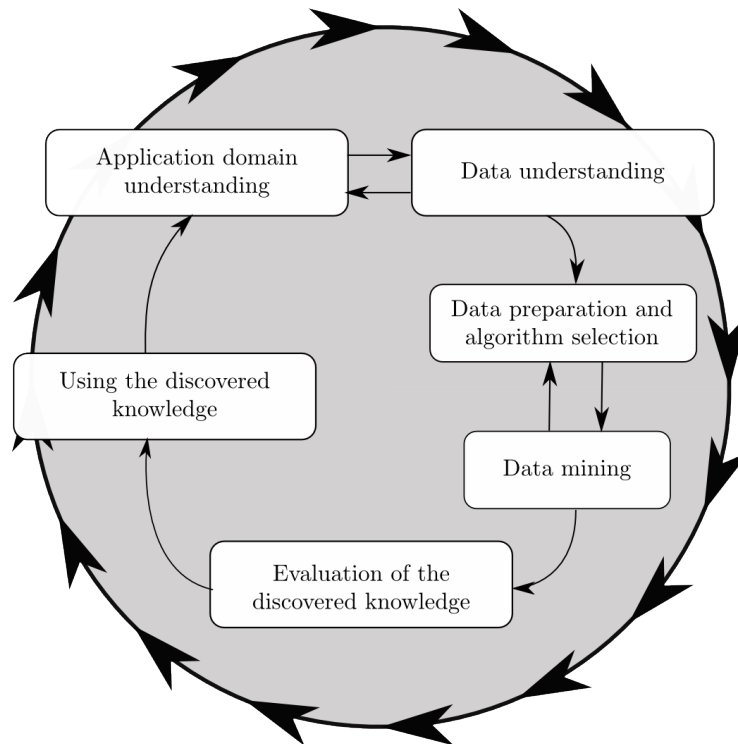


Figure 1.1: The KDDM process model as proposed by Kurgan and Musilek (2006).

In the first step of the KDDM process, a general understanding of the application domain and the relevant prior knowledge is developed. During this step the data mining problem and the objectives of the knowledge discovery and data mining endeavour are defined. The second step involves the identification and acquisition of appropriate data sources, data exploration, data sampling, as well as the selection of appropriate, relevant and interesting attributes. Data preparation, the third step, involves the preprocessing of the data set into the correct structure and form for use with the selected machine learning technique. During this step the appropriate machine learning technique or combination of machine learning techniques are identified in line with the data mining objectives set out during step one. Step four, the data mining step, involves the application of the selected machine learning techniques to the prepared data. In the context of the data mining objectives, the usefulness of the discovered patterns is evaluated and any alternative actions needed are identified during the fifth step. Useful knowledge learned is deployed for practical use in the final step.

The work presented in this dissertation represents many iterations through this KDDM process aimed at discovering useful patterns in the relationship between a file fragment's content and the file fragment's associated file type. This chapter already contributes to this process by providing a background of the application domain, by defining the data mining problem and objectives, and by briefly introducing the machine learning techniques used. The remainder of the dissertation is organized into three phases: a literature study phase, an experimental design phase and an empirical study phase.

An overview of supervised learning techniques as applied to the file object type identification problem, as well as an overview of the MLP neural network classifier and the linear programming discriminant classifier comprise the literature study. The literature study contributes to the KDDM process by not only providing an understanding of the application domain but also of the machine learning techniques applied. The literature study phase contributes towards steps one, two and three of the data mining process.

The approach followed to test the research hypothesis is outlined in the second phase of the dissertation. Experimental design includes a discussion with regard to the data set used, feature selection techniques applied, model selection techniques applied, as well as model evaluation and comparison methods chosen. The experimental design phase contributes towards steps two, three and four of the data mining process. The data mining activity is described in terms of four experiments.

In the last phase, the empirical results of the data mining endeavour are given, evaluated and interpreted. This phase contributes towards steps five and six of the data mining process.

1.5 Dissertation outline

This section outlines the structure of the dissertation and explains the purpose and content of each chapter.

A brief general overview of supervised learning as relevant to classification problems is given in Chapter 2. This chapter includes a review of the MLP neural network classifier and the linear programming-based discriminant classifier as these techniques are applied in later chapters. Ensemble techniques are also outlined in Chapter 2. Many principles referred to in Chapter 3 are described in Chapter 2.

In Chapter 3 prior work with regard to file type and file fragment type identification is presented. Whereas previous authors have outlined selective literature chronologically, this chapter organizes literature systematically under the various components of a supervised learning problem as identified in Chapter 2. By doing so, the diversity in literature is not only highlighted, but the various similarities and differences in the literature become apparent.

In Chapter 4 the research methodology used is outlined. This chapter includes a discussion regarding the data set used, feature selection techniques applied, model selection techniques applied, as well as model evaluation and comparison methods chosen. In Chapter 4 various models are formulated as experiments.

The empirical results of the experiments defined in Chapter 4 are presented in Chapter 5. The empirical study is evaluated and discussed.

Finally, in Chapter 6 the research findings are summarized, important findings are highlighted and research limitations are listed. A recommended path for future work is given, after research contributions are evaluated against the objectives set forth in Chapter 1.

1.6 Chapter summary

Chapter 1 served to provide background to this dissertation. A computer file and a computer file type were defined, whereafter content-based file object type identification was distinguished from traditional file object type identification. It was noted that the lack of integrity and the unavailability of traditionally used metadata created the need for content-based file object type identification tools.

The problem statement for the file fragment type identification problem was provided. Furthermore, the file fragment type identification problem has been identified as belonging to the group of data mining problems known as classification problems. These classification problems can be solved by using supervised learning techniques. Two specific supervised learning techniques have been introduced: an MLP neural network classifier and a linear programming-based discriminant classifier.

The KDDM process model, used to guide the knowledge discovery endeavour in this dissertation, has been outlined as part of the research methodology. This chapter already contributes towards the process by providing some insight into the data mining problem and objective, as well as the application domain. The required prior knowledge is further developed in Chapter 2.

SUPERVISED LEARNING FOR FILE OBJECT TYPE IDENTIFICATION

II

In Chapter 1 it was noted that the file fragment identification problem is an example of the more general data mining problem known as classification. Classification involves determining a function that maps an object into one of several predefined classes (Fayyad *et al.*, 1996). As such, a supervised learning algorithm can be used to infer a classification function from a set of example file objects for which the file type is known. This classification function can then be used to assign file types to a file object for which a file type is not known.

Supervised learning and the classification problem are formally defined in Section 2.1. The chapter continues to explore two specific supervised learning techniques: the multilayer perceptron (MLP) neural network classifier in Section 2.2 and the linear programming-based discriminant classifier in Section 2.3. In the discussion of both these techniques the underlying model is outlined, the associated induction algorithm is discussed and some advantages, as well as disadvantages associated with the classifier are listed. The k -nearest neighbour supervised learning technique is also briefly introduced in Section 2.4. This technique is outlined here because of its popularity in existing literature with regard to file object type identification as will be shown in Chapter 3, and because this technique is later proposed for use as a benchmark classifier in Chapter 4. Ensemble techniques are described in Section 2.5 for the reason that this dissertation focuses on combining a set of classifiers in an attempt to improve classification accuracy.

2.1 Supervised learning

The goal of supervised learning is to derive a mapping from a set of input variables to an output variable using a set of example input-output variable pairs. The resulting mapping can then be used to infer an output variable value for instances where the input variables are known, but the value of the output variable is not known (Kotsiantis *et al.*, 2007).

Formally, given a set of N exemplar tuples $(\mathbf{x}^{(t)}, y^{(t)})$, where each example $t = 1, \dots, N$ constitutes a predictor variable vector $\mathbf{x}^{(t)} = (x_1, x_2, \dots, x_r)$ and an associated predicted variable $y^{(t)}$. In more complex problems there may be several predicted variables denoted by $y_l^{(t)}$ with $l = 1, \dots, L$. The predicted variable is related to the predictor variables by some unknown function $y = f(\mathbf{x})$. Let the function $h(\mathbf{x})$ be a hypothesis of the true function $f(\mathbf{x})$ and let the loss function $L(y, h(\mathbf{x}))$ be some measure of the error between the actual dependent variable value $y = f(\mathbf{x})$ and the predicted dependent variable value $\hat{y} = h(\mathbf{x})$.

The objective of supervised learning is to find a function $h(\mathbf{x})$ that minimizes the generalization loss $R(h)$ over all possible tuples $(\mathbf{x}, y)$ in population ε ,

$$R(h) = \sum_{(\mathbf{x}, y) \in \varepsilon} L(y, h(\mathbf{x})) p(\mathbf{x}, y). \quad (2.1)$$

The probability distribution $p(\mathbf{x}, y)$ for the population ε is, however, not known. Therefore an estimate of the generalization loss, the empirical loss $R_{\text{emp}}(h)$, is calculated as the average loss over the exemplar tuple set,

$$R_{\text{emp}}(h) = \frac{1}{N} \sum_{t=1}^N L(y^{(t)}, h(\mathbf{x}^{(t)})) \quad (2.2)$$

An optimization algorithm searches through the space of all possible hypotheses $\mathbb{H}$ for the function h^* that minimizes this empirical loss,

$$h^* = \arg \min_{h \in \mathbb{H}} R_{\text{emp}}(h) \quad (2.3)$$

An exhaustive search of the hypothesis space quickly becomes computationally exhaustive, therefore, supervised learning involves heuristic approaches to search for a good enough function h in $\mathbb{H}$. These approaches are called induction or learning algorithms (Maimon and Rakach, 2010; Kotsiantis *et al.*, 2007).

Three primary components of a supervised learning technique can be identified: model representation, model evaluation and search method (Fayyad *et al.*, 1996). Model representation involves choosing a family of models to be considered. The chosen model representation defines the hypothesis space $\mathbb{H}$. Model evaluation involves choosing a loss function with which to evaluate a candidate hypothesis h . Model search involves searching for the optimal hypothesis in the chosen hypothesis space $\mathbb{H}$.

The task of finding the best hypothesis $h \in \mathbb{H}$ can be further divided into two separate subtasks: model selection and parameter search (Russel and Novig, 2010; Fayyad *et al.*, 1996). In model selection, also known as hyper-parameter search, different model specifications in the chosen family of model representations are considered. Given a fixed model specification, a parameter search is performed by some induction algorithm to find those parameters, which optimize the evaluation criteria over the example data set. Hyper-parameter search is implemented as an outer loop to the parameter search problem, within each loop the parameter search is focussed on a different region of the hypothesis space $\mathbb{H}$.

For example, consider fitting a function to a data set. First, select the type of function one wants to fit to the data, say a polynomial function. Secondly, select a loss function with which to measure all candidate polynomials considered, say the mean-squared-error loss function. Thirdly, find the polynomial function that minimizes the average mean-squared-error over the example data set. In looking for the optimal polynomial, consider polynomials of different degrees (hyper-parameter search), and consider polynomials with

the same degree but different term coefficients (parameter search). Various polynomial regression techniques (induction algorithms) can be used to find the optimal term coefficients, once the optimal degree has been chosen.

As stated, the objective of supervised learning is to find a hypothesis function $h \in \mathbb{H}$ that closely approximates the true function f . Two types of problems can be addressed using supervised learning techniques (Fayyad *et al.*, 1996): regression problems and classification problems. If the image of the true function f is an infinite set of real values then the supervised learning problem is called a regression problem and the hypothesis function h is called the regression function. Alternatively, if the image of the true function f is a finite set of integers (or class labels) then the supervised learning problem is called a classification problem and the hypothesis function h is called a classifier.

Several supervised learning techniques exist with which to derive a classification function from an exemplar data set, including decision trees, naive Bayes classifiers and Bayesian networks, neural networks, discriminant functions, k -nearest neighbour classifiers and support vector machines (Maimon and Rakach, 2010; Holmstrom *et al.*, 1997; Lippmann, 1991). After learning, each of these classifiers has the ability to return a class label $\hat{y}$, from a pre-defined set of class labels, when presented with an input pattern $\mathbf{x}$. There is, however, no golden rule with regard to the best supervised learning technique to use for a particular problem, as the performance of the various techniques is much influenced by the particular application domain (Michie *et al.*, 1994).

Results from the *StatLog* project¹ show that the k -nearest neighbour classifier, the MLP neural network classifier and the linear discriminant classifier consistently perform relatively well in most application domains (Michie *et al.*, 1994). A considerable amount of research has already gone into file object type identification using the k -nearest neighbour classifier as will be outlined in Chapter 3. For this reason, this dissertation focuses on using neural network classifiers and linear discriminant classifiers, specifically linear programming-based discriminant classifiers, to solve the file fragment type identification problem. These two classifiers are outlined in Sections 2.2 and 2.3. The classification problem is addressed further in the next subsection.

2.1.1 The classification problem

Many examples of classification problems can be listed in the field of marketing, telecommunication, healthcare, human resource management, finance and so forth (Olson and Shi, 2006). Examples include face detection, as well as facial recognition (Bhele and Mankar, 2012); speech recognition (Anusuya and Katti, 2009); bankruptcy and credit scoring (Lin *et al.*, 2012; Twala, 2010); spam detection (Caruana and Li, 2012); document classification

¹Project for comparative testing and evaluation of statistical and logical learning algorithms on large-scale applications to classification, prediction and control.

(Baharudin *et al.*, 2010); financial crime detection (Bhattacharyya *et al.*, 2011; Sudjianto *et al.*, 2010), as well as churn detection (Rashid, 2008).

In each of these problem domains a function h has to be found that accurately maps an observed object into a finite set of predefined class labels. In a n class classification problem the function h has to map an observed object into one of n possible class labels. In a binary classification problem the function h has to map an observed object into one of two possible class labels.

Multiple class classification is essentially more challenging than a binary class classification problem, since the induced classifier must learn how to distinguish between a larger number of classes simultaneously (Woods *et al.*, 1997). Therefore, it is often required to transform a multiple class classification problem into a set of binary classification problems because many supervised learning techniques have a disadvantage in efficiently separating multiple classes (Olson and Shi, 2006). A multiple class problem can be decomposed into a set of binary subproblems by either comparing one class against all other classes or by comparing one class against another class. In a one-against-all decomposition each class is opposed to all the others classes, thus giving n subproblems. In a one-against-one decomposition each class is opposed to each other class, thus giving $n(n - 1)/2$ subproblems.

For example, in a 6-class classification problem a multiple class classifier h will classify an instance into any one of the possible classes, with $h \rightarrow \{G_0, \dots, G_5\}$ (Figure 2.1(a)). A one-against-all binary classifier h will confirm or refute that an instance belongs to the modelled target class, for example $h \rightarrow \{G_1(\text{positive}), \text{not } G_1(\text{negative})\}$ (Figure 2.1(b)). A set of 6 classifiers can be constructed by using each class in turn as the target. A one-against-one binary classifier h will assign instances into one of two classes, for example $h \rightarrow \{G_0, G_1\}$ (Figure 2.1(c)). A set of 15 classifiers can be constructed, one classifier for each possible combination of classes.

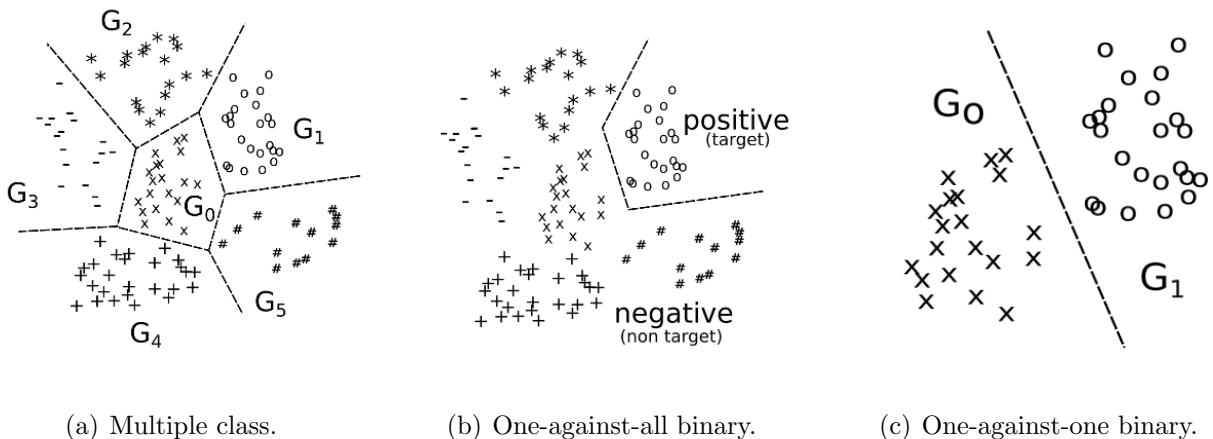


Figure 2.1: Possible formulations of the classification problem.

When transforming a multiple class problem into a set of binary subproblems a method is needed with which to combine the solutions of the binary subproblems into a solution

for the multiple class problem (Kuncheva, 2004). When using the one-against-all method the class with the highest output probability is usually chosen. In the one against-all strategy, the majority vote over the individual classifier outputs are taken (Furnkranz (2001) calls this technique round robin binarization). The best scheme to use varies according to the application domain and the supervised learning technique used (Milgram *et al.*, 2006; Klautau *et al.*, 2002).

This dissertation is interested in the classification problems known as file object type identification. As will be discussed in Chapter 3, the problem has been formulated as a one-against-all, one-against-one and multiple class classification problem within the existing literature. In the empirical phase of this dissertation different problem formulations are also used.

2.2 The neural network classification model

Neural networks imitate the cognitive function of the brain to approximate intelligent machines (Ramlall, 2010). The neuron, shown in Figure 2.2, is the fundamental building block of the human brain. The neuron is a special type of cell that processes and transmits information electrochemically. Individually each neuron receives nerve impulses from preceding connected neurons through connections called dendrites. Each input received is amplified or reduced according to the receiving neuron's learned sensitivity to inputs originating from each connected sender neuron. Within the cell body, the adjusted input signals are aggregated and an output signal is calculated. The output signal is transmitted on the axon. The axon in turn is the connection point for multiple successive neurons.

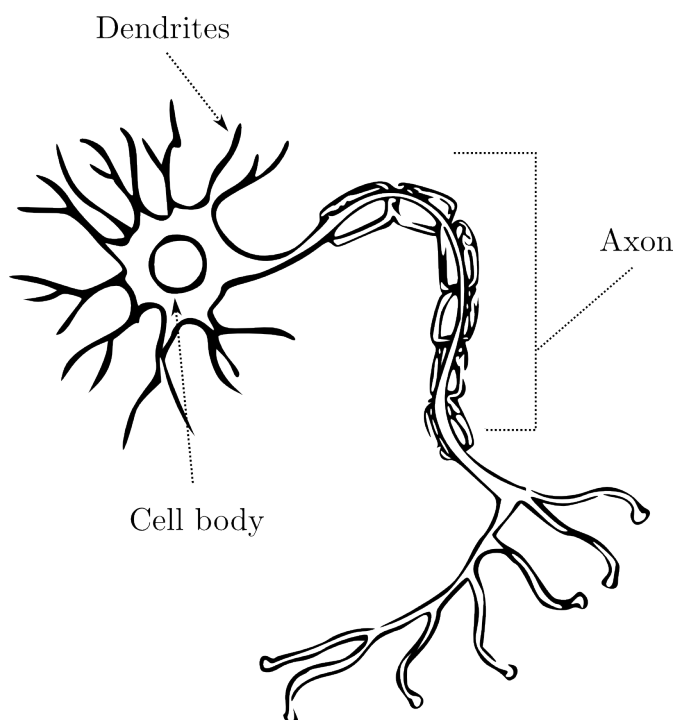


Figure 2.2: A biological neuron.

The ability of the human brain to encode knowledge is realized by signals sent within a complex network of roughly 100 billion neurons, each neuron connected to several thousand other neurons (Olson and Shi, 2006). Each neuron’s ability to alter its sensitivity to the various inputs it receives contributes a tiny bit to the overall network’s computational capacity.

Artificial neural networks originated from attempts to mimic the learning ability of the human brain (Zhang, 2010). In this section artificial neural networks are introduced. The first subsection outlines the perceptron and the multilayer perceptron neural network model. The second subsection introduces the back propagation algorithm, a parameter optimization method used to induce neural network classifiers. The third subsection introduces the concept of over fitting, as well as techniques that can be used to prevent over fitting. The last subsection lists advantages and disadvantages of MLP neural network classifiers.

2.2.1 Perceptrons and multilayer perceptrons

In the same manner as the human brain, an artificial neural network is composed of nodes connected by directed network links (Russel and Novig, 2010). The net-input u_j to some node j is computed by combining the input signals x_1 to x_r in a weighted sum and adding a node bias $-\theta_j$. This is,

$$u_j = \sum_{i=1}^r w_{ij}x_i - \theta_j \quad (2.4)$$

where the connection weight from input node i to node j is w_{ij} and the bias is θ_j . By introducing an extra dummy input signal x_{0j} equal to one, the combination function can be written as (Bishop, 1995)

$$u_j = \sum_{i=0}^r w_{ij}x_i \quad (2.5)$$

with $w_{0j} = -\theta_j$.

The node output a_j is 1 if the net-input u_j is greater than or equal to 0, else the node output is 0. This hard limit function, or step function, can be written as

$$a_j = \begin{cases} 0 & \text{if } u_j < 0 \Rightarrow \sum_{i=1}^r w_{ij}x_i < \theta_j \\ 1 & \text{if } u_j \geq 0 \Rightarrow \sum_{i=1}^r w_{ij}x_i \geq \theta_j \end{cases}. \quad (2.6)$$

Such a node, called a perceptron, is depicted in Figure 2.3.

A feed-forward neural network is composed of multiple perceptrons interconnected in such a way as to allow the output of one perceptron to be the input to another perceptron (Bishop, 1995). The connections in a feed-forward network only allow signals to pass in one direction. Specifically, the multilayer perceptron (MLP) neural network is the most widely studied and used feed-forward neural network model (Zhang, 2010). Such a neural network is structured as three or more distinct layers: an input layer, one or more hidden layers, and an output layer. Each layer constitutes a set of nodes, with nodes from different layers being highly interconnected in a feed-forward manner: All nodes from the input layer

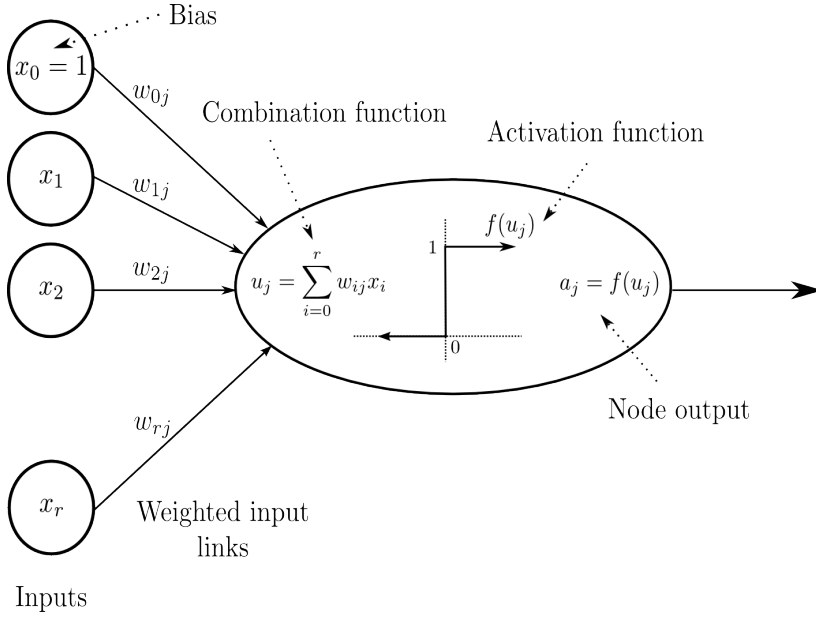


Figure 2.3: The perceptron.

are connected to all nodes in the first hidden layer, all nodes in the first hidden layer are connected to all nodes in the second hidden layer, and so forth, until all nodes in the last hidden layer are connected to all nodes in the output layer. The neural network maintains a vector of connection weights used to adjust the input signal as it propagates through the network towards the output layer. The structure of a three-layer MLP neural network is shown in Figure 2.4.

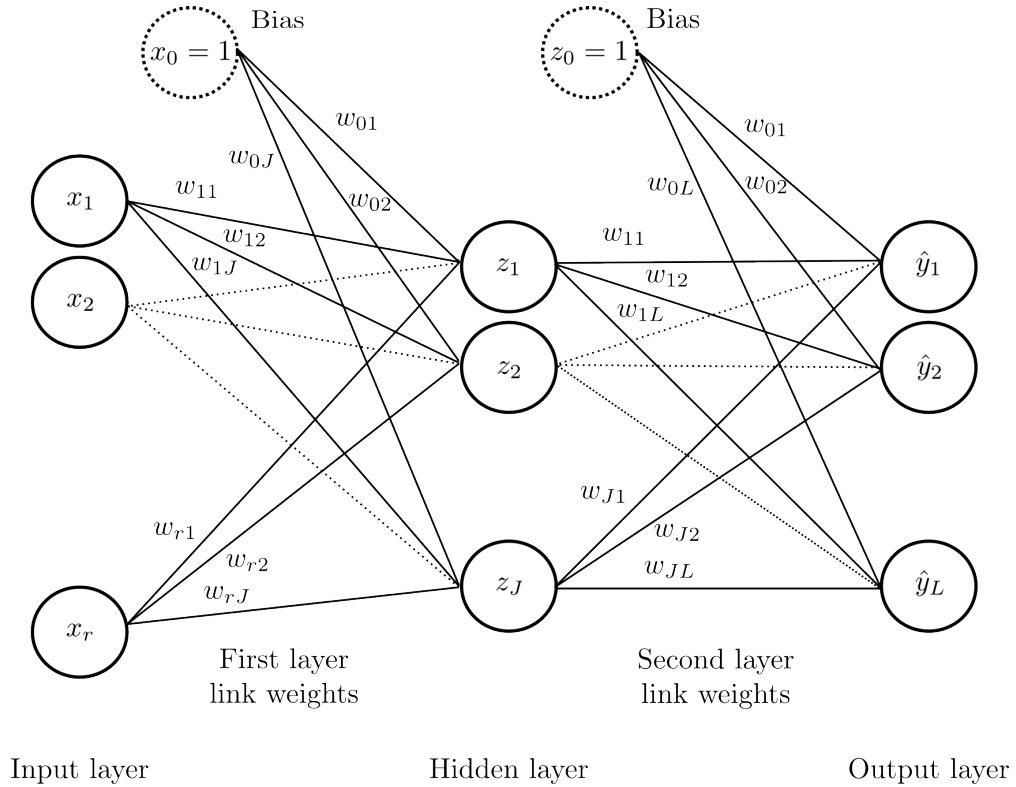


Figure 2.4: The multilayer perceptron neural network.

This MLP neural network receives $r + 1$ input signals at the input layer nodes. The input vector $\mathbf{x} \in R^{r+1}$ represents the r input features x_1 to x_r and the bias signal x_0 set equal to one. The input signal is propagated forward to J hidden layer nodes. Each hidden layer node functions like a perceptron. The node input is computed as the weighted sum of inputs received from all preceding nodes. That is $u_j = \sum_{i=0}^r w_{ij}x_i$ where w_{ij} is the weight of the connection between input node x_i and hidden node z_j .

The hidden layer's output is computed by passing the weighted sum of inputs u_j through an activation function $f^h(u_j)$. This activation function can be any monotonically increasing function, such as the symmetric sigmoid function (Steeb, 2005),

$$z_j = f^h(u_j) = \tanh(u_j) = \frac{1 + e^{-2u_j}}{1 - e^{-2u_j}}. \quad (2.7)$$

The output of the hidden layer is a vector $\mathbf{z} \in R^{J+1}$ with the bias signal z_0 set to one. The vector $\mathbf{z}$ supplies inputs to the output layer with L nodes.

Similar to the nodes in the hidden layer, each output layer node functions like a perceptron. Input signals from the hidden layer nodes are combined in a weighted sum $u_l = \sum_{j=0}^J w_{jl}z_j$ where w_{jl} is the weight of the connection from hidden node z_j to output layer node $\hat{y}_l$. The output layer produces a vector $\hat{\mathbf{y}} \in R^L$. Each output node produces output $\hat{y}_l$ by passing the weighted sum of inputs u_l through a monotonically increasing activation function $f^o(u_l)$. For example, a logistic activation function can be used,

$$\hat{y}_l = f^o(u_l) = \frac{1}{1 + e^{-u_l}}. \quad (2.8)$$

The entire MLP model can be represented as the following non-linear function,

$$\hat{y}_l = h_l(\mathbf{w}, \mathbf{x}) = f^o\left(\sum_{j=0}^J w_{jl}f^h\left(\sum_{i=0}^r w_{ij}x_i\right)\right) \quad (2.9)$$

for output $l = 1 \dots L$, where f^h and f^o are the activation functions in the output layer and hidden layer respectively, and $\mathbf{w} \in R^{J(r+1)+L(J+1)}$ is the vector of connection weights (Russel and Novig, 2010).

Each connection weight $w \in \mathbf{w}$ is used to adjust the input signal as it propagates through the network towards the output layer. By maintaining a weight for each connection in the network, a signal originating from a node can be very important in generating the output of one node and unimportant in generating the output of another node. It is through this process of connection specific signal weighting that a neural network has its predictive power. It has been shown that a three-layer MLP neural network with enough hidden layer nodes can approximate any continuous function with any desired degree of accuracy (Hornik *et al.*, 1989).

The process in which the optimal weights vector $\mathbf{w}^*$ is determined, hence the way the network “learns”, is commonly referred to as network training. The back propagation algorithm is a popular technique used for network training. Gaining a general understanding

of this algorithm is important due to its significance in neural network literature and because it is used in the empirical study of this dissertation. The back propagation algorithm is outlined in the next subsection.

2.2.2 The back propagation algorithm

The MLP neural network (2.9) has been described in the previous subsection. An infinite number of such unique models can be derived by adjusting the value of each connection weight $w \in \mathbf{w}$ and the value of each hyper-parameter r, J and L . Given a specific value for hyper-parameter r, J and L the supervised learning problem becomes a parameter optimization problem to find the weights vector $\mathbf{w}^*$ that minimizes a loss function across all outputs and all training instances (Russel and Novig, 2010),

$$\mathbf{w}^* = \arg \min_{\mathbf{w} \in R^{J(r+1)+L(J+1)}} \sum_{t=1}^N \sum_{l=1}^L L(y_l^{(t)}, \hat{y}_l^{(t)} = h_l(\mathbf{w}, \mathbf{x}^{(t)})). \quad (2.10)$$

The process in which optimal network weights $\mathbf{w}^*$ are determined is referred to as neural network training. A popular technique used to train a feed-forward neural network is known as the back propagation algorithm (Rumelhart *et al.*, 1995). The back propagation algorithm is essentially a form of gradient descent optimization, now applied to feed-forward neural network weight estimation (Russel and Novig, 2010; Bishop, 1995).

The back propagation algorithm starts by initializing each weight $w \in \mathbf{w}$ to a small random value. Using the current weights, the predicted network output $\hat{y}_l$ with $l = 1, \dots, L$ for a given input pattern $\mathbf{x}$ is calculated. This is done by propagating the input pattern forward through the hidden layer nodes and subsequently the output layer nodes. The predicted network output $\hat{y}_l$ and actual output y_l at each output node are compared using some loss function $L_l(y_l, \hat{y}_l)$. The loss function measures the error in the predicted network output.

After computing the error in the predicted network output for a given input pattern, gradient decent is applied to adjust weights in the direction that minimizes this error. That is, every weight $w \in \mathbf{w}$ is adjusted in proportion to the marginal rate of change in L_l with regard to this weight w over all the output nodes. The update rule can be written as,

$$w^{new} = w^{old} - \eta \sum_{l=1}^L \frac{\partial L_l}{\partial w} \quad (2.11)$$

where the scalar η is called the learning rate. The learning rate scales the effect of the marginal rate of change in correcting the old weight.

If the activation functions used in the hidden layer nodes and output layer nodes are differentiable then the chain rule can be applied to calculate $\frac{\partial L_l}{\partial w}$. The symmetric sigmoid activation function (2.7) and the logistic activation function (2.8) are both differentiable as,

$$\frac{\partial f^h(u_j)}{\partial u_j} = (1 - f^h(u_j))^2 \quad (2.12)$$

and

$$\frac{\partial f^o(u_l)}{\partial u_l} = f(u)(1 - f^o(u_l)) \quad (2.13)$$

respectively.

Therefore, if for example the squared-error loss function is used to measure the error in the predicted output

$$L_l(\hat{y}_l, y_l) = (y_l - \hat{y}_l)^2, \quad (2.14)$$

then the marginal rate of change in L_l with regard to the weight from hidden layer node j to output node l can be calculated as (Steeb, 2005),

$$\begin{aligned} \frac{\partial L_l}{\partial w_{jl}} &= \frac{\partial (y_l - \hat{y}_l)^2}{\partial w_{jl}} \\ &= -2(y_l - \hat{y}_l) \frac{\partial \hat{y}_l}{\partial w_{jl}} \\ &= -2(y_l - \hat{y}_l) \frac{\partial f^o(u_l)}{\partial w_{jl}} \\ &= -2(y_l - \hat{y}_l) f^o(u_l)(1 - f^o(u_l)) \frac{\partial u_l}{\partial w_{jl}} \\ &= -2z_j(y_l - \hat{y}_l) f^o(u_l)(1 - f^o(u_l)). \end{aligned} \quad (2.15)$$

Similarly, it can also be shown that the marginal rate of change in L with regard to the weight from input node i to hidden layer node j is

$$\frac{\partial L_l}{\partial w_{ij}} = -2w_{ji}(y_l - \hat{y}_l) f^o(u_l)(1 - f^o(u_l)) \frac{\partial z_j}{\partial w_{ij}}, \quad (2.16)$$

which can be expanded to

$$\begin{aligned} \frac{\partial L_l}{\partial w_{ij}} &= -2w_{ji}(y_l - \hat{y}_l) f^o(u_l)(1 - f^o(u_l)) \frac{\partial f^h(u_j)}{\partial w_{ij}} \\ &= -2w_{ji}(y_l - \hat{y}_l) f^o(u_l)(1 - f^o(u_l))(1 - f^h(u_j))^2 \frac{\partial u_j}{\partial w_{ij}} \\ &= -2w_{ji}x_i(y_l - \hat{y}_l) f^o(u_l)(1 - f^o(u_l))(1 - f^h(u_j))^2. \end{aligned} \quad (2.17)$$

The network weights are updated per training pattern presented to the learning algorithm using the update rule in Equation 2.11 once $\frac{\partial L_l}{\partial w}$ has been calculated for each network weight $w \in \mathbf{w}$ and each output node $l = 1, \dots, L$. This is called online learning (Ripley, 2008). Online learning is typically a continuous process, where training examples are given to the network as they become available.

Alternatively, in batch learning, all the training patterns are presented to the network first before the actual network weights are updated (Ripley, 2008). The change in the network weights after one cycle through all training patterns, called an epoch, is the sum of all the individual changes in network weights. The update rule for each weight $w \in \mathbf{w}$ in batch learning is,

$$w^{new} = w^{old} - \eta \sum_{t=1}^N \sum_{l=1}^L \frac{\partial L_l(\hat{y}^{(t)}, y^{(t)})}{\partial w} \quad (2.18)$$

where the scalar η is the learning rate. Once an epoch has been completed the empirical loss $R_{\text{emp}}(h)$ for the current network h is calculated. The empirical loss is tested for convergence against some predefined criterion with threshold τ . If the change in empirical loss from one epoch to the next is less than τ the training algorithm stops. If not, then the neural network algorithm continues for another epoch. Furthermore, the training algorithm also terminates once a predefined number of epochs ϵ has been completed. The back propagation algorithm for batch learning is summarized in Algorithm 2.1.

Algorithm 2.1 Back propagation algorithm for batch learning (Adapted from Steeb, 2005).

input: Set of N exemplar tuples $(\mathbf{x}^{(t)}, y^{(t)})$, learning rate $\eta > 0$, convergence criterion $\tau > 0$, and stop criterion $\epsilon > 0$

output: Optimal weight vector $\mathbf{w}^*$

begin algorithm

Initialize each weight $w \in \mathbf{w}$ to small random value

Initialize epoch counter e to 0

repeat

Increment epoch counter e .

For each $w \in \mathbf{w}$ initialise $\dot{w}$ to 0 ***Temporary weight update variable*

for $t = 1$ **to** N **do**

for $l = 1$ **to** L **do**

Calculate $y_l^{(t)} = h_l(\mathbf{w}, \mathbf{x}^{(t)})$.

Calculate $L_l(\hat{y}_l^{(t)}, y_l^{(t)})$ using predicted output $\hat{y}_l^{(t)}$ and actual output $y_l^{(t)}$

for each $w \in \mathbf{w}$ **do**

Calculate $\frac{\partial L_l(\hat{y}_l^{(t)}, y_l^{(t)})}{\partial w}$ using Equation 2.17 for a connection weight between an input layer node and a hidden layer node and Equation 2.15 for a connection weight between a hidden layer node and an output layer node

Update $\dot{w} \leftarrow \dot{w} + \frac{\partial L_l(\hat{y}_l^{(t)}, y_l^{(t)})}{\partial w}$

end for

end for

end for

For each weight $w \in \mathbf{w}$ update $w \leftarrow w - \eta \dot{w}$

Calculate R_{emp}

until $e \geq \epsilon$ **or** $\Delta R_{\text{emp}} \leq \tau$ ***Stop criterion and convergence criterion*

end algorithm

Even though the back propagation algorithm is a very slow optimization technique, it is still widely used. Several alternative algorithms have been proposed as improvements on the back propagation algorithm, such as the quickprop algorithm (Fahlman and Lebiere, 1990) and the RPROP algorithm (Riedmiller and Braun, 1993). Though these “improved” optimization algorithms have been shown to perform better than the back propagation algorithm on artificial data sets, this finding does not hold for real-world data sets (Schiffmann *et al.*, 1994). Furthermore, Lawrence *et al.* (1997) argue that the inefficiency in the back propagation algorithm might improve its ability to generalize patterns by inherently guarding against learning the idiosyncrasies present in the data. For these reasons the back propagation algorithm is used for neural network training in the empirical study of this

dissertation.

In conclusion, the back propagation algorithm is a parameter optimization technique that can be used to estimate the weight set $\mathbf{w}^*$ that defines the MLP neural network that best fits the exemplar data set. The hypothesis space $R^{J(r+1)+L(J+1)}$ in which the parameter optimization is performed is defined by selecting values for hyper-parameters r , L and J .

2.2.3 Learning and generalization

As stated in Section 2.1 the task of finding the best hypothesis $h \in \mathbb{H}$ can be divided into two separate subtasks: parameter search and hyper-parameter search. During hyper-parameter search the complexity of the hypothesis space $\mathbb{H}$ considered in parameter search is chosen. In the case of MLP neural networks, parameter search involves applying the back propagation algorithm to find an optimal weight vector $\mathbf{w}^*$ in the hypothesis space $R^{J(r+1)+L(J+1)}$ with $J(r+1) + L(J+1)$ dimensions (adjustable parameters). Hyper-parameter search involves choosing the values for r , J and L and therefore the number of adjustable parameters in the hypothesis space considered. As will be outlined in this subsection, hyper-parameter search is an important part of building a good MLP neural network classifier, because the complexity of the hypothesis space $\mathbb{H}$ influences the ability to which some function $h \in \mathbb{H}$ can learn and generalize the true function f .

Learning is the ability of a function h to approximate the training data, while generalization is the ability of function h to approximate the true function f that underlies the training data (Zhang, 2000). The number of adjustable parameters in the hypothesis space from which function h is induced determines how well such a function can learn and generalize the true function f . A hypothesis space with too few adjustable parameters might not be flexible enough to express the true function f while a hypothesis space with too many adjustable parameters will be flexible enough to fit the idiosyncrasies in the data.

Consider the example of fitting a polynomial function to a set of data points in the Cartesian plane depicted in Figure 2.5. Assume the true relationship between x and y is described by the fourth degree polynomial f . However, like in any real-world data set, every data point deviates from the true function with random noise ξ . A poor approximation of the true function is achieved by fitting a second degree polynomial h_1 because the hypothesis space of all second degree polynomials lacks the flexibility to express the true function. That is, a second degree polynomial is unable to learn the true function f . Poor approximation of the true function is also achieved by fitting a polynomial function h_2 with seven degrees. The hypothesis space defined by all polynomials with a degree of seven is too flexible, and as such, a seven degree polynomial is able to fit all the data points exactly by learning the random noise in the data. Though all example points are approximated perfectly, the hypothesised function h_2 and the true function f still differ to a large extent — this is known as over fitting. A polynomial with some degree between two and seven would have been better suited to learn the true function f .

A model that has been over fitted is characterized by good predictive power within the

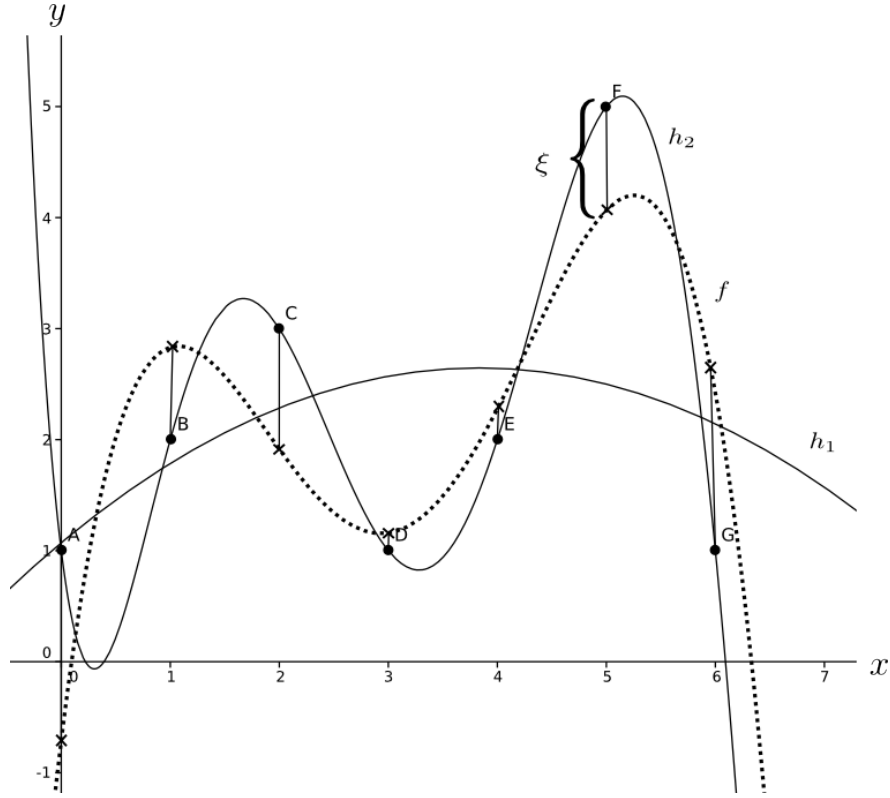


Figure 2.5: Fitting a polynomial function.

training examples and bad predictive power on new examples. Consider a model's empirical loss estimated on a training set and a validation set (see Figure 2.6). As the complexity of the hypothesis space is increased, an induced model is better able to generalize and learn the true underlying function from the training data. Consequently both the empirical loss in the training set and the empirical loss in the validation set are reduced. As the complexity of the hypothesis space is increased further, an induced model starts to fit the idiosyncrasies in the training data. After some point A , the induced models will continue learning the training data by memorizing the noise in the data, but because this noise is in actual fact random, the function learned will not generalize to the validation data. As soon as the model starts fitting the noise in the training data, the empirical loss on the validation set increases. The ideal complexity of the hypothesis space will be at point A , the point where the induced model has low empirical loss within the training set and a low empirical loss on a validation set (Bishop, 1995).

All predictive models suffer from the general phenomenon of over fitting. However, over fitting becomes much more likely as the complexity of the hypothesis space grows (Russel and Novig, 2010). As a result, care has to be taken when choosing the complexity of the hypothesised MLP neural network fitted to data. A network with too many adjustable parameters is not only computationally expensive to train but it is also subject to over fitting. The complexity of the hypothesis space from which an MLP neural network is induced is selected by setting the number of input nodes r , number of output layer nodes L and the number of hidden layer nodes J .

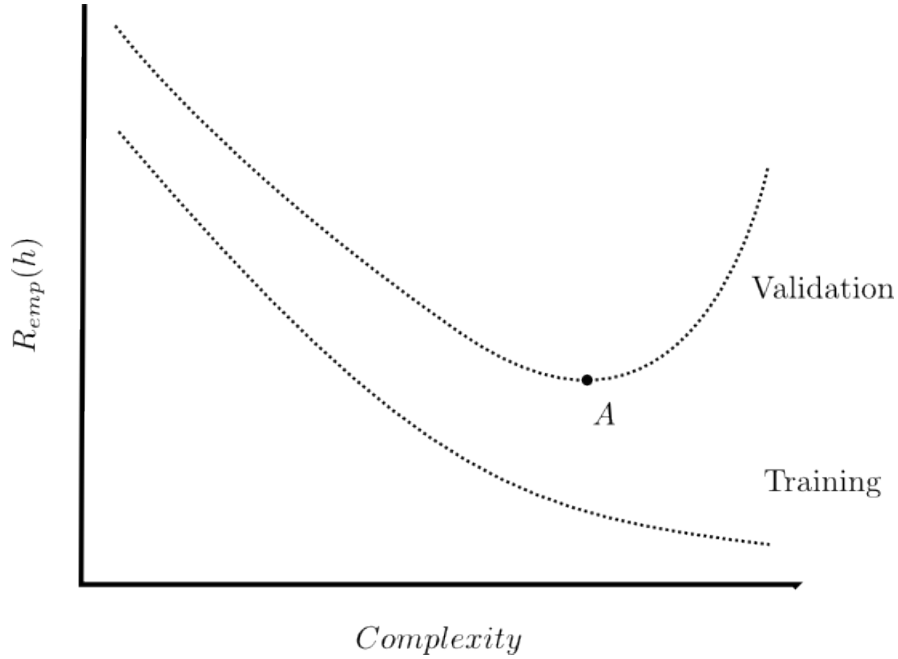


Figure 2.6: The empirical loss estimated on a training set and a validation set for models induced from increasingly complex hypothesis spaces.

The number of input nodes r used with a neural network is implied in the number of input attributes selected to form the input vector $\mathbf{x}$. The performance of a neural network can often be improved by excluding attributes that are irrelevant or redundant (Kwak and Choi, 2002). Feature selection is outlined in Section 4.1.

The number of output nodes L is implied by the formulation of the classification problem as outlined in Section 2.1.1. In the case of an MLP neural network, only one output node is needed to solve a binary classification problem (Bishop, 1995). The network output is coded $y = -1$ for instances belonging to the first class and $y = 1$ for instances belonging to the second class. An MLP neural network with n output nodes is usually employed to solve a n class classification problem. The network outputs are coded such that $y_l = 1$ for instances belonging to class l and $y_l = 0$ for instances not belonging to class l ($l = 1, \dots, n$).

One reason to transform a multiple class classification problem into a set of binary subproblems is that a binary classifier is usually simpler and therefore better suited to fit small data sets (i.e. less prone to over fitting). Furthermore, Furnkranz (2001) argues that a set of binary classifiers can provide better performance than one complex multiple class classifier because it is easier to learn how to distinguish between two classes than it is to learn how to distinguish between multiple classes.

The number of hidden layer nodes J is, however, not so simple to infer from the problem at hand. Choosing the appropriate number of hidden layer nodes is important because this hyper-parameter has a large influence on the number of adjustable parameters in the hypothesis space. As such, a neural network with too few hidden layer nodes will not be expressive enough to learn the underlining patterns in the data while a neural network

with too many hidden layer nodes will over fit the data (Ramlall, 2010). The appropriate number of hidden layer nodes depends on the inherent noise within the training data and the availability of training data; the number of input features used and the interdependence of these input features; the number of output nodes used; the network architecture selected; as well as the complexity of the true function f being approximated (Zhang, 2010).

One technique for selecting the appropriate number of hidden nodes is to compare a range of models each having a different number of hidden layer nodes. The empirical loss for each model is then estimated on a validation set, and the simplest model with the lowest empirical loss is accepted (Bishop, 1995). This technique is, however, computational and time intensive (Reed, 1993). Furthermore, the estimated empirical loss might not be as well behaved and smooth across models with different complexities as depicted in Figure 2.6. Identifying the optimal complexity of hypothesis space (point A) is difficult.

Alternatively, a complex neural network can be trained, and the insignificant hidden layer nodes removed from the network as part of network training (Reed, 1993). The general idea behind network pruning is to alternate between training the network and removing nodes from the network. After training the network for several epochs, nodes that do not contribute to the network's overall success are removed. A node's contribution in the network is approximated as the difference between the empirical loss of the network that includes the node and the empirical loss of the network that excludes the node, all else held constant.

Regularization is another approach with which to control model complexity (Bishop, 1995). Regularization is a class of methods that encourages the induction of smoother models during model training, because models that have a lot of curvature and structure tend to over fit the data. A measure of smoothness is introduced into estimated empirical loss by adding a penalty term Ω ,

$$\hat{R}_{\text{emp}}(h) = \frac{1}{N} \sum_{(\mathbf{x}, y) \in \mathbb{E}} L(y, h(\mathbf{x})) + \lambda \Omega. \quad (2.19)$$

The penalty term Ω is some measure of the hypothesised function's complexity, and λ is the extent with which Ω is to influence the solution. Regularization combines loss and complexity into one metric, allowing the training algorithm to find the best hypothesis in terms of learning and generalization. A function h that gives a good fit to the training data will have a small L value while one that is very smooth will have a small Ω value.

One of the simplest regularizers, called weight decay, is defined as the sum of the squares of the adjustable parameters (Krogh and Hertz, 1995),

$$\Omega = \frac{1}{2} \sum_{w \in \mathbf{w}} w^2. \quad (2.20)$$

Weight decay encourages the training algorithm to find the hypothesis h defined by the sparsest weight vector $\mathbf{w}$ that solves the learning problem. Weight decay supports generalization by keeping network weights small and therefore effectively removing insignificant

connections by slowly decaying connection weights towards zero (Bartlett, 1998). Experimental results show that the back propagation algorithm can be improved significantly by including a weight decay term in the loss function used (Gupta and Lam, 1998).

The concept of learning and generalization has been outlined in this subsection. It has been stated that the complexity of the hypothesis space determines, to a large extent, the ability of an induced classifier to learn and generalize the true function f . The complexity of a neural network model is determined by the number of input nodes r , number of hidden layer nodes J , and the number of output layer nodes L employed. Various techniques to reduce model complexity have been listed, some of which are employed in the empirical phase of this dissertation. Specific hyper-parameter optimization algorithms are outlined in Section 4.2.

2.2.4 Advantages and disadvantages of neural networks

Neural networks have been used for classification in various problem domains, including accounting and finance; healthcare and medicine; engineering and manufacturing, as well as marketing. A literature survey conducted by Paliwal and Kumar (2009) reveals that neural networks outperformed other methods in most application domains. However, there are several advantages and disadvantages to using a neural network. Advantages of neural networks include (Zhang, 2010; Ramlall, 2010):

- Neural networks are non-linear functions of their inputs, therefore more complex relationships between variables are modelled.
- MLP neural networks are universal approximators. Given a finite set of hidden layer nodes it can be used to approximate any continuous function to any desired degree of accuracy.
- There is no need to specify a functional form for the hypothesis function f . Any needed variable transformation and combination are done within the network.
- Once a network function has been induced, it can be used for prediction very quickly.
- Neural networks are free from statistical assumptions on the underlining data.

Disadvantages of neural networks include:

- The optimal value of several hyper-parameters have to be chosen, including the number of hidden layer nodes, which is directly related to a neural network's ability to generalize patterns. As with most parameters in machine learning, limited theory exists to guide the choice of hyper-parameters.

- Training neural networks is time consuming, and often involves the subjective art of trial and error.
- An MLP neural network is prone to over fitting. Building complex MLP neural networks therefore requires large samples of data. However, it has been shown that large neural networks do generalize well as long as weights are kept small.
- Neural networks are black boxes because it is difficult to trace the exact relationship between inputs and outputs.
- Neural networks have nonlinear loss functions, which means that they can converge to local optimal solutions rather than global optimal solutions.

2.3 Linear discriminant classification model

The use of neural networks for classification has its origins in artificial intelligence. Another popular methodology for deriving classification rules from a sample of labelled objects, based on observed attributes of those objects, is called discriminant analysis. Discriminant analysis has its origin in classical statistics, but is naturally defined as a supervised learning technique because it also uses a set of exemplar input-output pairs to derive a so-called discriminant function (Kuncheva, 2004).

Discriminant analysis produces a linear classifier with a hard threshold, such as

$$h(\mathbf{x}) = \begin{cases} 0 & \text{if } \sum_{i=1}^r w_{ij}x_i < b \\ 1 & \text{if } \sum_{i=1}^r w_{ij}x_i \geq b \end{cases}, \quad (2.21)$$

where 0 is the class label corresponding to group A, 1 is the class label corresponding to group B, $\sum_{i=1}^r w_{ij}x_i$ is the discrimination function and b is the group boundary value. The decision boundary $\sum_{i=1}^r w_{ij}x_i = b$ can be viewed as the hyperplane that separates two classes in the r -dimensional space (Russel and Novig, 2010). Figure 2.7 shows two hypothetical decision boundaries that can be used to separate instances belonging to one of two groups. Using the linear discriminant classifier h_1 a new instance $\tilde{\mathbf{x}}_1$ will be classified as belonging to group A because $h_1(\tilde{\mathbf{x}}_1) < b$. A new instance $\tilde{\mathbf{x}}_2$ will be classified as belonging to group B because $h_1(\tilde{\mathbf{x}}_2) \geq b$. A new instance $\tilde{\mathbf{x}}_3$ will be classified differently when linear discriminant classifier h_2 is used instead of linear discriminant classifier h_1 .

The objective of discriminant analysis is to find $\mathbf{w}^*$ and b^* that defines the decision boundary that optimally separates example instances belonging to two classes. By extension, such hyperplane can then be used to assign class membership to instances for which the class labels are not known. Fisher's linear discriminant analysis is a popular multivariate technique used to derive discriminant functions (Fisher, 1936). The ability of this technique to derive an optimal discriminant function relies on a multivariate normality assumption. A sub-optimal discriminant function is estimated when this assumption does not hold, as is the case in most real-world problems.

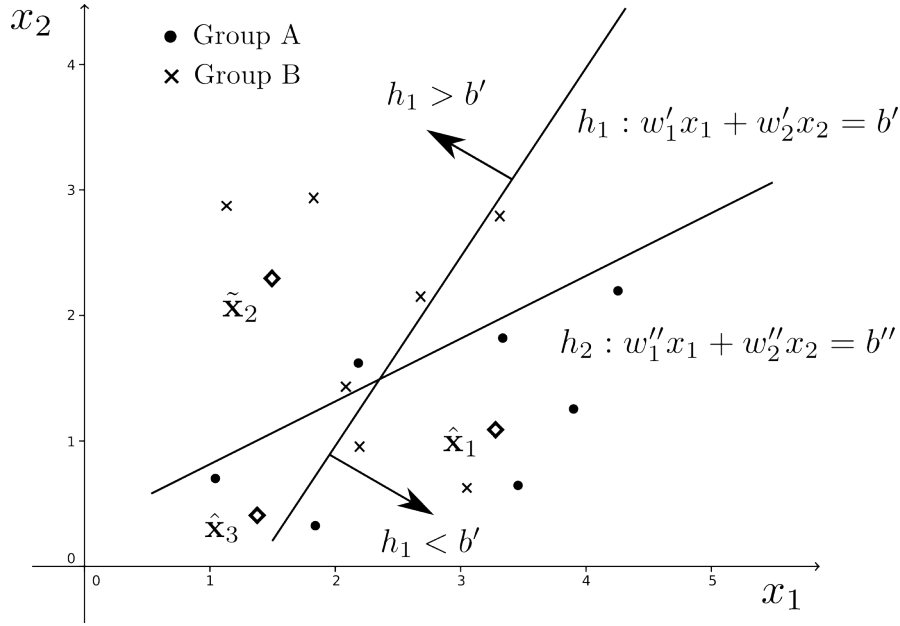


Figure 2.7: Discriminant analysis.

A linear programming-based discriminant analysis approach allows for efficient computation of an optimal discriminant function over large sets of sample data without relying on any assumptions about the data. Empirical studies have shown that linear programming-based discriminant classifiers (LPDCs) outperform Fisher's linear discriminant classifiers in situations where the normality assumption does not hold or where there are outliers present in the data (Youssef and Rebai, 2007; Lame and Moy, 1997).

In this section a method for deriving a linear discriminant classifier using linear programming is described. Linear programming is introduced in the first subsection. In the second subsection the simplex algorithm, an optimization technique used to solve linear programming problems, is outlined. In the third subsection a simple linear programming-based discriminant analysis model and its derivations are formulated. Some advantages and disadvantages of linear programming based discriminant classifiers are outlined in the last subsection.

2.3.1 Linear programming

Linear programming (LP) is concerned with the problem of minimizing or maximizing a linear function in the presence of linear inequalities (Gale, 2007). An LP problem is said to be in standard form if all constraints are equalities and all variables are non-negative, such as

$$\begin{aligned}
 & \underset{\mathbf{x}}{\text{maximize}} && z(\mathbf{x}) = \mathbf{c}^T \mathbf{x} \\
 & \text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b} \\
 & && \text{with } \mathbf{b}, \mathbf{x} \geq 0
 \end{aligned} \tag{2.22}$$

where $\mathbf{c}$ is the vector of objective function coefficients, $\mathbf{A}$ is the $(m \times n)$ matrix of

constraint coefficients and the vector $\mathbf{b}$ contains the non-negative right-hand side of each constraint. The collection of vectors $\mathbf{x}$ that satisfies the set of constraints is called the feasible region. Any vector $\mathbf{x}$ in the feasible region is called a feasible solution, and any feasible solution that optimizes the objective function is called an optimal feasible solution.

An LP problem that is not in standard form can be converted to standard form by (Bazaraa *et al.*, 1990):

- multiplying any constraint with $b_i < 0$ by -1 ;
- converting constraints given as $\sum_{i=1}^n a_{it}x_i \geq b_i$ to equalities by subtracting a slack variable x_{n+1} , to get $\sum_{i=1}^{n+1} a_{it}x_i = b_i$ with $a_{n+1} = -1$;
- converting constraints given as $\sum_{i=1}^n a_{it}x_i \leq b_i$ to equalities by adding a slack variable x_{n+1} , to get $\sum_{i=1}^{n+1} a_{it}x_i = b_i$ with $a_{n+1} = 1$;
- replacing any variables x_i with unrestricted sign by two non-negative variables x'_i and x''_i such that $x_i = x'_i - x''_i$; and by
- converting a minimization problem to a maximization problem by multiplying the cost vector by -1 . That is, *minimize* $\sum_{i=1}^n c_i x_i$ becomes *maximize* $-\sum_{i=1}^n c_i x_i$.

The simplex algorithm is an optimization technique that finds the optimal feasible solution for an LP problem in standard form if such a solution exists (Bronshtein *et al.*, 2007). This algorithm can be used to find the optimal discriminant boundary in the linear programming-based discriminant analysis model (LPDA) proposed in Section 2.3. The simplex algorithm is outlined in the next subsection.

2.3.2 Simplex algorithm

The simplex algorithm is an optimization technique that will solve any well-structured LP problem exactly in a finite number of steps. A well-structured LP problem is one that has a bounded feasible region and does not have any redundant constraints². The simplex algorithm observes that the optimal solution to an LP problem will be at an extreme point of the feasible region (Bazaraa *et al.*, 1990). Instead of inefficiently enumerating all extreme points, the simplex algorithm implements a guided search along the edge of the feasible region from one extreme point to the next extreme point towards the optimal extreme point. At each extreme point the effect that moving to any adjacent extreme point will have on the objective function value is evaluated. The simplex algorithm moves to the next extreme point only if such a move will improve the objective function value. The iterative procedure of moving step by step from one extreme point to an adjacent extreme point is continued until no further improvement in the objective function can be realized.

²If this assumption does not hold, special consideration has to be given to degenerate and unbounded LP problems (Bazaraa *et al.*, 1990).

To understand the simplex algorithm, first consider the following system of linear equations,

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad (2.23)$$

where $\mathbf{A}$ is a matrix with n columns and m independent rows, $\mathbf{x}$ is a column vector with n entries, $\mathbf{b}$ is a column vector with m entries and $m \leq n$.

The matrix $\mathbf{A}$ can be decomposed into a submatrix $\mathbf{B}$ (a basis for the column space of $\mathbf{A}$) and a submatrix $\mathbf{N}$ (the columns vectors in $\mathbf{A}$ not in $\mathbf{B}$)

$$\mathbf{A} = [\mathbf{B}, \mathbf{N}]. \quad (2.24)$$

In the same way, the column vector $\mathbf{x}$ can be decomposed to correspond to the columns in $\mathbf{B}$ and $\mathbf{N}$,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{bmatrix}. \quad (2.25)$$

There have to be m linearly independent columns in $\mathbf{A}$ because there are m linearly independent rows in $\mathbf{A}$ (Johnson *et al.*, 2002). If $\mathbf{B}$ is the submatrix consisting of the m independent columns in $\mathbf{A}$, it will be invertible such that,

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ [\mathbf{B}, \mathbf{N}] \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{bmatrix} &= \mathbf{b} \\ \mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N &= \mathbf{b} \\ \mathbf{B}\mathbf{B}^{-1}\mathbf{x}_B + \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N &= \mathbf{B}^{-1}\mathbf{b} \\ \mathbf{x}_B &= \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N \end{aligned} \quad (2.26)$$

It follows that an arbitrary solution for $\mathbf{x}_B$ can be obtained for any $\mathbf{x}_N \in \mathbb{R}^{(m-n)}$. If $\mathbf{x}_N$ is set equal to zero then the resulting solution for $\mathbf{x}$ is called a basic solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$. The non-zero variables in the basic solution correspond to the column vectors in basis $\mathbf{B}$. As such, the variables in $\mathbf{x}_B$ are called basic variables and the variables in $\mathbf{x}_N$ are called non-basic variables (Gale, 2007).

Now consider the set of all vectors $\mathbf{x}$ that satisfies the constraints in 2.22,

$$\mathbb{X} = \left\{ \mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0, \forall \mathbf{b} \in \mathbb{R}^n, \mathbf{A} \in \mathbb{R}^{m \times n} \text{ with } m \leq n \right\} \quad (2.27)$$

The set $\mathbb{X}$ is called the feasible region of the LP problem. A feasible point $\mathbf{x}^* \in \mathbb{X}$ where

$$z(\mathbf{x}^*) \geq z(\mathbf{x}) \forall \mathbf{x} \in \mathbb{X} \quad (2.28)$$

is called the optimal feasible solution.

According to Bazaraa *et al.* (1990) if $\mathbb{X}$ is non-empty, then there is an optimal solution to the LP problem in 2.22; this optimal solution is an extreme point of $\mathbb{X}$; each extreme point of $\mathbb{X}$ is a basic solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$ with $\mathbf{x} \geq 0$; therefore the optimal extreme point

also corresponds to a basic solution to $\mathbf{Ax} = \mathbf{b}$ with $\mathbf{x} \geq 0$. The simplex algorithm finds the optimal feasible solution to the LP problem in 2.22 by iteratively evaluating the basic solutions to $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{x} \geq 0$. Different basic solutions are found by considering different combinations of variables as basic variables.

The simplex algorithm can be explained algebraically as follows (Griffon, 2012; Bazaraa *et al.*, 1990): Choose an initial basic feasible solution $\mathbf{x} = [\mathbf{x}_B, \mathbf{x}_N]$. A basic feasible solution can be identified by setting $\mathbf{B}$ equal to any identity submatrix³ $\mathbf{I}_m$ in $\mathbf{A}$, and solving for $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$. The submatrix $\mathbf{N}$ consists of all column vectors in $\mathbf{A}$ not included in $\mathbf{B}$.

For the current basic feasible solution, the cost vector $\mathbf{c}$ can be divided into its basic and non-basic components $\mathbf{c} = [\mathbf{c}_B, \mathbf{c}_N]$ to correspond with the columns in $\mathbf{B}$ and $\mathbf{N}$. The objective function then becomes,

$$\begin{aligned} z(\mathbf{x}) &= \mathbf{c}^T \mathbf{x} \\ &= [\mathbf{c}_B^T, \mathbf{c}_N^T] \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{bmatrix} \\ &= \mathbf{c}_B^T \mathbf{x}_B + \mathbf{c}_N^T \mathbf{x}_N \end{aligned} \quad (2.29)$$

and by substituting 2.26 into 2.29,

$$\begin{aligned} z(\mathbf{x}) &= \mathbf{c}_B^T (\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N) + \mathbf{c}_N^T \mathbf{x}_N \\ &= \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{b} - \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N + \mathbf{c}_N^T \mathbf{x}_N \\ &= \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{b} + (\mathbf{c}_N^T - \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{N})\mathbf{x}_N. \end{aligned} \quad (2.30)$$

If $\mathbb{J}$ is the set of column indices, which corresponds to the non-basic variables in $\mathbf{A}$ then $\mathbf{A}_{.j}$ is the column vector in $\mathbf{A}$ that corresponds to non-basic variable x_j , then 2.30 can be written as,

$$z(\mathbf{x}) = \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{b} + \sum_{j \in \mathbb{J}} (c_j - \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{A}_{.j})x_j. \quad (2.31)$$

The marginal rate of change in z with respect to variable x_j can then be derived by differentiation,

$$\begin{aligned} \frac{\partial z(\mathbf{x})}{\partial x_j} &= c_j - \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{A}_{.j} \\ &= -(\mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{A}_{.j} - c_j) \\ &= -(z_j - c_j) \\ -\frac{\partial z(\mathbf{x})}{\partial x_j} &= z_j - c_j. \end{aligned} \quad (2.32)$$

The value $-\frac{\partial z(\mathbf{x})}{\partial x_j}$ is called the reduced cost value for variable x_j , or the rate with which z will decrease per unit increase in x_j . If the reduced cost value is negative for any non-basic variable x_j then z can be increased by increasing the value of x_j .

³If no such submatrix exists, artificial variables can be added to create an identity matrix. By assigning these artificial variables a very large negative cost coefficient the simplex algorithm will be forced not to include them in any basic feasible solution (Bazaraa *et al.*, 1990).

Select to increase the value of non-basic variable x_j with the most negative reduced cost value⁴ This is called Dantzig's pivot rule (Gale, 2007). From 2.26 it is clear that the value of $\mathbf{x}_B$ will change if the value of $x_j \in \mathbf{x}_N$ is changed,

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{A}_{\cdot j}x_j. \quad (2.33)$$

Setting $\bar{\mathbf{b}} = \mathbf{B}^{-1}\mathbf{b}$ and $\bar{\mathbf{a}}_j = \mathbf{B}^{-1}\mathbf{A}_{\cdot j}$ the new value of each variable $x_{B_i} \in \mathbf{x}_B$ is,

$$\begin{aligned} \mathbf{x}_B &= \bar{\mathbf{b}} - \bar{\mathbf{a}}_j x_j \\ \begin{bmatrix} x_{B_1} \\ x_{B_2} \\ \vdots \\ x_{B_m} \end{bmatrix} &= \begin{bmatrix} \bar{b}_1 \\ \bar{b}_2 \\ \vdots \\ \bar{b}_m \end{bmatrix} - \begin{bmatrix} \bar{a}_{j_1} \\ \bar{a}_{j_2} \\ \vdots \\ \bar{a}_{j_m} \end{bmatrix} x_j \\ &= \begin{bmatrix} \bar{b}_1 - \bar{a}_{j_1} x_j \\ \bar{b}_2 - \bar{a}_{j_2} x_j \\ \vdots \\ \bar{b}_m - \bar{a}_{j_m} x_j \end{bmatrix} \end{aligned} \quad (2.34)$$

It is given that $\bar{\mathbf{b}} \geq 0$, therefore if $\bar{a}_{j_i} \leq 0$ then $\bar{b}_i - \bar{a}_{j_i} x_j \geq 0$ regardless of how large x_j becomes. If this is the case then the feasible region of the LP problem is unbounded. If $\bar{a}_{j_i} \geq 0$ then $\bar{b}_i - \bar{a}_{j_i} x_j$ will approach zero as x_j is increased. In order to ensure $\mathbf{x}$ remains non-negative, x_j cannot increase beyond the point that will decrease the value of any variable x_{B_i} below zero. The maximum increase in x_j allowed on each x_{B_i} is given by,

$$\begin{aligned} x_{B_i} &\geq 0 \\ \bar{b}_i - \bar{a}_{j_i} x_j &\geq 0 \\ -\bar{a}_{j_i} x_j &\geq -\bar{b}_i \\ \bar{a}_{j_i} x_j &\leq \bar{b}_i \\ x_j &\leq \frac{\bar{b}_i}{\bar{a}_{j_i}} \end{aligned} \quad (2.35)$$

for all x_{B_i} where $\bar{a}_{j_i} \geq 0$. Thus, the largest possible value allowed on x_j , while ensuring all variables remain positive, is

$$x_j = \min \left\{ \frac{\bar{b}_i}{\bar{a}_{j_i}}, i = 1, \dots, m \text{ and } \bar{a}_{j_i} \geq 0 \right\}. \quad (2.36)$$

This is called the minimum ratio test (Gale, 2007). If 2.36 solves to $x_j = \frac{\bar{b}_i}{\bar{a}_{j_i}}$ for $i = k$ then variable x_j will be included in the basis $\mathbf{B}$ at the expense of variable x_{B_k} . The variable x_{B_k} will become a non-basic variable and x_j will become a basic variable. This moves the system to a new extreme point with a higher objective function value. The new basis $\mathbf{B}$ is

⁴Some authors, such as Render *et al.* (2011), select to increase the value of non-basic variable x_j with the greatest positive value for $c_j - z_j$. However, the term reduced cost refers to $-\frac{\partial z(\mathbf{x})}{\partial x_j} = z_j - c_j$

constructed by replacing the columns vector k in the previous basis with column vector j in $\mathbf{A}$. The new matrix $\mathbf{N}$ is all the column vectors in $\mathbf{A}$ not included in the new basis $\mathbf{B}$.

At the new extreme point, the basic feasible solution is,

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} \quad (2.37)$$

and the new object function value is,

$$z(\mathbf{x}) = \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{b}. \quad (2.38)$$

If there exists a non-basic variable x_j in the new $\mathbf{N}$ with a negative reduced cost value, pivot this variable into the basis $\mathbf{B}$ and pivot the variable x_{B_i} , with the smallest positive ratio $\frac{\bar{b}_i}{\bar{a}_{ji}}$, out of the basis $\mathbf{B}$. The simplex algorithm, summarized in Algorithm 2.2, continues until no non-basic variable has a negative reduced cost value. In Appendix B an example LP problem is solved using the simplex algorithm in algebraic form.

Algorithm 2.2 The simplex algorithm (Adapted from Griffon, 2012).

input: An LP problem in standard form with cost vector $\mathbf{c}$, coefficient matrix $\mathbf{A}$ and right-hand side vector $\mathbf{b}$

output: Optimal basic feasible solution and optimal objective function value

begin algorithm

Obtain an initial basic feasible solution by any means

if Initial basic feasible solution exists **then**

Let $\mathbb{J}$ be the set of indices of non-basic variables in $\mathbf{A}$

else

exit

end if

Compute the reduced cost $z_j - c_j$ of each variable x_j with $j \in \mathbb{J}$

while $z_j - c_j \leq 0$ for any $j \in \mathbb{J}$ **do**

Select a non-basic variable x_j with most negative reduced cost value $z_j - c_j$ to become a basic variable ***Datzig's pivot rule*

For the selected variable x_j select the corresponding column vector $\bar{\mathbf{a}}_j$ from $\mathbf{B}^{-1}\mathbf{N}$

Compute $\bar{\mathbf{b}} = \mathbf{B}^{-1}\mathbf{b}$

Find the index i that solves $\min\{\frac{\bar{b}_i}{\bar{a}_{ji}}, i = 1, \dots, m \text{ and } \bar{a}_{ji} \geq 0\}$ ***Minimum ratio test*
 x_j in $\mathbf{A}$ becomes basic and x_i in current $\mathbf{B}$ becomes non-basic. Update $\mathbb{J}$, $\mathbf{B}$ and $\mathbf{N}$

Compute the reduced cost $z_j - c_j$ of each variable x_j with $j \in \mathbb{J}$

end while

Compute the optimal basic feasible solution and the optimal objective function value

end algorithm

2.3.3 Linear programming-based discriminant classifiers

LP can be used to address the discriminant analysis problem. Within an LP based discriminant analysis (LPDA) model the decision variables are the vector of weights $\mathbf{w}$, as well as the class boundary b . Optimal values for these decision variables define the linear classifier

h (Equation 2.21) that best fits the exemplar data set. This linear classifier is called a linear programming-based discriminant classifier (LPDC) in this dissertation.

The instances in the exemplar set might, however, not be linearly separable, therefore the objective of the LPDA model is to find the decision boundary that minimizes the extent of misclassification over the training data set. A solution to the LPDA model is found by applying the simplex algorithm outlined in the previous subsection.

Freed and Glover (1986) proposed the “minimize the sum of deviations” (MSD) model as a solution to the binary classification problem. Within this model the extent of misclassification of an instance $(\mathbf{x}^{(t)}, y^{(t)})$ belonging to group A is measured as the deviation d_t in the discriminant score $\sum_{i=1}^r w_i x_i^{(t)}$ above the group boundary b . Similarly, the extent of misclassification of an instance belonging to group B is measured as the deviation in the discriminant score below the group boundary. The objective of the MSD model is to find $\mathbf{w}^*$ and b^* such that the sum of deviations is minimized over all the input-output patterns $(\mathbf{x}^{(t)}, y^{(t)})$ in the training set with N patterns. Denote the set of all instances in group A as G_0 , denote the set of all instances belonging to group B as G_1 , and let r be the number of attributes in the input pattern $\mathbf{x}$, then the MSD model is formulated as,

$$\underset{\mathbf{w}, b}{\text{minimize}} \quad \sum_{t=1}^N d_t \quad (2.39)$$

$$\text{subject to} \quad \sum_{i=1}^r w_i x_i^{(t)} - d_t \leq b \quad \forall (\mathbf{x}^{(t)}, y^{(t)}) \in G_0, \quad (2.40)$$

$$\sum_{i=1}^r w_i x_i^{(t)} + d_t \geq b \quad \forall (\mathbf{x}^{(t)}, y^{(t)}) \in G_1, \quad (2.41)$$

$$\text{with} \quad d \geq 0, \mathbf{w} \in \mathbb{R}^r, b \in \mathbb{R}, \text{ and } t = 1, \dots, N. \quad (2.42)$$

The MSD model can also be generalized for an n -class classification problem (Freed and Glover, 1981). Let b_j^l and b_j^u be the lower and upper bounds on discriminant scores for instances in set G_j ; and let d_j be the maximum deviation of all instances in group j where the deviation is the extent with which the discrimination score calculated for an observation in group j is greater than b_j^u or lower than b_j^l .

$$\underset{\mathbf{w}, \mathbf{b}}{\text{minimize}} \quad \sum_{t=1}^N d_t \quad (2.43)$$

$$\text{subject to} \quad \sum_{i=1}^r w_i x_i^{(t)} - d_j \leq b_j^u \quad \forall (\mathbf{x}^{(t)}, y^{(t)}) \in G_j \quad (2.44)$$

$$\sum_{i=1}^r w_i x_i^{(t)} + d_j \geq b_j^l \quad \forall (\mathbf{x}^{(t)}, y^{(t)}) \in G_j \quad (2.45)$$

$$b_j^u \leq b_{j+1}^l \quad (2.46)$$

$$\text{with} \quad \mathbf{d} \geq 0, \mathbf{w} \in \mathbb{R}^r, b \in \mathbb{R}^n, j = 1, \dots, (n-1), t = 1, \dots, N \quad (2.47)$$

A parameter optimization algorithm, such as the simplex algorithm, finds the optimal weight vector $\mathbf{w}^*$ and optimal boundary vector $\mathbf{b}^*$. The parallel discriminant boundaries

defined by the optimal feasible solution is used to classify unlabelled instances. The multiple class MSD model has two inherent shortcomings. Firstly, the order of the discriminant values for each group, as specified by constraint 2.46, has to be determined as an outer loop to parameter optimization. Secondly, one discriminant function with multiple group boundaries might not be flexible enough to solve a complex classification task (Freed and Glover, 1981). Transforming the multiple class problem into a set of binary subproblems is therefore recommended.

The MSD model has problems related to unbounded solutions, trivial solutions (the zero solution), improper solutions (the discriminant score of all instances is equal to the group boundary value) and the instability of solutions when using different input pattern transformations (Kwak *et al.*, 2002). More recent models have successfully overcome these problems.

One recent LPDA model, that has been shown to outperform Fisher’s linear discriminant analysis and other LPDA models, especially in the presence of outliers, is the PHASER model (Lame and Moy, 1997; Stam and Ragsdale, 1992). The PHASER model implements a two-phase classification approach. In the first phase, a classification gap is allowed in the range of discriminant scores. Those instances with discriminant scores in this classification gap are not yet deemed classifiable, and are therefore not considered when calculating the extent of misclassification associated with any given discriminant function. During the second phase, the discriminant function is refined by focusing only on separating patterns with discriminant values within the classification gap, while ignoring those outlier instances that have been misclassified in the first phase.

The first phase is modelled as follows,

$$\underset{\mathbf{w}}{\text{minimize}} \quad \sum_{t=1}^N d_t, \quad (2.48)$$

$$\text{subject to} \quad w_0 + \sum_{i=1}^r w_i x_i^{(t)} - d_t \leq 0 \quad \forall \quad (\mathbf{x}^{(t)}, y^{(t)}) \in G_0, \quad (2.49)$$

$$w_0 + \sum_{i=1}^r w_i x_i^{(t)} + d_t \geq 1 \quad \forall \quad (\mathbf{x}^{(t)}, y^{(t)}) \in G_1, \quad (2.50)$$

$$\text{with} \quad d_t \geq 0, \quad \mathbf{w} \in \mathbb{R}^{r+1}, \quad t = 1, \dots, N. \quad (2.51)$$

By including a bias term w_0 , the model allows for a flexible classification gap in the discriminants scores between $0 - w_0$ and $1 - w_0$. Solving the model produces optimal weights $\mathbf{w}^*$. The discriminant function defined by these weights is used to calculate a discriminant score for each of the training samples. Denote the set of instances that is classified as belonging to group A by having a discriminant score of less than zero with the label G'_0 , denote the set of instances classified as belonging to group B by having a discriminant score of more than one with the label G'_1 ; and denote the set of all patterns with a discriminant score between zero and one as G'_f . The set of correctly classified instances in group A and group B can be then denoted as $G_0 \cap G'_0$ and $G_1 \cap G'_1$ respectively.

Using the results from the first phase, the discriminant function is refined to minimize the extent of misclassification of those instances in the classification gap, without sacrificing any correct classifications made in phase 1. Any instances incorrectly classified in phase 1 are ignored in phase 2, as these instances are deemed outliers. By excluding these instances in phase 2, they are precluded from distorting the objective function value. The second phase can be formulated as follows,

$$\underset{\mathbf{w}, b}{\text{minimize}} \quad \sum_{(\mathbf{x}^{(t)}, y^{(t)}) \in G_0 \cap G_f} d_t + \sum_{(\mathbf{x}^{(t)}, y^{(t)}) \in G_1 \cap G_f} d_t, \quad (2.52)$$

$$\text{subject to} \quad w_0 + \sum_{i=1}^r w_i x_i^{(t)} \leq 0 \quad \forall (\mathbf{x}^{(t)}, y^{(t)}) \in G_0 \cap G'_0 \quad (2.53)$$

$$w_0 + \sum_{i=1}^r w_i x_i^{(t)} \geq 1 \quad \forall (\mathbf{x}^{(t)}, y^{(t)}) \in G_1 \cap G'_1 \quad (2.54)$$

$$w_0 + \sum_{i=1}^r w_i x_i^{(t)} - d_t \leq b \quad \forall (\mathbf{x}^{(t)}, y^{(t)}) \in G_0 \cap G_f \quad (2.55)$$

$$w_0 + \sum_{i=1}^r w_i x_i^{(t)} + d_t \geq b \quad \forall (\mathbf{x}^{(t)}, y^{(t)}) \in G_1 \cap G_f \quad (2.56)$$

$$0 \leq b \leq 1 \quad (2.57)$$

$$\text{with} \quad d \geq 0, \mathbf{w} \in \mathbb{R}^{r+1} \quad t = 1, \dots, N. \quad (2.58)$$

The optimal weights $\mathbf{w}^*$ and optimal group boundary b^* found by solving the second phase model define the LPDC used to classify new unlabelled instances.

Except for the empirical comparison of LPDA models by Youssef and Rebai (2007) and Lame and Moy (1997), these models are yet to be used widely on real-world applications. Kwak *et al.* (2002) compare the performance of the k -nearest neighbour algorithm, Fisher's linear discriminant analysis and the PHASER model in the classification of the symptomatic status of HIV-AIDS patients. In this application domain, the PHASER model outperformed the other classification methods on a holdout sample. Falangis (2007) compares the performance of the MSD model with that of the k -nearest neighbour algorithm, Fisher's linear discriminant analysis and neural network models on three credit scoring datasets. Results show that the MSD model perform similar to or better than the neural network models and better than the k -nearest neighbour algorithm. The PHASER model is used to derive the LPDCs used to solve the file fragment classification problem in this dissertation.

2.3.4 Advantages and disadvantage of LPDA

Advantages of LPDA include (Ramlall, 2010; Olson and Shi, 2006; Glover *et al.*, 1988):

- Compared with other machine learning tools for classification, such as decisions trees, k -nearest neighbour classifiers and neural networks, the LPDA approach is simple, direct, and free of statistical assumptions

- Various objective functions can be considered, including assigning different cost coefficients to each deviation variable.
- LPDA models are less prone to over fitting because there are fewer adjustable parameters in the hypothesised decision boundary.
- Optimal solutions to the linear programming problems are found efficiently and exactly for any given data set.
- LPDA models are inherently less sensitive to outliers because they use an L_1 metric (absolute deviation) rather than an L_2 metric (squared deviation) .

The obvious disadvantage of LPDA is that only linear relationships between the variables and the predicted class value are considered. Furthermore, the LPDA models do not extend gracefully to multiple class classification problems.

2.4 k -nearest neighbour algorithm

The MLP neural network classifier and LPDC have been outlined in the previous two sections. In this section the k -nearest neighbour algorithm is briefly introduced because of its popularity in existing literature with regard to file object type identification will be shown in Chapter 3, and because this technique is later proposed for use as a benchmark classifier in Chapter 4.

The principle of the k -nearest neighbour algorithm is that instances belonging to the same class will generally exist in close proximity to one another within the r dimensional space, where each dimension corresponds to one of the r attributes in the feature vector $\mathbf{x}$. The multidimensional space is populated by the reference set of labelled training instances $(\mathbf{x}^{(t)}, y^{(t)})$. A distance measure $D(\mathbf{x}^{(t)}, \tilde{\mathbf{x}})$ is used to measure the proximity between each instance in the reference set and an unlabelled instance $\tilde{\mathbf{x}}$. The unlabelled instance is classified by observing the class labels of its closest k neighbours within the reference set, and assigning a class label by popularity vote or distance weighted popularity vote over the set of observed class labels.

Figure 2.8 shows how the k -nearest neighbour algorithm performs classification. Within the two-dimensional feature space a 3-nearest neighbour algorithm will classify instance $\tilde{\mathbf{x}}_1$ as belonging to group A . This classification is done by observing that all three closest training set instances have the class label A . The instance $\tilde{\mathbf{x}}_2$ is classified as belonging to group B because two out of three neighbouring training instances have the class label B .

The k -nearest neighbour algorithm is called a lazy supervised learning technique because classification is based directly on the reference set, and not on an induced function h . Therefore, classification using this technique can be computationally and memory intensive, especially when the reference set is large. Considering that the computational and memory requirements of the k -nearest neighbour algorithm grow with the number of instances in

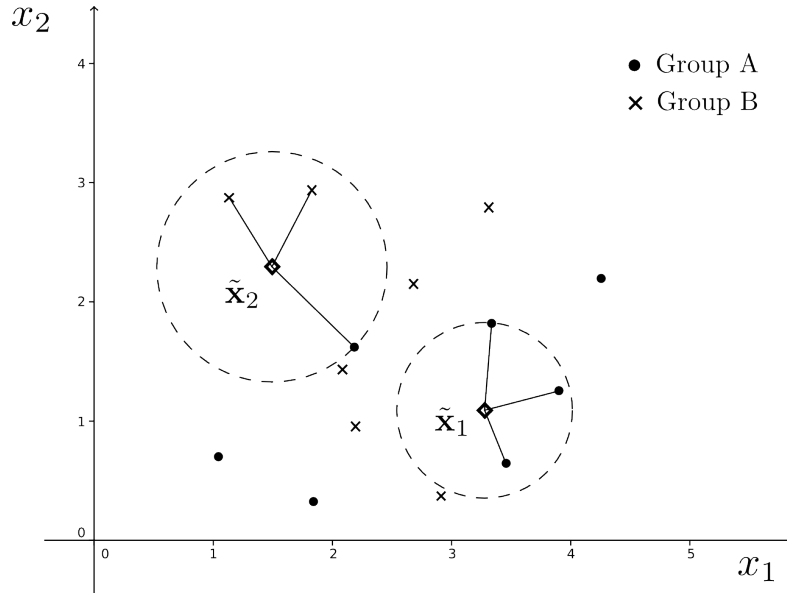


Figure 2.8: The k -nearest neighbour algorithm.

the reference set, several instance selection methods can be applied to reduce the size of the reference set (Reinartz, 2002). Instance selection methods, also known as prototyping methods, try to reduce the number of instances used in the reference set below that of the original training set. Reducing the size of the reference set not only decreases the computational and memory requirements of the k -nearest neighbour algorithm but also filters out the noise in the original training set (Aha *et al.*, 1991).

Prototyping assumes that a small number of instances can be found to represent the entire set of training instances. In general, one prototype per class can be constructed by averaging over all the example instances belonging to the same class. Alternatively, multiple prototypes can be constructed by looking for subsets of homogeneous instances within the set of instances belonging to the same class. Averaging over each identified subset produces a prototype. Clustering, an unsupervised learning technique, can be applied here to find homogeneous groups of instances. An introduction to clustering techniques can be found in Nisbet *et al.* (2009).

The k -nearest neighbour algorithm has the advantage of only having one hyper-parameter, the number of nearest neighbours k , that needs to be optimized. However, it has been shown that this supervised learning technique is intolerant to noisy data, intolerant to redundant features and sensitive to the choice of similarity measure used (Aha *et al.*, 1991). Therefore, though the k -nearest neighbour algorithm is very simple to understand and implement, its performance varies considerably across implementations.

2.5 Ensemble techniques

So far supervised learning methods in which a single hypothesis function h is used to make a prediction have been considered. Ensemble techniques, on the other hand, use a collection

of hypothesis functions to make a collective decision. Ensemble techniques are described in this section for the reason that this dissertation focuses on combining a set of classifiers in an attempt to improve classification accuracy.

Consider a voting system where each voter has the probability p of being correct and a majority vote of all the individual voters' opinions is correct with probability P . The Condorcet jury theorem (Lam, 2000), assuming only two voting outcomes, states that if each voter is at least weakly correct such that $p > 0.5$ and the set of voters is diverse then P approaches 1 as the number of voters in the system increases towards infinity. This principle is exploited by ensemble learning methods: If a diverse collection of weakly correct classifiers, i.e. each classifier can make mistakes, however, not all at the same time, can be constructed, it is possible to achieve highly accurate classification decisions by combining the predictions of the individual classifiers (Woods *et al.*, 1997).

Dietterich (2000a) argues that an ensemble is able to improve classification accuracy beyond that of a single classifier by addressing three problems inherent to single classifiers: the statistical problem, the computational problem and the representational problem. The statistical problem (Figure 2.9(a)) arises when a number of different hypotheses h_i can be fitted to the training data, each with a reasonable empirical loss. It is difficult to choose the single best hypothesis because the empirical loss estimated for each hypothesis may vary over different data sets. Combining the outputs of the individual hypotheses can provide a better generalization of the true function f .

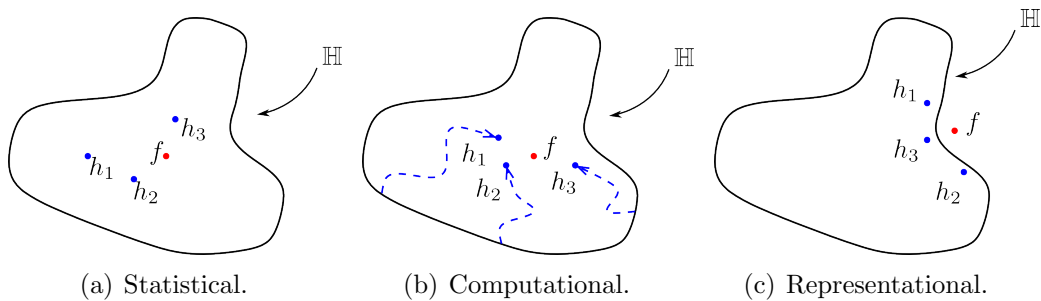


Figure 2.9: Three fundamental reasons why an ensemble of classifiers may work better than a single classifier (Adapted from Dietterich, 2000a).

The computational problem (Figure 2.9(b)) arises from the fact that many parameter search algorithms produce local optimal solutions instead of the global optimal solutions. Again, various different hypotheses h_i can be fitted to the training data, each hypothesis representing a different local optimal solution. Combining the outputs of various “local optimal” hypotheses might provide a better approximation of the “global optimal” hypotheses.

The representational problem (Figure 2.9(c)) arises when the hypothesis space $\mathbb{H}$, in which parameter search is performed, does not contain the true function f . The true function f can then be approximated by combining various hypotheses taken from the hypothesis space.

In agreement, Kuncheva (2004) uses a simulation experiment to show that an ensemble of classifiers can improve classification accuracy considerably beyond that of the best single base classifier in the ensemble. In one simulation for example, an ensemble of five negatively correlated classifiers achieved a classification accuracy 20 percent higher than the single best individual classifier included in the ensemble. Kuncheva (2004) concludes that an ensemble of negatively correlated classifiers provides greater improvements in classification accuracy than an ensemble of independent classifiers or an ensemble of positively correlated classifiers.

Two important components of any ensemble can be identified. Firstly, the technique used to generate diverse classifiers and secondly, the technique used to combine the outputs of the various individual classifiers into one output (Xu *et al.*, 1992).

2.5.1 Techniques for generating diverse classifiers

An ensemble has to be constructed from classifiers that are as marginally correct; however, if they make classification errors these errors should be made on different objects. Several mechanisms to construct a set of diverse classifiers have been proposed, including (Kuncheva, 2004; Woods *et al.*, 1997)

- manipulating the training data;
- manipulating the induction algorithm;
- partitioning the search space;
- using multiple model representations; or
- transforming the target variable.

Bagging and boosting are two popular techniques used to construct diverse classifiers. Both these techniques create a temporary training set for each induced classifier by sampling the original training set. In bagging, a temporary training set is constructed by drawing N instances, randomly with replacement, from the original training set with N instances (Breiman, 1996). Each temporary training set is unique because some instances from the original training set are included more than once, while other instances from the original training set are not included at all. A unique classifier is induced from each temporary training set sampled.

Similarly, in boosting, a temporary training set is also sampled for every classifier included in the ensemble (Freund and Schapire, 1996). However, in this method the temporary data sets are not constructed independently from one another as in bagging. In boosting, each induced classifier is used to adapt the weighting of each instance in the training data set before drawing the temporary data set over which the next classifier will be induced. Instances for which the target variable was accurately predicted by the previously induced classifier are weighted less while instances for which the target variable was incorrectly predicted are weighted more. In this way, each newly induced classifier focusses more on those instances that are harder to learn.

As an alternative to re-sampling the training set, the available data can simply be partitioned into various disjoint subsets. Each subset is then used to induce one classifier to be included in the ensemble. Chawla *et al.* (2004) propose dividing a large training set into a large number of disjoint and small partitions. Hundreds of classifiers are then constructed by training one classifier over each partition. This technique has been shown to be fast, accurate and very scalable.

Various techniques exist with which to manipulate the induction algorithm in an attempt to produce diverse classifiers. For example, a committee of neural networks is formed by training neural networks with different hyper-parameter specifications (Maclin *et al.*, 1995; Partridge and Yates, 1995). Different neural networks can also be obtained by altering the initial connection weights used in the back propagation algorithm (Pollack, 1990). In this way, the hypothesis space is transversed from different starting points and is therefore likely to converge to different local optimal solutions. Breiman (2001) and Dietterich (2000b) propose randomizing the internal decisions process in the induction algorithm used to induce classifiers.

The search space can be partitioned into several subspaces by simply making a different subset of the available features available to each classifier. The induction algorithm produces the hypothesis that best fits the training data described by the features provided to it. The features given to each classifier are selected randomly or the feature set can be decomposed into disjoint subsets. For example, Ho (1998) built an ensemble of decision trees by inducing multiple decision trees, each using only a randomly selected subset of the available features. This technique has been shown to be appropriate in classification tasks where a large number of features is available (Rokach, 2008). Tsymbal *et al.* (2003) showed that an ensemble of neural networks, each using different subsets of features, is able to achieve expert levels of accuracy when tested on a scientific image analysis task.

Classifiers with different model representations can also be combined. For example, Saha and Ekbal (2013) show how an ensemble of multiple base classifiers with heterogeneous model representations can improve significantly over a single base classifier when applied to named entity recognition (a subproblem in natural language processing). Woods *et al.* (1997) show that a marginal improvement in accuracy can be achieved when using different supervised learning techniques as base classifiers in an ensemble. Similarly, Lee and Elder (1997) compare the accuracy of four base classifiers, each with a different model representation, against an ensemble of these classifiers on five application domains. They conclude that the classification accuracy of the ensemble improved over the single best classifier in some application domains while not in others. A comparative study by Duin and Tax (2000) found that combining the outputs of one type of classifier constructed over many feature sets is more effective than combining the results of various types of classifiers on the same feature set.

Related to ensembles that combine base classifiers with different model representations are the so-called hybrid supervised learning techniques. Such a technique integrates several

model representations into one. For example, Lazkano and Sierra (2003) propose a hybrid classifier that combines a Bayesian network algorithm with the nearest neighbour algorithm; Zhou and Chen (2002) show how decision trees and neural networks can be combined and Lee *et al.* (2012) combine naive Bayesian classifiers and support vector machines.

Lastly, diverse models can be constructed by transforming the target variable in the training data set. In a technique called error correction output coding, the classification task is decomposed into a set of binary classifiers that is trained to discriminate between two pseudo classes (Dietterich and Bakiri, 1995). Each pseudo class can represent an actual class or a combination of actual classes. A n class classification problem can be decomposed into $2^{(n-1)} - 1$ unique dichotomies. An appropriate number of these dichotomies is chosen to form a matrix of codewords. For each of the chosen dichotomies, a classifier is induced after the class label variable is recoded. The output pattern produced by the ensemble of classifiers is compared to each codeword with distance measure. An unlabelled instance is assigned the class label associated with the closest codeword.

For example, the complete set of dichotomies for a 4-class problem is given in Table 2.1. Assume a classifier $h_i \rightarrow \{1, 0\}$ is constructed for each dichotomy D_i with $i = 1, \dots, 7$. Each classifier is used to assign a label to an unlabelled instance $\tilde{\mathbf{x}}$. Say the output pattern of the ensemble is $(h_1(\tilde{\mathbf{x}}), h_2(\tilde{\mathbf{x}}), \dots, h_7(\tilde{\mathbf{x}})) = (0, 1, 1, 0, 1, 0, 1)$. The instance $\tilde{\mathbf{x}}$ will be assigned class label 3 because the ensemble's output pattern is closest to the codeword of class 3 when measured by the Hamming distance⁵.

Table 2.1: The code matrix for a 4-class problem with all possible dichotomies used.

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
Codeword for class 1	0	0	0	1	0	1	1
Codeword for class 2	0	0	1	0	0	0	0
Codeword for class 3	0	1	0	0	1	0	1
Codeword for class 4	1	0	0	0	1	1	0

As part of the empirical investigation of this dissertation, several techniques are used to introduce model diversity. Firstly, by inducing various MLP neural network classifiers, each having a different hyper-parameter configuration, secondly, by training LPDCs on different feature sets and thirdly, by combining MLP neural network classifiers and LPDCs.

2.5.2 Techniques to combine the output of multiple classifiers

After constructing a diverse set of classifiers the outputs of the individual classifiers are combined into a single output. Output selection and output fusion are two general approaches to combine the output of multiple classifiers.

⁵The Hamming distance between the output pattern of the ensemble and a codeword is the number of positions at which the corresponding bits are different (Hamming, 1950)

In output selection, a higher level classifier is used to decide which base classifiers in the ensemble should be used for the classification of a given input pattern. For example, Jacobs *et al.* (1991) induce a set of neural network classifiers, each with a different hyper-parameter specification. Due to the different hyper-parameter specification, each neural network is better suited to predict values in different regions of the input space. Another neural network, called a gating network, uses the same input vector as the base classifiers, to determine which one of the individual network's predictions to use as a final prediction. Gutta *et al.* (2000) built an ensemble of radial basis functions and use inductive decision trees and support vector machines as a gating classifier for deciding which of the classifiers should be used to determine the final classification output.

In output fusion, the outputs provided by the individual classifiers are combined into an ensemble output. A set of discrete class labels can be fused using majority vote (Kuncheva, 2004). In this combination scheme an unlabelled instance is given the class label most frequently assigned to it in the set of base classifiers. If each base classifier produces a probability estimate for each class, then the set probability estimates can be fused by summing across all base classifiers. The unlabelled instance is given the class label with the highest aggregated probability estimate. Kittler *et al.* (1998) compared various such techniques theoretically and concluded that the sum rule is theoretically optimal while Duin and Tax (2000) showed empirically that there is no overall best combination rule.

As an alternative to using a static combination rule, a meta-level classifier can be used to combine the class probability estimates of each individual classifier into a final prediction (Lee and Elder, 1997; Ting and Witten, 1999). Such a meta-level classifier uses the outputs of all the individual classifiers in the ensemble as a training set, and induces a high level classification function that can be used to assign a class label. Alternatively, Langdon *et al.* (2002) showed how genetic programming can be applied to find the most optimal combination rule for the given training data set.

In this dissertation model fusion is used to combine the output of the individual classifiers into a single output.

2.5.3 Advantages and disadvantages of ensembles

It might be difficult to find a single classifier that performs as well as an ensemble of classifiers, however, the cost for improved performance is a loss in model transparency and comprehensibility (Kotsiantis *et al.*, 2006). Furthermore, all the classifiers in the ensemble have to be included in the implemented classification system. In order to classify a new instance, all the individual classifiers in the ensemble need to be processed, and some combination scheme needs to be applied. This requires greater memory and computational resources. From a modelling perspective, building a good ensemble can be time intensive and requires a greater effort (Nisbet *et al.*, 2009).

Most notably, ensemble techniques have been criticized for solving the problem of finding

the best feature set and best classifier, by introducing the problem of finding the best set of classifiers and the best combination technique (Ho, 2002). However, ensembles have been shown to be very successful in solving complex problems, and are therefore applied in this dissertation.

2.6 Chapter summary

As part of the knowledge discovery and data mining process, this chapter developed a general understanding of supervised learning as relevant to classification problems. Not only were supervised learning and the classification problem defined but three supervised learning techniques were also outlined. The MLP neural network classifier and LPDC will be proposed as a solution to the file fragment classification problem in Chapter 4 and Chapter 5. The k -nearest neighbour algorithm will be implemented in Chapter 4 and Chapter 5 as a benchmark.

The concept of using an ensemble of supervised learning classifiers for improved classification accuracy was outlined. It was noted that ensembles of marginally accurate but diverse classifiers are usually better at classification compared to individual classifiers. One method to build a diverse ensemble is to combine classifiers with different model representations. The rationale behind this dissertation's research hypothesis is that the MLP neural network classifier and the LPDC have different model representations and should therefore be good candidates to participate in an ensemble.

The terminology introduced in this chapter is the foundation for the literature review regarding the use of supervised learning techniques in the domain of file object type identification that follows in Chapter 3.

LITERATURE REVIEW OF FILE OBJECT TYPE IDENTIFICATION

III

In the previous chapter supervised learning has been defined and three supervised learning techniques have been outlined. In this chapter, the focus moves from developing a general understanding of supervised learning towards developing an understanding of supervised learning techniques as applied to file object type identification. Chapter 3 constitutes a major contribution towards the literature review phase of this dissertation.

Five broad topics relating to any supervised learning endeavour can be identified from the discussion in the previous chapter: the object being observed; the input pattern used to characterize the object under observation; the size of the classification problem; the exemplar set being used; as well as the model representation chosen. A systematic summary of the existing literature relating to file type identification and file fragment type identification (collectively referred to as file object type identification) is provided in tabular form under these five headings. A tabular form is used in an attempt to highlight the similarities and differences as well as in an attempt to avoid repetition. This systematic presentation of the literature also outlines the variety in the literature with regard to file object type identification.

3.1 Instance under observation

The file object type identification problem revolves around assigning one of various predetermined file types to some observed file object. In the literature, the object under observation has either been a complete file, a file fragment, or specifically a file header fragment. The chosen size of a file fragment also differs across studies.

Though this distinction might seem subtle, the file object under observation has an implication when interpreting the research results. In general, supervised learning techniques have been more successfully used for file type identification than for file fragment type identification. The latter problem is more difficult because the pattern that underlies a file type's common data structure varies depending on the position of a file fragment in a file (Hall and Davis, 2006). Furthermore, identification of file fragments belonging to container file formats, such as **pdf** files and **zip** files, are particularly problematic. Such container files allow different base files to coexist in one logical computer file. Identification of these file types on a fragment level is troublesome because a file fragment might, for all practical reasons, contain data consistent with another file type.

The various file objects for which classification functions have been induced in the existing literature is summarized in Table 3.1. For example, this table shows that Li *et al.* (2010)

focused on the classification of a 4096-byte file fragment into one of the predetermined file types.

Table 3.1: Summary of various file objects under observation.

File object	Literature
4096-Byte file fragments	Li <i>et al.</i> (2010); Veenman (2007); Karresand and Shahmehri (2006a,b)
1024-Byte file fragments	Conti <i>et al.</i> (2010); Calhoun and Coles (2008)
512-Byte file fragments	Sportiello and Zanero (2011); Mayer (2011); Axelsson (2010); Harris (2007)
File header fragments	Li <i>et al.</i> (2006, 2005); Calhoun and Coles (2008)
Complete files	Ahmed <i>et al.</i> (2010); Cao <i>et al.</i> (2010); Kattan <i>et al.</i> (2010); Amirani <i>et al.</i> (2008); Hall and Davis (2006); Li <i>et al.</i> (2005); McDaniel and Heydari (2003); McDaniel (2001)

3.2 The input pattern

Within the supervised learning endeavour, the objects under observation are characterized by some set of attributes. The chosen collection of attributes forms the input pattern $\mathbf{x}$ associated with each instance in the exemplar set. The attributes included in the input pattern are chosen in a way that is relevant to the task at hand. Various attributes have been proposed with which to describe a file object as summarized in Table 3.2. From this table, it can be inferred that the byte frequency histogram is the most frequently used attribute with which to characterize a file object. Note that various authors, such as McDaniel and Heydari (2003) use more than one attribute in their attempt to solve the file object type identification problem.

Table 3.2: Summary of attributes used to describe a file object.

Attribute	Description	Literature
Byte frequency histogram (1-gram model)	A file object is a stream of byte values, where each byte value is a numerical token in the range of 0 to 255. A file object’s byte frequency histogram (BFH) is computed by counting the number of occurrences of each possible byte value within the byte stream. The result is an array of 256 frequency values. The BFH, also known as a 1-gram model, is part of the broader family of characteristics called n -gram models. Examples of BFHs can be found in Appendix D.	Cao <i>et al.</i> (2010); Ahmed <i>et al.</i> (2010); Li <i>et al.</i> (2010); Amirani <i>et al.</i> (2008); Veenman (2007); Karresand and Shahmehri (2006a,b); Li <i>et al.</i> (2005); McDaniel and Heydari (2003); McDaniel (2001)

Table continues on next page.

Table 3.2: Summary of attributes used to describe a file object (continued).

Attribute	Description	Literature
Gram frequency histogram (n -gram model)	A n -gram model is the frequency histogram of each possible n -gram in a stream of tokens (Russel and Novig, 2010). To compute the n -gram model for a file object, count the number of occurrences of each unique sequence of n consecutive bytes within the byte stream. The result is an array of 256^n frequency values.	Mayer (2011); Cao <i>et al.</i> (2010); Calhoun and Coles (2008); Li <i>et al.</i> (2006)
Cross-correlation histogram	The cross-correlation histogram tracks the pairwise correlation in the number of occurrences of byte values within a byte stream. The result is an array of 256×256 correlation values.	McDaniel and Heydari (2003); McDaniel (2001)
File header-trailer byte frequency histogram	The file header-trailer byte frequency histogram tracks the frequency with which each possible byte value occurs at a specific position within a file. This method only looks at the first n_h byte position and the last n_t bytes in a file. The result is a $256 \times n_h + 256 \times n_t$ array.	McDaniel and Heydari (2003); McDaniel (2001)
Rate of change histogram	The rate of change histogram is the set of frequency values for all possible absolute differences between two consecutive byte pairs. The absolute difference between byte values can range from 0 if two consecutive bytes have the same byte value, to 255 if the one byte has value 255 and the other has value 0. Counting the number of occurrences of each absolute difference between byte values produces an array with 255 frequency values.	Karresand and Shahmehri (2006a,b); Li <i>et al.</i> (2005)
All byte values	The complete file object can be used as input pattern. A n byte file object will constitute a feature vector $\mathbf{x}$ with n attributes, each attribute corresponding to the value of each byte in the file object.	Axelsson (2010); Kattan <i>et al.</i> (2010); Calhoun and Coles (2008); Harris (2007); Karresand and Shahmehri (2006a)
Statistical moments	Statistical moments of all byte values within the file object, including the average of byte values, the kurtosis of byte values, the standard deviation of byte values, the distribution of averages and the distribution of standard deviations.	Conti <i>et al.</i> (2010); Calhoun and Coles (2008); Veenman (2007); Li <i>et al.</i> (2005)
Byte stream entropy	Shannon’s entropy measures the randomness or uncertainty in a stream of tokens (Shannon, 2001). A byte stream with a predictable structure would have low entropy, while a byte stream in which one byte is relatively independent of another will have a high entropy value. The entropy over all possible byte values v in a file object V is defined as, $H_V = - \sum_{v=0}^{255} p(v) \log_2 p(v)$ where the frequency of each byte value v in the file object V is used to estimate the probability $p(v)$.	Conti <i>et al.</i> (2010); Veenman (2007)

Table continues on next page.

Table 3.2: Summary of attributes used to describe a file object (continued).

Attribute	Description	Literature
Compressibility	Compressibility is related to entropy, however, here the Lempel–Ziv–Welch compression algorithm is used to estimate the randomness in data (Welch, 1984). Information sparse files will have a high compression ratio, because a compression algorithm will find many repeated sequences in the file to compress. A file with high information content will compress badly as a compression algorithm will find few repeated sequences to compress.	Veenman (2007); Hall and Davis (2006)
Hamming weight of bits	Here, the file object is seen as a bit stream. The hamming weight of bits is calculated as the number of one-bits in the file object divided by the total number of bits in the file object.	Conti <i>et al.</i> (2010)
Chi-square fit statistic	The chi-square statistic can be used to summarize how closely the observed distribution of byte values in a file object matches a uniform distribution of byte values. The chi-square value is defined as $\chi^2 = \sum_{i=0}^r \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$ Using the chi-square value the chi-square fit statistic is calculated as the upper probability of the chi-square distribution. For a description of the Chi-square fit statistic see Rice (2007).	Conti <i>et al.</i> (2010)
Sliding window plots of various statistics	A sliding window plot is constructed by computing various statistics for a fixed number of consecutive bytes at various intervals across the file object. Measurements are taken at approximately equal intervals across the file object, and the set of measurements produce a sliding window plot used to characterize a file. Various statistics, such as those defined above, can be calculated for each block of bytes.	Moody (2008); Erbacher and Mulholland (2007); Hall and Davis (2006)

3.3 The size of the classification problem

The size of the classification problem is determined by the formulation of the classification problem and by the number of file types n included in the study. The multiple class classification problem was distinguished from the binary class classification problem in Section 2.1.1. It was noted that a multiple class classification problem is essentially more challenging than a binary class classification problem, since the induced classifier must learn how to distinguish between a larger number of classes. The probability of making a classification error grows as the number of classes in the classification problem increase.

The file object type identification problem has been approached as a multiple class, one-against-all binary and one-against-one binary classification problem. A summary of the problem formulations in the literature is provided in Table 3.3. It should be noted that of the six authors that attempted a binary decomposition of the file fragment identification

problem, only Li *et al.* (2010) attempted to combine the binary classification subproblems into a solution for the multiple class problem.

Table 3.3: Summary of the classification problem formulations.

Decomposition	Description	Literature
Multiple class (No decomposition)	To distinguish n file types from one another, a classification function h is induced to approximate a function $f \rightarrow \{G_0, \dots, G_{n-1}\}$, where G_i is the class label associated with file type i , for $i = 1, \dots, n$. The induced classifier can assign a file object of unknown type to any of the n file types included.	Mayer (2011); Ahmed <i>et al.</i> (2010); Axelsson (2010); Cao <i>et al.</i> (2010); Kattan <i>et al.</i> (2010); Li <i>et al.</i> (2010); Amirani <i>et al.</i> (2008); Harris (2007); Veenman (2007); Hall and Davis (2006); Li <i>et al.</i> (2005); McDaniel and Heydari (2003); McDaniel (2001)
One-against-all	To distinguish file type G_i from all other file types, a classification function h is induced to approximate a function $f \rightarrow \{\text{positive}, \text{negative}\}$, where positive corresponds to a file object belonging to the file type with label G_i and negative corresponds to a file object not belonging to the file type with label G_i . The induced classifier can assign a file object of unknown type either as belonging to file type G_i or not belonging to file type G_i .	Sportiello and Zanero (2011); Veenman (2007); Karresand and Shahmehri (2006a,b); Li <i>et al.</i> (2006)
One-against-one	To distinguish file type G_i from file type G_j where $i \neq j$, classification function h is induced to approximate a function $f \rightarrow \{G_i, G_j\}$ where G_i and G_j are the class labels associated with file type i and j respectively. The induced classifier can assign a file object of unknown type as either belonging to file type G_i or file type G_j .	Li <i>et al.</i> (2010); Calhoun and Coles (2008)

Apart from using different problem formulations, each study reviewed considered different file types, as well as a different number of file types. Together with the formulation of the classification problem (multiple class or binary class), the number of file types used in each study should be considered when trying to interpret and compare research results. For example, the prior probability of identifying a file object correctly as belonging to one of five different file types is 20 percent, which is double that of identifying a file object correctly as belonging to one of ten different file types. The set of file types considered also influences the results as some file types are more difficult to distinguish from one another. The effect of container files has already been outlined in Section 3.1.

Table 3.4 summarizes the total number of file types and what file types were included in each study surveyed. For brevity, only the popular file types are listed. For example, Li *et al.* (2010) considered the jpeg, pdf, mp3 and dll file types in their study. The jpeg file type has been most frequently included in the various studies surveyed.

Table 3.4: Summary of popular file types considered.

Literature	Nr. of file types	jpeg/jpg	pdf	gif	doc/docx	bmp	exe	html	xls/xlsx	txt/text	ppt/pptx	mp3	zip	dll	png	ps	gz
Mayer (2011)	28	1	1	1	1	1		1	1	1	1		1		1	1	1
Sportiello and Zanero (2011)	9	1	1	1	1	1	1				1	1					
Li <i>et al.</i> (2010)	4	1	1									1		1			
Ahmed <i>et al.</i> (2010)	10	1	1	1	1		1	1	1	1		1					
Axelsson (2010)	28	1	1	1	1	1		1	1	1	1		1		1	1	1
Cao <i>et al.</i> (2010)	4	1	1		1								1				
Conti <i>et al.</i> (2010)		14 primitive data types															
Kattan <i>et al.</i> (2010)	5	1	1	1			1			1							
Amirani <i>et al.</i> (2008)	6	1	1	1	1		1										
Moody (2008)	8	1				1	1	1	1	1				1			
Calhoun and Coles (2008)	4	1	1	1		1											
Veenman (2007)	11	1	1	1	1	1	1	1			1		1				
Harris (2007)	5	1		1		1									1		
Li <i>et al.</i> (2006)	5	1	1		1				1								
Karresand and Shahmehri (2006a)	52	1	1	1	1	1	1	1				1	1	1	1	1	
Karresand and Shahmehri (2006b)	49	1	1	1	1	1	1	1				1		1	1	1	
Hall and Davis (2006)		Two datasets with 454 and 108 file types respectively. File types not listed.															
Li <i>et al.</i> (2005)	8	1	1	1	1		1		1		1			1			
McDaniel and Heydari (2003)	30	1	1	1	1	1	1	1	1	1	1	1	1			1	1
Total		17	15	13	12	10	10	8	7	6	6	6	6	5	5	5	3

3.4 Exemplar set and file types

In the existing research, the set of exemplar file objects used in the data mining endeavour was either sourced from the author’s personal computer, by querying public search engines, or by using the publicly available file corpus provided by Garfinkel *et al.* (2009). Fetherston and Gollins (2012) argue that using a publicly available corpus of file objects allows new file object type identification tools to be evaluated in an objective and repeatable manner. Furthermore, the confidence in any results reported is increased if a publicly available file object corpus is used because other researchers are able to conduct their own validation and

testing on the same file objects. The sources of exemplar files in the literature surveyed is summarized in Table 3.5.

Table 3.5: Summary of the various sources of exemplar file objects.

Source of exemplar file objects	Literature
Not given	Li <i>et al.</i> (2010); McDaniel and Heydari (2003); McDaniel (2001); Calhoun and Coles (2008); Cao <i>et al.</i> (2010); Kattan <i>et al.</i> (2010)
Personal files	Karresand and Shahmehri (2006b,a)
Public (search engines)	Li <i>et al.</i> (2006); Veenman (2007); Amirani <i>et al.</i> (2008); Harris (2007); Li <i>et al.</i> (2005); Sportiello and Zanero (2011)
Govdoc dataset (Garfinkel <i>et al.</i> , 2009)	Mayer (2011); Axelsson (2010)
Various sources	Ahmed <i>et al.</i> (2010); Conti <i>et al.</i> (2010); Hall and Davis (2006)

3.5 Model representation

Model representation involves choosing a family of models to be considered. The chosen model representation defines the hypothesis space $\mathbb{H}$ from which a classification function can be induced. In dealing with the file object identification problem, many different model representations have been used to map the file object input pattern to a file type, and each model has been tested with varying success. The choice of the model representation not only defines the hypothesis space but also which learning algorithms can be used to find a good hypothesis in the hypothesis space. A summary of the classification models used in the existing literature is provided in Table 3.6.

Table 3.6: Summary of classification models used.

Model representation	Literature
K-nearest neighbour algorithm	Axelsson (2010); Cao <i>et al.</i> (2010); Ahmed <i>et al.</i> (2010); Conti <i>et al.</i> (2010); Kattan <i>et al.</i> (2010); Hall and Davis (2006); Karresand and Shahmehri (2006a,b); Li <i>et al.</i> (2005); McDaniel and Heydari (2003); McDaniel (2001)
Linear discriminant analysis	Calhoun and Coles (2008); Veenman (2007)
Support vector machines	Sportiello and Zanero (2011); Li <i>et al.</i> (2010, 2006)
Artificial neural network	Ahmed <i>et al.</i> (2010); Harris (2007); Amirani <i>et al.</i> (2008)

Each method in Table 3.6 is described in more detail in the remainder of this section. It should be clear that due to the complexity of the problem domain a direct comparison of results is of limited value because no two studies use similar data sets, include the same file types or number of file types, nor report the results for the same classification problem formulation. Only general results are therefore provided in each subsection.

3.5.1 k -nearest neighbour algorithm

The k -nearest neighbour algorithm has been introduced in Section 2.4. From Table 3.6 it follows that this supervised learning technique has been the most popular approach used with regard to the file object type identification. Within the k -nearest neighbour algorithm a file object with unknown file type is compared to a file object in a reference set by means of some distance measure. The reference set contains exemplar instances of every file type under observation.

As stated in Section 2.4 the k -nearest neighbour algorithm is very sensitive to the reference set used, the value chosen for hyper-parameter k , as well as the distance measure used. It is therefore not unexpected that each study surveyed differs in terms of the instance reduction methods used, the various distance measures applied as well as the various values chosen for the hyper-parameter k .

There have been three approaches with regard to selecting instances to form the reference set. Firstly, a single prototype per file type is constructed by averaging over the input pattern of all exemplar instances with a common file type. The set of prototypes forms the reference set. Secondly, multiple prototypes per file type are constructed by applying k -means clustering over all exemplar instances with the same file type. A prototype is constructed as the average over the input patterns of all instances in a given cluster. Again, the set of prototypes forms the reference set used for classification. Thirdly, no prototypes are constructed, and therefore all available exemplar instances are included in the reference set. In each case, a file object with unknown type is assigned the file type of the closest prototype in the reference set.

A summary of the different reference sets used is given in Table 3.7. For example, in this table it shows that Li *et al.* (2005) applied a nearest neighbour algorithm using multiple prototypes per file type as reference set. Ahmed *et al.* (2010) also use multiple prototypes per file type as reference set, but apply the k -nearest neighbour algorithm for various values of k . When looking at the results in Table 3.9 that follows later in this subsection, it would seem that classification using prototypes as reference set is less accurate than classification using all available exemplar instances as reference set.

Table 3.7: Summary of reference set used within the k -nearest neighbour model.

Reference set	Decision rule	Literature
Single prototypes per file type	1-nn	Cao <i>et al.</i> (2010); Ahmed <i>et al.</i> (2010); Conti <i>et al.</i> (2010); Kattan <i>et al.</i> (2010); Karresand and Shahmehri (2006b,a); Li <i>et al.</i> (2005); McDaniel and Heydari (2003); McDaniel (2001)
Multiple prototypes per file type	1-nn	Li <i>et al.</i> (2005)
Multiple prototypes per file type	k -nn	Ahmed <i>et al.</i> (2010)
No prototypes. All exemplar file objects used as reference set	1-nn	Li <i>et al.</i> (2005)
No prototypes. All exemplar file objects used as reference set	k -nn	Kattan <i>et al.</i> (2010); Axelson (2010)

Various distance measures have been employed within the k -nearest neighbour algorithm to measure the similarity between an input pattern with unknown file type and each pattern in the reference set. The various distance measures used are summarized in Table 3.8. Each distance measure is defined for a given input pattern vector $\tilde{\mathbf{x}}$ and a prototype pattern vector $\mathbf{x}$ each with r attributes, where σ_i is the standard deviation in the i th attribute of the prototype pattern $\mathbf{x}$ calculated from all the instances used to construct this prototype.

Table 3.8: Distance measures used with the k -nearest neighbour algorithm.

Distance measure	Description	Literature
Manhattan (point-to-point delta)	$D(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{i=0}^r x_i - \tilde{x}_i $	Conti <i>et al.</i> (2010); Hall and Davis (2006); Li <i>et al.</i> (2005)
Euclidean	$D(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{i=0}^r \sqrt{(x_i - \tilde{x}_i)^2}$	Conti <i>et al.</i> (2010)
Cosine	$D(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{\mathbf{x} \cdot \tilde{\mathbf{x}}}{ \mathbf{x} \tilde{\mathbf{x}} }$	Ahmed <i>et al.</i> (2010); Cao <i>et al.</i> (2010); Karresand and Shahmehri (2006b)
Weighted quadratic	$D(\mathbf{x}, \tilde{\mathbf{x}}) = \sum_{i=0}^r \frac{(x_i - \tilde{x}_i)^2}{\sigma_i}$	Karresand and Shahmehri (2006a,b)
Simplified Mahalanobis (also known as normalized Euclidean)	$D(\mathbf{x}, \tilde{\mathbf{x}}) = \sqrt{\sum_{i=0}^r \frac{(x_i - \tilde{x}_i)^2}{\sigma_i^2}}$	Ahmed <i>et al.</i> (2010); Li <i>et al.</i> (2005)

Table continues on next page.

Distance measures used with the k -nearest neighbour algorithm (continued).

Distance measure	Formula/Description	Literature
Normalized compression	If $C(\mathbf{x})$ and $C(\tilde{\mathbf{x}})$ are the compressed length of data vector $\mathbf{x}$ and $\tilde{\mathbf{x}}$ by some compression algorithm, and $C(\mathbf{x}, \tilde{\mathbf{x}})$ is the compressed length of the concatenation of data vector $\mathbf{x}$ and $\tilde{\mathbf{x}}$, then the normalized compression distance between data vector $\mathbf{x}$ and $\tilde{\mathbf{x}}$ is defined as, $D(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{C(\mathbf{x}, \tilde{\mathbf{x}}) - \min(C(\mathbf{x}), C(\tilde{\mathbf{x}}))}{\max(C(\mathbf{x}), C(\tilde{\mathbf{x}}))}$	Axelsson (2010)
Pearson's rank order correlation	This method calculates how well points in two vectors correlate, therefore tracking how well the values of the data points in two vectors rise and fall together. A higher correlation implies the two vectors have similar form. A small or negative correlation value indicates that the two vectors do not have similar form. For a description of Pearson's rank order correlation see Rice (2007).	Hall and Davis (2006)

Empirical results reported for the k -nearest neighbour algorithm vary. A brief summary of results is given in Table 3.9.

Table 3.9: Summary of results when applying the k -nearest neighbour algorithm.

Literature	Results
Ahmed <i>et al.</i> (2010)	Ranked the attributes in the byte frequency histogram according to frequency values. Approximately the top 10 percent of the ranked attributes was used to calculate the distance measure. General accuracies of approximately 70 percent and 65 percent were reported respectively when using multiple prototypes per file type and a single prototype per file type as reference set.
Axelsson (2010)	Solved the 28-class problem with a classification accuracy varying between 36 to 32 percent as k values were varied from 1 to 10.
Cao <i>et al.</i> (2010)	Proposed using a feature selection evaluation function to rank the attributes in the 1-gram and 2-gram frequency histogram. The evaluation function measures both how frequent a gram is encountered within files of the same file type, as well as how distributed this frequency is across files of the same file type. Through experimentation 100 1-grams and 256 2-grams were found to be optimal predictors of file types. Classification accuracies of 67.44 percent and 90.34 percent were reported using the 1-gram model and the 2-gram model respectively for the 4-class problem.
Conti <i>et al.</i> (2010)	Attempted to classify file fragments into 14 primitive data types. The data types included random, encrypted, compressed (6-subtypes), encoded (2-subtypes), machine code (2-subtypes), bitmap and text data. A classification accuracy of 63 percent was reported.

Table continues on next page.

Summary of results when applying the k -nearest neighbour algorithm (continued).

Literature	Results
Kattan <i>et al.</i> (2010)	Used genetic programming to extract two features from a file's byte stream. The objective of the genetic program was to find the two features that optimize the performance of the k -nearest neighbour algorithm in the two dimensional feature space. A classification accuracy of 85, 88, 85 and 70 percent was reported for the 2-class, 3-class, 4-class and 5-class file type identification problem respectively.
Karresand and Shahmehri (2006b)	Average detection rate of <code>jpeg</code> file fragments was 97.90 percent. It should be noted that the high <code>jpeg</code> detection rate reported for this method is achieved by including a wrapper that identifies file fragments with high frequency <code>jpeg</code> specific markers (Karresand and Shahmehri, 2006b).
Karresand and Shahmehri (2006a)	In the various experiments performed, the detection rate of <code>jpeg</code> file fragments was at least 86 percent. For <code>exe</code> file fragments the detection rates were between 45 and 85 percent and for <code>zip</code> file fragments the detection rates were at most 13 percent. Interestingly, in one experiment the output of two proposed methods was combined using an exclusive or operator. The improvement in the classification accuracy achieved by combining the two approaches was negligible.
Hall and Davis (2006)	A file fragment was reported as correctly detected if the correct file type was encountered in any of the 10 closest neighbouring reference set file fragments. No average detection rate was offered. However, the detection rate for each file type given varies considerably across the two data sets used.
Li <i>et al.</i> (2005)	Reported a classification accuracy of 82 percent for the 5-class problem using a single prototype per file type. Using ten prototypes per file type an accuracy of 89 percent was reported. When using the complete set of exemplar files as reference set an accuracy of 93.8 percent was achieved.
McDaniel and Heydari (2003); McDaniel (2001)	Reported results for the 30-class file type identification problem using three input patterns. When using the byte frequency histogram the file type of 23 percent of the test files was correctly predicted. When using the byte frequency cross-correlation and the file header-footer histogram 37 percent and 96 percent of test files were assigned the correct file type respectively. It should be noted that the header-footer histogram approach is identical to traditional file type identification using header and footer metadata signatures.

3.5.2 Discriminant analysis

Discriminant analysis has been introduced in Section 2.3. No literature could be found that uses LP-based discriminant analysis for file object type identification. However, Calhoun and Coles (2008) and Veenman (2007) proposed using Fisher's linear discriminant analysis to derive a dissension boundary that separates instances of two file types.

Table 3.10: Summary of results when using discriminant analysis.

Literature	Results
Calhoun and Coles (2008)	The classification accuracies for the various one-against-one discriminant models varied between 60 to 88 percent depending on the input pattern used.

Table continues on next page.

Summary of results when using discriminant analysis (continued).

Literature	Results
Veenman (2007)	An average classification accuracy of 45 percent was reported for all one-against-one discriminant models. The ROC measure reported for the one-against-all binary classifiers ¹ . This ROC measure is 0.72, 0.72, 0.77, 0.86 and 0.85 respectively when using entropy, complexity, entropy and complexity, the byte frequency histogram and all characteristics as input pattern. Veenman (2007) also showed that the detection rate of some file types can be improved by considering the file type of adjacent file fragments. In their experiments, the ROC measure improved from 0.76 to 0.85 when taking the file types of 16 adjacent file fragments into consideration.

3.5.3 Neural networks

The MLP neural network model was introduced in Section 2.2. MLP's have been used to solve the multiple class file object type identification problem. In general neural networks have performed well in file type identification. However, the results found by Harris (2007) for file fragment type identification have been modest. A summary is provided in Table 3.11.

Table 3.11: Summary of results when using neural networks.

Literature	Results
Ahmed <i>et al.</i> (2010)	Proposed reducing the complexity of the multiple class problem by segmenting the training set into multiple smaller sets by means of clustering over the exemplar set. After clustering, a neural network is used to predict the correct file type within each cluster. For each cluster an MLP neural network is trained with 256 input nodes, 6 hidden nodes, and the same number of output nodes than the number of file types remaining in the cluster. A file is first assigned to a cluster and then the neural network, trained for the selected cluster, is used for predicting a file type. A reported 90.1 percent of test files was typed correctly using this divide-and-conquer approach.
Kattan <i>et al.</i> (2010)	Compared a genetic programming approach for file type classification to a neural network approach. No information with regards to the neural network used is given. However, accuracies varied between 39.33 percent to 74.17 percent for a 2-class, 3-class, 4-class and 5-class classification problem. Kattan <i>et al.</i> (2010) found that the genetic programming approach is superior to the neural network approach.
Amirani <i>et al.</i> (2008)	Used an MLP neural network to solve the multiple class problem after reducing the dimensionality of the byte frequency histogram. Sixty features are extracted by the principle component analysis, which is then reduced to fifteen features using non-linear principal component analysis. The 15 features extracted are used as inputs for an MLP with 25 hidden nodes and 6 output nodes. A classification accuracy of 98.33 percent was reported. A description of non-linear principal component analysis can be found in Kramer (1991).

Table continues on next page.

¹The ROC measure has a minimum value of 0 and a maximum value of 1, and higher values are preferred over lower values (Fawcett, 2006).

Summary of results when using neural networks (continued).

Literature	Results
Harris (2007)	Showed how an MLP neural network can be used to classify 512-byte file fragments. All 512 bytes were used as input pattern in one experiment and the BFH calculated for the 512 byte fragment was used as input pattern in another experiment. Hidden layer nodes were varied between 10, 20 and 30. In neither experiment were any of the file types considered were correctly identified more than 60 percent of the time. Harris (2007) only considered five image file types, which may be more difficult to distinguish from one another.

3.5.4 Support vector machines

Support vector machines are not implemented in this dissertation, therefore a complete description of this technique is out of scope. A support vector machine is a supervised learning technique that has the ability to map a set of linear inseparable instances into a higher dimensional space where these instances are linearly separable. Therefore, a non-linear separation problem in the original feature space becomes a linear separation problem in the new feature space. A decision plane is constructed to partition the new feature space such that all elements of a class are grouped into the same partition and the margin between partitions are maximized. An optimal hyperplane is found by quadratic programming. Such a hyperplane is then used as a classification rule to predict the class label of unlabelled instances. For a complete introduction to support vector machines see Fletcher (2009).

Sportiello and Zanero (2011), as well as Li *et al.* (2010) apply support vector machines to solve the file fragment identification problem.

Table 3.12: Summary of results using support vector machines.

Literature	Results
Sportiello and Zanero (2011)	Reported an average detection rate of 90.4 percent for nine one-against-all binary classification models developed.
Li <i>et al.</i> (2010)	The SVM classifiers were able to detect 98 percent, 91 percent and 89 percent of <code>jpeg</code> file fragments correctly when paired against <code>ddl</code> , <code>pdf</code> and <code>mp3</code> file fragments respectively. Li <i>et al.</i> (2010) also combined the various one-against-one binary problems into a solution for the multiple class classification problem by majority vote. A classification accuracy was reported as 87 percent for a 3-class problem and 81 percent for a 4-class problem.
Li <i>et al.</i> (2006)	Detection rates of 92 percent, 97 percent, 87 percent, 98 percent and 88 percent were reported respectively when using <code>doc</code> , <code>jpg</code> , <code>xls</code> , <code>c++</code> and <code>pdf</code> as target file types in a one-against-all SVM classifiers .

3.5.5 Other approaches

Roussev and Garfinkel (2009) argued that a supervised learning approach towards solving the file fragment classification problem has been generally unsuccessful because these methods fail to take into account the definitive characteristics of the internal structure found in modern file types. They concluded that specialized approaches aimed at specific data formats can be superior to generalized file type identification. Accordingly, Garfinkel *et al.* (2010) recommended building file discriminators, to identify files based on some threshold of an observed n -gram or specific file type characteristic. As an example, three discriminators were constructed for **pdf**, **mp3** and **zip** files. Each of these discriminators were defined by looking at file specifications and examining a hexadecimal representation of exemplar files. These discriminators were able to correctly identify 98 percent, 99 percent and 49 percent of **pdf**, **mp3** and **zip** test files respectively.

In support, Mayer (2011) showed how long n -grams can be extracted automatically from a set of exemplar files. A list of the most common n -grams per file type was maintained, and a file with unknown type assigned to the file type with the most mutual n -grams. Using long n -grams, some file types can be detected with a precision of 66 percent or better. However, not all file types are filled with distinctive long n grams, and when distinctive n -grams do exist, they are sparsely located across a file. Mayer (2011) argued that in such cases, more general approaches to file type detection are required. Mayer (2011) proposes that n -grams be used as input to more robust classification algorithms, such as support vector machines, k -nearest neighbour algorithms, linear discriminant classifiers and neural networks.

3.6 Chapter summary

A systematic summary of the existing literature with regard to the use of supervised learning techniques for file object type identification has been given in this chapter. This chapter, as well as the previous chapter, served to guide further experimentation in the use of an ensemble of MLP neural network classifiers and LPDCs as a solution to the file fragment type identification problem.

As part of the knowledge discovery and data mining process model followed, Chapter 2 and Chapter 3 contributed towards developing an understanding of the application domain and a sense of the relevant data and prior research. In the next chapters the knowledge discovery endeavour is continued by providing a description of the research methodology and experimental design.

RESEARCH METHODOLOGY AND EXPERIMENTAL DESIGN

IV

The file object type identification problem has been identified as a classification task, which can be solved using supervised learning techniques. Three supervised learning techniques have been introduced in Chapter 2 and a literature summary with regard to the use of supervised learning methods for file object type identification was provided in Chapter 3. The remainder of this dissertation focuses on the use of MLP neural network classifiers and LP-based discriminant classifiers (LPDCs) as applied to the file fragment type identification problem. The abilities of these two classification models, used individually and as part of an ensemble, to correctly predict the file type associated with a file fragment is tested empirically.

This chapter gives an overview of various topics related to research methodology and experimental design. Specifically, the feature selection technique applied is outlined in Section 4.1; techniques for hyper-parameter optimization are outlined in Section 4.2; and a statistical technique used to estimate a model’s performance is introduced in Section 4.3. Furthermore, the model evaluation measures used are defined in Section 4.4 and a technique used to compare the performance of different models is outlined in Section 4.5. In Section 4.6 the experimental design is outlined.

4.1 Feature selection by mutual information

Feature selection is a process whereby a subset of the available features is selected for use with a classifier. The best subset contains features that contribute towards the induced classifier’s accuracy and excludes features that are irrelevant or redundant (Yu and Liu, 2004). The objectives of feature selection are to improve the performance of an induced classifier by avoiding the curse of dimensionality and the phenomenon of over fitting that is associated with complex classifiers; to eliminate noisy features; to provide faster and more cost-effective training and utilization of classifiers; and to facilitate data visualization and comprehensibility (Nisbet *et al.*, 2009; Guyon and Elisseeff, 2003; Dash and Liu, 1997). As noted in Section 2.2.3, feature selection is of particular importance when constructing MLP neural networks because the number of input features used has an impact on the ability of an induced neural network to learn and generalize the true function that underlies the data (Battiti, 1994).

The feature selection technique used in this dissertation is known as “greedy” mutual information-based feature ranking (Kwak and Choi, 2002; Battiti, 1994). This technique ranks predictor variables based on each predictor variable’s mutual information with respect to the predicted variable and with respect to the already-ranked predictor variables.

The mutual information between two variables X_i and X_j is defined as the difference between the uncertainty in X_i and the uncertainty in X_i remaining after knowing X_j (Guyon and Elisseeff, 2003). The uncertainty in variable X_i and variable X_j , with realizations x and $\dot{x}$ respectively, are quantified in terms of Shannon's entropy (Shannon, 2001) and is defined as,

$$H(X_i) = - \sum_{x \in X_i} p(x) \log(p(x)) \quad (4.1)$$

$$H(X_j) = - \sum_{\dot{x} \in X_j} p(\dot{x}) \log(p(\dot{x})) \quad (4.2)$$

$$H(X_i, X_j) = - \sum_{x \in X_i} \sum_{\dot{x} \in X_j} p(x, \dot{x}) \log(p(x, \dot{x})) \quad (4.3)$$

which is related as,

$$H(X_i, X_j) = H(X_j, X_i) = H(X_j|X_i) + H(X_j) \quad (4.4)$$

to calculate $H(X_i)$ and $H(X_i, X_j)$ the probability distributions $p(x)$ and $p(\dot{x})$, as well as the joint probability distribution $p(x, \dot{x})$ are estimated by relative frequency count.

Using Shannon's entropy as a measure of uncertainty, the mutual information $I(X_j, X_i)$ between variable X_i and variable X_j can be written as,

$$\begin{aligned} I(X_j, X_i) &= H(X_j) - H(X_j|X_i) \\ &= H(X_j) + H(X_j) - H(X_i, X_j) \\ &= - \sum_{x \in X_i} p(x) \log(p(x)) - \sum_{\dot{x} \in X_j} p(\dot{x}) \log(p(\dot{x})) + \sum_{x \in X_i} \sum_{\dot{x} \in X_j} p(x, \dot{x}) \log(p(x, \dot{x})) \\ &= \sum_{x \in X_i} \sum_{\dot{x} \in X_j} p(x, \dot{x}) \log\left(\frac{p(x, \dot{x})}{p(x)p(\dot{x})}\right) \end{aligned} \quad (4.5)$$

Equation 4.5 can be used to calculate the mutual information $I(X_j, X_i)$ between every pair of predictor variables X_j and X_i in the input pattern $\mathbf{x}$, as well as the mutual information $I(Y, X_i)$ between predicted variable Y and every predictor variable X_i . For any two variables the estimated mutual information will approach zero if the two variables are more independent of one another, while the estimated mutual information will become relatively larger if the variables are more dependent on one another¹.

The set R of c features that maximize $I(Y, R)$ is used to construct a classifier. However, computing $I(Y, R)$ for all possible combinations of c features in the set of r available features is computationally infeasible because the number of subsets that has to be considered is very large². Instead of enumerating all possible feature subsets, features can be selected iteratively for inclusion in R . During each iteration, the feature X_i that maximizes the gain

¹If two variables X_j and X_j are independent then $p(X_i, X_j) = p(X_i)p(X_j)$ (Rice, 2007). The value of Equation 4.5 will be zero because $\log(1) = 0$.

²The number of subsets to consider is $\binom{r}{c}$. The number of attributes used in this dissertation is $r = 259$ as will be outlined in Section 4.6.2

in mutual information is selected. The gain in mutual information achieved by adding feature X_i to the set of already selected features R can be approximated by the information that X_i provides with regard to the predicted class variable Y , adjusted for by the information already available in R (the redundant information provided by X_i). That is,

$$I(Y, \{R, X_i\}) \approx I(Y, X_i) - \beta \sum_{X_j \in R} I(X_i, X_j) \quad (4.6)$$

where the parameter $\beta \in (0, 1)$ weighs the importance of excluding redundant features in the feature selection process.

The “greedy” mutual information-based feature ranking algorithm is summarized in Algorithm 4.1 and an example is given in Appendix C. Predictor variables are ranked in the order in which they are included in the set R . The c top ranked features are the c features that collectively maximize the mutual information about the class denoting variable Y . An arbitrary number of c top ranked features can be used to construct a classifier or the optimal number of top ranked features can be found as part of model selection. Battiti (1994) proposes terminating the algorithm when the gain in mutual information by adding another feature to R falls below some predefined stopping criterion, and using those features already in R to construct a classifier. Determining the optimal stopping criterion for a given classifier is, however, problematic.

Algorithm 4.1 Mutual information-based feature ranking (Adapted from Battiti, 1994).

input: Data set D

List of r features $F = \{X_1, X_2, \dots, X_r\}$

Value for parameter β

output: List of ranked features R

begin algorithm

$R = \{\}$ ***Initialize list of ranked features*

Compute $p(X_i = x), p(Y = y), p(X_i = x, Y = y)$ for each $X_i \in F$

Compute $p(X_i = x, X_j = \dot{x})$ for each $X_i, X_j \in F$ where $i \neq j$

Use estimated probability distributions to compute $I(Y, X_i)$ for each feature $X_i \in F$

Use estimated probability distributions to compute $I(X_i, X_j)$ for each $X_i, X_j \in F$ where $i \neq j$

repeat

 Compute $I(Y, \{R, X_i\},)$ for each $X_i \in F$ ***Use Equation 4.6*

 Add feature with highest $I(Y, \{R, X_i\})$ to R

 Remove feature added to R from F

until Feature list F is empty

end algorithm

“Greedy” mutual information feature ranking is implemented in this dissertation because it is suitable for complex classification tasks where feature ranking methods based on linear relations are prone to mistakes (Battiti, 1994). This technique has also been shown to outperform traditional feature combination techniques, such as principle component analysis and feature selection techniques, such as random dimension reduction (Kwak and Choi, 2002). Feature selection is applied to the data set used in Subsection 4.6.3.

4.2 Hyper-parameter optimization

The objective of supervised learning is to find a hypothesis function h that closely approximates the true function f , which underlies the data. Three primary components of a supervised learning problem have been identified in Section 2.1: model representation, model evaluation and search method. Once the model representation and the model evaluation criteria have been selected, then the knowledge discovery endeavour is reduced to an optimization task of finding the parameters that define the hypothesis, which best fits the data (Fayyad *et al.*, 1996).

Both the model parameters and the hyper-parameters are considered (Hutter, 2009). An induction algorithm typically selects the best-fitting hypothesis by selecting optimal model parameter values from the hypothesis space using either closed-form solutions or some sort of induction algorithm. Hyper-parameters on the other hand determine the hypothesis space in which the induction algorithm operates (model complexity parameters), as well as fine-tunes the induction algorithm used to search the hypothesis space (algorithmic parameters).

For example, consider inducing an MLP neural network classifier by applying the back propagation algorithm. Hyper-parameter optimization will involve choosing the number of input features and the number of hidden layer nodes. Furthermore, the learning rate and strength of weight regularization used in the back propagation algorithm can also be selected. Given a hyper-parameter configuration, model parameter search involves finding the optimal hypothesis h in the hypothesis space defined by the hyper-parameter configuration selected.

The problem of identifying a good hyper-parameter configuration λ from the set of all possible hyper-parameter configurations Λ is called hyper-parameter optimization or model selection (Bergstra and Bengio, 2012). Hyper-parameter optimization can be summarized as,

$$\lambda^* = \arg \min_{\lambda \in \Lambda} R_{\text{emp}}(h^\lambda) \quad (4.7)$$

$$= \arg \min_{\lambda \in \{\lambda^{(1)}, \dots, \lambda^{(p)}\}} R_{\text{emp}}(h^\lambda), \quad (4.8)$$

where $\lambda^{(1)}, \dots, \lambda^{(p)}$ are the p hyper-parameter configurations evaluated and $R_{\text{emp}}(h^\lambda)$ is the empirical loss³ estimated for an induced model h with hyper-parameter configuration λ .

In most cases it is not computationally possible to evaluate each hyper-parameter configuration $\lambda \in \Lambda$, therefore the set of hyper-parameter configurations $\lambda^{(1)}, \dots, \lambda^{(p)}$ to evaluate has to be chosen. Three basic approaches exist to select the hyper-parameter configurations considered: manual search, grid search and random search (Bergstra and Bengio, 2012).

In a manual search each hyper-parameter configuration λ is chosen by the modeller. Manual search is often used to provide insight into how learning algorithms respond to changes in hyper-parameter values. For expert users this is a useful technique to identify

³Empirical loss was defined in Section 2.1 as the average prediction error over the set of exemplar instances, where the prediction error is measured by some loss function $L(y, \hat{y})$.

regions of interest in the hyper-parameter configuration space. However due to the subjective nature of manual search it is not easily reproducible, automatable or justifiable (Bergstra and Bengio, 2012).

A grid search requires the preselection of a set of possible values for each of the Q hyper-parameters in the hyper-parameter vector $\lambda^{(Q \times 1)}$. The grid search is performed by inducing hypotheses for each possible combination of hyper-parameter values. The number of configurations tested in a grid search is $\prod_{q=1}^Q L_q$ where L_q is the number of values selected for hyper-parameter q . Though grid search is reliable in low dimensional spaces, it becomes prohibitively computationally expensive in higher dimensional spaces (larger values of Q) and at finer hyper-parameter value resolutions (large values of L_q). Furthermore, selecting a manageable and appropriate resolution for each hyper-parameter is problematic and extending the grid requires commitment to a much larger experiment.

Within a random search, hypotheses are induced independently, each model using a hyper-parameter configuration selected randomly from the predefined range of allowable hyper-parameter values. Bergstra and Bengio (2012) argue that a random search can find hyper-parameter configurations that are at least as good as those found in traditional grid search approaches, within a small fraction of the computational time. This is because the empirical loss estimated for a hypothesis may be highly sensitive to changes in some hyper-parameters and only weakly sensitive to changes in other hyper-parameters. A random search will explore more distinct values of the important hyper-parameter, and therefore more distinct empirical loss estimates (points on the empirical loss surface).

For example, consider the two-dimensional hyper-parameter spaces shown in Figure 4.1. Assume the empirical loss estimated for a model is highly sensitive to changes in hyper-parameter one and only weakly sensitive to changes in hyper-parameter two. A grid search will waste trials by searching across parameter two, poorly covering the empirical loss surface while a random search will cover more points on the empirical loss surface by considering more values for both the important parameter and the unimportant parameter.

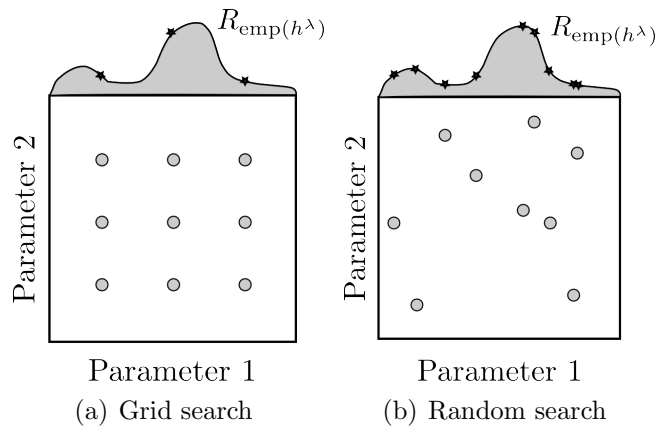


Figure 4.1: Grid search vs. random search.

A random search also has the advantage that a search can be stopped and restarted at

any time; searches can be carried out asynchronously; and because trials are independent new trials can be added and failed trials can be restarted as needed.

Grid search and random search are used in this dissertation. The traditional grid search is applied where it is computationally feasible to test every possible hyper-parameter combination while a random search is applied when it is not feasible.

4.3 Cross-validation

Cross-validation is a statistical method of evaluating and comparing hypothesis functions by dividing data into two segments. One segment is used to induce the hypothesis function and the other segment used to test the hypothesis function (Refaeilzadeh *et al.*, 2009). Various cross-validation techniques exist, including hold-out cross-validation, k -fold cross-validation and leave-one-out cross-validation. In this dissertation, k -fold cross-validation is used to calculate a robust estimate of the the empirical loss associated with a given hypothesis function as suggested by Bouckaert (2003).

The empirical loss is a measure of how well the hypothesis function h approximates the true function f (Section 2.1), calculated as

$$R_{\text{emp}}(h) = \frac{1}{N} \sum_{t=1}^N L(y^{(t)}, h(\mathbf{x}^{(t)})) \quad (4.9)$$

where the loss function $L(y, h(\mathbf{x}))$ measures the extent with which the predicted class variable value $\hat{y} = h(\mathbf{x})$ differs from the actual class variable value y . Many different loss functions exist (Bishop, 1995), such as the mean-squared-error loss function,

$$L(y, \hat{y} = h(\mathbf{x})) = (y - \hat{y})^2 \quad (4.10)$$

or the cross-entropy loss function,

$$L(y, \hat{y} = h(\mathbf{x})) = y \ln\left(\frac{\hat{y}}{y}\right) + (1 - y) \ln\left(\frac{1 - \hat{y}}{1 - y}\right). \quad (4.11)$$

In this dissertation, the 0-1 loss function is used to calculate the empirical loss associated with a hypothesis function presented,

$$L(y, \hat{y} = h(\mathbf{x})) = \begin{cases} 0 & \text{if } h(\mathbf{x}) = y \\ 1 & \text{if } h(\mathbf{x}) \neq y \end{cases}. \quad (4.12)$$

Using cross-validation provides valuable insight into the stability of the empirical loss associated with a given hypothesis function. Empirical loss may vary due to the choice of the testing set; the choice of the training set; the internal randomness of the induction algorithm; as well as the random classification error due to having mislabelled objects in the testing data (Kuncheva, 2004).

In model selection, k -fold cross-validation is employed to estimate the empirical loss associated with a given hypothesis function as summarized in Algorithm 4.2. Several classifiers are trained, each with a different hyper-parameter configuration λ . The best hyper-parameter configuration λ^* is selected in order to minimize the empirical loss $R_{\text{emp}}(h^\lambda)$. During cross-validation the dataset D is split into k subsets $S_1, S_2, \dots, S_k$ each with an approximately equal number of example instances. The set $T_i = D - S_i$ with $i = 1..k$ is divided equally into a training subset T_{i1} and validation subset T_{i2} . The subset T_{i1} is used for training the classifier, the subset T_{i2} is used to calculate empirical loss of the induced classifier h_i^λ , while subset S_i is not used. The hypothesis h^λ is trained and validated k times, each time producing an empirical loss estimate. The k individual empirical loss estimates are averaged to calculate the reported empirical loss for classifier h^λ .

Algorithm 4.2 k -Fold cross validation for model *selection*.

input: Data set D

output: Best hyper-parameter configuration λ^*

begin algorithm

Randomly divide data set D into k partitions $S_1, \dots, S_k$ each having an approximately equal number of instances

repeat

Select a hyper-parameter configuration λ ***By manual, grid or random search*

Initialize $\bar{R}_{\text{emp}}(h^\lambda)$ to 0

for $i = 0$ **to** k **do**

Use temporary data set $T_i = D - S_i$.

Partition temporary data set T_i into a training set T_{i1} and a validation set T_{i2} .

Induce hypothesis h_i^λ over training set T_{i1} .

Estimate empirical loss $R_{\text{emp}}(h_i^\lambda)$ over validation set T_{i2} .

$\bar{R}_{\text{emp}}(h^\lambda) = \bar{R}_{\text{emp}} + R_{\text{emp}}(h_i^\lambda)$.

end for

$\bar{R}_{\text{emp}}(h^\lambda) = \bar{R}_{\text{emp}}(h^\lambda)/k$

Tabulate $\bar{R}_{\text{emp}}(h^\lambda)$

until Hyper-parameter optimization is complete

Select hyper-parameter configuration λ with lowest $\bar{R}_{\text{emp}}(h^\lambda)$ ***This is λ^**

end algorithm

After finding the best hyper-parameter configuration λ^* (the one with the lowest associated empirical loss), the empirical loss for hypothesis h^λ with $\lambda = \lambda^*$ is estimated using the set T_i as training set, and the set S_i as testing set as outlined in Algorithm 4.3. The hypothesis h^λ is trained and tested k times, each time producing an empirical loss estimate. The k individual empirical loss estimates are averaged to calculate the reported empirical loss for classifier h^λ .

In this way, model selection is done independently from model evaluation, because S_i is not used during model selection but to estimate the performance of the classifier only after a hyper-parameter configuration was selected. The phenomenon of peaking, where the same data set is used for both selecting and evaluating a hypothesis, is therefore avoided (Russel and Novig, 2010).

Algorithm 4.3 k -Fold cross validation for model *evaluation*.

input: Data set D Hyper-parameter configuration λ **output:** Estimate empirical loss for hypothesis h with hyper-parameter configuration λ **begin algorithm**Randomly divide data set D into k partitions $S_1, \dots, S_k$ each having an approximately equal number of instancesInitialize $\bar{R}_{\text{emp}}(h^\lambda)$ to 0**for** $i = 0$ **to** k **do**Use data set $T_i = D - S_i$ as training set and data set S_i as testing set.Induce hypothesis h_i^λ over training set T_i .Estimate empirical loss $R_{\text{emp}}(h_i^\lambda)$ over testing set S_i . $\bar{R}_{\text{emp}}(h^\lambda) = \bar{R}_{\text{emp}} + R_{\text{emp}}(h_i^\lambda)$.**end for** $\bar{R}_{\text{emp}}(h^\lambda) = \bar{R}_{\text{emp}}(h^\lambda)/k$ **end algorithm**

4.4 Model evaluation measures

A model evaluation measure is a quantitative statement of how well a particular classifier meets the goals of the supervised learning endeavour (Fayyad *et al.*, 1996). In this research project the performance of each hypothesised classifier is reported by means of several evaluation measures: the confusion matrix, classification accuracy, precision and recall. Each reported evaluation measure is estimated using 10-fold cross-validation (as outlined in Algorithm 4.3). That is, an evaluation measure is calculated as the average of 10 individual evaluation measures estimated on independently trained and tested classifiers. The evaluation measures used in this dissertation are outlined in the next three subsections.

4.4.1 Confusion matrix

A confusion matrix is a matrix of size $n \times n$, where n is the number of classes considered in the classification problem, that shows the relation between the predicted class labels and actual class labels (Kohavi and Provost, 1998). The confusion matrix is particularly useful to determine the extent of individual class misclassification. It might be possible that a classifier performs extremely well in correctly classifying instances belonging to one class, and extremely poorly in correctly classifying instances belonging to another class. The value of a classifier that performs well in the identification of classes and worse in the identification of other classes depends on the application domain.

A confusion matrix for a three-class classification problem is shown in Table 4.1. The number of instances actually belonging to class 1 that was also predicted as belonging to class 1 is denoted a ; the total number of instances in the testing set actually belonging to class 1 is denoted $t_{r1} = a + b + c$; while the total number of instances in the testing set predicted to belong to class 1 is denote $t_{c1} = a + d + g$.

Table 4.1: Example of a confusion matrix

Actual\Predicted	Class 1	Class 2	Class 3	Total	Precision
Class 1	a	b	c	t_{r1}	π_1
Class 2	d	e	f	t_{r2}	π_2
Class 3	g	h	i	t_{r3}	π_3
Total	t_{c1}	t_{c1}	t_{c1}		
Recall	ρ_1	ρ_2	ρ_3		

4.4.2 Precision and recall

There are four possible classification outcomes for every instance in the data set (Fawcett, 2006). A classifier can assign an instance to class i if it does in actual fact belong to class i , this is counted as a true positive (tp_i). If the classifier assigns the instance to class i if it does not actually belong to class i it is counted as a false positive (fp_i). An instance not assigned to class i that does not belong in class i is counted as a true negative (tn_i) whereas an instance that is not assigned to class i that does belong to class i is counted as a false positive (fn_i).

For every class i in the n class classification problem a precision rate can be calculated as (Ozgun *et al.*, 2005),

$$\pi_i = \frac{tp_i}{tp_i + fp_i} \quad (4.13)$$

and a recall rate can be calculated as,

$$\rho_i = \frac{tp_i}{tp_i + fn_i} \quad (4.14)$$

The precision rate summarizes a classifier's ability to identify a pattern as belonging to a class i if it does in fact belong to class i . The precision rate for class 1 in Table 4.1 can be calculated as $\pi_1 = a/t_{r1}$. The recall rate measures the probability that an instance classified as belonging to class i is in fact a class i instance. The recall rate for class 1 in Table 4.1 can be calculated as $\rho_1 = a/t_{c1}$. Similarly, precision and recall can be calculated for each class. The average precision and recall rate for a classifier is calculated as the average over all the individual class precision and recall rates.

4.4.3 Model accuracy

Accuracy, sometimes called the correct classification rate, can be calculated as the number of correct classifications made by the model over the test set, divided by the total number of instances in the test set (Olson and Shi, 2006). From the confusion matrix in Table 4.1, the classification accuracy of the classifier can be calculated as $(a + e + i)/(a + b + \dots + i)$.

The classification accuracy is related to the empirical loss (with a 0-1 loss function) estimated in cross-validation as outlined in Section 4.3. The empirical loss measures the rate of misclassification whereas the classification accuracy $\bar{A}$ measures the rate of correct

classification, formulated as,

$$\begin{aligned}
\bar{A} &= 1 - \bar{R}_{\text{emp}}(h) \\
&= 1 - \text{mean}_{S_1, S_2, \dots, S_{10}} R_{\text{emp}}(h) \\
&= 1 - \text{mean}_{S_1, S_2, \dots, S_{10}} \frac{1}{|S_i|} \sum_{(\mathbf{x}, y) \in S_i} \begin{cases} 0 & \text{if } h(\mathbf{x}) = y \\ 1 & \text{if } h(\mathbf{x}) \neq y \end{cases}.
\end{aligned} \tag{4.15}$$

4.5 Model comparison

After evaluating each hypothesized classifier built as part of the knowledge discovery endeavour, the performance measures can be compared to select the best model. In comparing the performance of two classifiers, the best classifier can be selected based on the perceived difference in the observed evaluation measures. Alternatively, a more scientific approach can be taken to statistically test the significance of the observed performance difference between two classifiers (Kumar, 2005). A statistical hypothesis test tries to determine whether one can expect the same results from a repeated study or whether the observed results can be attributed to chance.

To compare the performance of two classifiers h^a and h^b a 10-fold cross-validation t -test is used to test the research hypothesis that the two classifiers perform equally (Bouckaert, 2003; Salzberg, 1997). That is, the null hypothesis states that the results obtained are purely random, whereas the alternative hypothesis states that the results are extraordinary. If the two algorithms are statistically different then the null hypothesis will be rejected in favour of the alternative hypothesis.

As stated in Section 4.4, the classification accuracy of an induced classifier is estimated using the k -fold cross-validation. For each subset S_i an accuracy measure $A_i^a = 1 - R_{\text{emp}}(h_i^a)$ and $A_i^b = 1 - R_{\text{emp}}(h_i^b)$ is estimated. The difference in the estimated accuracies of classifier h^A and h^B on subset S_i is $d_i = A_i^a - A_i^b$. The difference in the estimated classifier accuracies is assumed to be normally distributed with mean,

$$\bar{d} = \frac{1}{k} \sum_{i=1}^k d_i \tag{4.16}$$

and variance

$$\hat{\sigma}_d^2 = \frac{\sum_{i=1}^k (d_i - \bar{d})^2}{k - 1} \tag{4.17}$$

Under the null hypothesis $H_0 : d = 0$ the test statistic,

$$t_d = \frac{1}{\sqrt{k}} \frac{\bar{d}}{\sqrt{\hat{\sigma}_d^2}} \tag{4.18}$$

has a Student's t -distribution with $k-1$ degrees of freedom. From the Student's t -distribution the probability p of obtaining a test statistic greater than or equal to the the observed statistic, when assuming the null hypothesis is true, can be obtained. If the two-tailed p -value

is less than the chosen level of significance α then the null hypothesis H_0 can be rejected in favour of the two-tailed alternative hypothesis $H_A : p \neq 0$. That is, reject the null hypothesis if $p(|t| \geq t_\alpha | H_0) \leq \alpha$. The alternative hypothesis states that the accuracy of classifier h^a and accuracy of classifier h^b are statistically different from one another. Instead of explicitly specifying a level of significance, usually chosen as $\alpha = 0.05$, the p -value can also be interpreted as the smallest significance level at which the null hypothesis will be rejected (Rice, 2007).

As an alternative to the 10-fold cross-validation t -test the McNemar’s test, the 5×2 -fold cross-validation test and the 10×10 -fold cross-validation test were also considered. The McNemar’s test is not suitable as it does not consider variability due to the choice of training set or due to the internal randomness in the induction algorithm (Kuncheva, 2004). Neither the 10×10 -fold cross-validation test nor the 5×2 cross-validation test have been widely adopted in the data mining field, with the 10-fold cross-validation test remaining the most popular procedure (Refaeilzadeh *et al.*, 2009). Therefore the 10-fold cross-validation hypothesis test is used in this dissertation.

4.6 Experimental design

Not only is the ability of an MLP neural network classifier and LPDCs to solve the multiple class file fragment type identification problem investigated in this dissertation, but also the hypothesis that an ensemble of MLP neural network classifiers and LPDCs will be able to solve the file fragment identification problem with improved accuracy over the individual classifiers. The rationale behind this hypothesis is that the two classifiers with different model representations are diverse, and would therefore perform better collectively than individually (Lee and Elder, 1997; Woods *et al.*, 1997).

The data set used in this empirical investigation is described in Subsection 4.6.1. The input pattern used to characterize a file fragment is defined in Subsection 4.6.2. In Subsection 4.6.3 the “greedy” mutual information-based feature ranking algorithm is applied to the data set to derive the feature ranking used in this dissertation. Four experiments used to test the research hypothesis of this dissertation are briefly introduced in Subsection 4.6.4. A more in-depth discussion of each experiment is given in Chapter 5.

4.6.1 Data preparation: The data set

The set of raw files used in the empirical study were taken from the *govdoc* file corpus (Garfinkel *et al.*, 2009). This publicly available file corpus consists of approximately one million files of various file types provided to support forensic research involving the analysis of files or file fragments. The complete corpus is made available in subsets consisting of 1 000 files each. The set of exemplar files used in this dissertation are sourced from 24 of these subsets.

Only fragments belonging to the `html`, `gif`, `ps`, `xls`, `ppt`, `jpg`, `doc`, `txt`, `gz` and `pdf` file types were considered. These ten file types were selected partly because they are user-centric files, which is of forensic importance (Garfinkel, 2007), partly because they are popular file types considered in the literature (see Tabel 3.4) and partly because the number of fragments that could be generated from files of these file types was numerous enough to support the data mining activity.

The descriptive statistics for the set of exemplar files used are given in Table 4.2. A total of 20 069 files were included in the file set, amounting to approximately 14 GBs of data. The average size of files in the exemplar file set is 0.69 MB. Files smaller than 8192 bytes were discarded, therefore the smallest file in the data set has a file size of 0.0078 MB. The largest file used in this research was a `txt` file with a file size of approximately 1.5 GB.

Table 4.2: Descriptive statistics of the file set used.

File type	Number of files	File size (in Mbs)			
		Total	Mean	Min	Max
html	4 890	437.7	0.0895	0.0078	39.31
gif	373	20.7	0.0556	0.0079	0.56
ps	374	601.7	1.6087	0.0090	41.59
xls	999	832.4	0.8332	0.0086	66.13
ppt	1 300	3 157.5	2.4289	0.0132	234.25
jpg	1 587	235.9	0.1486	0.0078	9.08
doc	1 800	1 401.1	0.7784	0.0107	158.51
txt	2 484	3 372.9	1.3579	0.0078	1 557.08
gz	223	125.8	0.5639	0.0080	6.02
pdf	6 039	3 593.5	0.5950	0.0078	30.37
All	20 069	13 779.2	0.6906	0.0078	1 557.08

Each file in the exemplar file set was broken down into 4096-byte file fragments using a Java program developed for this purpose. The program loops across each file, storing each 4096-byte fragment separately. This file fragment size was chosen to correspond with the 4096-byte disk sector that has become the standard disk sector size within the storage industry (Garfinkel *et al.*, 2010). In line with related studies, the header and trailer file fragments were excluded from the fragment set (Li *et al.*, 2010; Veenman, 2007; Karresand and Shahmehri, 2006a,b). It can be argued that header and footer fragments contain the traditionally used metadata that this study does not want to exploit. Furthermore, any underutilized footer fragments might distort the analysis as they are either padded with null values or contain orphaned data. The exemplar fragment set consisted of more than 3.5 million file fragments. The number of fragments per file type is given in Table 4.3.

From the exemplar fragment set, 4 000 file fragments per file type were selected randomly without replacement for inclusion in the final data set used. For example, 4 000 `html` file fragments were sampled from the available 104 724 file fragments. An input pattern is calculated for each exemplar fragment after which each feature in the input pattern is standardized. Each input pattern is labelled to correspond to the file type as denoted by the file extension of its parent file. The result is the final data set used in data mining.

Table 4.3: Number of file fragments per file type

File type	Number of fragments
html	104 724
gif	4 744
ps	153 465
xls	211 545
ppt	806 300
jpg	58 036
doc	355 869
txt	859 794
gz	31 851
pdf	910 860
Total	3 497 188

This process is summarized in Figure 4.2. The input pattern is defined in Subsection 4.6.2. The feature ranking for this data set is given in Subsection 4.6.3.

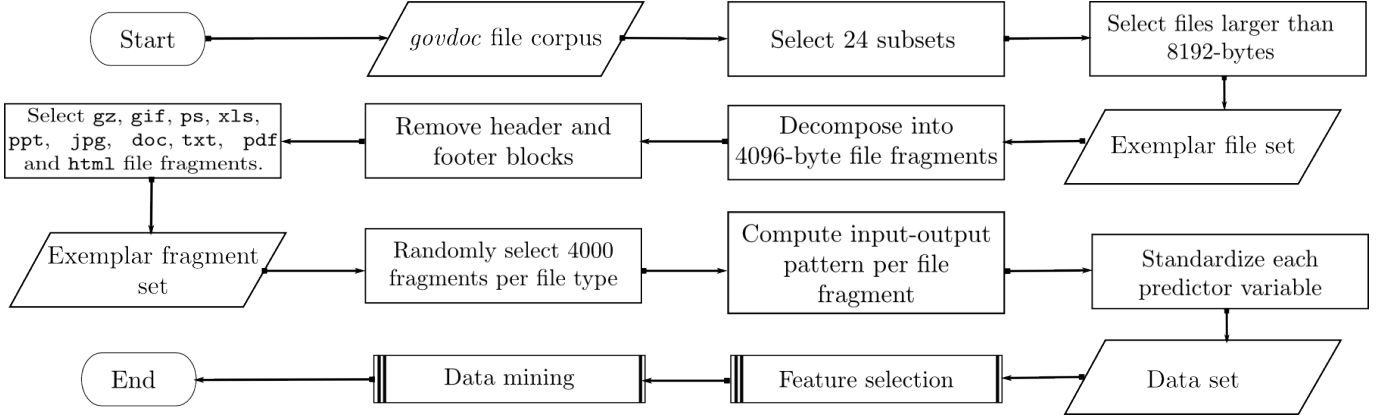


Figure 4.2: Process followed to construct the base data set used in data mining.

4.6.2 Data preparation: The input pattern

The byte frequency histogram (BFH) is used as input pattern to describe a file fragment. The BFHs are constructed using a Java program developed for this purpose. This program sequentially reads each byte in a file fragment while counting the number of occurrences of each possible byte value 0 to 255. The result is an array of 256 frequency values. An example BFH is shown in Figure 4.3 and more examples are given in Appendix D.

The BFH was chosen due to its popularity in the literature (see Table 3.2). Assuming that each file type has an underlining internal data structure, this data structure will at least be somewhat represented in the aggregate BFH.

The input pattern is further augmented by including the mean of the byte values in the file fragment, standard deviation of the byte values in the file fragment, and the entropy of the byte values in the file fragment. The mean byte value in a file fragment is used to

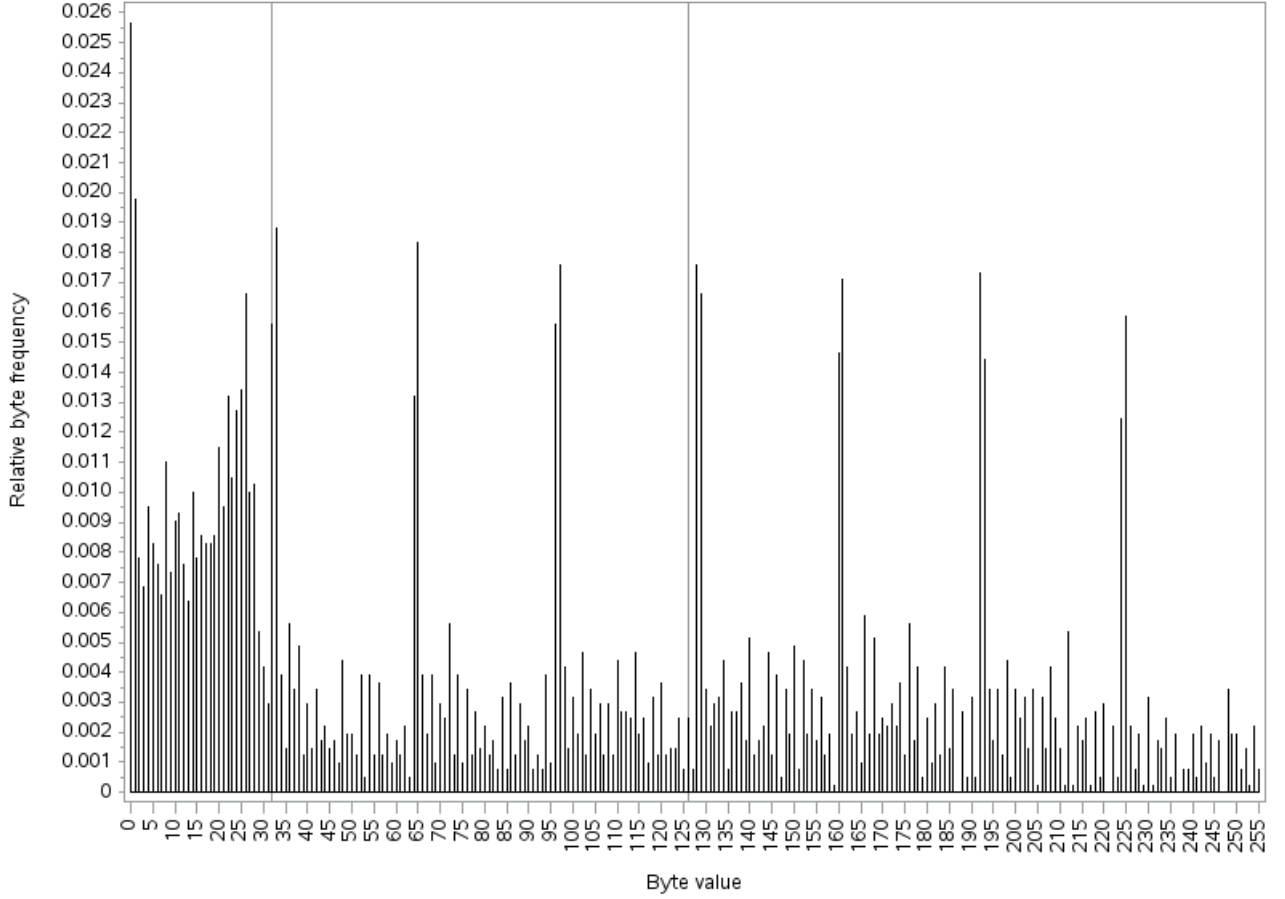


Figure 4.3: The BFH of a pdf file fragment.

measure the centre of the byte frequency histogram. The mean of all byte values b_i in a 4096-byte file fragment is defined as,

$$\mu = \frac{1}{4096} \sum_{i=1}^{4096} b_i. \quad (4.19)$$

The standard deviation of byte values in a file fragment is used to measure the variance of byte values from the mean byte value. The standard deviation of all byte values b_i in a 4096-byte file fragment is defined as,

$$\sigma = \sqrt{\frac{1}{4096} \sum_{i=1}^{4096} (b_i - \mu_b)^2}. \quad (4.20)$$

Entropy measures the randomness of byte values within a file fragment. The entropy in a file fragment is calculated as,

$$H = - \sum_{v=0}^{255} p(v) \log(p(v)) \quad (4.21)$$

where the probability distribution $p(v)$ is the relative frequency of each possible byte value v ($v = 0, \dots, 255$) in the the given 4096-byte file fragment.

These measures summarize the BFH’s form, and would arguably be able to provide a classifier with some information regarding the whole BFH even if only a subset of the BFH is made available to the classifier. The mean byte value, standard deviation of byte values and entropy of byte values are attractive attributes because they can be computed directly from the BFH without having to reprocess the complete file fragment.

The BFH, as well as the calculated mean byte value, standard deviation of byte values and entropy of byte values constitute the 259-feature input pattern calculated for each file fragment included in the data set⁴.

4.6.3 Data preparation: Feature selection

As outlined in Section 4.1 a classifier’s accuracy can be improved by reducing the number of input features made available to it. As such, only a subset of the 259 features are made available to each classifier tested. The “greedy” mutual information-based feature ranking algorithm outlined in Section 4.1 is used to rank the 259 features in the input pattern.

The data set used for feature ranking is sampled from the exemplar file set as shown in Figure 4.4. A random sample of 1000 instances per file type is selected from the exemplar fragment set. The input pattern outlined in Section 4.6.2 is calculated per file fragment. Each feature in the input pattern is discretized into 10 bins with equal frequency (Ying *et al.*, 2010). That is, the top 10 percent of values in the feature is assigned to value 1, the second 10 percent of values in the feature is assigned value 2, and so forth.

A β parameter of 0.7 was used to obtain the feature ranking given in Table 4.4⁵. For brevity, the complete feature ranking obtained is not presented here. The top c ranked features are the set of c features that provide the most information with regard to the predicted class variable. For example, the subset of 4 features, which provides the most information in distinguishing between all ten file types is the frequency value for bytes 0, 32, 10 and 107.

The number of input features made available to a classifier is determined as part of hyper-parameter optimization.

4.6.4 Data mining: The experiments

In preliminary work, Wilgenbus *et al.* (2012) showed that both the MLP neural network classifier and the LPDC have the ability to solve the one-against-all file fragment type classification problems with acceptable performance (see Appendix A). The experiments presented here focus on solving the multiple class file fragment type classification problem. The usefulness of MLP neural network classifiers and LPDCs deployed independently and

⁴The 256-byte frequency counts in the BFH, the mean byte value, the standard deviation of byte values and the entropy of byte values.

⁵Kwak and Choi (2002) suggest using a β parameter value between 0.5 and 1, where larger β values are used to exclude redundant parameter more aggressively. Estimating the optimal parameter value is difficult as no prior knowledge with regard to the true optimal feature ranking is available.

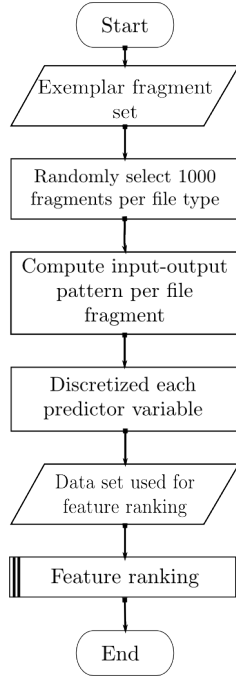


Figure 4.4: Process followed to construct the data set used for feature ranking.

Table 4.4: Feature ranking by mutual information.

Ranking	Feature
1	Byte 0
2	Byte 32
3	Byte 10
4	Byte 107
5	Byte 52
6	Byte 65
7	Byte 13
8	Byte 118
9	Byte 95
10	Byte 66
⋮	⋮
258	Byte 149
259	Byte 205

together in an ensemble is tested empirically. The results are compared statistically to test the hypothesis that the ensemble performs better than the individual classifiers. With this goal in mind, four experiments are defined.

Experiment 1 A benchmark model, using the k -nearest neighbour algorithm, is proposed. This experiment is provided in an attempt to make the research results of this dissertation more comparable to the research results reported by other researchers.

Experiment 2 A single MLP neural network classifier, as outlined in Chapter 2, is used to solve the 10-class file fragment type classification problem.

Experiment 3 The 10-class file fragment type classification problem is decomposed into 45 one-against-one binary classification subproblems. Each binary classification subproblem is solved using an LPDC. The two phase LPDA model, as outlined in Chapter 2, is used to derive each LPDC. The solutions to the binary subproblems are combined into a solution for the multiple class problem by majority vote. An instance is assigned to the file type with the most votes.

Experiment 4a An ensemble of ten MLP neural network classifiers are constructed. Each individual MLP neural network classifier in the ensemble has a different hyper-parameter configuration. The probability estimate

for each file type provided by each individual MLP neural network classifier is combined into an aggregate probability estimate for each file type. An instance is assigned to the file type with the highest aggregate probability estimate.

Experiment 4b An ensemble of 450 binary LPDCs are constructed. That is, ten classifiers are constructed for each pair of file types in the 10-class classification problem. Drawing an analogy from Experiment 4a, ten composite LPDCs are constructed, where each composite classifier consists of 45 binary one-against-one classifiers. Every composite LPDC uses a different subset of the input pattern. The total number of votes towards each file type over the set of 450 votes is determined. The probability of a file fragment belonging to file type i is estimated as the percentage of total votes in favour of file type i . An instance is assigned to the file type with the highest probability estimate (i.e. an instance is assigned to the file type with the most votes).

Experiment 4c An ensemble of ten MLP neural network classifiers and ten composite LPDCs is constructed. That is, the aggregate probability estimates of Experiment 4a and Experiment 4b are combined. An instance is assigned to the class with the highest aggregate probability estimate.

A more in-depth description of the experiments and the results of each experiment is given and evaluated in the next chapter. The results of the experiments are compared to test the research hypothesis.

4.7 Chapter summary

The dissertation hypothesizes that an ensemble of MLP neural network classifiers and LPDCs can provide improved classification accuracy when applied to the file fragment type identification. In this regard, the effectiveness of MLP neural network classifiers and LPDCs deployed independently and in an ensemble are tested empirically. In this chapter an overview of the methodology followed in the empirical investigation was given. Furthermore, four experiments with which to test the research hypothesis were defined. The next chapter gives the results of these experiments.

In the framework of the knowledge discovery and data mining process model being followed, this chapter introduces the data mining activity described in terms of four experiments. Specific attention has also been given to data understanding and data preparation. The data mining activity cumulates into the results given in the next section.

In the previous chapter four experiments were outlined. In each experiment, the 10-class file fragment type identification problem was solved using a different approach. The 10-class labels refer to the 10 file types considered in this dissertation, as outlined in Section 4.6.1. The results of these experiments are presented in this chapter.

In Sections 5.2 to 5.5 each experiment that has been introduced in the previous chapter are formulated in more detail. For each experiment, the result of hyper-parameter optimization and the best model, as identified in hyper-parameter optimization, are reported. The models are compared statistically and conclusions are drawn in Section 5.6. Before getting into the results of the first experiment, a general comment on the choice of hyper-parameter values are made in Section 5.1.

5.1 A note on hyper-parameter optimization

As outlined in Section 4.2, the problem of identifying a good hyper-parameter configuration from the set of all possible hyper-parameter configurations is called hyper-parameter optimization. Testing all possible hyper-parameter configurations is computationally prohibitive; therefore, only a subset of all configurations is considered.

In this dissertation, grid search and random search are used to select the subset of configurations tested in each experiment. The use of these approaches makes hyper-parameter optimization reproducible, automatable and justifiable (Bergstra and Bengio, 2012). However, the data mining practitioner still has to select the grid resolution and grid size that bound the grid search as well as the range of values considered during a random search. As stated by Nisbet *et al.* (2009), it is in the selection of these values where data mining becomes an art rather than a science.

Hyper-parameter optimization performed in this dissertation has been set up after taking the parameters used in similar experiments (Bergstra and Bengio, 2012; Axelsson, 2010; Cao *et al.*, 2010; Amirani *et al.*, 2008; Harris, 2007), as well as the results of exploratory experiments, performed as part of this dissertation, into careful consideration. In this way, the results presented in this chapter are the result of a non-linear process where the output of one cycle was used to refine the next cycle. In each cycle the hyper-parameter optimization problem was refined by eliminating hyper-parameter values that did not seem promising, giving more attention to those that do. As stated in Section 1.4, the knowledge discovery and data mining process is exactly this activity of cycling through the process model using the output of one cycle to provide feedback to the next cycle.

Each hyper-parameter configuration was evaluated using a validation set, which is independent from the testing set. The best hyper-parameter configuration, based on classification accuracy, for each experiment is used in the final model for which results are reported. All the results reported in this chapter were estimated with 10-fold cross-validation. This procedure was outlined in Section 4.3.

5.2 Experiment 1: k -nearest neighbour

A major shortcoming in the file object type identification research domain is the comparability of research results. As stated in Chapter 3, a comparison between the results reported by two researchers is of limited value due to differences in the problem formulation used, the size of the classification problem studied, the file types considered and the file object under observation. As such, Experiment 1 is an attempt to make the results reported in this dissertation somewhat comparable to other approaches found in the literature. A benchmark model, deemed representative of the current “state of the art” in file object type identification, is provided as reference point. A simple k -nearest neighbour classifier is proposed as a benchmark due to its popularity in the literature (Table 3.6).

The k -nearest neighbour algorithm was used to solve the 10-class file fragment type identification problem. As outlined in Section 2.4, the k -nearest neighbour algorithm assigns a file fragment to a file type based on a majority vote of the file types observed over the k -nearest file fragments in the reference set provided.

The SAS[®]PROC DISCRIM procedure was used to implement the nearest neighbour algorithm (SAS, 2009). The Mahalanobis distance measure was used to measure the similarity between two file fragments. This distance measure is defined as,

$$d(\mathbf{x}, \tilde{\mathbf{x}}) = \sqrt{(\mathbf{x} - \tilde{\mathbf{x}})^T \mathbf{S}^{-1} (\mathbf{x} - \tilde{\mathbf{x}})} \quad (5.1)$$

where $\mathbf{x}$ is an input pattern in the reference set, $\tilde{\mathbf{x}}$ is the unlabelled input pattern and $\mathbf{S}$ is the covariance matrix calculated over all instances in the reference set.

All available file fragments in the training set were included in the set of file fragments referenced by the k -nearest neighbour algorithm. Instance selection (as introduced in Section 2.4) was not implemented because exploratory experiments showed that classification using prototype file fragments as reference set was less accurate than classification using all available exemplar file fragments as reference set. This finding is supported by the literature represented in Subsection 3.5.1.

The k -nearest neighbour algorithm is sensitive to ties for any k values larger than one. That is, the majority vote over the set of nearest neighbours observed for a given file fragment can be indecisive. Such a tie needs to be resolved in order to assign an instance to a particular file type. In the results reported here, tied instances were deemed incorrectly classified. However, a weighted k -nearest neighbour classifier with a higher classification accuracy may exist (Macleod *et al.*, 1987). A weighted k -nearest neighbour classifier was

not used as benchmark in this dissertation; firstly, in order to keep the proposed benchmark model as simple as possible and secondly, because no weighted k -nearest neighbour algorithm has been applied to file object type identification in the literature surveyed. Therefore, a simple k -nearest neighbour algorithm is representative of the current “state of the art” in file object type identification.

5.2.1 Hyper-parameter selection

In this implementation of the k -nearest neighbour algorithm, the number of top-ranked input features c to use and the value of algorithmic parameter k had to be configured. Optimal values were determined by a grid search (Section 4.2). The c features used to describe a file fragment are the c top-ranked features in the ranking presented in Subsection 4.6.3. The value of c was alternated between 50, 100, 150, and 200 while the value of k was varied from 1 to 31 by 1. The grid size and grid resolution was deemed reasonable after taking into account the literature surveyed in Chapter 3, as well as the results of exploratory experiments (Section 5.1). In total 124 different hyper-parameter configurations were tested, for which the results are given in Figure 5.1.

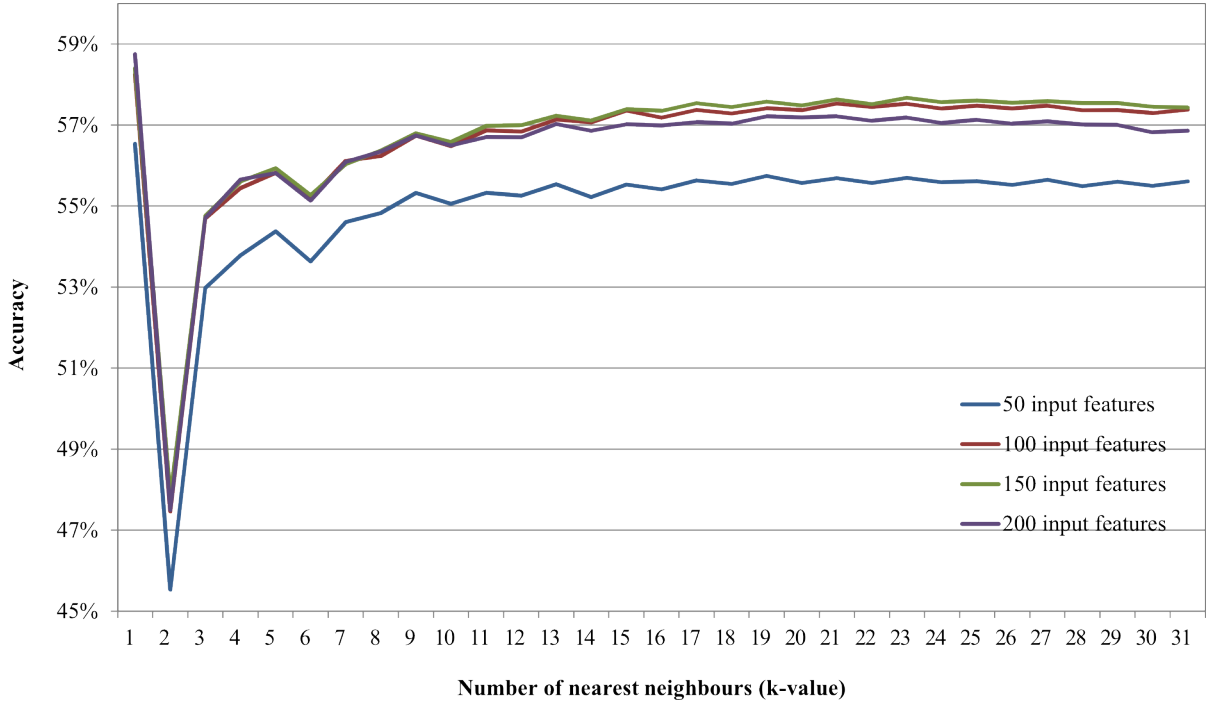


Figure 5.1: Classification accuracy for different hyper-parameter configurations.

The highest classification accuracy is observed at $k = 1$. A noticeable increase in classification accuracy is observed when using 100, 150 or 200 input features as opposed to 50 input features. The change in classification accuracy when using 100, 150 and 200 features respectively is negligible. Therefore, $k = 1$ and $c = 100$ was used in the final model for which the results are reported in the next subsection.

5.2.2 Model evaluation

The classification accuracy of the 1-nearest neighbour algorithm using 100 input features is 66.2 percent with an average precision of 66.8 and an average recall of 66.2. The classification accuracy, precision and recall rate are calculated from the confusion matrix as outlined in Section 4.4.

A confusion matrix for Experiment 1 is given in Table 5.1. To ease interpretation, the values in this table are given as the percentage of the total number of instances in the testing set instead of actual number of instances in the testing set traditionally reported in confusion matrices. The confusion matrix can be interpreted as outlined in in Section 4.4. For example, 9.5 percent of file fragments in the testing set is actual `html` file fragments which was correctly classified. In total 10 percent of the testing set is `html` file fragments, giving a correct `html` classification rate of 95.0 percent. A total 10.13 percent of all the instances in the testing set was classified as belonging to the `html` file fragments. Similarly, ten percent of the testing set is `gif` file fragments. In total 6.67 percent of the testing set was actual `gif` file fragments correctly predicted to be `gif` file fragments. That is, 66.7 percent of `gif` file fragments was classified correctly.

Table 5.1: Cross-validated confusion matrix for the nearest neighbour algorithm. Note: *Zero cells represent some very small value which rounds to zero, whereas empty cells represent actual zero values.*

Actual/ Predicted	html	gif	ps	xls	ppt	jpg	doc	txt	gz	pdf	Total	Recall
html	9.50		0.01	0.07	0.00		0.18	0.24		0.01	10	94.96
gif		6.67		0.02	0.75	0.30	0.61		0.90	0.75	10	66.70
ps	0.04	0.01	9.67	0.03	0.01	0.00	0.01	0.11	0.00	0.14	10	96.73
xls	0.05	0.01	0.01	9.33	0.13	0.04	0.31	0.04	0.04	0.04	10	93.31
ppt		0.53	0.01	0.12	2.80	1.25	1.64		2.07	1.57	10	28.03
jpg		0.28	0.00	0.05	1.27	5.44	1.04		0.88	1.06	10	54.42
doc	0.16	0.30	0.01	0.32	1.48	0.94	4.21	0.06	1.37	1.17	10	42.09
txt	0.39		0.04	0.07			0.05	9.45		0.00	10	94.51
gz		0.44	0.00	0.03	1.81	0.51	1.47		4.01	1.73	10	40.11
pdf	0.01	0.35	0.10	0.05	1.26	0.79	1.03	0.01	1.34	5.07	10	50.74
Total	10.13	8.58	9.85	10.06	9.52	9.27	10.55	9.90	10.61	11.54	100	
Precision	93.75	77.76	98.22	92.72	29.45	58.68	39.91	95.50	37.80	43.97		

The nearest neighbour algorithm performed well in predicting some file types and considerably worse in predicting other file types. Though approximately 95.0, 96.7, 93.3, and 94.5 percent of `html`, `ps`, `xls` and `txt` file fragments were correctly identified only 28.0, 42.1 and 40.1 percent of `ppt`, `doc` and `gz` file fragments were correctly identified. A large percentage of `ppt`, `doc` and `pdf` file fragments was mistaken for `gz` files by the k -nearest neighbour algorithm, resulting in low recall for these file types and a low precision for the `gz` file type. A low precision for `ppt` file fragments was observed because a high frequency of actual `jpg`, `doc`, `gz` and `pdf` file fragments was identified as `ppt` file fragments.

5.3 Experiment 2: Single MLP neural network classifier

In this experiment a single MLP neural network classifier was used to solve the 10-class file fragment identification problem. The SAS[®]PROC NEURAL procedure was used to implement the neural network (SAS, 2009). The default three-layer MLP architecture was selected, which includes linear combination functions in the hidden and output layer nodes, symmetric sigmoid activation functions in the hidden layer nodes, and softmax activation functions in the output layer nodes (Sarma, 2007). The softmax activation function is a generalization of the logistic function to L output nodes, where the node output is calculated as,

$$\hat{y}_l = \frac{e^{u_l}}{\sum_{i=1}^L e^{u_i}} \quad (5.2)$$

with u_i and u_l the node input of the i^{th} and l^{th} output node respectively. The softmax activation function allows the network output to be interpreted as posterior probabilities (Bishop, 1995). An instance was assigned to the class with the highest probability.

The cross-entropy loss function, the default loss function in PROC NEURAL for multiple nominal output nodes, was used. This cross-entropy loss function for a neural network with L output nodes is defined as,

$$L(\mathbf{y}, \hat{\mathbf{y}} = h(\mathbf{x})) = \sum_{l=1}^L y_l \ln\left(\frac{\hat{y}_l}{y_l}\right) + (1 - y_l) \ln\left(\frac{1 - \hat{y}_l}{1 - y_l}\right) \quad (5.3)$$

The back propagation algorithm, as outlined in Algorithm 2.1, was used to train the neural network classifier with a default convergence criterion of $\epsilon = 10^{-8}$ and the maximum number of iterations set to $\tau = 20000$. The maximum number of iterations was set above the default SAS[®]PROC NEURAL value to allow the back propagation algorithm to converge even when using small learning rate values. The back propagation algorithm was used for the reasons outlined in Section 2.2.2.

5.3.1 Hyper-parameter selection

In the implementation of the multiple class MLP neural network classifier, four hyper-parameters were considered: the optimal number of input features c , the appropriate number of hidden layer nodes h , the optimal learning rate η and the appropriate strength of weight regularization λ . The strength of weight regularization λ determines the extent to which weight decay influences the estimated empirical loss for a given neural network (Section 2.2.3).

A grid search of this four-dimensional hyper-parameter space was computationally prohibitive, therefore a random search was performed instead (Section 4.2). For each trial in the random search the number of input features c , was set with equal probability, to either

100 or 200; the number of hidden layer nodes h was drawn geometrically¹ between 5 and 300; the learning rate was drawn exponentially² as a value between 0.0001 and 0.1; and the strength of the weight regularization λ was draw exponentially in the range of 0.0001 and 0.5. By drawing values geometrically and exponentially, simpler models are favoured by oversampling the lower end of the hyper-parameter range. The performance of the MLP neural network classifier was evaluated for 400 unique hyper-parameter configurations. The predefined range of allowable values for each hyper-parameter, as well as the number of trials used were deemed reasonable when taking into account the literature surveyed in Chapter 3, the results of exploratory experiments (Section 5.1) and the random search performed by Bergstra and Bengio (2012).

The top five hyper-parameter configurations found, based on classification accuracy, are given in Table 5.2. The difference in the estimated classification accuracy between these models is negligible, therefore in line with Occam’s razor (Domingos, 1999) the least complex neural network (based on the number of hidden layer nodes) was used to implement the MLP neural network classifier. That is, $(c, h, \eta, \lambda) = (200, 56, 0.098, 0.0002)$ was used in the final model for which the results are reported in the next subsection.

Table 5.2: Classification accuracy, $\bar{A}$, for the five best MLP neural network classifiers found.

c	h	η	λ	$\bar{A}$
200	106	0.097	0.0004	0.7242
200	56	0.098	0.0002	0.7184
200	166	0.07	0.0072	0.7166
200	110	0.07	0.0001	0.7153
200	70	0.07	0.0041	0.7145

5.3.2 Model evaluation

The optimal parameter found in the random hyper-parameter search was used to implement the MLP neural network classifier. The classification accuracy of this neural network is 73.5 percent, with an average precision and recall of 74.0 percent and 73.6 percent respectively. The confusion matrix is given in Table 5.3. Again, while the neural network classifier was able to correctly predict the file type of many file fragments, a low classification accuracy is observed for **ppt**, **doc**, **gz** and **pdf** file fragments. The neural network’s ability to correctly detect **doc**, **ppt** and **pdf** file types is low with a recall of approximately 34, 33 and 47 percent respectively. The precision in detecting **gz** and **ppt** file types is approximately 45 and 39

¹To draw geometrically between A and B means to draw a value uniformly between $\log(A)$ and $\log(B)$, exponentiating to get a number between A and B, and rounding to the nearest integer (Bergstra and Bengio, 2012).

²The same as drawing geometrically without rounding the result to the nearest integer.

percent respectively. The single MLP neural network performed better than the 1-nearest neighbour algorithm. A comparison is given in Section 5.6.

Table 5.3: Cross-validated confusion matrix for the MLP neural network classifier.

Actual/ Predicted	html	gif	ps	xls	ppt	jpg	doc	txt	gz	pdf	Total	Recall
html	9.50		0.01	0.01			0.03	0.44		0.01	10	94.95
gif		8.99			0.19	0.05	0.05		0.50	0.23	10	89.90
ps	0.07	0.00	9.79	0.00	0.00		0.00	0.10	0.01	0.03	10	97.87
xls	0.02	0.01	0.01	9.34	0.09	0.06	0.38	0.02	0.06	0.02	10	93.40
ppt	0.02	0.54	0.02	0.08	3.33	1.90	0.62	0.00	2.92	0.59	10	33.25
jpg	0.04	0.21		0.01	0.69	8.13	0.24		0.30	0.38	10	81.33
doc	0.05	0.34	0.01	0.28	1.58	1.68	3.42	0.04	2.17	0.45	10	34.15
txt	0.34		0.05	0.03		0.00	0.04	9.54		0.01	10	95.38
gz		0.73		0.01	1.32	0.27	0.26		6.78	0.64	10	67.80
pdf	0.02	0.33	0.22	0.02	1.17	1.12	0.28	0.02	2.06	4.77	10	47.73
Total	10.06	11.16	10.10	9.77	8.35	13.20	5.31	10.15	14.79	7.12	100	
Precision	94.41	80.59	96.88	95.57	39.82	61.60	64.34	93.99	45.86	67.03		

5.4 Experiment 3: LP-based discriminant classifiers in round robin

In Experiment 3 the 10-class file fragment type identification problem was decomposed into 45 one-against-one binary classification subproblems — one subproblem for each unique pair of file types in the set of file types considered. Each binary subproblem was then solved using a LP-based discriminant classifier (LPDC). The majority vote was taken over all 45 discriminant classifiers to classify a file fragment into one of the 10 file types considered. The set of 45 discriminant classifiers can therefore be seen as one composite classifier because all 45 discriminant classifiers need to be evaluated to make a final prediction. Furnkranz (2001) refers to this form of problem decomposition as round robin binarization.

The two-phase LP-based discriminant analysis (LPDA) model outlined in Section 2.3.3 was used to derive the discriminant classifier that best separates file fragments belonging to file type i and file fragments belonging to file type j . Algorithm 5.1 describes the process followed to derive the set of discriminant classifiers using a training set T and to test the set of discriminant classifiers on testing set S . Each LPDA model was solved in the SAS[®]PROC OPTMODEL mathematical programming environment using the simplex algorithm outlined in Algorithm 2.2.

Algorithm 5.1 Linear programming-based discriminant classifiers in round robin.

input: Training set T , testing set S , and number of file types n

output: Testing set S with each instance classified as one of the n predefined file types

begin algorithm

$r=0$ ***Initialize counter for the number of file type pairs*

for $i = 0$ **to** $n - 1$ **do**

for $j = i$ **to** n **do**

$r++$

 Create temporary training set T' by selecting all instances in T belonging to file type i (denoted G_i) and all instances belonging to file type j (denoted G_j)

 Derive discriminant function weights $\mathbf{w}_1^*$ by solving phase 1 of the LPDA model presented in Section 2.3.3 over temporary training set T'

 Use $\mathbf{w}_1^*$ to compute a discriminant score for each instance in the temporary training set T'

 Label instances in temporary training set T' as either G'_i (classified as file type i), G'_j (classified as file type j) or G'_f (in classification gap) based on the discriminant scores

 Derive discriminant function weights $\mathbf{w}_2^*$ and boundary value b^* by solving phase 2 of the LPDA model presented in Section 2.3.3 over temporary training set T'

 Use $\mathbf{w}_2^*$ to compute a discriminant score for each instance in testing set S

 Use the calculated discriminant score and the boundary value b^* to classify each instance in testing set S as either file type i or file type j in a new class variable $vote_r$ ***Testing set S now has new column $vote_r$*

end for

end for

 Calculate the total number of votes towards file type i (for $i = 1, \dots, n$) over all the individual class labels $vote_r$ (for $r = 1, \dots, \frac{n(n-1)}{2}$).

 Classify each instance in testing set S as the file type $G_1, \dots, G_n$ with the highest aggregate vote.

end algorithm

5.4.1 Hyper-parameter selection

The only hyper-parameter that had to be configured was the number of top-ranked input features c used to describe each file fragment. A one-dimensional grid search was performed to test c values of 50, 100, 150, and 200. The grid size and grid resolution were deemed reasonable when taking into account the literature surveyed in Chapter 3, as well as the results of exploratory experiments (Section 5.1).

Figure 5.2 shows the classification accuracy for each value of c . A relatively large increase in classification accuracy is observed when using 100 input features instead of 50 input features. A marginal change in classification accuracy is observed when using 150 input features as opposed to 100 input features. Therefore, in line with Occam's razor, 150 input features were used.

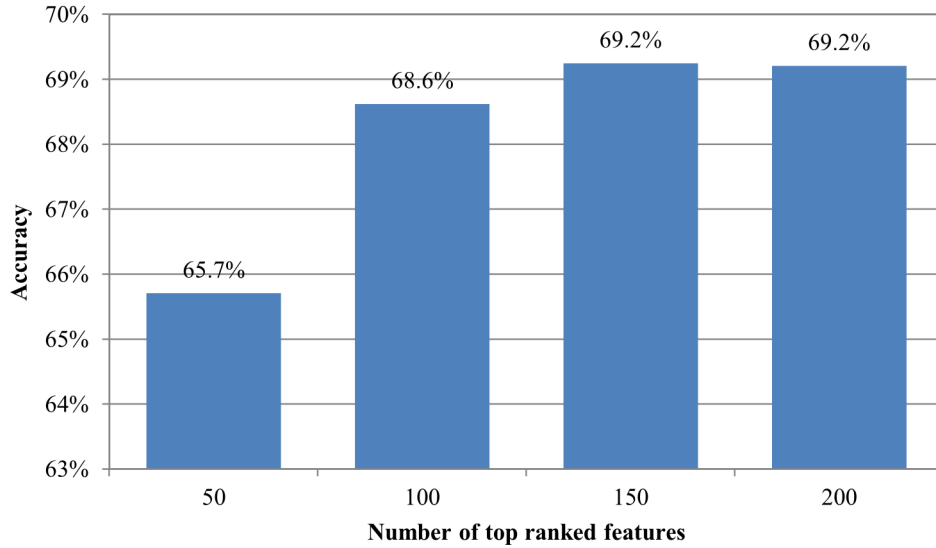


Figure 5.2: Classification accuracy for different hyper-parameter configurations.

5.4.2 Model evaluation

The 10-class problem was solved using a set of LPDCs and 150 input features. The classification accuracy of the composite classifier is 71.8 percent, with an average precision of 71.3 percent and an average recall of 71.8 percent. The confusion matrix is given in Table 5.4. The composite LPDC failed to accurately predict the file types of **ppt**, **doc**, **gz** and **pdf** file fragments, in much the same fashion as the k -nearest neighbour algorithm and the MLP neural network classifier. The single composite LPDC performed better than the 1-nearest neighbour algorithm and marginally worse than the MLP neural network classifier. A comparison is given in Section 5.6.

Table 5.4: Cross-validated confusion matrix for the composite LPDC.

Actual/ Predicted	html	gif	ps	xls	ppt	jpg	doc	txt	gz	pdf	Total	Recall
html	9.06		0.07	0.05	0.01	0.03	0.06	0.59	0.01	0.14	10	90.60
gif		8.20			0.52	0.04	0.30		0.53	0.42	10	82.00
ps	0.03		9.52	0.03	0.01	0.02	0.01	0.15	0.02	0.22	10	95.20
xls	0.01	0.01	0.01	9.01	0.13	0.12	0.52	0.10	0.05	0.06	10	90.10
ppt		0.48	0.01	0.05	2.67	1.87	1.40		2.63	0.90	10	26.70
jpg		0.11			0.86	7.75	0.56		0.10	0.62	10	77.50
doc	0.02	0.18		0.32	1.47	1.50	3.66	0.11	2.08	0.67	10	36.60
txt	0.20		0.05	0.01			0.02	9.70		0.03	10	97.00
gz		0.39	0.01		1.21	0.11	0.42		6.84	1.02	10	68.40
pdf		0.24	0.13	0.01	0.95	1.01	0.43	0.01	1.85	5.37	10	53.70
Total	9.32	9.61	9.80	9.48	7.83	12.45	7.38	10.66	14.11	9.45	100	
Precision	97.21	85.33	97.14	95.04	34.10	62.25	49.59	90.99	48.48	56.83		

5.5 Experiment 4: Ensembles

The research hypothesis of this dissertation is that more accurate file fragment type identification can be achieved by using MLP neural network classifiers and LPDCs in an ensemble instead of using these classifiers individually. Experiments 2 and 3 test the usefulness of a single MLP neural network classifier and a single composite LPDC. The usefulness of several of these models used together in an ensemble is tested in Experiment 4. Three different ensembles are outlined in Subsections 5.5.1, 5.5.2 and 5.5.3; after which the results are given in Subsection 5.5.4.

5.5.1 Experiment 4a: Ensemble of multiple MLP neural networks

In Experiment 2 a single MLP neural network classifier was constructed to solve the 10-class file type identification problem. The best neural network classifier was found using random hyper-parameter search. In this experiment ten such MLP neural networks, each using a different hyper-parameter configuration, were combined. Different hyper-parameter configurations were used in an attempt to introduce diversity into the ensemble (Maclin *et al.*, 1995). The hyper-parameter configuration of each model was selected randomly from the set of fair hyper-parameter configurations found as part of the random search implemented in Experiment 2. A hyper-parameter configuration was deemed fair if it had an associated classification accuracy higher than 30 percent (i.e. a classification accuracy 3 times better than assigning a file fragment to a random file type). The reason for enforcing a lower bound of 30 percent was to ensure that all classifiers included in the ensemble was at least weakly accurate. The probability estimate for each file type provided by each individual MLP neural network classifier was summed into an aggregate probability estimate for each file type (Kuncheva, 2004). An instance was then assigned to the file type with the highest aggregate probability estimate.

5.5.2 Experiment 4b: Ensemble of multiple LP-based discriminant classifiers

In Experiment 3, a set of 45 binary LPDCs was used to solve the 10-class classification problem. All 45 discriminant classifiers were evaluated to classify an instance into one of the ten file types considered. The 45 discriminant classifiers can therefore be seen as one composite classifier. In this experiment ten such composite classifiers, or 450 binary LPDCs, were constructed and combined in an ensemble. The available feature set was partitioned into 10 disjoint subsets, and every composite classifier was trained on a different subset of features in an attempt to introduce diversity into the ensemble (Chawla *et al.*, 2004). The

reason for partitioning the available features into 10 disjoint subsets was to provide each individual classifier included in the ensemble with enough features in order to be at least weakly accurate.

The total number of votes towards each file type over the set of 450 votes was determined. The probability of a file fragment belonging to file type i is estimated as the percentage of total votes in favour of file type i . An instance was then assigned to the file type with the highest probability estimate (i.e. an instance was assigned to the file type with the most votes).

5.5.3 Experiment 4c: Combining multiple MLP neural network and multiple LP-based discriminant classifiers

In Experiment 2 and 3, the 10-class file fragment classification problem was solved using a single neural network and a single composite LPDC. In Experiment 4a multiple MLP neural networks were combined in an ensemble. Diversity was introduced into this ensemble by using a different hyper-parameter configuration in each network. In Experiment 4b multiple composite LPDCs were combined in an ensemble. Diversity was introduced into this ensemble by training each composite classifier on a disjoint subset of the available features.

In this experiment, diversity was introduced into an ensemble by combining classifiers with different model representations (Saha and Ekbal, 2013; Lee and Elder, 1997; Woods *et al.*, 1997). The aggregate probability estimates of the ensemble in Experiment 4a and the aggregate probability estimates in the ensemble in Experiment 4b were summed and an instance was then assigned to the file type with the highest aggregate probability estimate (Kuncheva, 2004).

5.5.4 Model evaluation

The classification accuracy for each of the ensembles is given in Figure 5.3. The ensemble of MLP neural network classifiers and LPDCs (Experiment 4c), with a classification accuracy of 67.3 percent, performed marginally better than the other ensembles.

Though the classification accuracies of the various ensembles are close, there seems to be a difference in the distribution of misclassified instances over the file types considered. Figure 5.4 shows the relative portion of the total misclassification rate attributed to each file type in each ensemble. There is a large variance in the portion of misclassified file fragments that belong to the `ppt`, `jpeg`, `gz` and `pdf` file types. For example, 23.9 percent of misclassified instances is `pdf` fragments in Experiment 4a while only 17.1 percent of misclassified instances is `pdf` fragments in Experiment 4b. In Experiment 4c 21.4 percent of misclassified instances was `pdf` file fragments.

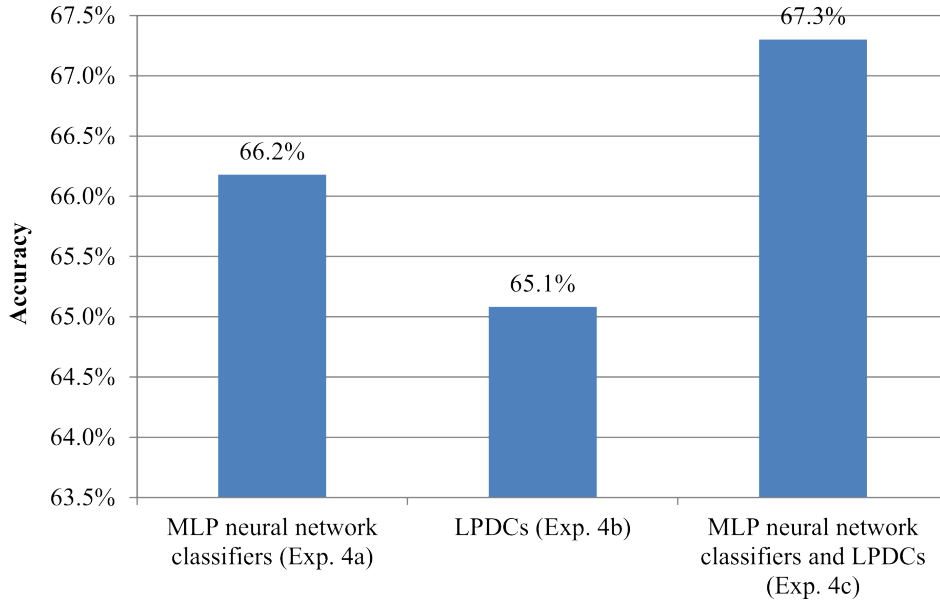
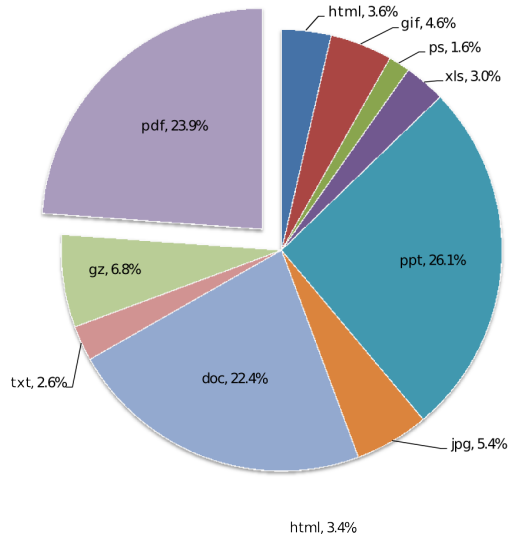


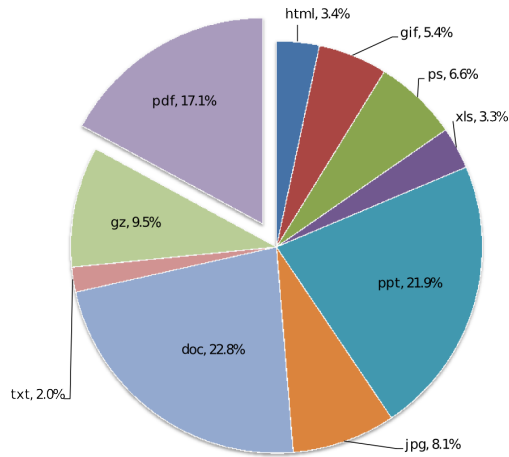
Figure 5.3: Classification accuracy achieved by each of the ensembles tested.

The classification accuracies of the ensembles tested are less than expected. This can be attributed to a lack of diversity between the individual classifiers making up the ensembles. Figure 5.5(a) shows the pairwise correlation in the file type predicted by each individual classifier (labelled LPDA1 to MLP10) included in the ensembles. For example, the file types predicted by MLP9 and MLP3 were very highly correlated. A correlation coefficient of nearly 0.8 was calculated for this pair of classifiers. The same conclusion is drawn when measuring model diversity in terms of the probability that two classifiers will predict the same file type for any given file fragment. Such pairwise probabilities are given in Figure 5.5(b). For example, the probability that MLP7 and LPDC5 will be assigning a file fragment to the same file type is lower than the probability that MLP5 and MLP1 will be assigning a file fragment to the same file type.

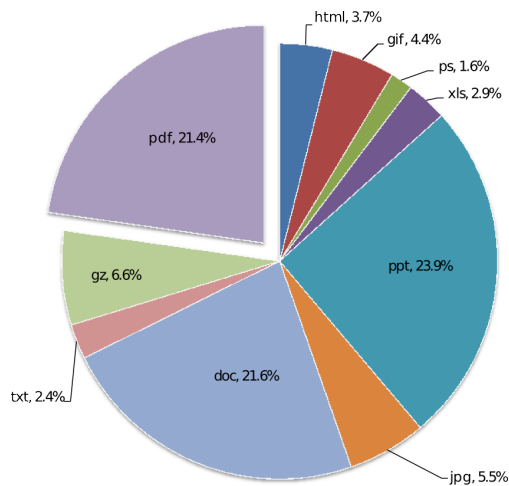
No two classifiers are negatively correlated. The average pairwise correlation of all composite LPDCs (LPDC1 to LPDC10) is 0.52 while the average pairwise correlation of all MLP neural network classifiers (MLP1 to MLP10) is 0.65. The average pairwise correlation of all classifiers (LPDC1 to MLP10) is 0.57. The ensemble of composite LPDCs (Experiment 4b) is therefore more diverse than the other two ensembles. Therefore, as shown in Table 5.5, the improvement over the single best individual classifier is more in the ensemble of composite LPDCs (Experiment 4b) than in the other ensembles. The best single composite LPDC in the ensemble of composite LPDCs has a classification accuracy of 61.4 percent. The classification accuracy of the ensemble of composite LPDCs is 3.7 percent higher than the classification accuracy of the best single discriminant classifier included the ensemble. The ensemble of MLP neural network classifiers (Experiment 4a) improved over the best individual classifier in the ensemble with -0.7 percent while the ensemble of MLP neural network classifiers and LPDCs (Experiment 4c) improved over the best individual classifier



(a) Ensemble of MLP neural network classifiers (Exp. 4a)

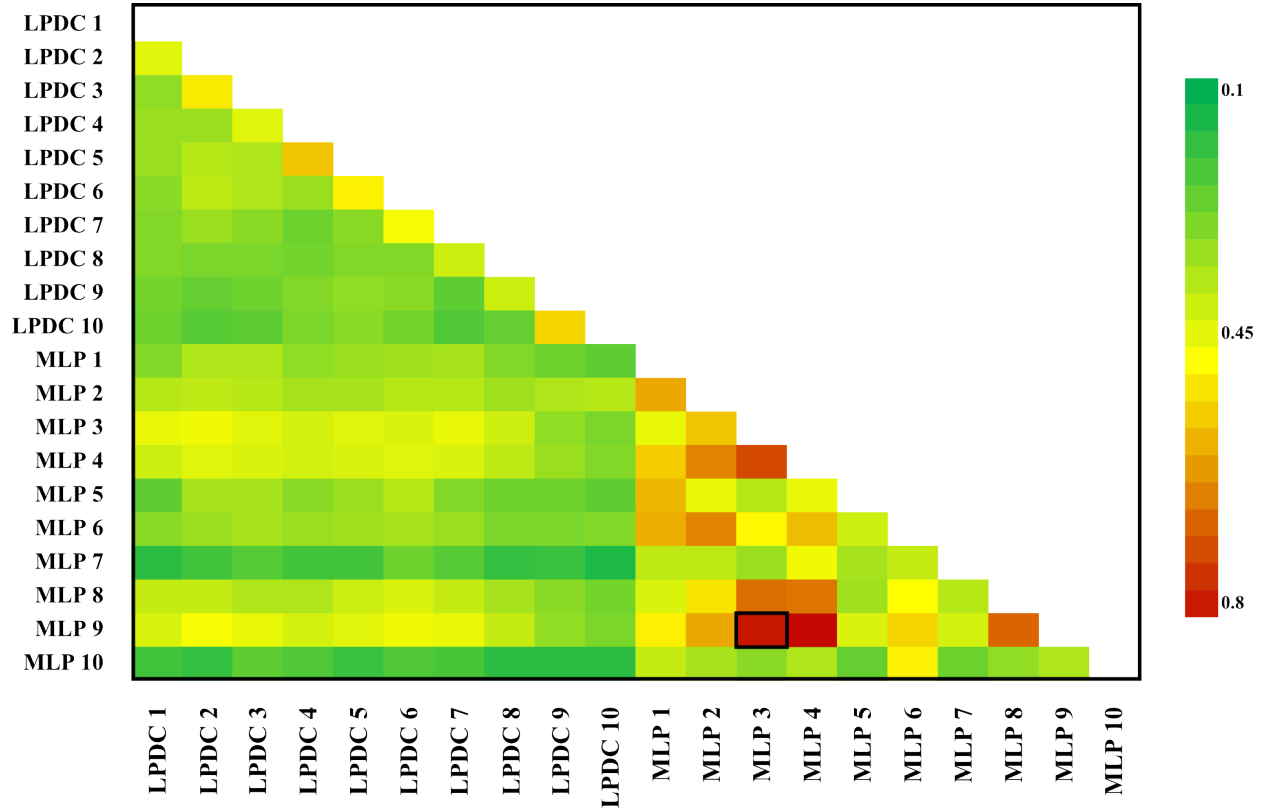


(b) Ensemble of LPDCs (Exp. 4b)

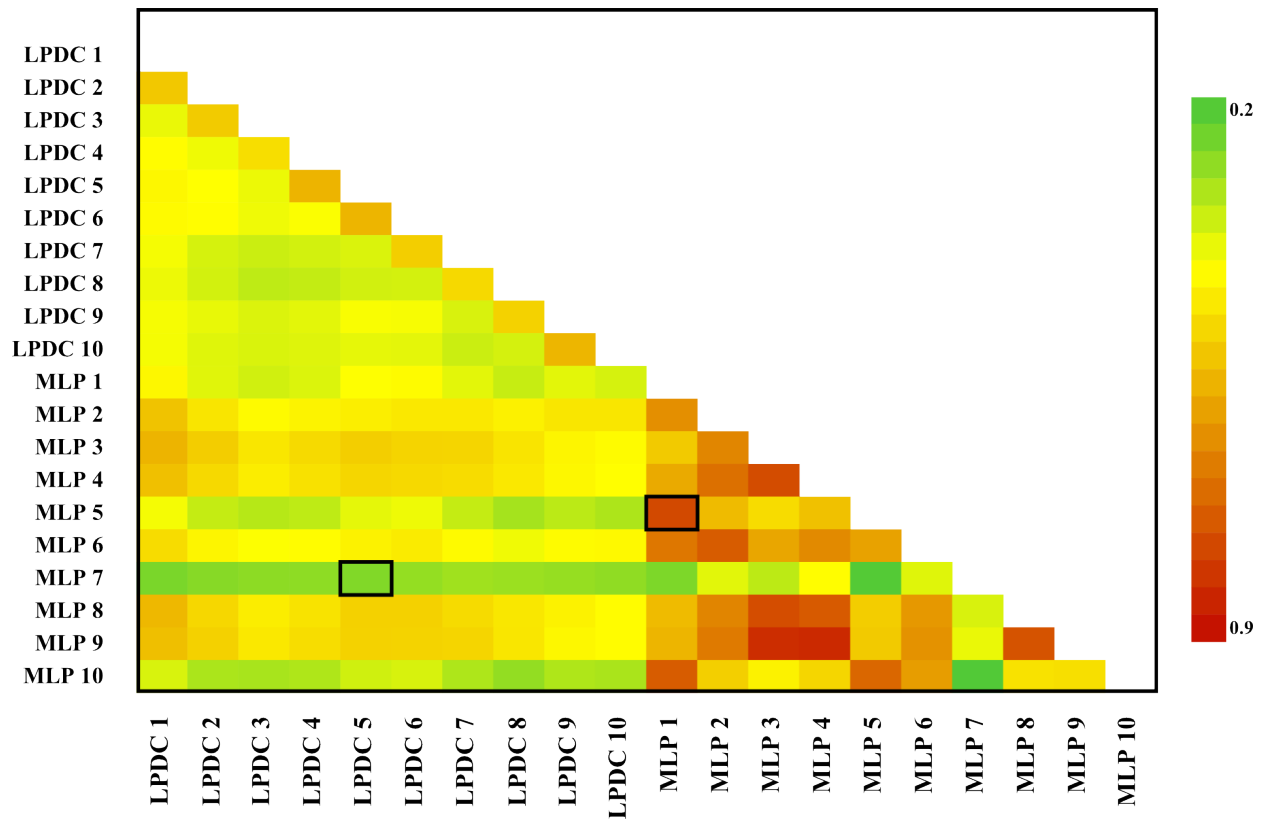


(c) Ensemble of MLP neural network classifiers and LPDCs (Exp. 4c)

Figure 5.4: The profile of misclassified fragments per file type per ensemble.



(a) Correlation in predictions



(b) Probability to make the same classification

Figure 5.5: Diversity of the ensembles

in the ensemble with 0.5 percent.

Table 5.5: The improvement in classification accuracy over the best individual classifier included in a ensemble.

	Average pairwise correlation	Average accuracy of individual classifiers in ensemble	Accuracy of best individual classifier in ensemble	Ensemble accuracy	Improve- ment over the best individual classifier in ensemble
Exp. 4a (MLPs)	0.65	54.9	66.9	66.2	-0.7
Exp. 4b (LPDCs)	0.52	53.3	61.4	65.1	3.7
Exp. 4c (MLPs & LPDCs)	0.57	54.1	66.9	67.3	0.5

5.6 Model comparison and conclusion

The results reported in each experiment are summarized in Figure 5.6. The highest classification accuracy of approximately 74 percent was achieved by using a single MLP neural network classifier (Experiment 2). Therefore, if a well-selected single MLP neural network configuration can be found it would appear to be fairly well suited for file fragment type identification.

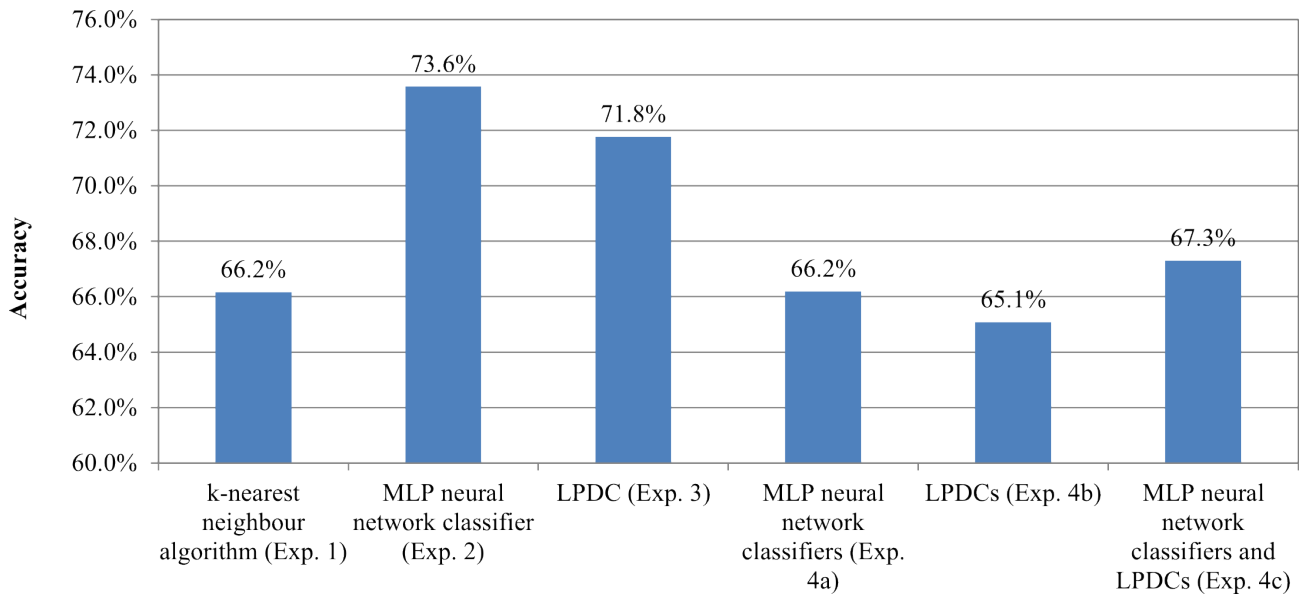


Figure 5.6: Classification accuracy achieved in each experiment.

Though a single MLP neural network classifier (Experiment 2) seemed to outperform the other approaches on average, this is not the case when evaluating the performance of the proposed approaches over each file type individually. The classification accuracy of each experiment per file type is summarized in Figure 5.7. The k -nearest neighbour algorithm (Experiment 1), the single composite LPDC (Experiment 3) and the ensemble of MLP neural network classifiers (Experiment 4a) seem to have performed better than a

single MLP neural network classifier (Experiment 2) in the identification of some file types. For example, the ensemble of composite LPDCs (Experiment 4a) outperformed the single MLP neural network classifier (Experiment 2) in detecting `gz` file fragments with a margin of approximately 10 percent. Furthermore, the single composite LPDC (Experiment 3) outperformed the single MLP neural network (Experiment 2) in correctly classifying `txt` and `pdf` file types, whereas the k -nearest neighbour algorithm (Experiment 1) outperformed the other approaches in correctly identifying `html` and `doc` files. This finding gives credibility to the idea that different classifiers are better suited to detect file fragments belonging to different file types.

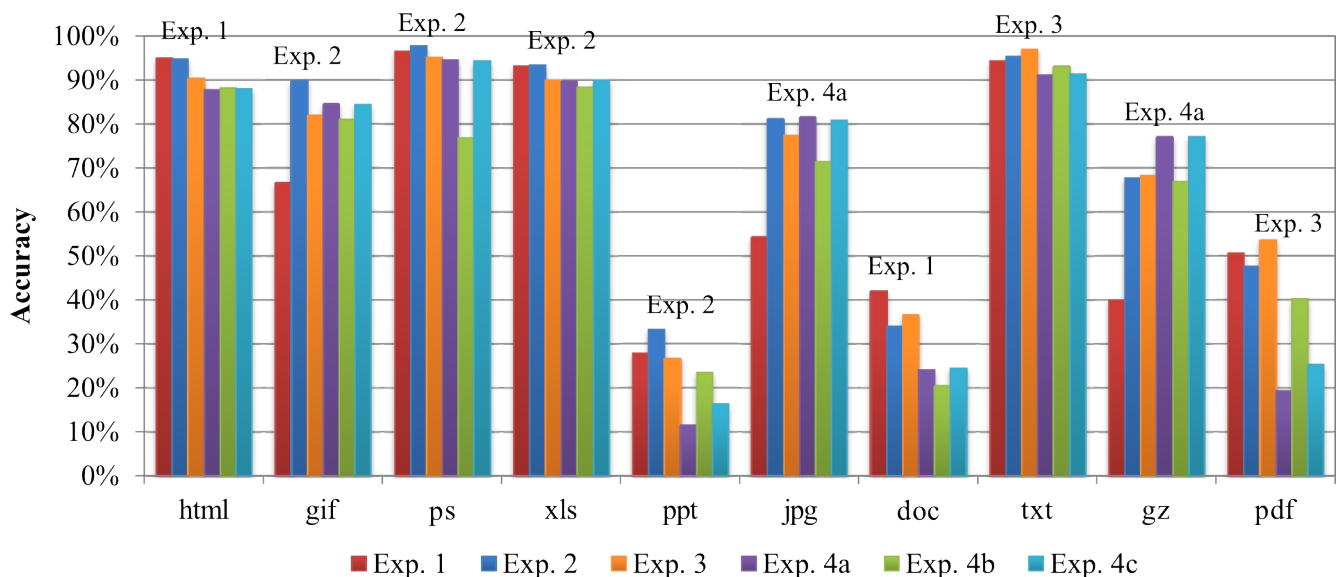


Figure 5.7: Classification accuracy achieved in each experiment per file type considered. The experiment in which the highest classification accuracy was obtained for each file type is given in the labels.

The expected classification accuracy achieved when randomly assigning a file fragment to any one of the ten file types considered is 10 percent. The fact that the classification accuracy of each file type is higher than 10 percent gives credibility to the idea that there is an underlying data pattern in the content of a file fragment, which can be found by machine learning algorithms.

When comparing the average recall and average precision in classifying file fragments belonging to different file types, as summarized in Figure 5.8, three patterns emerge: Firstly, `html`, `gif`, `ps`, `xls` and `txt` file patterns were classified with high precision and high recall. A great deal of trust can be placed in a file fragment assigned to one of these file types because the supervised learning techniques presented here were able to identify file fragments belonging to these file types correctly and reliably. Secondly, `ppt`, `doc` and `pdf` files were predicted with relatively high precision and low recall. That is, if a file fragment was predicted as belonging to one of these file type, it most likely does belong to the predicted file type. A large percentage of file fragments that does belong to these

file types was, however, not correctly classified. Thirdly, `jpg` and `gz` files were predicted with relatively high recall albeit low precision. That is, a large percentage of these files was predicted correctly. However, only a few of the file fragments predicted as belonging to `jpg` and the `gz` file type do in actual fact belong to these file types.

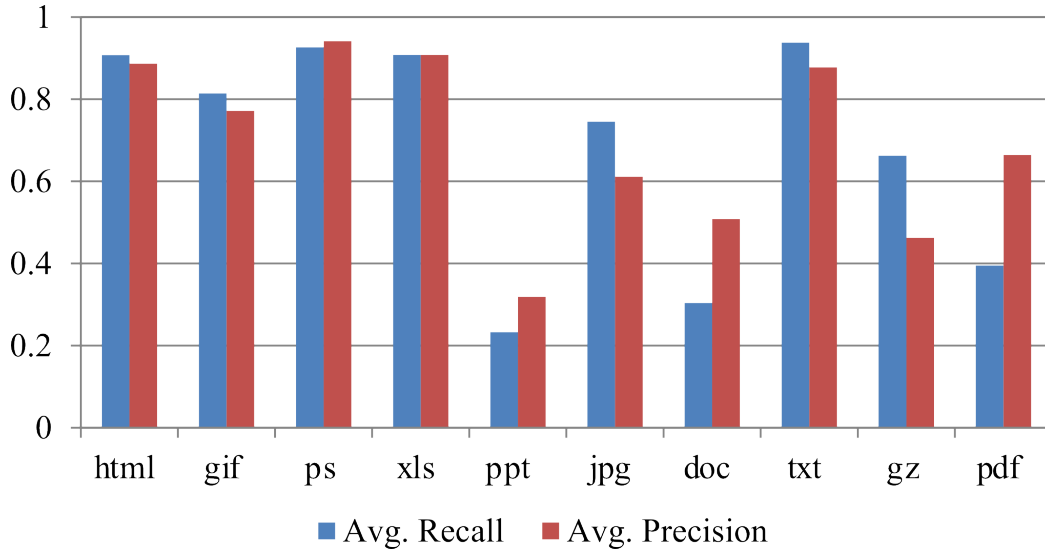


Figure 5.8: Average precision and recall per file type over all the experiments (excluding Experiment 1).

The identification of container files is problematic. As noted in Section 3.1, container files allow for different base files to coexist in one logical computer file. Identification of these file types on a fragment level is troublesome because a file fragment might, for all practical reasons, contain data consistent with another file type. For example, `pdf` file fragments were often confused with `jpg` file fragments. However, `pdf` files may encapsulate `jpg` files, therefore, it can be expected that some file fragments in the data set labelled as `pdf` file fragments are for all practical reasons actually `jpg` file fragments. These “incorrectly” labelled file fragments that exist in the data set influence the classification accuracy in two ways. Firstly, any induced classifier will learn to incorrectly identify `jpg` file fragments as `pdf` file fragments because some `jpg` file fragments are labelled as `pdf` file fragments in the *training* data set. Secondly, some actual `jpg` file fragments will be counted as incorrectly classified when assigned to the `jpg` file type, because they are labelled as `pdf` file fragments in the *testing* data set. The same argument holds for container file types such as, `ppt` and `doc`, which can also encapsulate image file fragments, and `gz` files that can encapsulate fragments of any file type.

When comparing the performance of two classifiers, the best classifier can be selected based on the perceived difference in the observed classification accuracy. Alternatively, a more scientific approach can be taken to statistically test the significance of the observed difference in the classification accuracy between two classifiers (Kumar, 2005). As such, the results of each experiment performed in this dissertation was compared statistically using a Student’s *t*-test as outlined in Section 4.5. For each pair of experiments, the *p*-values is

given in Table 5.6. The p value can be interpreted as the minimum level of significance at which the null hypothesis can be rejected. For example, the null hypothesis that the classification accuracy of the k -nearest neighbour algorithm (Experiment 1) is statistically equal to the classification accuracy of the composite LPDC (Experiment 3) can be rejected at a significance level of 0.02 in favour of the two-tailed hypothesis, which states that the performance of the two classifiers is not equal³. The best individual MLP neural network classifier in the ensemble of MLP neural network classifiers (Experiment 4a) and the best base composite LPDC in the ensemble of LPDCs (Experiment 4b) is also included in Table 5.6 for comparison.

Table 5.6: Model comparison using the two-tailed t -test. The null hypothesis is rejected in favour of the alternative hypothesis, at a significance level of 0.05, for all classifier pairs in bold. The classification accuracy, $\bar{A}$, for each experiment is included as reference from Figure 5.6

	$\bar{A}$	Exp. 1	Exp. 2	Exp. 3	Exp. 4a	Exp. 4b	Exp. 4c	Best MLP	Best LPDC
Exp. 1 (k -NN)	66.2		0.00	0.02	0.99	0.79	0.56	0.05	0.67
Exp. 2 (MLP)	73.6	0.00		0.28	0.00	0.04	0.02	0.00	0.00
Exp. 3 (LPDC)	71.8	0.02	0.28		0.00	0.02	0.03	0.00	0.01
Exp. 4a (MLPs)	66.2	0.99	0.00	0.00		0.75	0.28	0.03	0.66
Exp. 4b (LPDCs)	65.1	0.79	0.04	0.02	0.75		0.52	0.28	0.56
Exp. 4c (MLPs & LPDCs)	67.3	0.56	0.02	0.03	0.28	0.52		0.02	0.74
Best LPDC	61.4	0.05	0.00	0.00	0.03	0.28	0.02		0.03
Best MLP	66.9	0.67	0.00	0.01	0.66	0.56	0.74	0.03	

From Table 5.6 and Figure 5.6 the following conclusions may be drawn (assuming a significance level of 0.05 (Kumar, 2005)):

- The benchmark k -nearest neighbour algorithm (Experiment 1) performed statistically worse than the single MLP neural network classifier (Experiment 2, $p = 0.00$) and the single composite LPDC (Experiment 3, $p = 0.02$). This provides support for using more complex machine learning techniques in the attempt to solve the file fragment identification problem. Using an MLP neural network classifier or a composite LPDC instead of the k -nearest neighbour algorithm also has some other advantages as listed in Chapter 2.
- The single MLP neural network classifier (Experiment 2) seems to have outperformed the single composite LPDC (Experiment 3) by a margin of 1.82 percent, though this difference is not statistically significant ($p = 0.28$). Therefore a composite LPDC may be a valid alternative to an MLP neural network classifier, especially when considering that an LPDC is simpler to implement and to optimize. Various other advantages of LPDCs have been listed in Chapter 2.

³It follows that the null hypothesis can also be rejected in favour of the one-sided hypothesis that the composite LPDC (Experiment 3) performs better than the k -nearest neighbour algorithm (Experiment 1) at a significant level of 0.01 (0.02/2)

- While the ensemble of MLP neural network classifiers and composite LPDCs (Experiment 4c) performed marginally better than the other ensembles (Experiment 4a and Experiment 4b), these performance differences are not statistically significant ($p = 0.75$ and $p = 0.28$ respectively).
- The ensemble of MLP neural networks classifiers and composite LPDCs (Experiment 4c) performed marginally better than the k -nearest neighbour algorithm (Experiment 1) though these performance differences are not statistically significant ($p = 0.56$).
- The single MLP neural network classifier (Experiment 2) statistically outperformed all ensembles (Experiment 4).
- The single composite LPDC (Experiment 3) statistically outperformed all ensembles (Experiment 4).
- Though the ensemble of composite LPDCs (Experiment 4b) did not outperform the single composite LPDC (Experiment 3). The ensemble did, however, statistically outperform the best individual LPDC included in the ensemble.
- The ensemble of MLP neural network classifiers (Experiment 4c) did not statistically outperform the best individual MLP neural network classifier included in the ensemble.

These findings lead to the conclusion that an ensemble of classifiers may be able to improve over the best individual classifier included in the ensemble, however, in this problem domain an ensemble of classifiers may not be able to improve over a single optimized classifier. It is important to notice, that the individual classifier included in the ensembles presented here were, sometimes highly, positively correlated. Even though different hyper-parameter specifications were used in the ensemble of MLP neural network classifiers, different input features were used in the ensemble of composite LPDCs and different model representations were combined the ensembles were not diverse. As stated by Kuncheva (2004), an ensemble becomes more successful if the individual classifiers included in the ensemble are negatively correlated. It would seem that, in order to outperform the single optimized classifier, an ensemble of highly accurate and very diverse classifiers or a large ensemble of marginally accurate and very diverse classifiers needs to be constructed. This is, however, no feeble task (Woods *et al.*, 1997). Most actual ensembles consist of marginally accurate and positively correlated classifiers.

This study also found that a single MLP neural network classifier (Experiment 2) or a set of LPDCs in a round robin (Experiment 3) can outperform the k -nearest neighbour algorithm, which has been used extensively in the literature. Both MLP neural network classifier and LPDCs should be considered in this problem domain. It was found that the performance of LPDCs in a round robin may be comparably to the performance of an MLP neural network classifier. LPDCs in a round robin may therefore be considered as a valid and simple alternative to complex MLP neural networks.

5.7 Chapter summary

The data mining and knowledge discovery process accumulated to the results given and evaluated in this chapter. The results of four experiments have been discussed. Each experiment has been outlined, the selection of hyper-parameters described and the results evaluated using appropriate figures and tables. An overall comparison of the different experiments, from which conclusions were drawn, was also provided. The next chapter furnishes some final remarks.

An important task in digital forensics and security is to identify the true type of a file or file fragment without relying on traditionally used metadata. This study builds on existing literature regarding the use of supervised learning techniques for content-based file object type identification, and explores the combined use of MLP neural network classifiers and linear programming-based discriminant classifiers (LPDCs) as a solution to the multiple class file fragment type identification problem.

The purpose of this study was to investigate and compare the use of a single MLP neural network classifier, a single LPDC and a combined ensemble of these classifiers in the field of file type identification. The ability of each individual classifier and the ensemble of these classifiers to accurately predict the file type to which a file fragment belongs were tested empirically.

Chapter 6 presents the final comments and concluding remarks of the study. The objectives of the study and their achievement will be summarised. New problems and opportunities for further study that were presented during the research project will also be outlined.

6.1 Research objectives

Though the capacity of supervised learning algorithms to solve the file type classification problem has been shown in other literature, using an ensemble of classifiers to solve the multiple class file fragment classification problem is unique. As such, the primary research objective of this project was to describe how MLP neural network classifiers and LPDCs can be used in an ensemble to solve the multiple class file fragment type classification problem.

To accomplish the primary research objective, a list of secondary research objectives, which contribute to achieving the primary research objective was defined in Section 1.3 (Chapter 1). A summary of these objectives and how each objective was addressed follows:

1. *Give a brief overview of supervised learning as relevant to classification problems.*

The supervised learning problem was formally defined and a description of the classification problem was given in Section 2.1 (Chapter 2).

2. *Describe the MLP neural network classifier and LPDC.*

The MLP neural network classifier and the LPDC were outlined in Sections 2.2 and 2.3 (Chapter 2). In the discussion of both these techniques each model was outlined, the associated induction algorithm was dis-

cussed and some advantages, as well as disadvantages associated with the classifier were listed.

3. *Give an overview of literature relating to file type and file fragment type identification.*

File object type identification, consisting of file type identification and file fragment type identification, was introduced in Chapter 1. An overview of literature relating to file type and file fragment type identification was given in Chapter 3. Not only did Chapter 3 provide a context for this dissertation but it also served to guide experimentation in the use of an ensemble of MLP classifiers and LPDCs as a solution to the file fragment type identification problem.

4. *Outline how MLP neural network classifiers and LPDCs can be used individually and in an ensemble to solve the file fragment type identification problem.*

Ensemble techniques were described in Section 2.5 (Chapter 2). It was noted that a set of diverse classifiers might theoretically perform better than an individual classifier. Various techniques for generating an ensemble of diverse classifiers and techniques for combining these classifiers in an ensemble were outlined.

One of the techniques listed was to combine classifiers of various model representations. It was noted that the rationale behind this dissertation's research hypothesis is that the MLP neural network classifier and the LPDC have different model representations and should therefore be good candidates to participate in an ensemble.

In Experiments 2 and 3 (Chapter 5), the MLP neural network classifier and the LPDC were applied individually as a solution to the 10-file type identification problem. The purpose of these experiments was to establish a baseline of how these classifiers perform when applied individually.

In Experiment 4 (Chapter 5), three possible ensembles consisting of MLP neural network classifiers and LPDCs were outlined. The purpose of these experiments was to determine whether ensembles can be effective tools in the domain of file fragment type identification.

5. *Empirically test the effectiveness of individual classifiers and the ensemble of classifiers in the domain of file fragment type identification*

Chapter 4 gave an overview of the methodology followed in the empirical investigation presented in this dissertation. Various modelling techniques were discussed, including random hyper-parameter search, k -fold cross-validation and mutual information-based feature ranking.

In Section 4.4 various model evaluation measures were outlined. Each experiment was evaluated based on these measures. A 10-fold cross-

validation t -test was outlined in Section 4.5. This statistical test was used to compare the performance of each model configuration proposed. These techniques were applied in Chapter 5 in the empirical investigation presented.

6. *Compare the classification accuracy of classifiers used in the existing literature to the classification accuracy of the proposed ensemble*

As stated in Chapter 3, a comparison between the results reported by two researchers is often of limited value due to differences in the problem formulation used, the size of the classification problem studied, the file types considered and the file object under observation. For this reason, the performance of the standard k -nearest neighbour algorithm was also tested in Experiment 1 (Chapter 5). The k -nearest neighbour algorithm was selected due to its popularity in the literature, and because it is a fair representation of the current “state of the art” in file object type identification. It is hoped that other researchers will also follow this approach to make meta-comparisons of research reported possible.

As such the primary objective of this dissertation was achieved: It was shown how MLP neural networks and LPDCs may be used individually and in an ensemble to solve the multiple file type file fragment identification problem.

6.2 Research methodology

To achieve the research objectives in this dissertation a knowledge discovery and data mining (KDDM) process model, repeated in Figure 6.1, was used. This process model consists of a set of processing steps needed to complete a KDDM project.

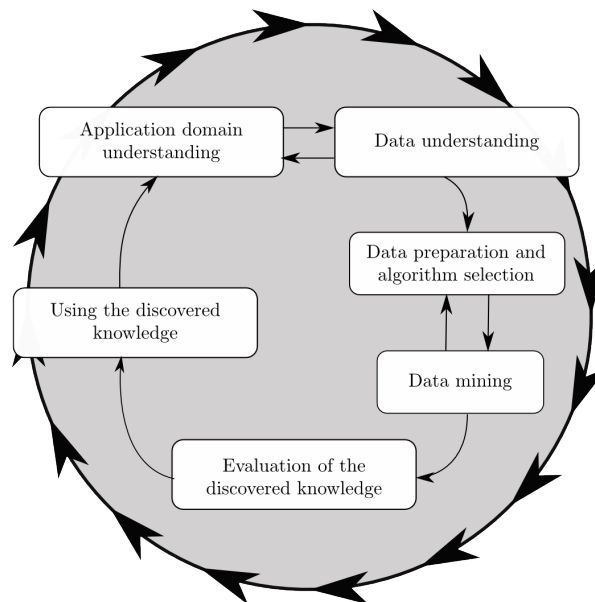


Figure 6.1: KDDM process model.

Chapter 1 contributed to the KDDM process by providing some insight into the data mining problem and objectives. The application domain was also introduced. The required prior knowledge was further developed in Chapter 2 and Chapter 3. Specifically, Chapter 2 provided a general description of supervised learning but also the machine learning techniques applied in this dissertation. In Chapter 3 a general description of supervised learning as relevant to the file object type identification problem was given. The data mining activity was introduced in Chapter 4. The discussion with regard to experimental design addressed various topics such as the data set used, feature selection techniques applied, model selection techniques applied, as well as model evaluation and comparison methods chosen. The results of the knowledge discovery endeavour were given and evaluated in Chapter 5. This dissertation focussed on evaluating the usefulness of MLP neural networks and LPDCs for file fragment type identification. The actual implementation of the techniques in practical scenarios is left as a future research problem.

6.3 Research results

The research hypothesis of this dissertation was as follows: more accurate file fragment type identification can be achieved by using MLP neural network classifiers and LPDCs in an ensemble instead of using these classifiers individually.

When applied individually, both models seemed to perform well in solving the multiple class file fragment type identification problem. Both models seem to be a valid alternative to the k -nearest neighbour algorithm, which is frequently used for file object type identification in the literature. The composite LPDCs seems to be a valid and promising alternative to MLP neural networks because it is simple and efficient to solve.

The results of combining MLP neural network classifiers and LPDCs in an ensemble were not better than those of the single optimized classifiers. Even though the MLP neural network model and LP based discriminant model have different model representations, the file type predicted by each model was very correlated. It would seem that better improvement in classification accuracy could be achieved if one combines more diverse classifiers.

The ensemble of classifiers did outperform the best individual classifier included in the ensemble. The classification accuracy of a single optimized classifier is, however, better. Ideally, one would like to combine highly accurate and diverse classifiers in order to outperform the single optimized classifier. The problem of how to build a set of accurate and diverse classifiers remains a challenge.

This study introduced the idea of ensemble learning to the domain of file fragment type identification, and it is hoped that further research in this regard will follow.

6.4 Research limitations

Limitations to this dissertation, as well as this research domain, can be identified:

The identification of file fragments belonging to container files remains a limiting factor in the successful implementation of file fragment type identification problems. Further research is needed as to how this problem should most likely be addressed. The question that needs to be answered is: Should a file fragment be classified in line with the file type of its parent container file or in line with its base file type? The answer to this question will most likely be determined by application area. If the objective is to identify file content of a specific type, then assigning a file fragment its base file type would seem correct. If the objective is to apply these techniques in file reconstruction algorithms, then assigning a file fragment to its parent file type would seem correct.

Only popular file types found in the literature were included in this study. Further research needs to be done on what type of file fragments are truly of interest to security, network and digital forensic specialists.

There are various techniques that can be used to introduce diversity into an ensemble. Only three different approaches were tested in this dissertation, therefore the results may not be suitable for generalization. However, ensemble learning is a broad and active research area in machine learning. Testing every technique proposed in the literature was not part of the scope of this dissertation. Other approaches might provide interesting results.

6.5 Future research

Various opportunities for further research exist:

1. Other approaches for introducing diversity into the ensemble can be tested. For some techniques, such as bagging and boosting it may be possible to construct larger ensembles, which might be able to improve classification accuracy.
2. The use of alternative predictors than the byte frequency histogram employed here may also be examined. The use of longer n -grams (bi-grams or tri-grams) might be considered.
3. Longer n -grams analyses, as proposed for file type identification by Harris (2007), combined in ensembles with other supervised learning techniques should be examined.
4. Taking the file types of neighbouring file fragments (for example on a disk image) into consideration before assigning a file type to a file fragment could be explored. This approach might be useful in identifying file fragments belonging to container files.
5. As shown in this dissertation, different supervised learning techniques might be well suited for the identification of some file types and not for others. A divide-and-conquer approach, as proposed for file type identifi-

cation by Ahmed *et al.* (2010), combined with ensembles might improve classification accuracy of some file types.

6. In terms of LP-based discriminant analysis (LPDA) other alternative model formulations can be investigated. Also, assigning different cost coefficients in the objective functions of the LPDA models might be a way to introduce more diversity in the ensemble.

6.6 Chapter summary

This final chapter presented a summary of the initial objectives and how the objectives were addressed. In conclusion, the problems experienced and possible future research opportunities were outlined.

RELATED RESEARCH

A

The following paper was presented at the International Conference of Computational and Mathematical Methods in Science and Engineering held in Murcia, Spain on 2-5 July 2012, and published in the refereed volumes of this proceedings as,

Wilgenbus, E.F., Kruger, H.A., du Toit, J.V., Jul. 2012. File fragment classification: An application of a neural network and linear programming based discriminant model. In: Vigo-Aguiar, J., eds. 2012 International Conference of Computational and Mathematical Methods in Science and Engineering (CMMSE), Murcia, Spain. Vol 3. pp. 1237-1248. ISBN: 978-84-615-5392-1.

Research that forms part of this work has also been presented at the following conferences,

Wilgenbus, E.F., Kruger, H.A., du Toit, J.V., Jul. 2012. The application of neural networks and linear programming discriminant models on the file fragment classification problem. Paper presented at the 25th European Conference on Operational Research (Euro 2012), Vilnius, Lithuania.

Wilgenbus, E.F., Kruger, H.A., du Toit, J.V., Sep. 2012. A supervised learning approach for file fragment classification. Paper presented at 41st ORSSA Annual Conference (ORSSA 2012), Johannesburg, South Africa.

Wilgenbus, E.F., Kruger, H.A., du Toit, J.V., Sep. 2012. A supervised learning based framework for file fragment classification. Poster presented at The Southern Africa Telecommunication Networks and Applications Conference (SATNAC 2012), George, South Africa.

File fragment classification: An application of a neural network and linear programming based discriminant model

Erich Wilgenbus¹, Hennie Kruger¹ and Tiny du Toit¹

¹ *School of Computer, Statistical and Mathematical Sciences., North-West University,
Potchefstroom Campus. Private Bag X6001. Potchefstroom, 2520. South Africa.*

emails: `Erich.Wilgenbus@nwu.ac.za`, `Hennie.Kruger@nwu.ac.za`,
`Tiny.DuToit@nwu.ac.za`

Abstract

The file fragment classification problem is important in data recovery, memory diagnostics and network and forensic analysis. Security and forensic specialists need reliable automated tools to identify the true file type of a file or a file fragment. The design of efficient and accurate classifiers have become a mayor challenge in digital forensics for which several approaches have been proposed. This paper investigates the feasibility of a neural network model and a linear programming based discriminant model to solve a binary file type classification problem. Labelled byte frequency histograms, computed from a set of file blocks, are used to construct and test each classifier. The study finds that both neural networks and linear programming discriminant models are able to predict the file type of a given file fragment with reasonable accuracy. A two phase linear programming model seems particularly capable of solving this problem.

Key words: Linear Programming, Discriminant Models, Neural Network, Multi Layer Perceptrons, File type identification, File fragment classification.

1 Introduction

Computer technology plays an increasingly important role in the processing, storage and transmission of data. Digital media has become the preferred data and information storage medium. As the use of computers to store both legal and illegal information grows, so does the need for specialized tools that can monitor, control and even recover this data [1]. Digital forensic and security specialists require trusted and automated tools to identify the

true nature of files and file fragments. Not only might such tools help to identify hidden or misrepresented data, but also aid in the reconstruction of files.

Traditionally files are identified by means of meta-data such as file extensions, file header signatures or operating system specific meta-data tables [2]. File type classification that relies on meta-data is not always reliable because the meta-data can be unavailable or could have been tampered with.

As fragmentation is a natural occurrence in the computer sciences, the need to identify the file type to which a file fragment belongs is a recurrent task in digital security and forensics. Fragment type identification is difficult as the fragment might not contain the meta-data used in traditional file type classification techniques. Data recovery of forensically important files such as pictures, documents and other end-user centric files is plagued by high fragmentation [3].

The file fragment classification problem involves identifying the true type of a computer file fragment [2]. A model has to be developed that is able to determine the file type of a given file fragment based on the file fragment's content. This paper evaluates the performance of a neural network and a linear programming discriminant model in classifying file fragments. A brief review of related work is offered in Section 2. The methods under investigation are discussed in Section 3. Empirical results are given in Section 4, a conclusion is drawn and future work is discussed in the final section.

2 Related work

Several solutions to the file type identification problem have been proposed in literature [2, 4, 5]. In general each proposed method tries to generalize some representative characteristic for a given file type from a sample of files. New files or file fragments can then be compared to the representative characteristics to predict their file type.

One characteristic used often is the byte frequency histogram (BFH), which is the frequency histogram of bytes contained in a file over all the possible byte values from 0 to 255 [2]. A centroid BFH is constructed as the representative characteristic of a file type and a new file is then compared to different centroids by means of some distance measure.

The use of single centroids [2], multiple centroids [5, 6], and exemplar files [7] have been proposed to improve distance based file type identification using a k -nearest neighbour classification algorithm. The byte frequency histogram analysis has also been extended to byte-pair frequency analysis in the same fashion [8].

Other characteristics utilized include various statistical moments [9, 10]; entropy and complexity [9]; rate of change histograms [11]; as well as longest common subsequence analysis [12]. The compression distance between files of a known file type and files of an unknown file type has also been tested [7].

Support vector machines [13, 14], Fisher's linear discriminant analysis [9, 12], as well as

multi-layer perceptron (MLP) neural networks [6, 15] have been used to predict file types with varying success.

The use of feature selection techniques is relevant in training classifiers for file type prediction due to the high dimensionality of the BFH employed as input pattern. Feature selection techniques used include filtering based on average feature value [6]; filtering based on average feature value after taking into account the consistency of this value [8]; as well as feature compression by means of principal component analysis (PCA) and non-linear component analysis (NLPCA) [15].

This research paper builds on a well established foundation of research contributions by other authors. A mutual information measure is used to select the feature set used in constructing a neural network to predict whether a file fragment belongs to a given file type. In addition, a linear programming based discriminant model is also used for file type classification.

3 Preliminaries: Methods used

The file fragment classification problem under investigation is modelled as a general binary classification problem. Supervised learning techniques are applied to derive a generalized hypothesis from an exemplar dataset that can be used to make future predictions [16].

Formally, let $(\vec{x}_t, y_t)$ with $t = 1 \dots k$ be a training set of k training patterns where each feature vector $\vec{x}_t$, of r attributes $x_1, x_2, \dots, x_r$, is associated with a predefined class label $y_t \in \{G_0, G_1\}$. The objective of the binary classification problem is to derive a classifier $f \rightarrow \{G_0, G_1\}$ which minimizes the difference between the given class label y_t and the predicted class label $\hat{y}_t$.

In this section a brief overview of two techniques, neural networks and linear programming discriminant analysis, for deriving a classifier f is given. These techniques are used in Section 4 to predict whether a file fragment belongs to a certain type of file.

3.1 Neural Networks

Neural networks imitate the cognitive function of the brain to approximate intelligent machines [17, 18]. The multilayer perceptron (MLP) neural network consists of multiple neurons in a highly connected pattern. The neurons are separated into three distinct layers. The input layer where the predictors are received, the output layer where the predicted values are found, and a layer of hidden units in-between. All units of the input layer are connected to all units in the hidden layer while all units of the hidden layer is in turn connected to all units of the output layer.

The neural network receives an input vector $\vec{x}_t \in \mathbb{R}^{r+1}$ representing the r input features as well as a bias signal x_0 set to one [19]. Each value $x_0, x_1, \dots, x_r$ in the input vector enters

the network at the corresponding input node. Each input node x_i passes its value to each hidden neuron z_j where $i = 0$ to r and $j = 1$ to J .

Each neuron z_j in the hidden layer combines the inputs received from each preceding interconnected input node as a weighted sum of inputs $u_j = \sum_{i=0}^r w_{ji}x_i$ where w_{ji} is the weight of the connection between input node x_i and hidden neuron z_j . The hidden neuron output z_j is computed by resolving an activation function $a^h(u_j)$ with the computed weighted sum u_j . The activation function a^h used in each hidden neuron can be any non-linear monotonically increasing function that is differentiable in u_j , such as the symmetric sigmoid function. The output of the hidden layer is a vector $\vec{z} \in \mathbb{R}^{J+1}$ with the bias signal z_0 set to one. The vector $\vec{z}$ supplies inputs to the output layer with one neuron in the case of a binary classification problem. That is the output layer produces a scalar $\hat{y} \in \mathbb{R}$. The network output $\hat{y}$ is determined as $a^o(u)$ where $u = \sum_{j=0}^J w_j z_j$ is the weighted sum of all connections between hidden neurons z_j with $j = 0 \dots J$, and a^o another non-linear monotonically increasing activation function.

The number of input nodes and the number of output neurons depend on the problem under investigation, however the optimal number of hidden neurons is not as forthcoming. The number of hidden neurons establish the predictive ability of the network: An MLP, with enough hidden neurons, can approximate any function from $\mathbb{R}^r$ to $\mathbb{R}$ with any desired degree of accuracy [20].

The correct number of hidden neurons depends on the characteristics inherent to a specific problem such as the noise in the dataset, the number of dimensions, the availability of training data, and the interdependence of features [18]. An overly simplified network with too few hidden neurons has little capacity to learn knowledge, while a complex network with too many hidden neurons will have more weights than can be supported by the available training data. Such a model will over-fit the training data while under performing on the test data.

Once the architecture of a neural network is defined, it has to be trained before it can be used for prediction. An MLP stores knowledge in the set of weights $w = \{w_j, w_{ij}\}$ where each weight represents the importance of a specific incoming signal in computing the outgoing signal for each neuron in the network. It is through this network of connections with each connection having a specific weight that a network has its predictive power.

A training algorithm adjusts each weight in the set of weights w in order to make the computed network output $\hat{y}_t$ as close as possible to the desired network output y_t for each training pattern output tuple $((\vec{x}_t, y_t), \hat{y}_t)$ with $t = 1 \dots k$. The backpropagation algorithm (BPROP), quick propagation algorithm and the RPROP algorithm are some algorithms used to train neural networks [21]. A complete description of the backpropagation algorithm can be found in [19].

A neural network is a practical classifier because it is free of statistical assumptions regarding the distribution of input data. In addition, a neural network does not require a

functional form to be specified *a priori*, as a non-linear functional form is derived implicitly during network training.

3.2 Linear Programming Based Linear Discriminant

Discriminant analysis (DA) is a popular method used in deriving a classification rule from a sample of labelled patterns. If $M = \{(\vec{x}_t, y_t) | \vec{x}_t \in R^r, y_t \in \{G_0, G_1\}\}$ is a training set of k training patterns where each feature vector $\vec{x}_t$ of r attributes $x_1, x_2, \dots, x_r$ is associated with a predefined class label $y_t \in \{G_0, G_1\}$. Define a linear discriminant function as $f(\vec{w}, \vec{x}_t) = w_0 + \sum_{i=1}^r w_i x_{ti}$ and a class boundary b such that $f(\vec{w}, \vec{x}_t) \leq b$ for all $(\vec{x}_t, y_t)$ where $y_t \in G_0$ and $f(\vec{w}, \vec{x}_t) \geq b$ for all $(\vec{x}_t, y_t)$ where $y_t \in G_1$. The objective of discriminant analysis is to find $\vec{w}^*$ and b^* that minimize the extent of misclassification. Optimal values of $\vec{w}^*$ and b^* are then used to assign a class label to an unlabelled pattern $\vec{x}'$ by calculating a classification score $f(\vec{w}, \vec{x}')$, and assigning $\vec{x}'$ label G_0 if $f(\vec{w}, \vec{x}'_i) \leq b$ as G_1 if $f(\vec{w}, \vec{x}'_i) \geq b$.

A linear programming discriminant analysis (LPDA) approach allows for efficient computation of an optimal discriminant function over large sets of sample data without relying on any assumptions about the data. Empirical studies have shown that LPDA models outperforms traditional discriminant methods, such as Fisher's linear discriminant analysis and Smith's quadratic discriminant analysis in situations where the normality assumptions of data do not hold [22]

One model that seems to perform better than the other LPDA models when a substantial number of outliers are present is the PHASER model which is implemented as a two step classification approach [22, 23]. Initially a discriminant rule is defined that allows for a classification gap, or fuzzy set, consisting of patterns for which the rule has not yet been established. In the first phase those patterns that lie close to the group boundary, and which is therefore most likely to be misclassified, are excluded in the construction of the discriminant function. During the second phase, the discriminant function is refined by focusing on the separation of patterns within the fuzzy area. The first phase can be modelled as follows [24]:

$$\begin{aligned} & \text{minimize } \sum_{t=1}^k d_t, \\ & \text{subject to } w_0 + \sum_{i=1}^r w_i x_{ti} - d_t \leq 0 \quad \forall \quad \vec{x}_t \in G_0, \\ & \quad \quad \quad w_0 + \sum_{i=1}^r w_i x_{ti} + d_t \geq 1 \quad \forall \quad \vec{x}_t \in G_1, \\ & \quad \quad \quad \text{with } d_t \geq 0 \quad \forall \quad \vec{x}_t \in G_0 \cup G_1, \\ & \quad \quad \quad w_i \in \mathbb{R} \text{ for } t = 1 \dots r. \end{aligned}$$

The area between $[0, 1]$ is defined as the fuzzy area and d_t denotes the distance of each pattern $t = 1 \dots k$ from the group boundary which is zero and one for patterns in group G_0 and G_1 respectively.

Solving the model produce optimal weights $\vec{w}^*$ that is used to calculate a discriminant score for each of the training samples. The training sample is then re-labelled such that the label S_0 is assigned to all patterns classified as belonging to G_0 by having a discriminant score of less than zero; the label S_1 is assigned to all patterns classified as belonging to G_1 by having a discriminant score of more than one; and the label S_f is assigned to all patterns having a discriminant score lying between zero and one.

The second phase allows for more careful analysis of the subtly distinguishable patterns in S_f . The discriminant function is refined to minimize the misclassification in the fuzzy region without sacrificing any correct classifications made in phase one. Denote the correctly classified patterns of each target group after phase one as $G_0 \cap S_0$ and $G_1 \cap S_1$ respectively. The second phase can then be formulated as follows [24].

$$\begin{aligned}
& \text{minimize} \quad \sum_{\vec{x}_t \in G_0 \cap (S_1 \cup S_f)} d_t + \sum_{\vec{x}_t \in G_1 \cap (S_0 \cup S_f)} d_t, \\
& \text{subject to} \quad w_0 + \sum_{i=1}^r w_i x_{ti} \leq 0 \quad \forall \vec{x}_t \in G_0 \cap S_0, \\
& \quad \quad \quad w_0 + \sum_{i=1}^r w_i x_{ti} \geq 1 \quad \forall \vec{x}_t \in G_1 \cap S_1, \\
& \quad \quad \quad w_0 + \sum_{i=1}^r w_i x_{ti} - d_t \leq b \quad \forall \vec{x}_t \in G_0 \cap (S_1 \cup S_f), \\
& \quad \quad \quad w_0 + \sum_{i=1}^r w_i x_{ti} + d_t \geq b \quad \forall \vec{x}_t \in G_1 \cap (S_0 \cup S_f), \\
& \text{with } 0 \leq b \leq 1, \\
& \quad \quad \quad d_t \geq 0 \quad \forall \vec{x}_t \in G_0 \cup G_1, \\
& \text{and } w_t \in \mathbb{R} \text{ for } t = 1 \dots r.
\end{aligned}$$

The optimal weights $\vec{w}^*$ and optimal group boundary b^* found in the second phase are used for classifying new unlabelled patterns.

4 Experimental Methodology and Results

This section presents an overview of the research methodology followed and the results obtained when applying an MLP and LPDA model to the file fragment classification problem. The discussion includes a brief overview of the dataset used, the model evaluation

criteria used, the feature selection technique employed, and the method used to estimate parameters. Models were implemented using Base SAS, SAS Enterprise Miner software and SAS/OR software [25]. The standard MLP architecture was used with the backpropagation learning algorithm.

4.1 Dataset

Files used in this research were taken from the govdocs1 dataset [26]. Only file types *gz*, *gif*, *ps*, *xls*, *ppt*, *jpg*, *doc*, *txt* and *html* were considered. For each file type under investigation a binary classification model was built to determine whether a given file fragment belongs to the this type of file or not. The specific file type for which a model was built to identify is henceforth called the model's target file type.

Files were broken down into 4096-byte blocks, for each of which a byte frequency histogram (BFH) was computed by counting the number of occurrences of each possible byte value (from 0 to 255) in the block. Header and trailer blocks were excluded and the byte frequency histograms were standardized by dividing each byte frequency value by 4096. The base dataset consisted of more than 730 000 BFHs. Descriptive statistics for the base dataset are given in Table 1.

From the base dataset a balanced dataset was created for each file type under investigation. Such a balanced dataset consists of 2000 randomly selected BFHs of the specific file type, as well as 2000 randomly selected BFHs from all remaining file types. The dataset for each file type was further partitioned into a training and a testing dataset at a ratio of 2:1.

Table 1: Source file descriptive statistics

File type	Nr of files	Total file size (MB)	Mean file size (MB)	Std dev	Max size (MB)	Min size (MB)	Nr of BFHs (Blocks)
gz	73	34,580	473.7	784.1	4,892	4	2,109
gif	97	8,436	87.0	74.8	320	4	8,645
ps	111	198,588	1,789.1	4,324.4	26,944	8	21,599
xls	294	176,148	599.1	2,416.2	26,048	4	21,837
ppt	410	701,812	1,711.7	3,684.7	33,340	8	44,038
jpg	455	87,348	192.0	518.5	6,660	4	49,647
doc	550	263,380	478.9	1,421.4	16,868	4	65,845
txt	617	428,044	693.8	4,657.7	92,464	4	107,011
html	1,067	86,396	81.0	421.4	8,808	4	175,453
pdf	1,758	936,992	533.0	972.8	11,520	4	234,248
Total		2,921,724					730,432

4.2 Model Evaluation

The accuracy of each model in classifying the file type in the test set of file fragments is reported using several measures [14, 6, 8]. First a precision measure is used. Precision (P) measures a classifier's ability to identify patterns as part of the target class which is in fact part of the target class. P is defined as $P = tp/(tp + fp)$ where tp is the percentage of patterns in the target class classified as belonging to the target class, while fp is the percentage of patterns classified incorrectly belonging to the target class.

Recall is the second measure used. Recall (R) measures the probability that a target class pattern is classified correctly. R is defined as $R = tp/(tp + fn)$ where fn is the percentage of patterns in the target class classified as not belonging to the target class. A combined measure F is constructed as the harmonic mean between P and R with $F = 2RP/(P + R)$. F represents the model's ability to predict target class objects correctly and non target class object correctly.

A general accuracy measure (A) is provided as the percentage of patterns classified correctly as either part of the target class or not part of the target class, such that $A = tp+tn$ where tn is the percentage of patterns correctly identified as not belonging to the target class.

4.3 Mutual information

Feature selection is a critical step in reducing the input dimensionality of a classification problem by reducing the complexity of the classifier, improving the generalization ability of a classifier while also improving the efficiency of classifier training [27]. Feature selection is of particular importance when training neural networks [16].

One technique that has been shown to be effective in selecting features for use with neural networks is based on Mutual information [28]. Mutual information between the class-denoting variable Y and the set of features S is defined as the difference between the uncertainty in Y and the uncertainty remaining in Y conditioned on the set of features S . That is $MI(Y|S) = H(Y) - H(Y|S)$ where entropy $H(\cdot)$ is used to measure the uncertainty in a variable.

The features are ranked in such a way that the k top ranked features are those k features that maximizes the mutual information about the category denoting variable Y . A complete description of the method is given in [28]. Feature ranking for each file type as target class is given in Table 2. The top k features (byte values) were used as inputs to the neural network where the optimal k is estimated empirically as outlined in the next section. For example, the subset of 4 features which provides the most information in distinguishing the *xls* file type from other file types consists of the byte values 0, 48, 13 and 80.

Table 2: Feature ranking for each target file type.

Rank	gz	gif	ps	xls	ppt	jpg	doc	txt	html	pdf
1	0	192	0	0	48	0	55	46	97	106
2	101	98	70	48	13	13	10	4	4	13
3	68	68	9	13	68	68	65	60	66	68
4	10	10	13	80	121	118	13	10	40	121
:	:	:	:	:	:	:	:	:	:	:
255	199	130	211	235	143	233	215	142	219	233
256	233	196	94	233	218	211	211	211	211	211

4.4 Parameter estimation

In the implementation of neural networks, the optimal number of features c and the appropriate number of hidden layers h had to be estimated. [29] argues that a random search for the optimal neural network configuration is able to find models that are as good as those found in traditional grid search approaches within a small fraction of the computation time. A random search for near optimal parameters was performed: Values $h \in \{1 \dots 60\}$ and $c \in \{10 \dots 70\}$ were drawn randomly for which the performance of the neural network was estimated by means of 10-fold cross validation. For each file type, models were evaluated for 100 randomly drawn parameters (h, c) pairs. The results reported in the next section are for the neural networks model found to provide the most accurate results. Where two or more models specifications had similar results the simpler model was chosen.

4.5 Results

The suitability of neural network and linear programming discriminant analysis for file fragment type predictions were investigated. For each file type, a neural network was constructed to predict whether a given file fragments belongs to this file type or not. Similarly, for each file type, a discriminant function were derived using a two phase linear programming model described earlier.

The performance of each file-specific neural network is given in Table 3. Each neural network was constructed using the c top ranked features as inputs with h hidden layers. The procedure to determine the values of c and h was described in Section 4.4. For example, the results for the model classifying a fragment as either belonging to or not belonging to the doc file type has been constructed using the 59 top-ranked features and 28 hidden layers. This model is able to identify 74.9 percent of files correctly.

The results from the linear programming discriminant models is given in Table 4. For each file-specific model the derived discriminant function is used to predict whether a file

Table 3: Results of neural network with h hidden nodes and c input features

target	h	c	tp(%)	tn(%)	fp(%)	fn(%)	P(%)	R(%)	F(%)	A(%)
doc	28	59	39	35	15	11	73	79	75.8	74.9
gif	13	46	47	46	4	2	92	95	93.3	93.2
gz	26	44	47	29	21	3	69	94	79.6	76.0
html	13	52	48	48	2	2	97	95	96.0	96.0
jpg	9	56	45	36	14	5	76	89	82.3	80.8
pdf	9	58	46	29	21	4	69	92	79.0	75.5
ppt	32	59	48	27	23	2	68	96	79.4	75.2
ps	16	54	48	49	1	2	97	95	96.2	96.2
txt	13	57	49	47	3	1	94	98	96.1	96.0
xls	8	55	47	46	4	2	93	95	93.8	93.7
Average	17	54	46	39	11	4	83	93	87.2	85.8

fragment belongs to this type or not. The LPDA was initially applied using the same number of features that was used in the corresponding neural networks. The linear classifiers were not able to achieve similar predictive accuracy as the non-linear classifiers using the reduced feature sets. While the neural network does not scale well to larger dimensional spaces, the LPDA is efficient using all 256 features.

Table 4: Results of linear programming model with 256 input features

Target	tp(%)	tn(%)	fp(%)	fn(%)	P(%)	R(%)	F(%)	A(%)
doc	38	38	12	12	76	76	75.6	75.7
gif	44	45	5	6	89	89	89.0	89.0
gz	40	40	10	10	80	79	79.4	79.5
html	48	44	6	2	89	96	92.6	92.3
jpg	44	43	7	6	86	87	86.3	86.2
pdf	39	38	12	11	77	78	77.5	77.3
ppt	32	39	11	18	75	65	69.4	71.4
ps	48	44	6	2	89	96	92.5	92.2
txt	48	48	2	2	97	96	96.5	96.5
xls	48	43	7	2	88	95	91.4	91.0
Average	43	42	8	7	84	86	85.0	85.1

Both the neural network and linear programming models provide reasonable accurate classification results. The average classification accuracy between classifiers are similar, with the neural network model only slightly outperforming the linear programming based model. The neural networks are able to accurately identify the target file type with an accuracy of between 74 percent and 96 percent. The LPDAs are able to accurately identify

the target file type with an accuracy of between 71 percent and 96 percent.

While the general results of both classifiers are nearly identical, the per file type performances varies. At this stage the LPDA model seems to be the more pragmatic solution to the classification problem because the model can be solved more easily for larger datasets, while construction and training of the neural networks are tedious and time consuming. Further research will investigate the feasibility of using both techniques together for improved predictive capability.

5 Conclusion

An important task in digital forensics and security is to identify the true type of a file or file fragment without relying on file typing meta-data. This paper identifies two techniques which can be used as binary file type classifiers: A multilayer perceptron (MLP) neural network and Linear Programming Discriminant Analysis (LPDA). Neural networks seemed promising due to their inept ability to learn difficult non-linear patterns from complex datasets while LPDA seemed promising because it can be solved efficiently given a large number of variables.

Both models seemed to perform well in solving the binary file fragment type classification problem, however, the use of these techniques needs to be investigated further. In terms of pre-processing, various feature selection techniques need to be evaluated before applying neural networks. The use of alternative predictors more and above the byte frequency histogram employed here should also be examined. In terms of linear programming discriminant analysis other alternative model formulations can be investigated.

References

- [1] A. Pal and N. Memon, "The evolution of file carving," *Signal Processing Magazine, IEEE*, vol. 26, no. 2, pp. 59–71, 2009.
- [2] M. McDaniel and M. H. Heydari, "Content based file type detection algorithms," in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, Jan. 2003, p. 10.
- [3] S. L. Garfinkel, "Carving contiguous and fragmented files with fast object validation," *Digital investigation*, vol. 4, pp. 2–12, 2007.
- [4] L. Sportiello and S. Zanero, "File Block Classification by Support Vector Machine," in *Sixth International Conference on Availability, Reliability and Security (ARES)*, Aug. 2011, pp. 307–312.
- [5] W. Li, K. Wang, S. J. Stolfo, and B. Herzog, "Fileprints: identifying file types by n-gram analysis," in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, June 2005, pp. 64–71.

- [6] I. Ahmed, K. Lhee, H. Shin, and M. Hong, "Content-based File-type Identification Using Cosine Similarity and a Divide-and-Conquer Approach," *IETE Technical Review*, vol. 27, no. 6, pp. 465–475, 2010.
- [7] S. Axelsson, "The Normalised Compression Distance as a file fragment classifier," *Digital Investigation*, vol. 7, pp. S24–S31, 2010.
- [8] D. Cao, J. Luo, M. Yin, and H. Yang, "Feature selection based file type identification algorithm," in *IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, vol. 3, 2010, pp. 58–62.
- [9] C. J. Veenman, "Statistical Disk Cluster Classification for File Carving," in *Third International Symposium on Information Assurance and Security*, Aug. 2007, pp. 393–398.
- [10] G. Conti, S. Bratus, A. Shubina, B. Sangster, R. Ragsdale, M. Supan, A. Lichtenberg, and R. Perez-Aleman, "Automated mapping of large binary objects using primitive fragment type classification," *Digital Investigation*, vol. 7, pp. S3–S12, 2010.
- [11] M. Karresand and N. Shahmehri, "File Type Identification of Data Fragments by Their Binary Structure," in *Information Assurance Workshop, IEEE*, June 2006, pp. 140–147.
- [12] W. C. Calhoun and D. Coles, "Predicting the types of file fragments," *Digital Investigation*, vol. 5, pp. S14–S20, 2008.
- [13] Q. Li, A. Ong, P. Suganthan, and V. Thing, "A Novel Support Vector Machine Approach to High Entropy Data Fragment Classification," in *Proceedings of the South African Information Security Multi-Conference (SAISMC)*, 2010.
- [14] B. Li, Q. Wang, and J. Luo, "Forensic Analysis of Document Fragment Based on SVM," in *Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia (IIH-MSP)*, 2006, pp. 236–239.
- [15] M. C. Amirani, M. Toorani, and A. A. Beheshti Shirazi, "A new approach to content-based file type detection," in *IEEE Symposium on Computers and Communications (ISCC)*, 2008, pp. 1103–1108.
- [16] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, pp. 249–268, 2007.
- [17] G. P. Zhang, "Neural Networks For Data Mining," *Data Mining and Knowledge Discovery Handbook*, pp. 419–444, 2010.
- [18] I. Ramlall, "Artificial Intelligence: Neural Networks Simplified," *International Research Journal of Finance and Economics*, no. 39, pp. 105–120, 2010.
- [19] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [20] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [21] W. Schiffmann, M. Joost, and R. Werner, "Optimization of the backpropagation algorithm for training multilayer perceptrons," University of Koblenz: Institute of Physics, Rheinau 1. 56075 Koblenz, Tech. Rep., 1994.

- [22] K. Fung Lam and J. W. Moy, “An experimental comparison of some recently developed linear programming approaches to the discriminant problem,” *Computers and Operations research*, vol. 24, no. 7, pp. 593–599, 1997.
- [23] N. Freed and F. Glover, “Applications and Implementation: A linear programming approach to the discriminant problem,” *Decision Sciences*, vol. 12, no. 1, pp. 68–74, 1981.
- [24] A. Stam and C. T. Ragsdale, “On the classification gap in mathematical programming-based approaches to the discriminant problem,” *Naval Research Logistics (NRL)*, vol. 39, no. 4, pp. 545–559, 1992.
- [25] SAS, *Base SAS ®9.2 Procedures Guide.*, Cary, NC: SAS Institute Inc., 2009.
- [26] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, “Bringing science to digital forensics with standardized forensic corpora,” *Digital investigation*, vol. 6, pp. S2–S11, 2009.
- [27] M. Dash and H. Liu, “Feature selection for classification,” *Intelligent data analysis*, vol. 1, pp. 131–156, 1997.
- [28] N. Kwak and C. H. Choi, “Input feature selection for classification problems,” *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 143–159, 2002.
- [29] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

THE SIMPLEX ALGORITHM: AN EXAMPLE

B

This appendix shows how to solve a simple linear programming (LP) problem graphically and by using the simplex algorithm outlined in Section 2.3.2. Similar examples can be found in Griffon (2012) and Bazaraa *et al.* (1990).

B.1 The example LP problem

For the purpose of this example, consider the following production problem,

$$\begin{aligned} \text{maximize} \quad & z(x_1, x_2) = 3x_1 + 2x_2 \\ \text{subject to} \quad & x_1 \leq 3 \\ & x_1 + x_2 \leq 6 \\ & 2x_1 + x_2 \leq 8 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{B.1}$$

The objective of this LP problem is to find non-negative values for the decision variables x_1 and x_2 that maximize the value of the objective function $z(x_1, x_2)$, within the feasible region defined by the three constraints.

B.2 Isoprofit graphical solution method

The graphical solution method works only when there are two decision variables. The steps are as follows (Render *et al.*, 2011):

1. Plot the feasible region defined by the constraints.
2. Plot a contour line of the objective function for any chosen value of z .
3. For a maximization problem, identify the contour line (isoprofit line) with the largest objective function value z that intersects the feasible region. This contour line will be parallel to the one drawn in point 2 and will pass through one of the feasible region's corner points.
4. For a minimization problems, identify the contour line (isocost line) with the smallest objective function value z that intersects the feasible region. This contour line will be parallel to the one drawn in point 2 and will pass through one of the feasible region's corner points.
5. The corner point identified will be the optimal basic feasible solution.

The feasible region defined by the constraints in Equation B.1 is shown in Figure B.1 as area $ABCDE$. An infinite number of contour lines could have been drawn by selecting any real number values for z . The contour lines shown are for objective function values $z = 9, 12, 13$ and 14 . Each

of these contour lines intercepts one of the corner points defined by the feasible region. Point C lies on the highest possible contour line that intersects the feasible region, therefore point C is the optimal solution with $z = 14$, $x_1 = 2$ and $x_2 = 4$.

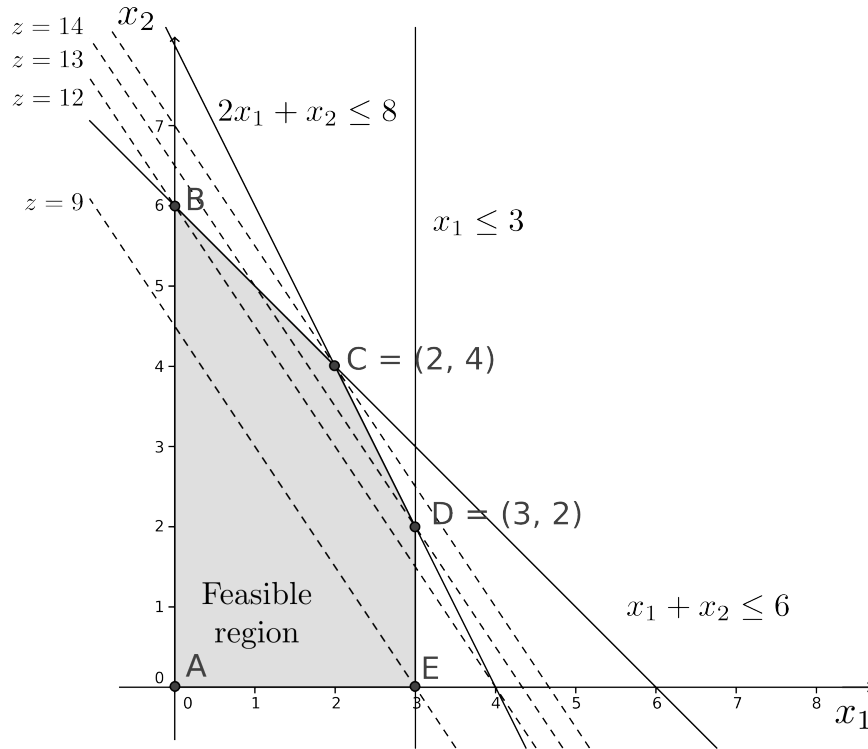


Figure B.1: The graphical representation of the LP problem defined in B.1.

B.3 Simplex Algorithm: Algebraic solution

The LP problem defined in Equation B.1 can be solved using the simplex algorithm described in Algorithm 2.2. The LP problem is converted to standard form by adding slack variables s_1 , s_2 and s_3 ,

$$\begin{aligned}
 \max \quad & z(x_1, x_2) = 3x_1 + 2x_2 \\
 \text{subject to} \quad & x_1 + s_1 = 3 \\
 & x_1 + x_2 + s_2 = 6 \\
 & 2x_1 + x_2 + s_3 = 8 \\
 & x_1, x_2, s_1, s_2, s_3 \geq 0
 \end{aligned} \tag{B.2}$$

The LP problem in Equation B.2 defines the matrices,

$$\mathbf{c} = \begin{bmatrix} 3 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 3 \\ 6 \\ 8 \end{bmatrix} \tag{B.3}$$

Initial basic feasible solution

Matrix $\mathbf{A}$ can be decomposed into matrix $\mathbf{B}$ and matrix $\mathbf{N}$ by selecting the three unit vectors corresponding to s_1, s_2 and s_3 as the initial basic solution,

$$\mathbf{B}_{s_1, s_2, s_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{N}_{x_1, x_2} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 2 & 1 \end{bmatrix} \quad (\text{B.4})$$

Therefore,

$$\mathbf{x}_B = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \mathbf{c}_B = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{x}_N = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mathbf{c}_N = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad (\text{B.5})$$

Matrix $\mathbf{B}$ is the unit matrix, therefore $\mathbf{B} = \mathbf{B}^{-1} = \mathbf{I}$. Set the non-basic variables $\mathbf{x}_N = \mathbf{0}$ then the basic feasible solution is,

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 3 \\ 6 \\ 8 \end{bmatrix} \quad (\text{B.6})$$

with an objective function value,

$$z(\mathbf{x}) = \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{b} = 0 \quad (\text{B.7})$$

This basic feasible solution corresponds to point A in Figure B.1.

Determine if this is the optimal solution by calculating the reduced cost for every x_i in $\mathbf{x}_N$. With

$$\mathbf{B}^{-1}\mathbf{N} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 2 & 1 \end{bmatrix} \quad (\text{B.8})$$

the reduced cost vector is,

$$\mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{N} - \mathbf{c}_N^T = [0, 0] - [3, -2] = [-3, -2] \quad (\text{B.9})$$

Step 1

Observe the negative reduce cost entries, and derive that,

$$\frac{\partial z}{\partial x_1} = -(\mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{A}_{.1} - c_1) = 3 \quad (\text{B.10})$$

$$\frac{\partial z}{\partial x_2} = -(\mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{A}_{.2} - c_2) = 2 \quad (\text{B.11})$$

The objective value z can therefore be increased with 3 units for each unit increase in x_1 and 2 units for each unit increase in x_2 . Choose x_1 to enter the basic solution. This requires one variable to exit the basis, that is, one current basic variable's value is reduced to zero. Investigate how much increase in x_1 each of the non-basic variables allows by performing the minimum ratio test using each positive element in $\mathbf{B}^{-1} \mathbf{A}_{.1}$ and the corresponding element of $\mathbf{B}^{-1}\mathbf{b}$. Calculate,

$$\mathbf{B}^{-1} \mathbf{A}_{.1} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} \quad (\text{B.12})$$

and,

$$\Delta \mathbf{x}_1 = \min \left\{ \frac{3}{1}, \frac{6}{1}, \frac{8}{2} \right\} \quad (\text{B.13})$$

In this case $\frac{3}{1}$ is the minimum ratio. Therefore x_1 enters the basis and variable s_1 leaves the basis. The new basic and non-basic variables are,

$$\mathbf{x}_B = \begin{bmatrix} x_1 \\ s_2 \\ s_3 \end{bmatrix} \quad \mathbf{c}_B = \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{x}_N = \begin{bmatrix} s_1 \\ x_2 \end{bmatrix} \quad \mathbf{c}_N = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad (\text{B.14})$$

By simply swapping out the columns of x_1 and s_1 in the basis matrix $\mathbf{B}$ and the non-basis matrix $\mathbf{N}$,

$$\mathbf{B}_{x_1, s_2, s_3} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \quad \mathbf{N}_{s_1, x_2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \quad (\text{B.15})$$

Derive the inverse of matrix $\mathbf{B}$ through elementary row operation such that $[\mathbf{B}|\mathbf{I}] \sim \dots \sim [\mathbf{I}|\mathbf{B}^{-1}]$. For the new basic feasible solution solve,

$$\mathbf{x}_B = \mathbf{B}^{-1} \mathbf{b} = \begin{bmatrix} 3 \\ 3 \\ 2 \end{bmatrix} \quad (\text{B.16})$$

and for the new objective function value solve,

$$z(\mathbf{x}) = \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{b} = 9 \quad (\text{B.17})$$

This new basic feasible solution corresponds to the point E in Figure B.1. Check whether this is the optimal solution by investigating the reduced cost values of the non-basic variables. With

$$\mathbf{B}^{-1} \mathbf{N} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ -2 & 1 \end{bmatrix} \quad (\text{B.18})$$

the reduced cost vector is calculated as,

$$\mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{N} - \mathbf{c}_N^T = [3, 0] - [0, -2] = [0, -2] \quad (\text{B.19})$$

Step 2

From the negative reduced cost entry it follows that,

$$\frac{\partial z}{\partial x_2} = -(\mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{A}_{.2} - c_2) = 2 \quad (\text{B.20})$$

Let x_2 enter the basic solution. Investigate how much increase in x_2 each of the non-basic variables allows. Perform the minimum ratio test using each positive element $\mathbf{B}^{-1} \mathbf{A}_{.2}$ and each corresponding element of $\mathbf{B}^{-1} \mathbf{b}$. That is, calculate

$$\mathbf{B}^{-1} \mathbf{A}_{.2} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad (\text{B.21})$$

and

$$\Delta \mathbf{x}_1 = \min \left\{ \frac{3}{1}, \frac{2}{1} \right\} \quad (\text{B.22})$$

In this case $\frac{2}{1}$ is the minimum ratio. Therefore x_2 enters the basis and variable s_3 leaves the basis. The new basic and non-basic variables are,

$$\mathbf{x}_B = \begin{bmatrix} x_1 \\ s_2 \\ x_2 \end{bmatrix} \quad \mathbf{c}_B = \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix} \quad \mathbf{x}_N = \begin{bmatrix} s_1 \\ s_3 \end{bmatrix} \quad \mathbf{c}_N = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{B.23})$$

and,

$$\mathbf{B}_{x_1, s_2, x_2} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 2 & 0 & 1 \end{bmatrix} \quad \mathbf{N}_{s_1, s_3} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (\text{B.24})$$

The new basic feasible solution that corresponds to point D in Figure B.1,

$$\mathbf{x}_B = \mathbf{B}^{-1} \mathbf{b} = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} \quad (\text{B.25})$$

with an objective function value of,

$$z(\mathbf{x}) = \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{b} = 13 \quad (\text{B.26})$$

Investigate the reduced cost value for each non-negative variable to determine whether the current feasible solution is optimal,

$$\mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{N} - \mathbf{c}_N^T = [-1, 2] - [0, 0] = [-1, 2] \quad (\text{B.27})$$

Step 3

From the negative reduced cost entry derive that the z can be increased further by increasing the value of s_1 . Let s_1 enter the basic solution. The increase allowed in s_1 by each of the basic variables is calculated in the minimum ratio test using each positive element in $\mathbf{B}^{-1} \mathbf{A}_{.3}$ and each corresponding element of $\mathbf{B}^{-1} \mathbf{b}$. Calculate,

$$\mathbf{B}^{-1} \mathbf{A}_{.3} = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix} \quad (\text{B.28})$$

and,

$$\Delta \mathbf{s}_1 = \min \left\{ \frac{3}{1}, \frac{1}{1} \right\} \quad (\text{B.29})$$

It follows that s_1 enters the basis and variable s_2 leaves the basis. The new basic and non-basic variables are,

$$\mathbf{x}_B = \begin{bmatrix} x_1 \\ s_1 \\ x_2 \end{bmatrix} \quad \mathbf{c}_B = \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix} \quad \mathbf{x}_N = \begin{bmatrix} s_2 \\ s_3 \end{bmatrix} \quad \mathbf{c}_N = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{B.30})$$

and,

$$\mathbf{B}_{x_1, s_1, x_2} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{bmatrix} \mathbf{N}_{s_2, s_3} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (\text{B.31})$$

The new basic feasible solution that corresponds to point C in Figure B.1 is,

$$\mathbf{x}_B = \mathbf{B}^{-1} \mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix} \quad (\text{B.32})$$

with an objective function value of,

$$z(\mathbf{x}) = \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{b} = 14 \quad (\text{B.33})$$

This is an optimal solution because the reduced cost of both non-basic variables is positive,

$$\mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{N} - \mathbf{c}_N^T = [1, 1] - [0, 0] = [1, 1] \quad (\text{B.34})$$

In these three steps, the simplex algorithm iterated from one extreme point to the next extreme point until the optimal basic feasible solution was found as shown in Figure B.2.

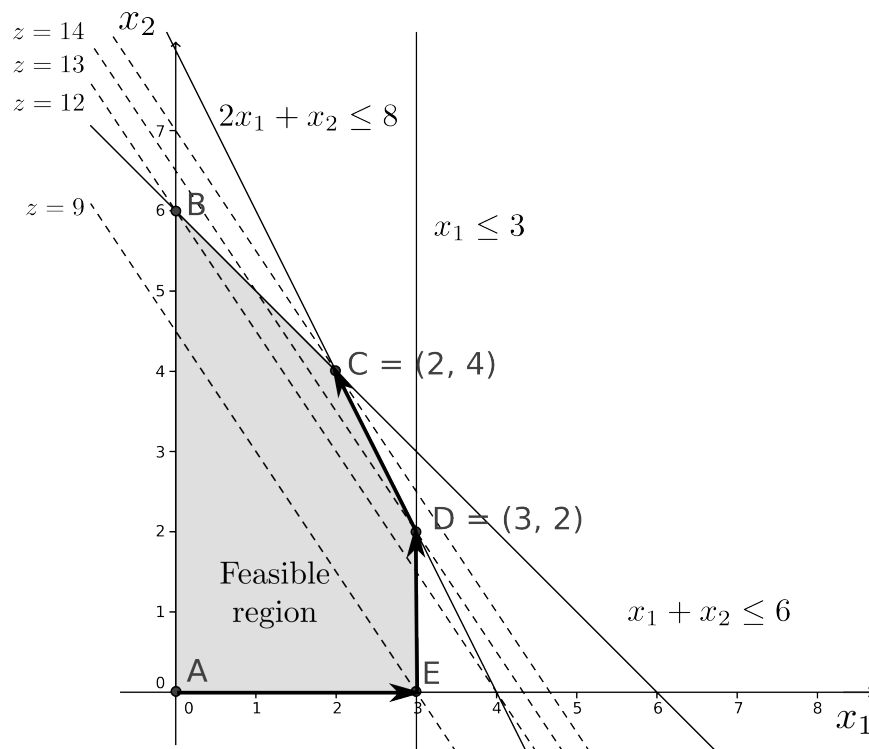


Figure B.2: The graphical representation of the simplex algorithm as it cycles through extreme points.

“GREEDY” MUTUAL INFORMATION BASED FEATURE RANKING: AN EXAMPLE

C

This appendix contains an example of mutual information feature ranking as outlined in Algorithm 4.1. This example was adapted from the one provided by Kwak and Choi (2002).

In this example it is assumed that a fictional data set of 1000 instances was generated. Each instance consists of four features $X_1 = a$, $X_2 = a - b$, $X_3 = b^2$ and $X_4 = c$ where the value of a , b and c are drawn uniformly between $[-0.5, 0.5]$. The class denoting variable Y is defined as,

$$Y = f(a, b) = \begin{cases} 0 & \text{if } a + 0.2b < 0 \\ 1 & \text{if } a + 0.2b \geq 0 \end{cases} \quad (\text{C.1})$$

For example, consider the random generate number tuple $(a, b, c) = (0.45, -0.29, -0.05)$. From these values a hypothetical data set instance can be constructed as,

$$(X_1 = a, X_2 = a - b, X_3 = b^2, X_4 = c, Y = f(a, b)) = (0.45, 0.74, 0.0841, -0.05, 1).$$

In order to implement the “greedy” first mutual information-based feature ranking algorithm, the continuous feature values need to be mapped into discrete feature values. Each feature is discretized into 10 bins with equal frequency. That is, the top 10 percent of values in the feature is assigned to value 1, the second 10 percent of values in the feature is assigned value 2, and so forth, until the bottom 10 percent of values is assigned value 10.

The probabilities $p(Y = y)$, $p(X_i = x)$ and the joint probability $p(Y = y, X_i = x)$ are estimated by relative frequency count using the discretized data set¹. The estimated probabilities are omitted for brevity. However, they are used to calculate the mutual information of each feature X_i with the class denoting variable Y , as

$$I(X_i, Y) = \sum_{x \in X_i} \sum_{y \in Y} p(x, y) \log_2 \left(\frac{p(x, y)}{p(x)p(y)} \right). \quad (\text{C.2})$$

for $i = 1, \dots, 4$. The results are given in Table C.1. For example, the mutual information between predictor variable X_1 and the class denoting variable Y is estimated as 0.829.

Table C.1: The value of $I(X_i, Y)$

	$I(X_i, Y)$
X_1	0.829
X_2	0.258
X_3	0.010
X_4	0.003

¹Where $y = 0, 1$ is the realization the class variable Y and $x = 1, \dots, 10$ is the realization of feature X_i for $i = 1, \dots, 4$.

The probabilities $p(X_i = x)$, $p(X_j = \dot{x})$ and the joint probability $p(X_i = x, X_j = \dot{x})$ are estimated by relative frequency count using the discretized data set². The estimated probabilities are omitted. However from these probabilities the mutual information between feature X_i and feature X_j is calculated as,

$$I(X_i, X_j) = \sum_{x \in X_i} \sum_{\dot{x} \in Y_j} p(x, \dot{x}) \log_2 \left(\frac{p(x, \dot{x})}{p(x)p(\dot{x})} \right) \quad (\text{C.3})$$

for all $i = 1, \dots, 4$ and $j = 1, \dots, 4$ where $i \neq j$. The results are given in Table C.2. For example, the mutual information between predictor variable X_1 and predictor variable X_2 is 0.593.

Table C.2: The value of $I(X_i, X_j)$

$I(X_i, X_j)$	X_1	X_2	X_3	X_4
X_1		0.593	0.054	0.046
X_2	0.593		0.155	0.058
X_3	0.054	0.155		0.064
X_4	0.046	0.058	0.064	

Features are selected iteratively for inclusion in the feature ranking R . The feature X_i that provides the highest mutual information with regard to Y is selected first. From the results in Table C.1, the feature $X_1 = a$ with $I(X_1, Y) = 0.829$ is ranked first.

During each remaining step, the feature X_i that provides the highest mutual information with regard to Y adjusted for by information already contained in the ranking R is selected for inclusion in the ranking R . The adjusted mutual information for feature X_i is calculated as,

$$I(Y, \{R, X_i\}) \approx I(X_i, Y) - \beta \sum_{X_j \in R} I(X_i, X_j), \quad (\text{C.4})$$

Using a redundancy parameter $\beta = 0.4$, the adjusted mutual information for features X_2, X_3 and X_4 is calculated in Table C.3 using the results from Table C.1 and Table C.2. Feature X_2 is selected for inclusion in ranking R because it has the highest value $I(Y, \{X_1, X_2\}) = 0.021$.

Table C.3: Feature ranking: Step 2

	$I(X_i, Y)$	$I(X_i, X_1)$	$I(Y, \{X_1, X_i\})$
X_2	0.258	0.593	0.021 ³
X_3	0.010	0.054	-0.011
X_4	0.003	0.046	-0.015

The next feature is selected by computing $I(Y, \{X_1, X_2, X_i\})$ for each feature not yet included in ranking R . This calculation is given in Table C.4.

²Where $x = 1, \dots, 10$ is the realization of feature X_i and $\dot{x} = 1, \dots, 10$ is the realization of feature X_j for $i = 1, \dots, 4$ and $j = 1, \dots, 4$.

³ $I(Y, \{X_1, X_2\}) = 0.258 - 0.4 \times 0.593 = 0.021$

Table C.4: Feature ranking: Step 3

	$I(X_i, Y)$	$I(X_i, X_1)$	$I(X_i, X_2)$	$I(Y, \{R, X_i\})$
X_3	0.010	0.054	0.155	-0.073 ⁴
X_4	0.003	0.046	0.058	-0.039

In this example, feature X_4 with $I(Y, \{X_1, X_2, X_4\}) = -0.073$ is ranked third as it has the highest adjusted mutual information value. The last feature X_3 is ranked fourth by default.

The feature ranking with $\beta = 0.4$ is $R = \{X_1, X_2, X_4, X_3\}$. Feature $X_2 = a - b$ is preferred to $X_3 = b^2$ because $Y = f(a, b)$ is a linear combination of $X_1 = a$ and $X_2 = a - b$. Though $X_3 = b^2$ contains more information with regard to $Y = f(a, b)$ than $X_4 = c$, it is ranked below X_4 because all the information in X_3 is already contained in X_2 and X_1 .

If $\beta = 0$ was used, then features would have been ranked as $R = \{X_1, X_2, X_3, X_4\}$ because the algorithm would only consider mutual information with regard to the output variable. Features $X_1 = a$, $X_2 = a - b$ and $X_3 = b^2$ are all more related to the target variable $Y = f(a, b)$ than $X_4 = c$.

⁴ $I(Y, \{X_1, X_2, X_3\}) = 0.010 + 0.4 \times 0.054 + 0.4 \times 0.155 = -0.073$

EXAMPLE BYTE FREQUENCY HISTOGRAMS

D

This appendix contains examples of file fragment byte frequency histograms for all file types included in this study. The vertical axis of each graph shows the relative frequency of a byte value in a file fragment, whereas, the horizontal axis shows all possible byte values.

The byte frequency histogram, together with the mean byte value, standard deviation in byte values and a byte stream entropy value forms the set of predictor variables considered in this dissertation (see Section 4.6.2).

Note: The scale on the vertical axis is not the same across all graphs. The histograms were normalized by dividing each byte value by 4096, the total number of bytes in a file fragment. The file number in the header of each image corresponds to the file name in the Garfinkel *et al.* (2009) file corpus and the position of the 4096-file segment in the original file.

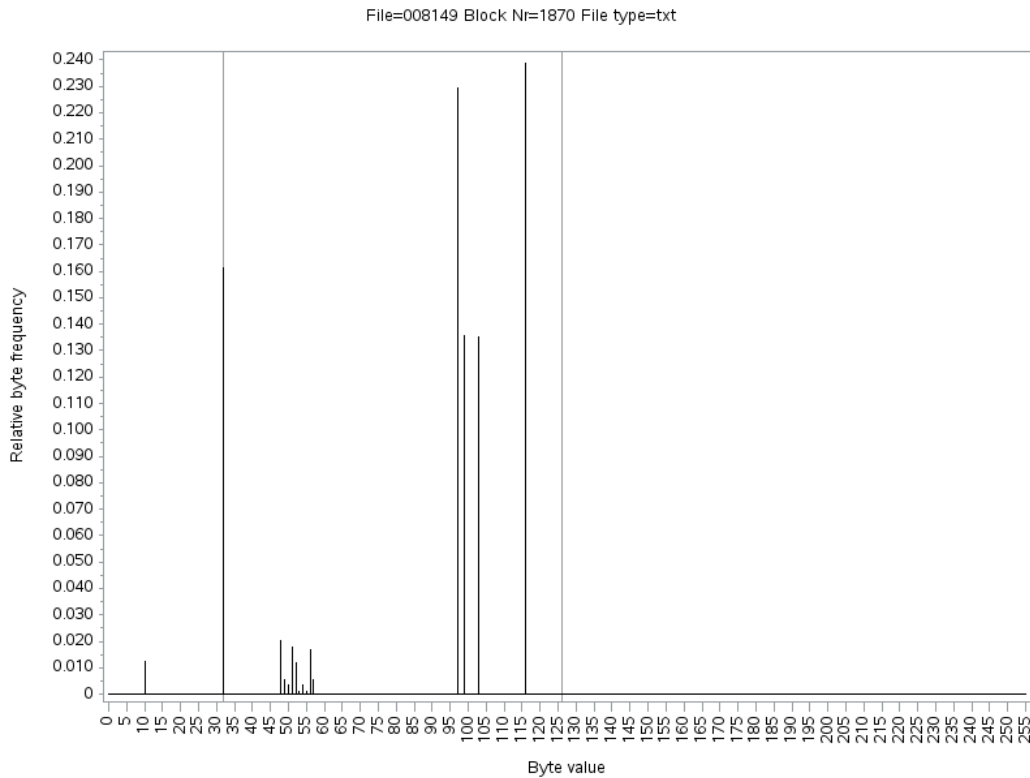


Figure D.1: The BFH of a `txt` file fragment. The file fragment has a mean byte value of 87.12, a standard deviation in byte values of 30.79, and a byte stream entropy value of 1.96 bits

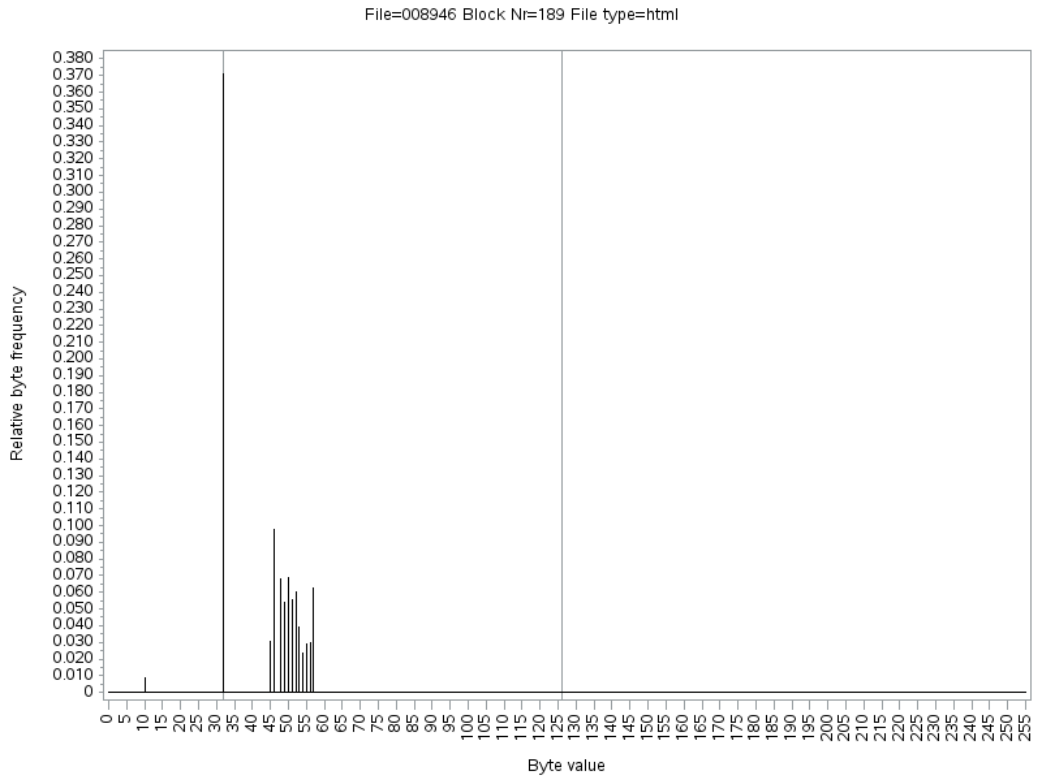


Figure D.2: The BFH of a **html** file fragment. The file fragment has a mean byte value of 43.38, a standard deviation in byte values of 9.97, and a byte stream entropy value of 2.19 bits.

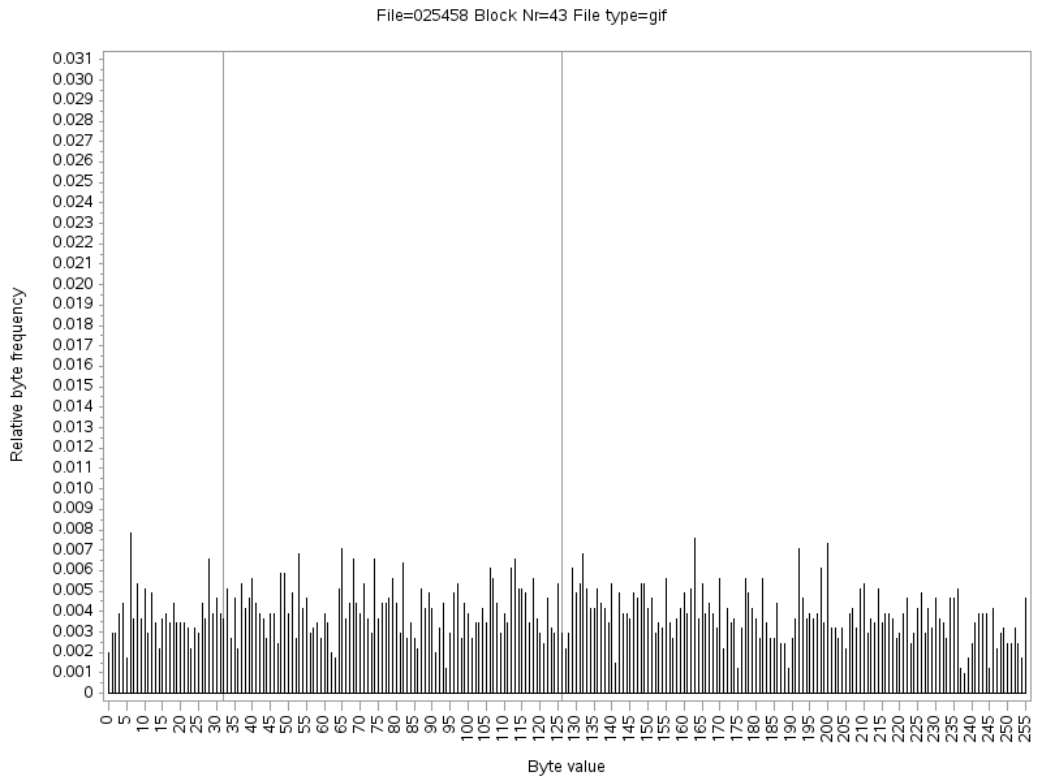


Figure D.3: The BFH of a **gif** file fragment. The file fragment has a mean byte value of 124.30, a standard deviation in byte values of 71.53, and a byte stream entropy value of 5.48 bits.

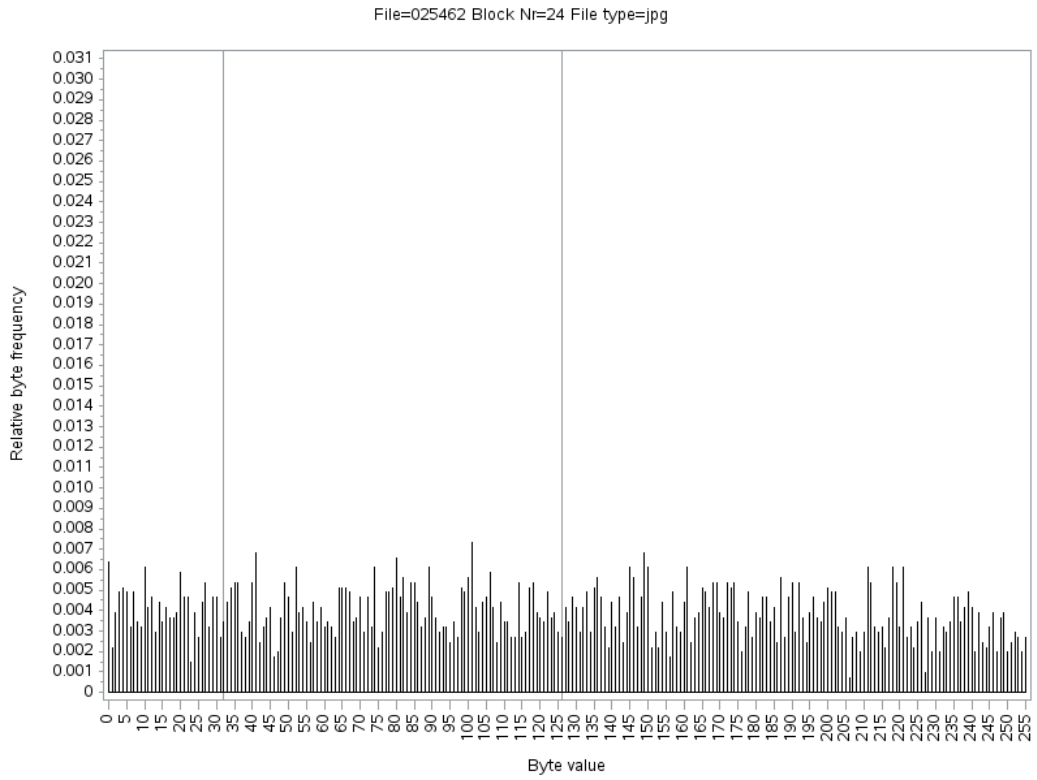


Figure D.4: The BFH of a jpeg file fragment. The file fragment has a mean byte value of 123.02, a standard deviation in byte values of 72.27, and a byte stream entropy value of 5.46 bits.

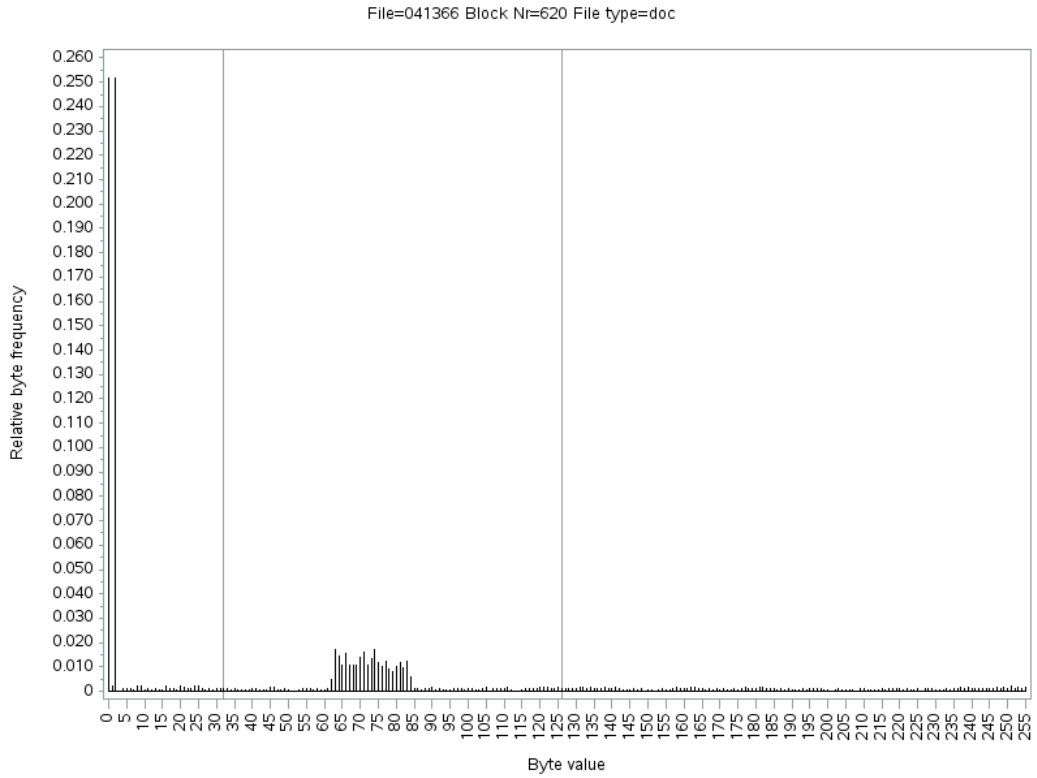


Figure D.5: The BFH of a doc file fragment. The file fragment has a mean byte value of 50.37, a standard deviation in byte values of 65.39, and a byte stream entropy value of 3.09 bits.

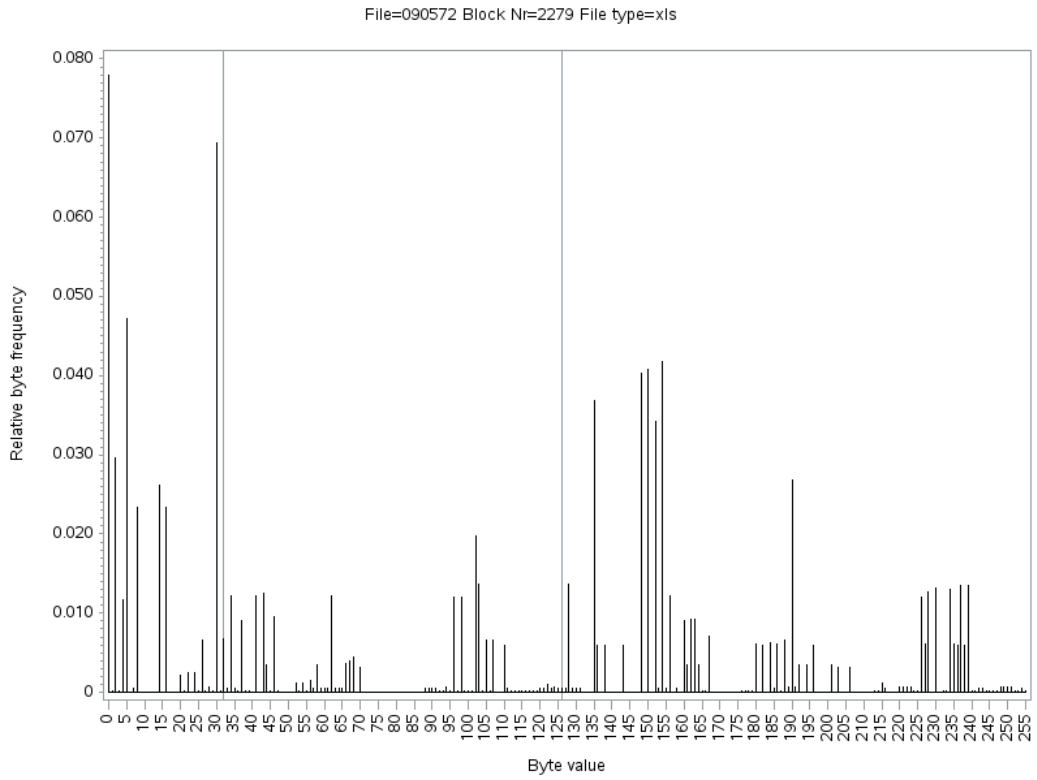


Figure D.6: The BFH of a **xls** file fragment. The file fragment has a mean byte value of 102.44, a standard deviation in byte values of 79.38, a byte stream entropy value of 3.84 bits.

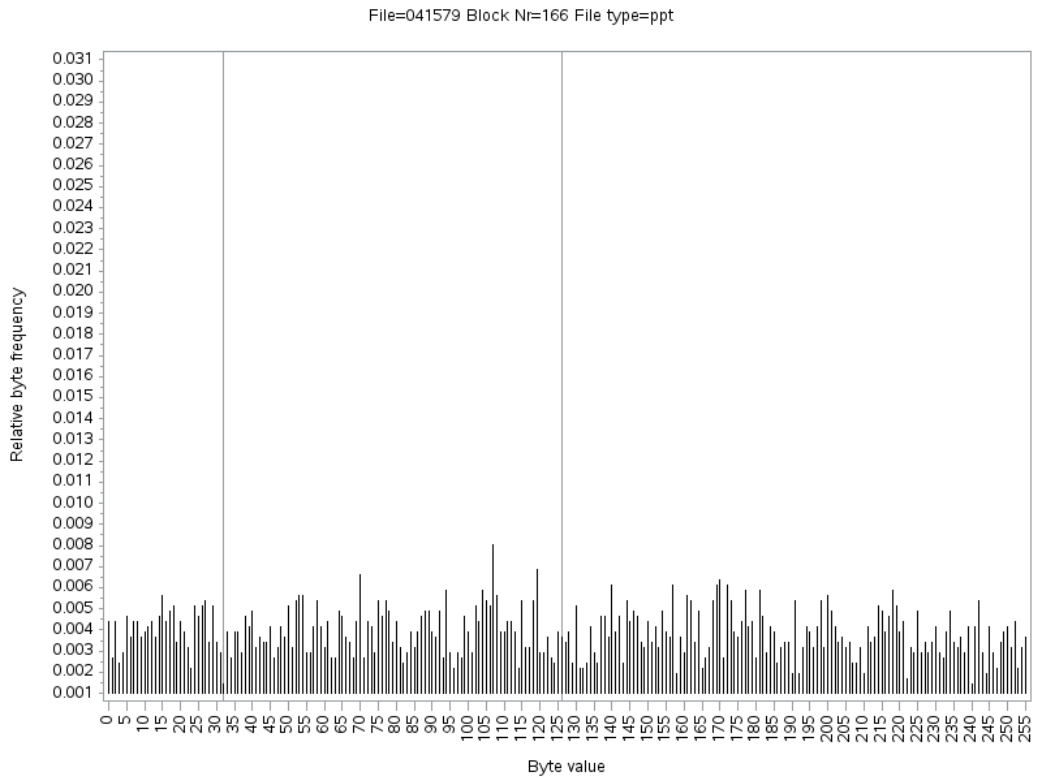


Figure D.7: The BFH of a **ppt** file fragment. The file fragment has a mean byte value of 125.151, a standard deviation in byte values of 72.77, a byte stream entropy value of 5.48 bits.

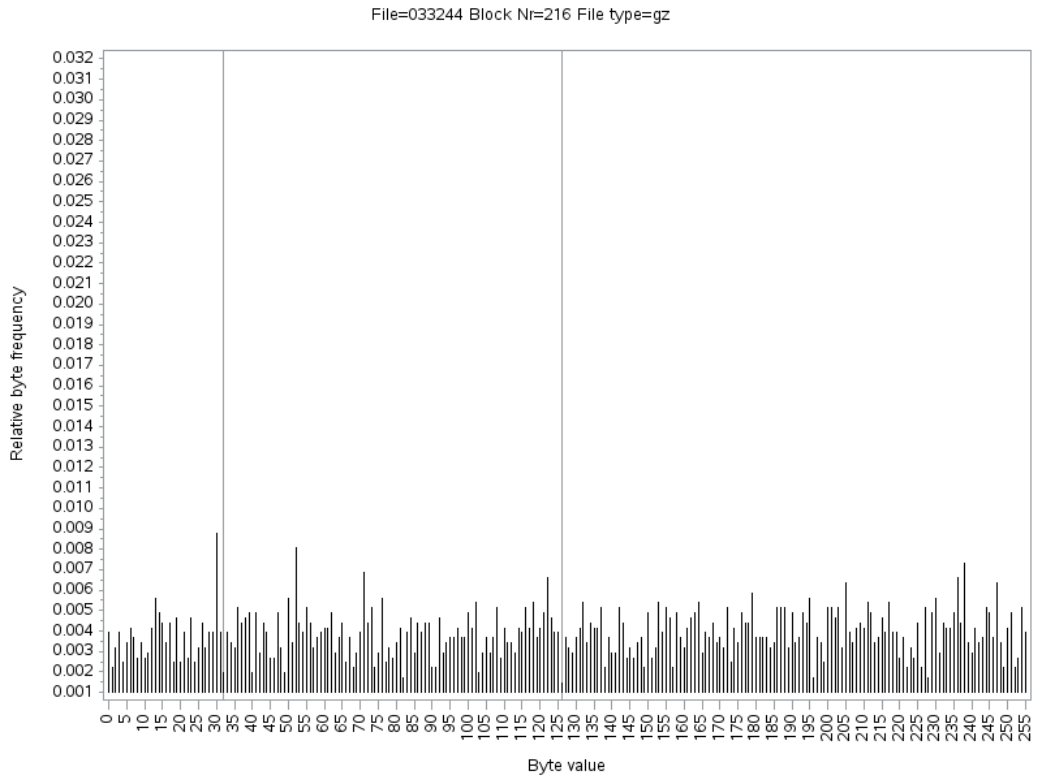


Figure D.8: The BFH of a **gz** file fragment. The file fragment has a mean byte value of 129.84, a standard deviation in byte values of 73.93, a byte stream entropy value of 5.48 bits.

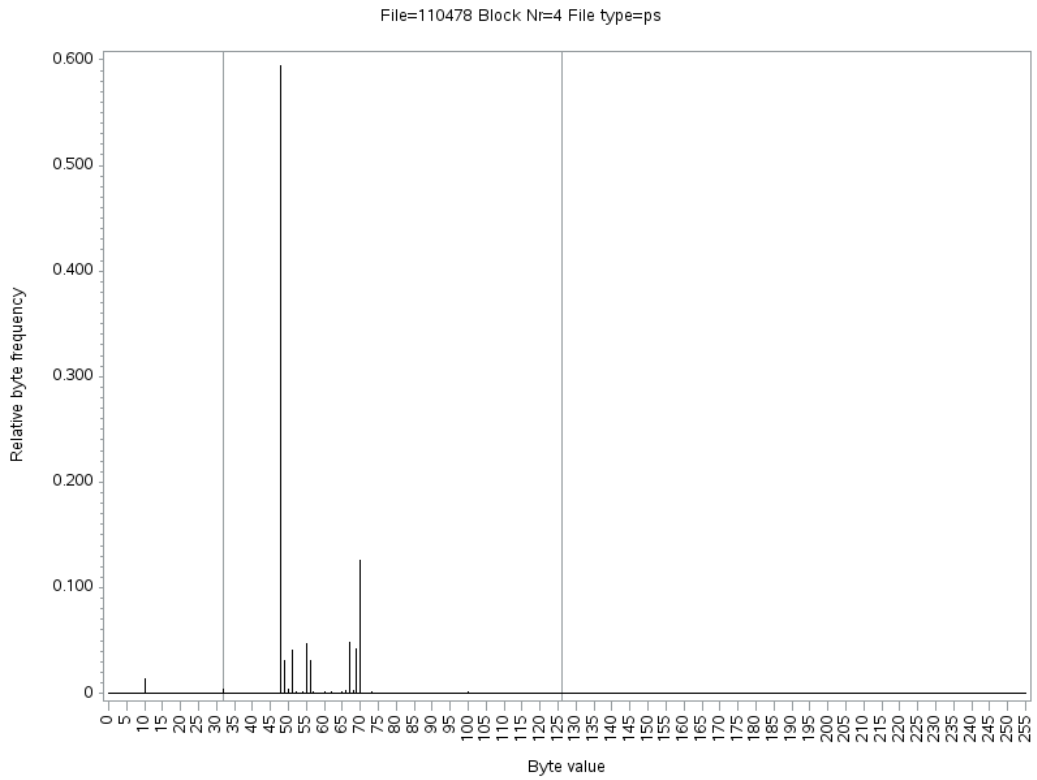


Figure D.9: The BFH of a **ps** file fragment. The file fragment has a mean byte value of 53.06, a standard deviation in byte values of 10.32, a byte stream entropy value of 1.56 bits.

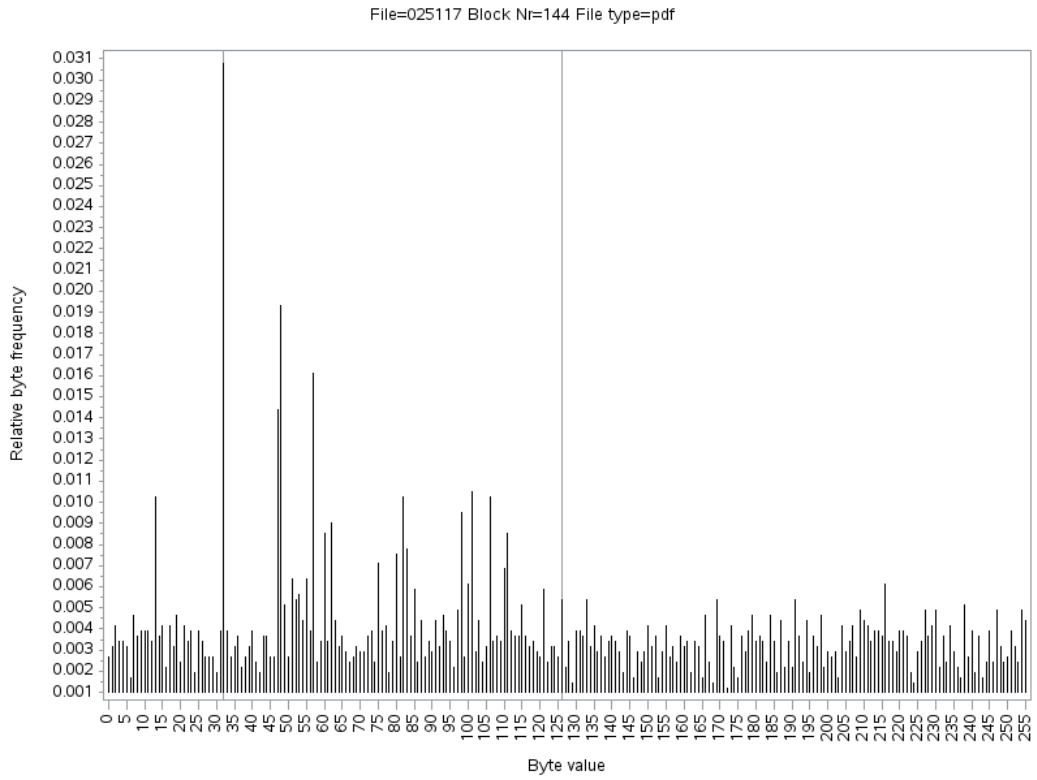


Figure D.10: The BFH of a pdf file fragment. Byte values have a mean of 117.78, a standard deviation in byte values of 72.95, a byte stream entropy value of 5.39 bits.

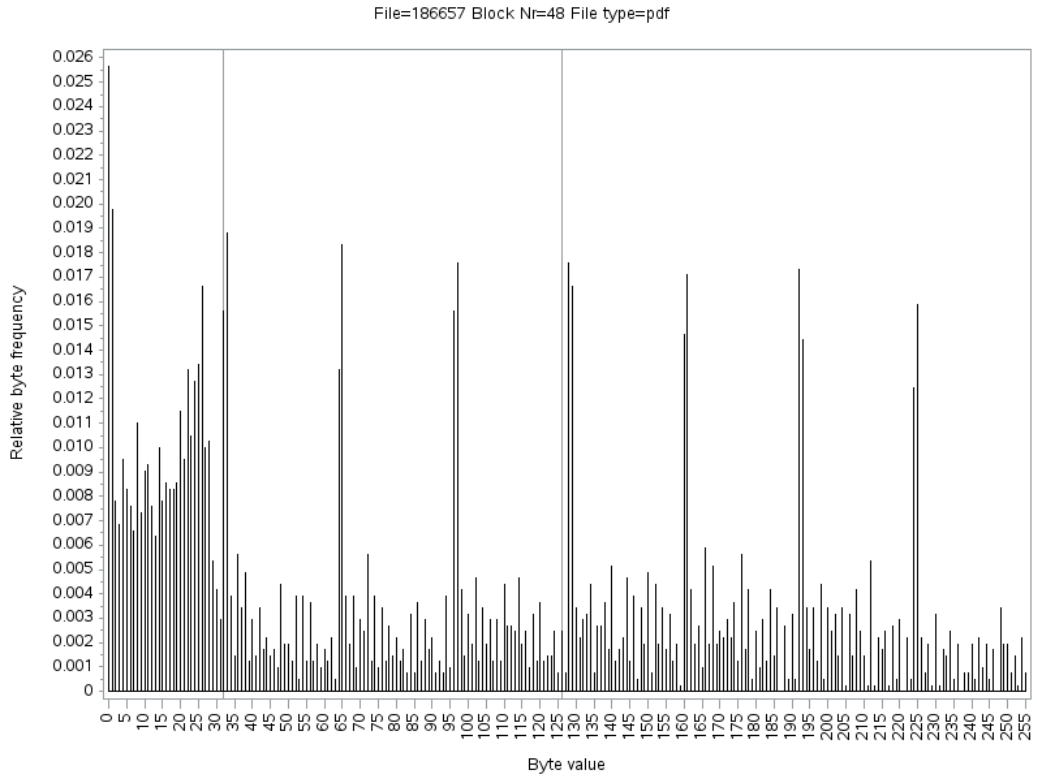


Figure D.11: The BFH of a pdf file fragment. The file fragment has a mean byte value of 95.95 , a standard deviation in byte values of 75.54, a byte stream entropy value of 4.99 bits.

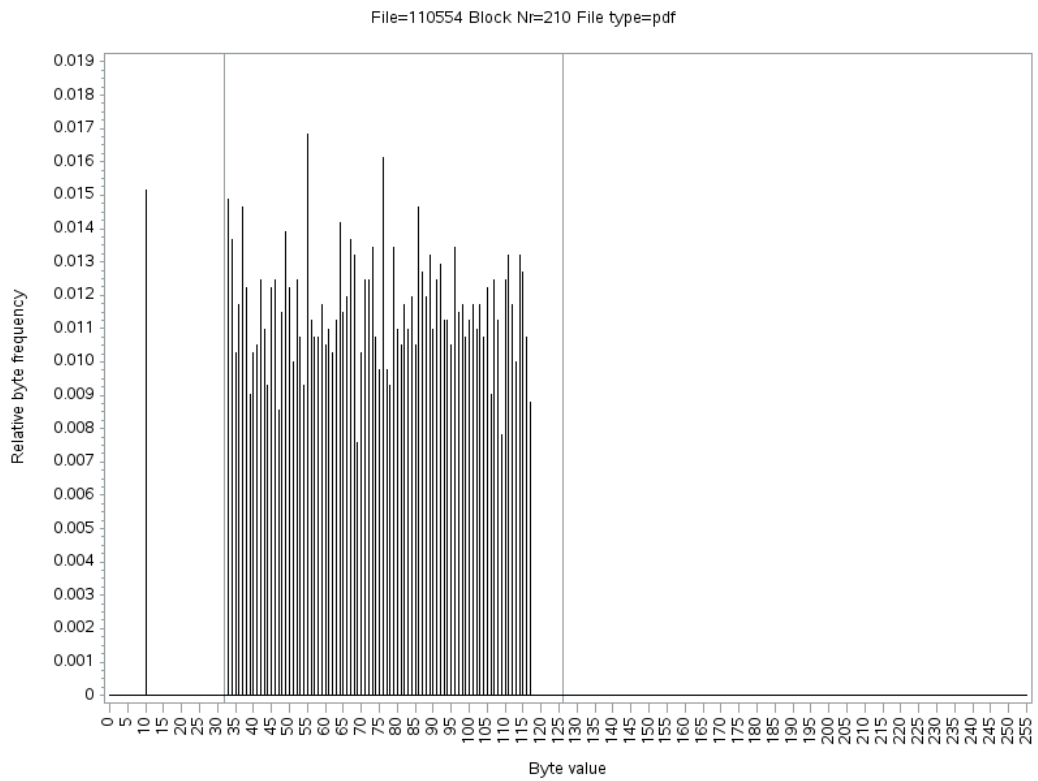


Figure D.12: The BFH of a pdf file fragment. The file fragment has a mean byte value of 73.76, a standard deviation in byte values of 25.55, a byte stream entropy value of 4.44 bits.

BIBLIOGRAPHY



- Aha, D. W., Kibler, D., Albert, M. K., 1991. Instance-based learning algorithms. *Machine learning* 6 (1), 37–66.
- Ahmed, I., Lhee, K., Shin, H., Hong, M., 2010. Content-based file-type identification using cosine similarity and a divide-and-conquer approach. *IETE Technical Review* 27 (6), 465–475.
- Amirani, M. C., Toorani, M., Beheshti Shirazi, A. A., 2008. A new approach to content-based file type detection. In: *IEEE Symposium on Computers and Communications*. Marrakech, Morocco, pp. 1103–1108.
- Anusuya, M. A., Katti, S. K., 2009. Speech recognition by machine: A review. *International Journal of Computer Science and Information Security* 6 (3), 182–205.
- Axelsson, S., 2010. The normalised compression distance as a file fragment classifier. In: *International Conference on Availability, Reliability, and Security*. pp. 641–646.
- Baharudin, B., Lee, L., Khan, K., 2010. A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology* 1 (1), 4–20.
- Bartlett, P. L., 1998. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory* 44 (2), 525–536.
- Battiti, R., 1994. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks* 5 (4), 537–550.
- Bazaraa, M. S., Jarvis, J. J., Sherali, H. D., 1990. *Linear programming and network flows*, 2nd Edition. John Wiley and Sons.
- Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, 281–305.
- Bhattacharyya, S., Jha, S., Tharakunnel, K., Westland, C., 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems* 50 (3), 602–613.
- Bhele, S. G., Mankar, V. H., 2012. A review paper on face recognition techniques. *International Journal of Advanced Research in Computer Engineering and Technology* 1 (8), 339–346.
- Bishop, C. M., 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.
- Bouckaert, R. R., 2003. Choosing between two learning algorithms based on calibrated tests. In: *Proceedings of the Twentieth International Conference on Machine Learning*. Washington, DC, United States of America, pp. 51–58.

- Box, G. E. P., Draper, N. R., 1987. Empirical model-building and response surfaces. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley.
- Breiman, L., 1996. Bagging predictors. *Machine Learning* 24 (2), 123–140.
- Breiman, L., 2001. Random forests. *Machine learning* 45 (1), 5–32.
- Bronshtein, I. N., Semendyayev, K. A., Musiol, G., 2007. Handbook of Mathematics. Springer London, Limited.
- Calhoun, W. C., Coles, D., 2008. Predicting the types of file fragments. *Digital Investigation: The International Journal of Digital Forensics and Incident Response* 5, S14–S20.
- Cao, D., Luo, J., Yin, M., Yang, H., 2010. Feature selection based file type identification algorithm. In: *IEEE International Conference on Intelligent Computing and Intelligent Systems*. Xiamen, China, pp. 58–62.
- Caruana, G., Li, M., 2012. A survey of emerging approaches to spam filtering. *ACM Computing Surveys* 44 (2), 6:1–6:42.
- Chawla, N. V., Hall, L. O., Bowyer, K. W., Kegelmeyer, W. P., 2004. Learning ensembles from bites: A scalable and accurate approach. *The Journal of Machine Learning Research* 5, 421–451.
- Conti, G., Bratus, S., Shubina, A., Sangster, B., Ragsdale, R., Supan, M., Lichtenberg, A., Perez-Aleman, R., 2010. Automated mapping of large binary objects using primitive fragment type classification. *Digital Investigation: The International Journal of Digital Forensics and Incident Response* 7, S3–S12.
- Dash, M., Liu, H., 1997. Feature selection for classification. *Intelligent data analysis* 1, 131–156.
- Dietterich, T., 2000a. Ensemble methods in machine learning. *Multiple classifier systems*, 1–15.
- Dietterich, T., Bakiri, G., 1995. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2, 263–286.
- Dietterich, T. G., 2000b. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning* 40 (2), 139–157.
- Domingos, P., 1999. The role of occam’s razor in knowledge discovery. *Data mining and knowledge discovery* 3 (4), 409–425.
- Duin, R., Tax, D., 2000. Experiments with classifier combining rules. *Multiple Classifier Systems. Lecture Notes in Computer Science* 1857, 16–29.
- Erbacher, R. F., Mulholland, J., 2007. Identification and localization of data types within large-scale file systems. In: *Second International Workshop on Systematic Approaches to Digital Forensic Engineering*. Seattle, Washington, United States of America, pp. 55–70.
- Fahlman, S. E., Lebiere, C., 1990. The cascade-correlation learning architecture. In: *Advances in Neural Information Processing Systems* 3. Denver, Colorado, United States of America, pp. 524–532.

- Falangis, K., 2007. The use of msd model in credit scoring. *Operational Research* 7 (3), 481–503.
- Fawcett, T., 2006. An introduction to roc analysis. *Pattern recognition letters* 27 (8), 861–874.
- Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., 1996. From data mining to knowledge discovery in databases. *AI magazine* 17 (3), 37.
- Fetherston, A., Gollins, T., 2012. Towards the development of a test corpus of digital object for the evaluation of file format identification tools and signatures. *International Journal of Digital Curation* 7 (1), 16–26.
- Fisher, R. A., 1936. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics* 7 (2), 179–188.
- Fletcher, T., 2009. Support vector machines explained. Tutorial, United College of London, London, United Kingdom.
URL www.tristanfletcher.co.uk/SVM%20Explained.pdf
- Freed, N., Glover, F., 1981. Simple but powerful goal programming models for discriminant problems. *European Journal of Operational Research* 7 (1), 44–60.
- Freed, N., Glover, F., 1986. Evaluating alternative linear programming models to solve the two-group discriminant problem. *Decision Sciences* 17 (2), 151–162.
- Freund, Y., Schapire, R. E., 1996. Experiments with a new boosting algorithm. In: *The 13th Annual International Conference of Machine Learning*. Bari, Italy, pp. 148–156.
- Furnkranz, J., 2001. Round robin rule learning. In: *Proceedings of the 18th International Conference on Machine Learning*. pp. 146–153.
- Gale, D., 2007. Linear programming and the simplex method. *Notices of the AMS* 54 (3), 364–369.
- Garfinkel, S., Farrell, P., Roussev, V., Dinolt, G., 2009. Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation: The International Journal of Digital Forensics and Incident Response* 6, S2–S11.
- Garfinkel, S., Nelson, A., White, D., Roussev, V., 2010. Using purpose-built functions and block hashes to enable small block and sub-file forensics. *Digital Investigation: The International Journal of Digital Forensics and Incident Response* 7, S13–S23.
- Garfinkel, S. L., 2007. Carving contiguous and fragmented files with fast object validation. *Digital Investigation: The International Journal of Digital Forensics and Incident Response* 4, 2–12.
- Glover, F., Keene, S., Duea, B., 1988. A new class of models for the discriminant problem. *Decision Sciences* 19 (2), 269–280.
- Griffon, C., 2012. Linear Programming: Penn State Math 484 Lecture Notes. Version 1.8.2.1. Pennsylvania, United States.
URL www.personal.psu.edu/cxg286/Math484_V1.pdf

- Gupta, A., Lam, S. M., 1998. Weight decay backpropagation for noisy data. *Neural Networks* 11 (6), 1127–1138.
- Gutta, S., Huang, J. R. J., Jonathon, P., Wechsler, H., 2000. Mixture of experts for classification of gender, ethnic origin, and pose of human faces. *IEEE Transactions on Neural Networks* 11 (4), 948–960.
- Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. *The Journal of Machine Learning Research* 3, 1157–1182.
- Hall, G., Davis, W., 2006. Sliding window measurement for file type identification. Tech. rep., Computer Forensics and Intrusion Analysis Group, ManTech Security and Mission Assurance. URL <http://www.mantechcfia.com/SlidingWindowMeasurementforFileTypeIdentification.pdf>
- Hamming, R. W., 1950. Error detecting and error correcting codes. *Bell System Technical Journal* 29 (2), 147–160.
- Harris, R., 2007. Using artificial neural networks for forensic file type identification. Master’s thesis, Purdue University, West Lafayette, United States.
- Ho, T. K., 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20 (8), 832–844.
- Ho, T. K., 2002. Multiple classifier combination: Lessons and next steps. *Series in Machine Perception and Artificial Intelligence* 47, 171–198.
- Holmstrom, L., Koistinen, P., Laaksonen, J., Oja, E., 1997. Neural and statistical classifiers-taxonomy and two case studies. *IEEE Transactions on Neural Networks* 8 (1), 5–17.
- Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2 (5), 359–366.
- Hutter, F., 2009. Automated configuration of algorithms for solving hard computational problems. Ph.D. thesis, The University Of British Columbia, Vancouver, Canada.
- Jacobs, R. A., Jordan, M. I., Nowlan, S. J., Hinton, G. E., 1991. Adaptive mixtures of local experts. *Neural computation* 3 (1), 79–87.
- Johnson, L. W., Riess, R. D., Arnold, J. T., 2002. *Introduction to Linear Algebra*. Addison-Wesley.
- Karresand, M., Shahmehri, N., 2006a. File type identification of data fragments by their binary structure. In: *Information Assurance Workshop*. IEEE. West Point, New York, United States of America, pp. 140–147.
- Karresand, M., Shahmehri, N., 2006b. Oscar - file type identification of binary data in disk clusters and ram pages. In: *Fischer-Hubner, S., Rannenber, K., Yngstorm, L., Lindskog, S. (Eds.), Security and Privacy in Dynamic Enviroments*. Vol. 201. Springer, Boston, pp. 412–424.

- Kattan, A., Galvan-Lopez, E., Poli, R., O'Neil, M., 2010. Gp-fileprints: File type detection using genetic programming. In: 13th European Conference on Genetic Programming. Istanbul, Turkey, pp. 134–145.
- Kittler, J., Hatef, M., Duin, R. P. W., Matas, J., 1998. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (3), 226–239.
- Klautau, A., Jevtic, N., Orlitsky, A., 2002. Combined binary classifiers with applications to speech recognition. In: Seventh International Conference on Spoken Language Processing. Denver, Colorado, United States of America, pp. 2469–2472.
- Kohavi, R., Provost, F., 1998. Glossary of terms. *Machine Learning* 30, 271–274.
- Kotsiantis, S. B., Zaharakis, I. D., Pintelas, P. E., 2006. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review* 26 (3), 159–190.
- Kotsiantis, S. B., Zaharakis, I. D., Pintelas, P. E., 2007. Supervised machine learning: A review of classification techniques. *Frontiers in Artificial Intelligence and Applications* 160, 3–24.
- Kramer, M. A., 1991. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal* 37 (2), 233–243.
- Krogh, A., Hertz, J. A., 1995. A simple weight decay can improve generalization. *Advances in neural information processing systems* 4, 950–957.
- Kumar, R., 2005. *Research Methodology: A Step-by-Step Guide for Beginners*. SAGE Publications.
- Kuncheva, L., 2004. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley and Sons.
- Kurgan, L. A., Musilek, P., 2006. A survey of knowledge discovery and data mining process models. *Knowledge Engineering Review* 21 (1), 1–24.
- Kwak, N., Choi, C. H., 2002. Input feature selection for classification problems. *IEEE Transactions on Neural Networks* 13 (1), 143–159.
- Kwak, N. K., Kim, S. H., Lee, C. W., Choi, T. S., Oct 2002. An application of linear programming discriminant analysis to classifying and predicting the symptomatic status of hiv/aids patients. *Journal of Medical Systems* 26 (5), 427–438.
- Lam, S. W. L., 2000. Theory and application of majority vote: From condorcet jury theorem to pattern recognition. In: *International Conference on Mathematics Education into the 21st Century: Mathematics for Living*. Amman, Jordan.
- Lame, K. F., Moy, J. W., 1997. An experimental comparison of some recently developed linear programming approaches to the discriminant problem. *Computers and Operations research* 24 (7), 593–599.
- Langdon, W., Barrett, S., Buxton, B., 2002. Combining decision trees and neural networks for drug discovery. *Genetic Programming*, 76–89.

- Lawrence, S., Giles, C. L., Tsoi, A. C., 1997. Lessons in neural network training: Overfitting may be harder than expected. In: The Fourteenth National Conference on Artificial Intelligence. Providence, Rhode Island, United States of America, pp. 540–545.
- Lazkano, E., Sierra, B., 2003. Bayes-nearest: a new hybrid classifier combining bayesian network and distance based algorithms. Progress in Artificial Intelligence. Lecture Notes in Computer Science. 2902, 171–183.
- Lee, L., Rajkumar, R., Isa, D., 2012. Automatic folder allocation system using bayesian-support vector machines hybrid classification approach. Applied Intelligence 36 (2), 295–307.
- Lee, S. S., Elder, J. F., 1997. Bundling heterogeneous classifiers with advisor perceptrons. Tech. rep., University of Idaho.
- Li, B., Wang, Q., Luo, J., 2006. Forensic analysis of document fragment based on SVM. In: The 2006 International Conference on Intelligent Information Hiding and Multimedia. Pasadena, California, United States of America, pp. 236–239.
- Li, Q., Ong, A., Suganthan, P., Thing, V., 2010. A novel support vector machine approach to high entropy data fragment classification. In: The South African Information Security Multi-Conference. Port Elizabeth, South Africa.
- Li, W., Wang, K., Stolfo, S. J., Herzog, B., 2005. Fileprints: identifying file types by n-gram analysis. In: The Sixth Annual IEEE Systems, Man and Cybernetics Information Assurance Workshop. West Point, New York, United States of America, pp. 64–71.
- Lin, W., Hu, Y., Tsai, C., 2012. Machine learning in financial crisis prediction: A survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42 (4), 421–436.
- Lippmann, R. P., 1991. A critical overview of neural network pattern classifiers. In: The IEEE Workshop on Neural Networks for Signal Processing. Princeton, New Jersey, United States of America, pp. 266–275.
- Macleod, J. E. S., Luk, A., Titterton, D. M., 1987. A re-examination of the distance weighted k -nearest neighbour classification rule. IEEE transaction on Systems, Man, and Cybernetics 4 (17), 689–696.
- Maclin, R., Shavlik, J. W., *et al.*, 1995. Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks. In: International Joint Conference on Artificial Intelligence. Montreal, Quebec, Canada, pp. 524–531.
- Maimon, O., Rakach, L., 2010. Supervised Learning, 2nd Edition. Data Mining and Knowledge Discovery Handbook. 2nd Edition. Springer Science and Business Media, Ch. 8, pp. 113–138.
- Mayer, R., 2011. Filetype identification using long, summarized n-grams. Master’s thesis, Naval Postgraduate School, Monterey, United States.

- McDaniel, M., 2001. An algorithm for content-based automated file type recognition. Master's thesis, James Madison University, Harrisonburg, United States.
- McDaniel, M., Heydari, M. H., 2003. Content based file type detection algorithms. In: The 36th Annual Hawaii International Conference on System Sciences. Big Island, Hawaii, United States of America, p. 331.
- Michie, D., Spiegelhalter, D. J., Taylor, C. C., Campbell, J., 1994. Machine learning, neural and statistical classification. New York: Ellis Horwood.
URL <http://www1.maths.leeds.ac.uk/~charles/statlog/whole.pdf>
- Milgram, J., Cheriet, M., Sabourin, R., et. al., 2006. "One Against One" or "One Against All": Which One is Better for Handwriting Recognition with SVMs? In: Tenth International Workshop on Frontiers in Handwriting Recognition. La Baule, Centre de Congres Atlantia, France.
- Moody, S., 2008. SÁDI - Statistical Analysis for Data Type Identification. In: Third International Workshop on Systematic Approaches to Digital Forensic Engineering. Berkeley, California, USA, pp. 41 – 54.
- Moore, A. W., Zuev, D., 2005. Internet traffic classification using bayesian analysis techniques. ACM SIGMETRICS Performance Evaluation Review 33 (1), 50–60.
- Nisbet, R., Elder, J., Miner, G., 2009. Handbook of Statistical Analysis and Data Mining Applications. Academic Press/Elsevier.
- Olson, D. L., Shi, Y., 2006. Introduction to Business Data Mining. Irwin-McGraw-Hill Series: Operations and Decision Sciences. McGraw-Hill.
- Ozgun, A., Ozgur, L., Gungor, T., 2005. Text categorization with class-based and corpus-based keyword selection. In: International Symposium on Computer and Information Sciences. Istanbul, Turkey, pp. 606–615.
- Pal, A., Memon, N., 2009. The evolution of file carving. IEEE Signal Processing Magazine 26 (2), 59–71.
- Paliwal, M., Kumar, U. A., 2009. Neural networks and statistical techniques: A review of applications. Expert Systems with Applications 36 (1), 2–17.
- Partridge, D., Yates, W. B., 1995. Engineering multiversion neural-net systems. Neural Computing 8, 869–893.
- Pollack, J. B., 1990. Backpropagation is sensitive to initial conditions. Complex Systems 4, 269–280.
- Ramlall, I., 2010. Artificial intelligence: Neural networks simplified. International Research Journal of Finance and Economics (39), 105–120.
- Rashid, T., 2008. Classification of churn and non-churn customers for telecommunication companies. International Journal of Biometrics and Bioinformatics 3 (5), 82.

- Reed, R., 1993. Pruning algorithms-a survey. *IEEE Transactions on Neural Networks* 4 (5), 740–747.
- Refaeilzadeh, L., Tang, L., Liu, H., 2009. Cross-validation. *Encyclopedia of Database Systems*. Springer, United States.
- Reinartz, T., 2002. A unifying view on instance selection. *Data Mining and Knowledge Discovery* 6 (2), 191–210.
- Render, B. E., Ralph M. Stair, J., Hanna, M. E., 2011. *Quantitative Analysis for Management*. Pearson Prentice Hall.
- Rice, J. A., 2007. *Mathematical Statistics and Data Analysis*. Duxbury Advanced Series. Brooks/Cole.
- Riedmiller, M., Braun, H., 1993. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In: *EEE International Conference on Neural Networks*. San Francisco , California, United States of America, pp. 586–591.
- Ripley, B. D., 2008. *Pattern recognition and neural networks*. Cambridge university press.
- Rokach, L., 2008. Genetic algorithm-based feature set partitioning for classification problems. *Pattern Recognition* 41 (5), 1676–1700.
- Roussev, V., Garfinkel, S. L., 2009. File fragment classification: The case for specialized approaches. In: *Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering*. Oakland, California, United States of America, pp. 3–14.
- Rumelhart, D. E., Durbin, R., Golden, R., Chauvin, Y., 1995. Backpropagation: The basic theory. *Backpropagation: Theory, Architectures and Applications*, 1–34.
- Russel, S., Novig, P., 2010. *Artificial Intelligence: A modern approach*, 3rd Edition. Pearson.
- Saha, S., Ekbal, A., 2013. Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition. *Data and Knowledge Engineering* 85 (0), 15–39.
- Salzberg, S. L., 1997. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery* 1 (3), 317–328.
- Sarma, K. S., 2007. *Predictive Modeling With SAS circledR Enterprise Miner: Practical Solutions for Business Applications*. SAS Press Series. Sas Institute, Cary, North Carolina, United States of America.
- SAS, 2009. *Base SAS ®9.2 Procedures Guid*. SAS Institute Inc., Cary, North Carolina, United States of America.
- Schiffmann, W., Joost, M., Werner, R., 1994. Optimization of the backpropagation algorithm for training multilayer perceptrons. Tech. rep., University of Koblenz: Institute of Physics, Koblenz, Germany.
- URL [www.cs.bham.ac.uk/ pxt/NC/schiffmann.bp.pdf](http://www.cs.bham.ac.uk/pxt/NC/schiffmann.bp.pdf)

- Shannon, C. E., 2001. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review* 5 (1), 3–55.
- Sportiello, L., Zanero, S., 2011. File block classification by support vector machine. In: *Sixth International Conference on Availability, Reliability and Security (ARES)*. pp. 307–312.
- Stam, A., Ragsdale, C. T., 1992. On the classification gap in mathematical programming-based approaches to the discriminant problem. *Naval Research Logistics* 39 (4), 545–559.
- Steeb, W. H., 2005. *The Nonlinear Workbook: Chaos, Fractals, Cellular Automata, Neural Networks, Genetic Algorithms, Gene Expression Programming, Support Vector Machine, Wavelets, Hidden Markov Models, Fuzzy Logic with C++, Java and Symbolic++ Programs*, 3rd Edition. World Scientific.
- Sudjianto, A., Nair, S., Yuan, M., Zhang, A., Kern, D., Cela-Diaz, F., 2010. Statistical methods for fighting financial crimes. *Technometrics* 52 (1), 5–19.
- Tanenbaum, A. S., 2008. *Modern operating systems*, 3rd Edition. Pearson.
- Ting, K., Witten, I., 1999. Issues in stacked generalization. *Journal of Artificial Intelligence Research* 10, 271–289.
- Tsybmal, A., Puuronen, S., Patterson, D., 2003. Ensemble feature selection with the simple bayesian classification. *Information Fusion* 4 (2), 87–100.
- Twala, B., 2010. Multiple classifier application to credit risk assessment. *Expert Systems with Applications* 37 (4), 3326–3336.
- Veenman, C. J., Aug. 2007. Statistical disk cluster classification for file carving. In: *Third International Symposium on Information Assurance and Security*. pp. 393–398.
- Welch, T. A., 1984. A technique for high-performance data compression. *IEEE Computer*, 8–19.
- Wilgenbus, E. F., Kruger, H. A., du Toit, J. V., 2012. File fragment classification: An application of a neural network and linear programming based discriminant model. In: *Vigo-Aguiar, J. (Ed.), International Conference of Computational and Mathematical Methods in Science and Engineering*. Vol. 3. Murica, Spain, pp. 1237–1248.
- Woods, K., Kegelmeyer, W., Bowyer, K., 1997. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19 (4), 405–410.
- Xu, L., Krzyzak, A., Suen, C., 1992. Methods of combining multiple classifiers and their applications to handwriting recognition. *Systems, Man and Cybernetics, IEEE Transactions on* 22 (3), 418–435.
- Ying, Y., Webb, G., Wu, X., 2010. *Discretization Methods. Data Mining and Knowledge Discovery Handbook*. 2nd Edition. Springer Science and Business Media, Ch. 6, pp. 101–115.

- Youssef, S. B., Rebai, A., 2007. Comparison between statistical approaches and linear programming for resolving classification problem. *International Mathematical Forum* 2 (63), 3125–3141.
- Yu, L., Liu, H., 2004. Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research* 5, 1205–1224.
- Zhang, G. P., 2000. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 30 (4), 451–462.
- Zhang, G. P., 2010. Neural Networks For Data Mining. *Data Mining and Knowledge Discovery Handbook*. 2nd Edition. Springer Science and Business Media, Ch. 12, pp. 419–444.
- Zhou, Z., Chen, Z., 2002. Hybrid decision tree. *Knowledge-Based Systems* 15 (8), 515–528.