# Conditional-Entropy Metrics for Feature Selection

*Iain Bancarz*

Doctor of Philosophy

Institute for Communicating and Collaborative Systems

School of Informatics

University of Edinburgh

2005

# Abstract

We examine the task of feature selection, which is a method of forming simplified descriptions of complex data for use in probabilistic classifiers. Feature selection typically requires a numerical measure or *metric* of the desirability of a given set of features. The thesis considers a number of existing metrics, with particular attention to those based on entropy and other quantities derived from information theory. A useful new perspective on feature selection is provided by the concepts of *partitioning* and *encoding* of data by a feature set. The ideas of partitioning and encoding, together with the theoretical shortcomings of existing metrics, motivate a new class of feature selection metrics based on conditional entropy. The simplest of the new metrics is referred to as *expected partition entropy* or EPE.

Performances of the new and existing metrics are compared by experiments with a simplified form of part-of-speech tagging and with classification of Reuters news stories by topic. In order to conduct the experiments, a new class of accelerated feature selection search algorithms is introduced; a member of this class is found to provide significantly increased speed with minimal loss in performance, as measured by feature selection metrics and accuracy on test data. The comparative performance of existing metrics is also analysed, giving rise to a new general conjecture regarding the *wrapper* class of metrics. Each wrapper is inherently tied to a specific type of classifier. The experimental results support the idea that a wrapper selects feature sets which perform well in conjunction with its own particular classifier, but this good performance cannot be expected to carry over to other types of model.

The new metrics introduced in this thesis prove to have substantial advantages over a representative selection of other feature selection mechanisms: Mutual information, frequency-based cutoff, the Koller-Sahami information loss measure, and two different types of wrapper method. Feature selection using the new metrics easily outperforms other filter-based methods such as mutual information; additionally, our approach attains comparable performance to a wrapper method, but at a fraction of the computational expense. Finally, members of the new class of metrics succeed in a case where the Koller-Sahami metric fails to provide a meaningful criterion for feature selection.

# Acknowledgements

I would like to thank my supervisor, Miles Osborne, for excellent advice, encouragement, and support. He has been a superb mentor and guide and I consider myself very fortunate to have worked with him.

Thanks are also due to a number of people at Edinburgh University who helped with my research: I would particularly like to mention Jason Baldridge, Julia Hockenmaier, Andrew Smith, Chris Callison-Burch, Markus Becker, and Chris Williams, as well as the departmental secretary Betty Hughes and the members of the Stats-NLP reading group. I would also like to thank Rob Malouf of San Diego State University for the code used to implement maximum-entropy models.

Many friends in Edinburgh and elsewhere helped keep me sane and happy during the course of my PhD. Special thanks to Alisdair, Patrick, Chris, Dave, Euan, Ellie, Graeme, Ruth, Duncan, Charlie, Clionadh, Lucy, Jessica, Justin, and Dominique; to everyone at Edinburgh University Creative Writing Society; and to my mother and grandmother.

My girlfriend Stephanie has helped me in more ways than I could ever describe here. She gives me love and care in good times and bad, and has my deepest thanks and love in return.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Iain Bancarz*)

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

‘There are some perceptions which do not call for any further exercise of thought, because sensation alone can judge them adequately; but others which demand the exercise of thought because sensation cannot give a trustworthy result.’ — Plato, *The Republic*

## 1.1 The Feature Selection Problem

### 1.1.1 Simple Descriptions of Complex Data

In attempting to understand the world, we are often faced with an overwhelming number of possible variables. A fully detailed description of our environment may contain more information than can be effectively processed by a computer, or understood by a human being.

For example, if carrying out image recognition one could easily have a million pixels, each with a thousand possible colours and a thousand possible intensities, for a total of $10^{12}$ conceivable images. If we wished to construct a classifier which compared two images – for instance, to see if they were photographs of the same person – there would be some $10^{24}$ possible image pairs. In physics, a macroscopic sample of a gas contains on the order of $10^{23}$ individual particles, each with a distinct position and velocity. For Web page classification, Google records in excess of $8 \times 10^{12}$ individual pages, each of which may itself contain a large quantity of data.

When dealing with complicated environments such as these, it is usually necessary to find a simplified representation of the data. We wish to obtain such a description without losing relevant information. There is a basic tension between seeking shorter descriptions, which are easier to process and understand; and longer descriptions, which can in principle contain more information about the data. The idea of finding a simple representation can be viewed as an application of Occam's Razor, which advocates adopting the simplest available theory that is consistent with our observations.

*Feature selection* involves choosing characteristics of the data which are relevant to a particular task – for instance, constructing a model for the classification of data points. We formalise the idea of a feature by defining feature *functions* which map the data space to some other set. The number of possible feature functions is typically very large. If our feature functions map a set of $N$ possible data points to another set with $k$ elements, then there are $k^N$ possible feature functions. In general, the set of all possible features is very large and unwieldy, and many features may be of little use for classification. We therefore seek a smaller and more informative subset of the available feature functions; but this is a challenging task in itself, as a set with $n$ elements will have $2^n$ distinct subsets. Nonetheless, effective feature selection techniques exist and can give rise to simple yet informative descriptions of data; this thesis aims to extend and improve existing methods of feature selection.

Feature selection is a very general technique applicable to a wide variety of domains: Computational linguistics, image recognition, medical diagnosis, statistical physics, bioinformatics, and even the detection of buried land mines [AZ00]. This thesis is primarily intended as a contribution to the mathematical foundations of feature selection. The theoretical work presented herein should, in principle, be applicable to many different domains, including but not limited to the ones just mentioned. In order to provide an empirical assessment of the effectiveness of the mathematical methods in this thesis, we also present experiments with a simplified form of part-of-speech tagging and with the classification of Reuters news articles by topic.

### 1.1.2 Basic Concepts in Feature Selection

In the context of feature selection, a *feature* can be thought of as an identifiable characteristic of the data. Finding a 'good' set of features will in general require three steps: *Definition*, *extraction*, and *selection*. Definition involves specifying the general form of our features; extraction, finding a comparatively large collection of available features which satisfy our definition and have at least some relevance to the data; and selection, choosing a subset of the initial pool of features which we find useful. 'Usefulness' may be defined qualitatively, as improving our understanding of the phenomenon at hand; or quantitatively, as giving rise to a faster or more accurate probabilistic model. In practice, careful feature selection can bring about significant improvements in all these respects.

For example, suppose that we wish to construct a probabilistic model for classifying web pages by subject; and we have a training set of pages which we believe to be a representative sample of the ones our classifier will encounter. We begin by defining our features to be the presence of particular words or phrases. Extraction of a broad pool of features might be accomplished by finding all distinct words, and all distinct phrases of four words or fewer, in a set of training data. This pool of features is likely to be inconveniently large. A model which took account of all the features present in a large set of training data would be comparatively slow and difficult to understand; it might also be less accurate than one which concentrated on a small number of more informative features.

The above example considers a *classification* task. Feature selection usually revolves around classification, and classifiers will be the primary focus of this thesis. However, there is no reason to believe that the results of the thesis will not generalise to other settings, such as sequencing.

**Irrelevant and Redundant Features:** Feature selection is the process of finding a 'good' set of features. Exactly how we may characterise a 'good' feature set is one of the key questions of feature selection, and will be one of the main topics addressed by this thesis. Generally speaking, we would like to avoid the inclusion of *irrelevant* and *redundant* features. Intuitively, an irrelevant feature is one which gives no useful information; for instance, the presence or absence of the word ‘the’ is most unlikely to help us determine the subject of a given document.

A redundant feature is one which, while it may be useful in isolation, does not provide much additional information given that a particular other feature is already present in our chosen subset. For example, the phrase ‘annual gross domestic product’ will be strongly associated with documents whose subject is ‘economics’; but it will be largely redundant for the purpose of document classification if we are already looking for the phrase ‘gross domestic product’. Broadly speaking, redundancy is to be avoided; but as we discuss in Section 4.5, a certain amount of redundancy may be desirable as an error-correcting measure.

Feature selection typically requires a numerical measure of the desirability of a feature set. A wide variety of such *metrics* have been proposed; most of them attempt to capture these intuitive ideas of irrelevance and redundancy in a precise, quantitative fashion. This thesis examines a number of existing measures and introduces new ones, as Section 1.2 describes in greater detail. The new measures are theoretically appealing due to their solid foundation in information theory, and prove to have superior empirical performance to a representative selection of existing metrics.

**Search Strategies:** Another important task in feature selection is the formulation of an appropriate *search strategy*. A set with $n$ elements will have $2^n$ distinct subsets; even with a modest number of available features, we will therefore have a very large number of possible subsets. Exhaustively evaluating all the possible subsets is generally impractical; however, effective algorithms exist for finding reasonably good subsets (according to our chosen definition of ‘good’) without an exhaustive search. Again, this thesis surveys existing search strategies and proposes some new ones, as described in Section 1.2. A member of a new class of accelerated search algorithms proves to significantly increase the speed of feature selection with minimal loss in performance.

**Definition and Extraction:** The initial stages of definition and extraction are seldom explicitly considered in the literature; the presence of a large pool of available features, from which we wish to select a smaller subset, is often assumed as a given. In some cases, there may be little or no choice to be made at the first two steps. For instance, if our pool of available features consists of *all* the possible features which fit our definition, then the extraction step has effectively been omitted; and similarly, there may be only one reasonable definition of a 'feature' for our chosen data set.

More often though, there is a genuine choice as to how we define our feature functions and how we select our initial pool. These decisions are usually much easier than the final stage of feature selection itself, and correspondingly less interesting; but they should not be ignored altogether. For instance, in the document-classification example above we could have restricted ourselves to single words, or expanded our scope to include larger phrases or entire sentences; and we could have chosen a more or less restrictive method of generating our pool of available features.

Thus, definition and extraction may constitute a 'pre-selection' of features according to some *a priori* criteria. Such pre-selection may itself be based on a rigorous, quantitative analysis of the data; but it is often conducted on a more qualitative basis, and it may be quite arbitrary. In the document-classification example, we did not have any compelling reason to restrict ourselves to phrases of four words or fewer; it merely 'seemed reasonable.' There is nothing wrong with making our initial choices on qualitative grounds; however we should be aware when we are doing it, and that the results of feature selection may be dependent on our pre-selection choices.

**Mathematical Framework:** The above discussion helps to motivate the feature selection idea, but it is somewhat vague and imprecise. It is very helpful to adopt a clear mathematical framework for feature selection. Such a framework has a number of benefits: It enables us to think more precisely about feature selection; and it permits the definition of very general methods of feature selection, which can be applied to practical tasks in many different domains. This thesis will review previous efforts to establish a theoretical framework for feature selection, and suggest some additions and clarifications.

One of the key elements of a rigorous treatment of feature selection is the definition of *feature functions*. A feature function maps the data space to some other set, usually but not necessarily a simpler one. The function takes values based on some characteristic of the data point which is considered to be important. Feature functions therefore serve as effective tools for formalising our intuitive notion of a feature as being an identifiable characteristic of a data point.

As discussed in Chapter 2, any feature selection problem can be represented in terms of feature functions. It is common to abuse terminology slightly by referring to the feature functions simply as features; this is convenient, but can cause some confusion if we do not clearly distinguish between feature *functions* and the characteristics of data points.

## 1.2 Outline of Thesis

### 1.2.1 Preliminaries and Literature Survey

The thesis begins by establishing some basic definitions and terminology in Chapter 2. Feature functions are defined, as are schemes for combining them and incorporating them into probabilistic models. In the process, some new concepts are introduced: Specifically the idea of a *non-informative value* taken by a feature when it obtains no significant information on a given data point, and a formal definition of the *model architectures* used to form feature-based probabilistic models. Two commonly used model architectures are the naive-bayes (NB) and maximum-entropy (ME) schemes, which are used for the experiments detailed in Chapters 5, 6, 7, and 8. Chapter 2 defines both the NB and ME model architectures and discusses their properties and previous applications. The chapter concludes with a discussion of the important subclass of *binary features*, which were employed in our experiments.

Chapter 3 surveys the existing literature on feature selection. It outlines the commonly accepted theoretical framework for the feature selection problem, including general classes of evaluation metric and search strategy. An especially important distinction is the division of evaluation metrics into *wrappers* and *filters*; a further subdivision of filters into *hierarchical* and *non-hierarchical* measures is proposed. Particular attention is paid to filter metrics drawn from information theory: Information gain, the Koller-Sahami (KS) information loss metric, and two distinct types of mutual-information measure are defined, and their theoretical properties are discussed. Finally, the survey briefly outlines two topics closely related to feature selection: Dimensionality reduction and model selection by minimum description length.

### 1.2.2 Theoretical Developments

Chapter 4 introduces new perspectives on feature selection based on *partitioning* and *encoding* of data by a feature set. These new ideas, together with the theoretical shortcomings of existing metrics, motivate a new class of feature selection metrics based on conditional entropy. The new metrics are similar to the Koller-Sahami (KS) information loss measure; they share its solid foundation in information theory, but are theoretically more appealing.

The KS metric can be thought of as taking the pool of available features as a reference point; but as noted in Section 1.1.2, this pool will in general be dependent on the ‘pre-selection’ choices of feature definition and extraction and in particular on the set of available training data. It is therefore doubtful whether the pool of available features is a sufficiently general reference point; the new metrics avoid this problem by adopting the uniform distribution as their benchmark. There are also theoretical reasons to expect that the conditional-entropy metrics will perform well in comparison with other existing measures, such as information gain, mutual information, and wrapper methods.

The simplest of the new metrics is referred to as *Expected Partition Entropy* or EPE; as the name suggests, it is based primarily on the concept of partitioning. Ideas drawn from coding theory, and a desire to more explicitly capture interactions between features, motivate the extension of EPE to a family of metrics called *Expected Covering Entropies* or ECEs. The ECE metrics provide an interesting perspective on the use of *redundant* features as an error-correcting measure. In addition, they can be expected to be more effective than EPE in environments with very sparse training data; this is demonstrated in Chapter 8, which describes situations in which ECE succeeds where EPE and the KS metric fail to provide meaningful criteria for feature selection.

### 1.2.3 Experiments and Analysis

The new conditional-entropy metrics are evaluated in two different experimental domains: A simplified form of part-of-speech (POS) tagging with data drawn from the Penn treebank, and classification by topic of news articles from the Reuters corpus.

The former setting is discussed in Chapters 5, 6, and 7, and the latter in Chapter 8.

**Part-of-Speech Tagging:** Chapter 5 outlines the general setting and parameters of the POS-tagging experiments, and introduces a new class of accelerated feature selection search algorithms. Feature selection is carried out using an accelerated search algorithm with the new EPE metric and a variety of existing ones: Mutual information, the KS measure, a simple frequency-based metric, and two different types of wrapper metric. (The ECE metrics are too slow to be effectively used for feature selection investigated in this setting. However, as discussed below, they are successfully evaluated for POS-tagging from a slightly different perspective in Chapter 7, and used for feature selection in Chapter 8.) Random feature selection, and evaluation of the set of all available features, are also carried out for comparison. The feature sets obtained in experiments are incorporated into NB and ME models, and the performance of the models on held-out test data is assessed in Chapter 6.

Although the primary goal of the POS-tagging experiments is to assess the performance of the EPE metric, some interesting subsidiary results arise. Brief investigation of the properties of the new accelerated search algorithms indicates that they give rise to significant decreases in computational expense with minimal loss of performance. It is also noted that wrapper metrics – which are fundamentally tied to particular model architectures – perform particularly well when selecting features for 'their own' type of model, but this good performance does not necessarily carry over to other evaluation methods. It is shown that feature sets selected by wrappers are in general closely attuned to a particular type of classifier, whereas filters obtain more broadly applicable feature sets.

The new EPE metric obtains very good results in comparison with two different types of wrapper metric, and with a selection of filter metrics: Frequency-based cutoff, mutual information, and Koller-Sahami information loss. A wrapper metric based on the NB learner outperforms EPE in selecting features for an NB model, in keeping with the 'specialisation' hypothesis mentioned above; but EPE obtains clearly better performance with an ME model. EPE and an ME wrapper give rise to comparable accuracies, but EPE is considerably faster.

While the frequency-based and mutual-information filters outperform EPE when selecting very small feature sets, EPE is significantly better when selecting larger feature subsets. For medium-sized subsets, EPE obtains accuracies up to 12% greater than its frequency-based and mutual-information counterparts. Finally, the theoretical concerns raised for the KS metric prove to be justified in this setting. The KS metric frequently attains its optimal value for feature sets which are far from optimal in terms of their performance on held-out test data, whereas EPE does not suffer from this problem; in this experimental setting, EPE has all the advantages of KS with none of its drawbacks.

The question implicitly addressed in Chapter 6 was whether feature sets selected by a conditional-entropy metric – specifically EPE – will give rise to better performance on held-out test data. Chapter 7 evaluates the conditional-entropy EPE and ECE metrics from a slightly different perspective. The values of EPE and three different variants of ECE are computed on feature sets obtained using EPE, the NB and ME wrappers, mutual information, and the KS metric. In other words, the family of conditional-entropy metrics are used for assessment of existing feature subsets, rather than selection of new subsets. Computation of the EPE and ECE metrics shows a strong correlation between improvement in EPE/ECE and greater accuracy on test data, regardless of the metric used to select the feature set. Hence, the conditional-entropy metrics are good general indicators of classification accuracy in this experimental setting.

**Document Classification:** Additional insight into the properties of the EPE/ECE and Koller-Sahami metrics is provided by experiments with classification of Reuters news articles by topic, detailed in Chapter 8. This setting is very similar to one used by Koller and Sahami to evaluate their information-loss metric. It is found that the KS metric suffers from limitations in the document-classification setting which did not become apparent in KS' initial experiments, such that when selecting subsets of more than about 40 features from an initial pool of about 2400, the KS metric is equivalent to choosing features at random. The EPE metric suffers from a similar problem; however its extension to ECE overcomes these difficulties, and proves to be a useful method of feature selection for the Reuters document-classification task. Feature selection by ECE gives rise to feature sets which, in comparison with the pool of all available features, are as little as one-tenth the size and give rise to an increase of up to 4.3% in classification accuracy on test data.

**Discussion:** The thesis concludes with Chapter 9, which discusses the theoretical developments and experimental results and suggests a number of topics for future research.

# Chapter 2

# Preliminaries

## 2.1 Introduction

In addressing the topic of feature selection in probabilistic modelling, it is natural to begin by defining what features are and how they relate to probabilistic models. Definitions in the literature are often very informal; often there is no explicit definition of a feature at all. Instead the word ‘feature’ is used in its obvious, intuitive sense, to mean some identifiable characteristic of individual data points [DL97]. ‘Feature selection,’ then, is the process of choosing the particular characteristics that interest us, and somehow incorporating our chosen feature set into a probabilistic model.

The intuitive definition often proves adequate to obtain interesting results, including a number of successful approaches to practical tasks. However, a more formal definition of features can be useful. In this chapter we follow John et al. in defining *feature functions* which allow us to formalise our intuitive notion of a feature [JKP94]. We also formally define means of incorporating features into a model; this includes definitions of *model architectures* and the *feature-based probabilistic models* produced by them, which are not part of the standard literature on feature functions but prove useful in later discussions. The formal system presented in this chapter aims to be compatible with intuitive ideas, while giving rise to interesting ways of extending them.

Creating a more formal framework for feature selection has a number of advantages. It enables us to think about features in a more precise and systematic way, and thereby gives rise to ideas which can prove very useful in practical feature selection tasks. Concepts developed in this chapter will provide useful terminology for discussing the existing methods surveyed in Chapter 3, as well as helping to motivate the novel feature selection techniques developed in Chapter 4.

The existence of formal definitions and notation also allows us to see connections between feature selection and other fields of mathematics (or computer science, depending on one's point of view). Such connections might not have been apparent in the absence of a rigorous mathematical framework, and can prove very enlightening and useful. Given our theoretical setting, we can see that feature selection is closely connected to several well-established areas of study, such as *quantization* and *coding theory*, as we discuss in Chapter 4.

In this chapter, we will begin by defining *feature functions*, which constitute a mathematical representation of our intuitive idea of a feature; and *feature-based probabilistic models*, which are methods of incorporating sets of feature functions into a probability distribution. We then consider various extensions and consequences of these ideas, which will set the stage for our discussion of existing feature selection algorithms in Chapter 3 and the development of new ones in Chapter 4.

## 2.2 Feature Functions

### 2.2.1 Data Points and Features

First, we establish some basic terminology and notation. We denote data points by $x$ and the set of all possible data points by $X$. We also have a set of training data; we denote the training set and its individual elements respectively by $\tilde{X}$ and $\tilde{x}$. The data set $X$ is often very large, and may be infinite, while the training set $\tilde{X}$ is typically a finite subset of $X$.

**Example 2.1:** If our data points are English sentences together with their parse trees, then the set of all possible data points is infinite; meanwhile, we have a finite set of parsed sentences available with which to construct a probabilistic model. □

A *feature function* $f : X \mapsto Y$ is a function that maps data points $x \in X$ to some other set $Y$. We refer to $Y$ as the *feature image set*. For the sake of brevity, we often refer to feature functions simply as *features*. We may think of the value of $f(x)$ as representing some identifiable characteristic of the data point $x$.

**Example 2.2:** Suppose that each 'data point' $x$ is a web page. We might then define a feature $f_\ell(x)$, whose value is the length in words of the web page $x$. Clearly, $f_\ell(x)$ can be any whole number greater than zero; hence, the feature image set $Y$ is the set of positive integers. □

If we are dealing with more than one feature, then for convenience we place them in some arbitrary order to form a *feature list* $F = (f_1, f_2, \ldots, f_n)$. The feature list can be thought of as a vector-valued function:

$$F : X \mapsto Y_1 \times Y_2 \times \ldots \times Y_n$$

where $Y_1, \ldots, Y_n$ are the respective feature image sets. Hence, feature lists are sometimes called *feature vectors*. It is usual to abuse notation slightly by using $F(x)$ (and the term 'feature vector') to refer to both the feature list function in general, and its value on a particular data point $x$.

## 2.2.2 Comments on the Definition of a Feature List

**Shared image sets:** Features will often share the same image set $Y$, so that $F(x)$ maps $X$ to $Y^n$ (or some subset of $Y^n$). For example, we could have each feature $f_i$ take values in the non-negative integers. If we suppose further that we have five features in total, then the feature vector of a given data point $x$ would be a list of five non-negative integers – it might look something like $(12, 0, 1, 18, 23)$. This is the most common situation, and it will henceforth be assumed to be the case unless otherwise stated. However, it is worth noting that other feature sets are possible. For instance, we might add a sixth feature that can take any value in the real numbers; a sample feature vector might then be $(8, 14, 3, 0, 22, -0.475)$.

**Feature 'vectors':** The set of possible feature lists is not a true vector space, because in general it is not meaningful to add the lists together or multiply them by scalars.

Indeed it is not necessary for the features to be ordered at all, as long as each is uniquely labelled in some fashion. Nonetheless the vector analogy is a useful one, and some related concepts – such as the scalar product – remain relevant in the feature list setting.

**Surjectivity:** Note that $F$ may not be surjective; in other words, there may be points in $Y^n$ which do not correspond to any point in $X$. More simply, just because a particular feature vector can be defined does not mean that it will actually be observed. Some feature vectors may simply be very unlikely: For instance, we are unlikely to find documents which mention both 'Britney Spears' and 'Hilbert space theory.' (As of 10 May 2005, there were no Google hits for this pair of phrases; of course, this paragraph contains a rare mention of both topics in the same document.) Other feature vectors may not appear for some deeper reason; for example, if our data points represent weather reports, we will not find snow coexisting with temperatures well above freezing. Examining 'forbidden' combinations of feature values can thus improve our qualitative understanding of the data.

**Injectivity:** The feature vector $F$ may also not be injective; that is, $F$ may send different points in $X$ to the same point in $Y^n$. If $F(x_1) = F(x_2) = y$, then the feature set represented by $F$ is unable to distinguish between the data points $x_1$ and $x_2$; it assigns them both the same 'description' $y$. This is quite a common situation. Again, it may be useful to examine data points which receive the same combination of features. Observing a pair of data points with identical feature vectors may motivate us to introduce additional features, so as to be able to distinguish these two points; or it may simply alert us to the fact that the two data points are in some sense quite similar.

### 2.2.3 Non-Informative Values

Individual features $f_i$ will often have a *non-informative value*. As the name suggests, a non-informative value is one indicating that a feature provides no significant information on a given data point.

**Example 2.3:** If our data points are web pages then we may define features which return the number of occurrences of a given key word: $f_{\text{logic}}(x)$ counts the occurrences of the word ‘logic’ on the web page denoted by $x$, and so on. We will often have $f_{\text{logic}}(x) = 0$; in these instances, knowing the value of the ‘logic’ feature does not bring us significantly closer to knowing the topic of the web page. □

In the above example, the non-informative value of the feature was zero, but that does not have to be the case. For instance, we may have a feature which looks at a company’s stock price over the last year, and classifies it as good (growth much better than the overall growth of the stock market), mediocre (roughly in line with average growth), or poor (much worse than average). In this case it is natural to think of ‘mediocre’ as the non-informative value. A non-informative value, then, can be viewed as a default output for our features. If a feature often takes a non-informative value, then we can think of it as being quite specialised, and only providing information on a few data points. Such specialised features can be very useful in practice.

However, problems may arise if *all* features take their non-informative value on a given data point – particularly if this occurs on a great many data points which are otherwise unrelated. For instance, it may be that none of the key words we are looking for occur in certain web pages, and the nature of these web pages varies radically. We refer to the feature list composed entirely of non-informative values as the *non-informative vector*.

Occurrence of the non-informative vector may not seriously trouble us; if we are interested in specifically classifying financial news, then web pages which do not contain any financial key words could simply be discarded as irrelevant. On the other hand, if we are trying to construct a more general document classifier it will be a significant handicap if we often find that none of our key words are present. Hence, very frequent occurrence of the non-informative vector may indicate that our feature set is inadequate for the task at hand.

## 2.3 Feature-Based Probabilistic Models

### 2.3.1 Model Architectures

We refer to a scheme for incorporating features into a probabilistic model as a *model architecture*. The resulting *feature-based probabilistic model* (FBPM) is a probability distribution over possible feature values. Most model architectures allow considerable flexibility in the choice of feature set.

A model architecture may also include a number of free parameters; if one parameter is assigned to each feature, then the parameters are known as *feature weights*. Feature weights are used to assign different relative importances to the members of a feature set. We usually denote the set of free parameters by $\lambda$ and its individual members by $\lambda_i$. The collection of all possible sets of parameters is denoted by $\Lambda$. As with features, it is usually convenient to place the parameters in some consistent order and think of them as a vector. In particular, if we have one parameter affecting each of our $n$ features then it is convenient to think of our parameter set as a vector:

$$\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$$

where the parameter $\lambda_i$ corresponds to the $i$th feature.

In practice, we would like to set the parameters to values which are in some sense optimal. There are many possible definitions of ‘optimal.’ One of the most common is the principle of *maximum likelihood* or ML, which dictates that we maximise the likelihood of the training data with respect to the model. The ML principle is used to train the maximum-entropy models described in Section 2.3.4. Another possible criterion is the principle of *minimum description length*, which is briefly discussed in Section 3.4.

Having defined the optimum parameter values, we require a method of achieving them in practice. Setting the parameters to the best possible values is a problem in $n$-dimensional optimisation, and is unlikely to have an analytic solution. Instead, we generally use an iterative *training algorithm* which we hope will approach the optimal parameter values.

In order to construct an FBPM, we require a feature set $F$. For convenience, we assume that the features have been ordered into a list:

$$F = (f_i : i = 1, \ldots, n)$$
$$f_i : X \mapsto Y$$

As noted in Section 2.2.1 above, it is useful to think of $F$ as a vector-valued function sending points in $X$ to points in $Y^n$. An important characteristic of many model architectures is the ability to incorporate a *variable* set of features into a probabilistic model of feature values. There are typically very few restrictions on the feature set. In the examples we consider in Sections 2.3.4, and 2.3.3, the number of features $n$ has a lower limit of zero and no upper limit; and the features themselves may take any form, as long as the possible feature values $y$ are real numbers.

A *feature-based probabilistic model* or FBPM arises when, having chosen a model architecture, we specify a particular feature set and any appropriate parameters. The resulting FBPM is a probability distribution over possible combinations of feature values $y \in Y^n$. More formally, let $G$ denote the FBPM and $\lambda \in \Lambda$ our free parameters (if any – if our model architecture includes no additional parameters, then $\lambda$ can of course be omitted from the notation). We then have:

$$G_\lambda(y) \ : \ \{Y^n \times \Lambda\} \mapsto \{0, 1\}$$

So given any appropriate parameters $\lambda$, our FBPM is a function $G_\lambda(y)$ that assigns a probability to each possible combination of feature values. Of course, the points $y$ themselves arise from the value taken by our feature set on particular data points. For each data point $x$, there exists some $y \in Y^n$ such that $y = F(x)$. Hence, the FBPM can be used to form a probability distribution over data points:

$$P_\lambda(x) = G_\lambda(F(x))$$

Note that in order for the probabilities $P_\lambda(x)$ to sum to 1, we must have $G_\lambda(y) = 0$ for any ‘forbidden’ vector $y$ which does not equal $f(x)$ for any $x$ in $X$.

It is worth emphasising that our FBPM is *not* directly modelling the data points, but instead models the combinations of feature values which can arise from them. It is inherently incapable of distinguishing between different data points which have the same vector of feature values.

Two popular and effective examples of model architectures are the *naive-bayes* and *maximum-entropy* schemes, which we discuss respectively in Sections 2.3.3 and 2.3.4. These two architectures were employed for the experiments outlined in Chapter 5. Before discussing them, we briefly consider the distinction between conditional and unconditional models.

## 2.3.2 Conditional Models

This thesis concentrates on the use of FBPMs for classification although, as noted in Chapter 1, feature selection can be applied to other tasks as well. In a feature-based classification task, the FBPM is a conditional probability distribution which we refer to as a *classifier*. We aim to assign a data point to one of several categories $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$. The set of categories is usually, but not necessarily, finite.

Each data point $x \in X$ then consists of a *predicate* $\pi$ and a *label* $c$. For instance, the predicate might be an English word, with a part-of-speech tag as its label; a data point would then be of the form (bank, NOUN). A conditional FBPM assigns a conditional distribution over labels to each predicate: For instance, we might have $\Pr(\text{NOUN}|\text{bank}) = 0.8$ and $\Pr(\text{VERB}|\text{bank}) = 0.2$. The features used in a conditional FBPM will usually have values dependent only on the predicate. The model can then be used to estimate the probable categories of unlabelled data points.

**Example 2.4:** If our data points are web pages, then we might define a number of features which take the value 1 if a particular key word is present and 0 if it is absent. (We are thus using *binary features*; see Section 2.4 below.) Suppose that we wish to classify web pages by subject; in this case, the predicate is the web page itself while the label is its subject.

For instance, we could define binary features $f_1$ and $f_2$ whose truth conditions are the occurrence of the words ‘bank’ and ‘river’ respectively. If the feature vector $F(x) = (1,0)$ – that is, the page contains ‘bank’ but not ‘river’ – there is a high probability that the category is ‘finance’. If $F(x) = (1,1)$ then the probability of the subject being finance is much lower. $\square$

### 2.3.3 Naive-Bayes

The Naive-Bayes model architecture gives rise to a family of simple feature-based classifiers. It arises from Bayes' rule for conditional probabilities, which may be expressed as follows for two events $A$ and $B$ [GW86]:

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)}$$

In the case of feature-based classification, we suppose as usual that the data space is partitioned into countably many classes $\mathcal{C}_i$, and we have a feature set $F = (f_1, f_2, \ldots, f_n)$ chosen from a pool of available features $\mathcal{F}$. We are interested in finding the category of data points $x$, given their vectors of feature values $y = F(x)$. Bayes' rule gives us:

$$\Pr(x \in \mathcal{C}_i | F(x) = y) = \frac{\Pr(F(x) = y | x \in \mathcal{C}_i)\Pr(x \in \mathcal{C}_i)}{\Pr(F(x) = y)}$$

The probabilities $\Pr(x \in \mathcal{C}_i)$ and $\Pr(F(x) = y)$ can be easily estimated from our set of training data – see Section 3.3.1 for further discussion of how this may be accomplished. However, the conditional probability $\Pr(F(x) = y | x \in \mathcal{C}_i)$ will in general be much more difficult to find. We simplify matters by assuming that the individual features $f_i$ are independent of one another; this is known as the *Naive-Bayes assumption*. The conditional probabilities can then be rewritten as follows:

$$\Pr(F(x) = y | x \in \mathcal{C}_i) = \prod_{i=1}^{n} \Pr(f_i = y_i | x \in \mathcal{C}_i)$$

where $y_i$ is the $i$th component of the vector of feature values $y$.

The Naive-Bayes assumption makes it much easier to estimate the required conditional probabilities. However, as the name suggests, it is in general only a rough approximation to the real situation. In practical tasks, we may well find that feature values are dependent on one another; indeed, modelling such dependencies may be crucial for effective classification. Nevertheless, naive-bayes techniques have been used extensively and with considerable success in text classification; see McCallum and Nigam for a survey [MN98]. A more general overview of the naive-bayes assumption in machine learning is provided by Lewis [Lew98].

### 2.3.4 Maximum Entropy

Our second model architecture is the maximum-entropy scheme. Maximum-entropy classifiers are a variant of the minimum-divergence, maximum-entropy (MDME) family of models. A typical MDME model takes the form:

$$p_\lambda(v|u) = \frac{1}{Z(\lambda)}\; q(v)\; \exp\left(\sum_{i=1}^{n} \lambda_i f_i(u,v)\right)$$

where $v$ is an event occurring in a context denoted by $u$, $q(v)$ is a regularising probability distribution, $f_i$ is the $i$th feature function, $\lambda_i$ is a real-valued free parameter or *weight*, $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$, and $Z(\lambda) = \sum_v \exp \sum_i \lambda_i f_i(u,v)$ is a normalisation constant chosen to ensure that the probabilities sum to 1. (Della Pietra et al. establish that the MDME scheme can be extended in a rigorous fashion to allow the $\lambda_i$ to equal 'minus infinity'; this allows the model to assign an expected value of zero to particular features [DPDPL97].)

MDME models have a number of appealing theoretical properties, as described by Della Pietra, Berger et al. [BDPDP96, DPDPL97]. In particular, it can be shown that there is a unique value for the vector of weights which gives rise to a model $p^*$ satisfying the following criteria:

1. The expected value of each feature $f_i$ with respect to the model is the same as its expected value on the training set.

2. Among all models satisfying constraint (1), $p^*$ has the greatest conditional entropy. The conditional entropy is defined in Section 3.3.2, and can be thought of as a measure of the uncertainty inherent in a conditional probability distribution. In a sense then, a maximum-entropy model is the hypothesis consistent with our data which makes as few assumptions as possible, in accordance with the principle of Occam's Razor.

3. The model $p^*$ is the model in the parametric family $p_\lambda(v|u)$ which maximises the likelihood of the training data.

Efficient algorithms exist to find a close approximation to the optimal model $p^*$; see Della Pietra et al. and Malouf for details [DPDPL97, Mal02]. MDME models have

have been applied to a very wide variety of problems; for their application specifically to text-based natural language processing, see Berger et al., Rosenfeld and Ratnaparkhi [BDPDP96, Ros96, Rat96, Rat98].

The maximum-entropy (ME) models used in the experiments detailed in Chapters 5, 6 and 7 take the following form:

$$\begin{aligned} p_\lambda(x \in C_i | F^{(i)}(x) = y) &= \frac{1}{Z(\lambda)} \exp\left(\sum_{j=1}^{n} \lambda_j f_j^{(i)}(x)\right) \\ &= \frac{1}{Z(\lambda)} \exp\left(\lambda \cdot F^{(i)}(x)\right) \end{aligned}$$

Notice that the sum $\sum_{j=1}^{n} \lambda_j f_j^{(i)}(x)$ can be naturally expressed as the scalar product of the 'vectors' $\lambda$ and $F^{(i)}$. As usual, $x$ is a data point, $C_i$ is one of countably many categories, $y$ is a vector of feature values, and $Z(\lambda)$ is a normalising constant. In our chosen setting, the data points are words and the categories are part-of-speech tags. A given feature $f_j^{(i)}$ takes the value 1 if the $j$th indicator (in our setting, a substring) is present on a data point which falls into the $i$th category, and 0 otherwise. These category-dependent features are used together with labelled training data to optimise the values of the parameters $\lambda_j$. The model can attempt to classify unlabelled data by assuming that all features $f_j^{(i)}$ are active in any word containing the $j$th substring, and assigning the category with the highest probability $p_\lambda(x \in C_i | F^{(i)}(x) = y)$.

**The Forbidden Vector Problem:** Notice that, in order for the probabilities $G_\lambda(F(x))$ in an ME model to sum to 1, any 'forbidden' combinations of feature values $y$ (that is, those that do not correspond to any data point $x$) must satisfy $G_\lambda(y) = 0$. In some cases our model may not be flexible enough to arbitrarily assign zero probability to particular combinations of features.

This can be illustrated as follows: Suppose that, as in Example 4 above, our data points are English words and we have binary features whose indicators are particular substrings. We then incorporate those features into an MDME model – for instance, a conditional model that assigns probabilities to a word's part-of-speech tags given the features active on the word.[1]

[1]This is the setting for the extensive experimental investigation of feature selection methods in Chap-

Recall that in MDME models, assigning a weight of ‘minus infinity’ to a feature means that any word containing this feature has probability zero. In this instance, we may not want the binary features ing], [z, ja, and (where the left and right square brackets are the beginning- and end-of-word markers, respectively) to have weights of minus infinity, but intuitively we would like the non-word ‘zjaing’ to have zero (or at least very low) probability. It may in fact be desirable to ensure that all possible combinations of feature values receive non-zero probabilities. In the case of English words, combinations of letters never before seen may well turn up in recorded utterances – as new words or perhaps as abbreviations. (They need not even be pronounceable, as the adoption of VCR and DVD in English demonstrates.)

For practical purposes, though, we would like combinations of feature values that never occur in our available data to receive extremely low probabilities. Unfortunately, arbitrarily giving ‘zjaing’ very low or zero probability cannot be achieved except by defining more features – such as [zja, which would be active on words beginning with the three letters ‘zja’ – and assigning them negative weights of large (or infinite) magnitude. There are of course infinitely many combinations of commonly occurring substrings which do not correspond to English words, so the ‘extra features’ strategy will not get us far. We must instead choose our features carefully, so as to avoid giving too much probability mass to feature combinations which do not occur in our data set. Solving this ‘forbidden vector problem’ may present a significant challenge in practical feature selection tasks. It is similar to the challenge of *smoothing* a model, in that we must decide how much probability mass to assign to unseen data points.

## 2.4 Binary Features

### 2.4.1 Definitions

The set of possible feature values $Y$ is typically quite simple. In particular, it is common for $Y$ to be the binary set $\{0,1\}$. A feature $f$ which maps data points $x$ to the binary set is said to be a *binary feature*. The feature then takes the value 1 on a given data point if that data point has a given characteristic, in which case the feature is said to be

ter 4.

*active*. Otherwise the feature takes the value 0 and is said to be *inactive*. (Obviously, we consider 0 to be the null value.) We refer to the quality whose presence or absence is used to decide whether a feature is active as the feature's *indicator*.

**Example 2.5:** Suppose that our data points are English words. We might define a feature $f_1$ which is active on words containing the three-letter substring `gre`. For instance, we have $f_1(green) = 1$ and $f_1(ideas) = 0$. The indicator is simply the presence of the string `gre`.

If we additionally define a feature $f_2$ which is active on words containing the substring `de`, then $F(\text{green}) = (1,0)$ and $F(\text{ideas}) = (0,1)$. $\square$

In the case of binary features, it is common to abuse terminology slightly by using the word 'feature' to refer to both the feature function itself and its indicator. Using the above example, it is natural to refer to the three-letter substring `gre` as a 'feature' of the word 'green'. It should be emphasised that the indicator is *not* a function in its own right; it is simply an identifiable characteristic of a data point. In this instance, our indicator is the substring `gre`. The substring is distinct from $f_1$, which is a function mapping words to the binary set.

It is sometimes useful to adopt other feature image sets which have only two values, for instance 'positive' and 'negative.' Such features are of course very closely related to true binary features (those whose image set is $\{0,1\}$). Hence, we will sometimes use the term 'binary feature' in a slightly looser sense to mean any feature whose image set contains only two elements. Although binary features are particularly common, it should be noted that for most model architectures there are very few *a priori* restrictions on the feature image set $Y$, and other image sets are entirely possible.

### 2.4.2 Approximation of Multi-valued Features

It is important to note that combinations of binary features can be used to mimic the effect of more complicated features. This can be very useful, as binary features are particularly simple to deal with.

If a feature takes finitely many values, our model architecture may permit us to reproduce its behaviour perfectly. This requires assigning one binary feature to each possible value of the original, multi-valued feature function. Suppose that our original function $f(x)$ takes one of $m$ different values $\{y_1, y_2, \ldots, y_m\}$. If $f$ is well-defined, then for each possible input $x$ there will be exactly one output $y_k = f(x)$. Hence, we can define subsets $X_k$ of the data space $X$ as follows: A given data point $x$ is a member of $X_k$ if and only if $f(x) = y_k$. Each data point $x$ then falls into one and only one subset $X_k$.[2] Let us also suppose that there is a null value, which we denote by $y_0$.

For each possible output $y_k$, then, we define a new feature $f_k(x)$:

$$f_k(x) = \begin{cases} f(x) = y_k & \text{if } x \in X_k \\ y_0 & \text{otherwise} \end{cases}$$

Some care is needed here, as the collection of binary features may not behave in exactly the same way as the original multi-valued feature. Whether we can duplicate the original feature exactly depends on the nature of our model architecture. In ME models, though, this scheme can work very well. Let the initial multi-valued function be denoted by $f(x)$ and its weight by $\gamma$. For each $x$, we are replacing the initial feature/weight pair:

$$\gamma f(x)$$

with the same weight and a sum of binary features:

$$\gamma \left( \sum_{k=1}^{m} f_k(x) \right)$$

In the MDME case, our non-informative value is 0. Thus, for any given $x$, all but one of the binary features $f_k$ is equal to zero and the sum takes the same value as the initial function. The $f_k$ are not 'true' binary features; each $f_k$ takes the values 0 and $y_k$, not 0 and 1. Since the $y_k$ are real numbers, we could of course replace each $f_k$ with a true binary feature $f'_k$, multiplied by the fixed real number $y_k$. The sum of binary features is then:

[2] A feature thus has the effect of *partitioning* the data space $X$ – it divides it into disjoint subsets $X_k$ whose union is the whole of $X$. (It can be shown that this happens even if the feature image set $Y$ is infi nite.) The feature set as a whole has a similar partitioning effect, which will be examined in depth in Chapter 4.

$$\begin{aligned}\gamma f(x) &= \gamma\left(\sum_{k=1}^{m} y_k f'_k(x)\right)\\ &= \sum_{k=1}^{m} \gamma y_k f'_k(x)\end{aligned}$$

Notice that, while the overall weight $\gamma$ may be allowed to vary, the individual ‘weights’ $y_k$ must remain fixed for the sum to be equal to the original feature function. Unless we are very careful, our training algorithm may view the quantities $\gamma y_j$ and $\gamma y_k$ (where $j \neq k$) as parameters which may be varied independently, irrespective of the need to keep $y_j$ and $y_k$ fixed. The process of training may thus distort our binary sum so that it is no longer an accurate duplication of the original, multi-valued feature function.

In other cases, it may be possible to use binary features to adequately approximate a feature function with an infinitely large image set. Continuing the previous example, we might divide the range of possible document lengths into a number of intervals and assign one binary feature to each interval; we would then have indicators such as ‘between 100 and 200 words’ or ‘more than 1000 words’. Obviously, we cannot perfectly duplicate a feature that takes infinitely many values using finitely many binary features; but we may be able to form a satisfactory approximation.

The above discussion helps illustrate that combinations of features can be powerful tools for modelling data, but must be treated with care as the features may interact in ways which are difficult to predict.

## 2.5 Summary

In this chapter we have discussed the need for, and advantages of, creating precise definitions of features and the models which may incorporate them. We have continued by presenting suitable definitions of *feature functions* and discussing their properties; and by formally defining *feature-based probabilistic models*. Finally, we have examined the commonly occurring special case of *binary features*, indicating how binary features relate to the more general setting of feature functions.

Chapter 3 will survey the existing literature on feature selection, and where appropriate will discuss it in terms of the theoretical framework developed in Chapter 2.

# Chapter 3

# Literature Survey

## 3.1 Introduction

In Chapter 2 we established formal definitions for features and their relationship to feature-based probabilistic models (FBPMs). Recall that a *model architecture* is a scheme for incorporating a set of feature functions into a probabilistic model; and that most model architectures place very few restrictions on the feature set. It is therefore natural to ask what constitutes a good set of features, and how we may find such a set.

In this chapter we carry out a survey of existing feature selection literature. We begin by defining the feature selection problem and describing a general classification system for feature selection algorithms, based on the one established by Dash and Liu and later updated by Liu and Yu [DL97, LY02]. We then examine some commonly employed feature selection techniques, concentrating on methods derived from information theory: Information gain, the Koller-Sahami criterion, and mutual-information measures.

Throughout this chapter, we will make use of the terminology presented in Chapter 2 where appropriate. In particular, many of our examples will employ the *binary features* defined in Section 2.4. Recall that a binary feature function takes the value 1 if its indicator is present for a given data point, and 0 if the indicator is absent. Many studies, such as that by Yang and Pedersen, restrict themselves to the binary case and use the word ‘feature’ for both the function and its indicator [YP97]. However, these are really two different things; an indicator is some identifiable characteristic of a data point, whereas a feature function is a mapping from the data space to the binary set. In this survey, we follow the more general approach of Blum and Langley and John et al., which allows the extension of our ideas to non-binary features [JKP94, BL97].

A number of general surveys of feature selection have been carried out. Studies by John et al., Blum and Langley, and Dash and Liu have already been mentioned; Yang and Pedersen provide an overview of feature selection for the specific area of text categorisation [JKP94, BL97, DL97, YP97]. An earlier examination of feature selection by Langley is also of interest [Lan94]. In addition, a useful survey of recent developments in feature selection – including application to domains with tens or hundreds of thousands of features – has been carried out by Guyon and Elisseeff [GE03].

## 3.2 General Concepts

The setting of the general feature selection problem is as follows: We assume that we have specified our data space $X$ and a particular model architecture, and that we possess a set of training data $\tilde{X}$. We also have a large and possibly infinite pool of available feature functions, which we denote by $\mathcal{F}$.

**Example 3.1:** Suppose that we are attempting to construct an FBPM for classifying Web pages. Our data space $X$ is the infinite set of all possible Web pages; the training set $\tilde{X}$ is a finite collection of pages, each labelled by subject. Now suppose that we wish to use binary features, which take the value 1 if a particular word or phrase is present in the page and 0 if it is absent. Our pool of possible features $\mathcal{F}$ may contain one feature function for each possible word or phrase which might be encountered in a Web page; so in principle, $\mathcal{F}$ is astronomically large. Naturally, for practical purposes we can only incorporate a smaller subset of these features into our model. □

Even if it is possible to incorporate all available features into the model, we may not wish to do so. The reasons for this may include a desire for greater computational efficiency; for greater accuracy of classification, and in particular the avoidance of overfitting; for a smaller and more easily understood feature set; or to exclude certain features which are known to be irrelevant or misleading. Choosing a good set of features will give rise to a simpler, faster and more accurate model.

We therefore need a method for selecting a smaller feature set $F$, a subset of the pool of available features $\mathcal{F}$. (For this reason, the problem of feature selection is sometimes known as feature *subset* selection.) We would like our chosen feature subset to be in some sense optimal; this usually requires a quantitative definition of how 'good' a given feature subset is.

Furthermore, there are typically a great many subsets to choose from. An infinite set will of course have infinitely many, and even a finite set with $a$ elements has $2^a$ distinct subsets. With a very modest pool of 30 possible features, we have in excess of $10^9$ distinct subsets; and it is common for the available features themselves to number in the thousands or millions. In general then, it is impossible to carry out an exhaustive assessment of the possible feature subsets, and we need a more sophisticated strategy for finding a good one.

Dash and Liu identify four distinct elements of a typical feature selection algorithm [DL97]:

1. **Generation procedure:** A means of generating candidate feature subsets.
2. **Evaluation function:** A function which produces a numerical 'score' for each candidate feature subset.
3. **Stopping criterion:** A means of deciding when to terminate the search.
4. **Validation procedure:** Any feature selection method must be *validated* by assessing the performance of the FBPM which arises from it.

The process of feature selection essentially consists of repeatedly generating a feature subset and assessing it using our evaluation function. If our stopping criterion is met by the current feature subset, then we halt our search; otherwise, we continue to generate and assess new feature subsets. Taken together, the first three items can be thought of as a strategy for navigating through the immense space of feature functions. Our choice of each one of these three will be influenced by our choices for the other two, and perhaps by our chosen model architecture as well. Validation of the model is not part of the feature selection process as such, but it is an essential component of any practical use of feature selection. We now consider these four elements in greater detail.

## 3.2.1 Generation Procedures

The generation procedure is a means of generating subsets of our pool of available features. It is worth stressing that the *generation* of feature subsets is distinct from their *evaluation*. In general the choice of a particular generation procedure does not tie us to a specific evaluation method, and vice versa. Failure to keep this distinction in mind can result in significant confusion, as noted in the discussion of work by Acuna in Section 3.2.2.1 and by Koller and Sahami in Section 3.3.4 [Acu03, KS96].

Blum and Langley note that a generation procedure requires us to specify two things: A starting point in the space of feature subsets, and a means of organising a search through the space [BL97].

**Starting Point:** We could begin with the empty feature set; at the other extreme, if our pool of candidate features $\mathcal{F}$ is finite, we could start with the set of all available features. Searches which begin with the empty set and iteratively add features are known as *forward selection*; those which begin with the maximal feature set and iteratively remove features are known as *backward elimination*. These two simple techniques can prove very effective in practice, and are quite popular in the literature; see Dash and Liu or Blum and Langley for details [DL97, BL97]. An empirical comparison between the two methods is carried out by Aha and Bankert [AB96]. They conclude that, as one might expect, forward selection is superior when the optimal number of features is small while backward elimination is more effective in selecting large feature sets.[1] However, their study is carried out in a comparatively restricted domain and is far from definitive.

Another possibility is to begin with a feature subset of some intermediate size, chosen randomly or by some other appropriate method. Such a subset may have been sampled uniformly from the collection of available features $\mathcal{F}$ (that is, all features in

[1]'Small' and 'large' are generally defined with respect to the total number of available features.

$\mathcal{F}$ are equally likely to be included in our initial subset); or we may introduce a bias into the sampling, in order to favour features which we expect to be useful on some *a priori* grounds.

**Example 3.2:** Consider an FBPM which classifies English words by their part-of-speech tag. Each feature is defined by a particular substring; it takes the value 1 on a word in which the substring is present, and 0 on a word in which it is absent.[2] We might wish to select our initial feature set at random, subject to the constraint that at least one feature in our initial subset should take the value 1 on each word in the training set. □

A more elaborate approach is to start with *multiple* feature subsets. This is the setting of the canonical genetic algorithm; see Goldberg for details [Gol89]. In a genetic search algorithm we apply a 'survival of the fittest' strategy: Feature subsets which are particularly good, with respect to our chosen evaluation function, are used to form the next generation of subsets. The new generation contains subsets made up of features from two or more of the 'fittest' sets from the preceding generation; it may also contain exact copies of the best previous feature sets.

Genetic algorithms have been applied to several feature selection tasks [YH98, ILS01, SYBL02]. Although they have produced effective results, they have the obvious drawback of requiring us to assess a very large number of candidate feature subsets. If our evaluation function is difficult to compute, search by a genetic algorithm may be impractical. This problem may be eased somewhat if the assessment of different subsets can be carried out in parallel on different machines or processors. Vafaie and De Jong report that search by a genetic algorithm can significantly reduce the chance of a sequential search (in this case, backward elimination) becoming 'stuck' in a less than optimal region, with minimal decrease in computational efficiency [VD93, VD95]. However, their experiments are carried out with a pool of only 30 candidate features; scaling genetic algorithms up to domains with tens of thousands of features – such as the setting for the experiments in this thesis – presents considerable difficulty.

[2]We are therefore using *binary features*; see Section 2.4.

**Search Strategy:** In addition to choosing a starting point, we must specify a strategy for adding and/or removing features with the aim of improving the initial subset. Generally speaking, the basic idea is as follows: We consider a number of local changes to the existing subset; adopt the best one (with respect to the evaluation function); and iterate until our stopping criteria are satisfied.

The simplest technique is to carry out a greedy search. If we are using forward selection, then at each step we assess all remaining candidates for addition to our existing feature set, and add the one which gives rise to the highest-scoring new set. With backward elimination we consider all features in our existing set and remove one, again with the aim of maximising the score of the new set.

Several variations on the simple, greedy search have been employed. One obvious variant is to allow our search to 'backtrack', by both adding and removing features. For example, in the forward selection case we might add $k$ features and then remove one. This would allow our search to discard features which initially seem promising, but later interact with other features in undesirable ways. Another possibility is to consider all possible features which may be added to or removed from our existing set, and add or remove the feature which gives rise to the best-scoring new set; this is known as a *stepwise* search [BL97].

We can also add or remove more than one feature at a time. This is especially useful for very large feature sets, where selecting features one at a time can be prohibitively slow. In Chapter 5, we introduce a new technique which we refer to as *accelerated* forward selection. This method obtains good results with sets of between 10000 and 40000 feature functions by randomly selecting several blocks of features, each roughly 1% the size of our existing feature subset, and adding the one which brings about the greatest improvement in our evaluation function; it will be discussed in greater depth in Section 5.5.2.[3]

Regardless of how the local changes are generated, there are two distinct ways of deciding how many to evaluate. One is to assess a fixed number of possible changes

[3]A much more extreme randomising approach is the 'Las Vegas' method employed by Liu and Setiono; at each step, a new feature subset is chosen *entirely* at random, compared to the existing one, and adopted if it is found to be superior [LS98]. This is somewhat more effi cient than an exhaustive search, but needless to say it is a poor way of navigating through very large spaces of feature subsets.

(possibly all of them) and choose the best one. Alternatively, we can simply generate and assess local changes one by one, and adopt the first one which gives rise to an improvement over the current feature set. The latter technique sacrifices a thorough investigation of possible improvements at each step in favour of speed. It also presupposes that we *expect* an improvement at each step, which is not necessarily the case; for instance, in backward elimination we may wish to force our algorithm to discard a feature at each step, so as to arrive at a smaller and more efficient feature set.

The generation procedure we choose to adopt will depend on a number of factors. The most crucial tend to be the size of our pool of available features; the expected size of our final feature subset relative to the collection of possible features; and the difficulty of computing our evaluation function.

### 3.2.2 Evaluation Functions

#### 3.2.2.1 Wrappers versus Filters

An evaluation function is a quantitative measure which enables us to assess the desirability of a given feature subset. It is typically a function which maps a given feature subset $F \subset \mathcal{F}$ to the non-negative real numbers:

$$\mu(F) : 2^{\mathcal{F}} \mapsto [0, \infty)$$

where $2^{\mathcal{F}}$ denotes the set of subsets of $\mathcal{F}$.[4] The evaluation function may also map $2^{\mathcal{F}}$ to some subset of $[0, \infty)$. Some evaluation functions are maximised for optimal feature sets; others are minimised. We must of course be clear and consistent about what our objective is. However, whether we are maximising a measure of ‘goodness’ or minimising a measure of ‘badness’ is not of much general significance. (Maximising $\mu$ is equivalent for most practical purposes to minimising $1/\mu$.) A more important distinction is the one between *wrapper* and *filter* evaluation functions, which we describe below.

**Example 3.3:** Suppose that each data point $x$ falls into exactly one of countably many categories $\mathcal{C}_i$. Assume that we have access to a collection of correctly labelled

[4]The notation is indicative of the fact that, if $\mathcal{F}$ is fi nite with $a$ elements, it has $2^a$ distinct subsets.

data points. We divide the collection into a *training set* which will be used to assign values to any free parameters in our model, and a *test set* which will be used to assess the performance of the model.

Now suppose that a given feature set $F$ gives rise to a classifier:

$$C_\lambda(x|F)$$

Having chosen a feature set $F$, we first set the free parameters $\lambda$ using the training set and an appropriate training algorithm. The classifier then assigns each data point $x$ to a particular category $\mathcal{C}_i$.

We define an evaluation function $w(F)$ to be the fraction of the test set which is correctly labelled by the trained classifier $C_\lambda(x|F)$. (Equivalently, $w(F)$ is the probability that a data point, randomly sampled from the test set according to the uniform distribution, will be correctly classified.) The evaluation function $w(F)$ will therefore be a real number between 0 and 1. If $w(F) = 0$ then our classifier does not correctly classify any point in the test set; if $w(F) = 1$ then it correctly classifies them all. Obviously, we wish to maximise the value of $w(F)$. $\square$

The evaluation function $w(F)$ is an example of a *wrapper* scheme. In a wrapper, the trained model itself serves as our method of assessing a candidate feature set; the feature selection is ‘wrapped’ in the process of training and assessing a feature-based probabilistic model. The more accurate the model, the better the feature set is deemed to be. Wrappers were first presented by John, Kohavi and Pfleger [JKP94]. They have been further developed and applied by John, Kohavi and many others [KJ97, KS95, KJ98, LS6a, SH98, LS94, Ska94].

Conversely, a *filter* method is independent of our model architecture; it attempts to ‘filter’ out irrelevant and redundant features before we attempt to train the model.

**Example 3.4:** A very simple example of a filter measure is *frequency-based cutoff* (FBC). When applied to binary features, frequency-based cutoff prefers features which frequently take the value 1 on the training data. The ‘score’ of a particular feature is simply the number of times it takes the value 1 in the training set; the score of a feature subset is the sum of the scores of its individual features.[5] $\square$

[5] If extended to features which can take more than two values, FBC prefers those which do not often

Examples of more sophisticated filters are discussed in Sections 3.2.2.3, 3.2.2.4, and 3.3. An interesting empirical study of filter evaluation metrics applied to text categorisation, including information-gain and frequency-based cutoff measures, has been carried out by Forman [For03].

An experimental comparison of filter and wrapper procedures has been carried out by Acuna [Acu03]. Regrettably, Acuna does not fully control for generation procedures. Only in the case of forward selection is there a direct comparison between filter and wrapper metrics operating under the same generation procedure; the fully randomised 'Las Vegas' generation procedure of Liu and Setiono[6] is evaluated for a filter measure only, whereas backward elimination is carried out for a wrapper only. A more systematic comparison between wrapper and filter methods of feature selection in combination with different generation techniques was conducted by Aha and Bankert [AB96]. Investigations such as these two, along with the detailed study of wrapper methods by Kohavi and Joun and broader surveys of feature selection carried out by Dash, Liu, Blum, Langley and others, allow us to draw some general conclusions about the differences between wrappers and filters [KJ98, DL97, BL97, LY02].

Wrapper methods consistently give rise to very accurate models, which is to be expected if the criterion used to assess the accuracy of the model is the same as the one used to select its features. However, the good performance of a wrapper in selecting features for its own learner will not necessarily carry over to other model architectures. This was the case in the experiments described in Chapter 5; the implications for feature selection are discussed further in Section 6.8.

Wrappers also tend to be rather slow and computationally expensive, as a model must be trained and evaluated for every feature set we wish to assess. Furthermore, each wrapper is by its nature tied to a particular model architecture. This means that few general lessons can be drawn from a wrapper; it does not significantly improve our understanding of feature selection in other domains, and often gives little qualitative indication of *why* one feature set performs better than another.

Filter methods are usually faster than a wrapper, and tend to be more broadly applicable; they may be able to partially compensate for any weaknesses in our model

take their null value. (Null values are defi ned in Section 2.2.3.)

[6]Las Vegas methods were fi rst presented in [LS98].

architecture; they often have a theoretical foundation which can help us understand why some features are better than others; but they frequently give rise to less accurate models. A wide variety of filter measures have appeared in the literature. As Dash and Liu note, filters fall into four general categories: Information measures, distance measures, correlation or dependence measures, and consistency measures [DL97]. For reasons described in Section 3.2.2.3, we will concentrate on information measures and consider the others only briefly. Before considering further examples of information metrics, we briefly discuss the distinction between *hierarchical* and *non-hierarchical* filters.

#### 3.2.2.2 Hierarchical versus Non-Hierarchical

Filter metrics for feature selection can be divided into *hierarchical* and *non-hierarchical* measures. (As we shall see, all wrapper metrics are non-hierarchical.) A *hierarchical* metric is one which places features in a fixed order of desirability, regardless of how they are combined in a feature subset. Hence, a hierarchical metric cannot effectively consider interactions between features. The frequency-based cutoff measure described in Section 3.2.2.1 is an example of a hierarchical measure; another is the naive mutual-information metric described in Section 3.3.5.

Conversely, a non-hierarchical metric takes account of how features are combined. Rather than assessing feature functions in isolation, it evaluates a feature subset as a whole. In many cases this is accomplished by assigning scores to possible combinations of feature values, as with the metrics discussed in Section 3.2.2.3 and Chapter 4. Implicitly or explicitly, a non-hierarchical metric is capable of considering the influence of features upon each other.

Wrappers also effectively consider interactions between features. Interestingly, as we discuss in Chapter 7, this holds true even for the naive-bayes model architecture – which formally assumes that features are independent of one another. If there are dependencies between features – as there almost always will be – then taking account of such dependencies gives rise to more accurate classification, in spite of the independence assumption incorporated in the model.

Hierarchical metrics in general give rise to much more rapid feature selection than non-hierarchical ones. It is usually quite easy to evaluate a hierarchical metric for every member of our collection of available features; then, the most desirable subset of *n* features is simply the one containing the *n* features with the highest individual scores. Conversely, it is almost never possible to evaluate a non-hierarchical metric for every possible feature subset, and we must resort to heuristic searches which cannot be guaranteed to find a global maximum for the metric. However, non-hierarchical metrics do have the advantage of being able to consider interactions between features. In the experiments detailed in Chapter 5, non-hierarchical metrics gave rise to considerably greater accuracy than their hierarchical counterparts, even with relatively crude search techniques.

#### 3.2.2.3 Information Measures

Information measures are based on *entropy* and related concepts drawn from information theory, such as Kullback-Liebler divergence and mutual information. (See Cover and Thomas for definitions of these quantities [CT91].) Such measures are very popular in the literature, and they have demonstrated significant practical success in a variety of applications. They are also theoretically appealing, due to their foundations in the well-developed field of information theory – which is itself closely connected to probability theory. Specifically, information measures can be seen as measuring the distances between probability distributions. (Exactly where these distributions come from will be discussed in detail in Section 3.3.1.) Because of their sound theoretical background, information measures are typically very general; they can be applied to wide classes of problems, instead of being closely tied to a particular model architecture.

The new evaluation functions presented in Chapter 4 and experimentally investigated in Chapters 5, 6, and 7 are based on information theory. Later in this chapter, Section 3.3 will consider existing information measures in greater depth.

#### 3.2.2.4 Other Types of Filter

**Distance Measures:** These are similar to information measures, but they do not define the distance between probability distributions in terms of information theory. For instance, they might use measures based on Euclidean distance or the $L^p$ distances of real analysis.[7]

**Correlation Measures:** Also known as dependence measures, these measure our ability to predict the value of one variable from the value of another. For instance, one might examine the correlation between particular binary features and a class we are interested in; if the event $f_i(x) = 1$ is more strongly correlated than with class $C$ than the event $f_j = 1$, then $f_i$ is preferred to $f_j$.

Dash and Liu note that dependence measures can themselves be divided between distance and information measures [DL97]. However, they argue that dependence measures should be given their own category, as they represent a distinct way of thinking about feature selection.

**Consistency Measures:** These make use of the Min-Features bias defined by Almuallim and Dietterich [AD91]. Consistency measures seek consistent hypotheses about the training data, using as few features as possible.

### 3.2.3 Stopping Criteria

Recall that a feature selection algorithm begins by choosing a starting point somewhere in the space of feature subsets. It then attempts to navigate through that space, guided by an evaluation function which gives a quantitative definition of how ‘good’ a given feature subset is. Naturally, we need some method of deciding when to terminate our search. Possibilities include:

**Number of Iterations:** We may simply halt our feature selection process after a predetermined number of steps.

**Subset Size:** Stop when our feature subset reaches a particular fixed size. This is particularly appropriate for forward selection or backward elimination.

[7] See Priestley for defi nitions and discussion of $L^p$ distances [Pri97].

**Optimum Subset:** In some domains we can reasonably expect to find a feature subset which is 'perfect' with respect to our evaluation function; it is natural to halt when this is achieved. For example, a wrapper method with a small test set may be able to correctly classify every member of the test set. In many cases though, it will be unlikely or impossible for our evaluation function to reach its optimum value within a reasonable amount of time.

**No Visible Improvement:** Stop when none of the local changes we have surveyed is better than our existing feature set. We may wish to generate some additional local changes before halting the feature selection process. For instance, if we cannot obtain a better subset by adding one feature, we might also consider adding pairs of features before we halt the process. Another slight variation is to stop when the improvement falls below a certain threshold; for instance, the Expected Partition Entropy metric presented in Chapter 4 can be expected to asymptotically approach zero for medium-sized feature sets, but is unlikely to attain it except for very large ones.

Notice that the first two measures are independent of our evaluation function, while the others are not. We may wish to combine more than one stopping criterion. For instance, we might halt when no improvement is visible or when our feature subset reaches a certain size, whichever comes first.

### 3.2.4 Validation

A validation process tests the success of an FBPM which arises from a given feature set, usually with reference to its accuracy on unseen test data. It is not part of the feature selection process itself, but is an important guide to whether the process has been successful. Wrapper methods already perform this to some extent. Another important aspect of validation is contrasting the effectiveness of our FBPM with that of other models for the same phenomenon, perhaps ones obtained using other feature selection methods.

## 3.3 Examples of Information Measures

In this section, we describe *information gain*, the *Koller-Sahami filter*, and *mutual information*, three popular examples of evaluation functions based on information theory. In order to do so, we first need to define *approximate distributions*. We then introduce the key concepts of *entropy*, *conditional entropy*, *relative entropy*, and *mutual information*, and discuss the evaluation functions which arise from them.

### 3.3.1 Background – Approximate Distributions

It is important to note that information theory, and the measures derived from it, fundamentally involve properties of and relationships between probability distributions. In order to employ an information measure, we need some idea of the probability distribution governing the data points and their associated feature values.

In practice of course, we do not have access to the 'true' distribution which is assumed to govern the phenomenon that we are modelling. Instead we have a set of training data $\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, \ldots \tilde{x}_N\}$; our training data points are assumed to be independently sampled from an identical distribution. We can use this data to form a relatively crude approximation to the 'true' distribution $p(x)$, which in turn can help us select features to form what should be a better model. (Some other methods of feature selection also require an approximate distribution; see Section 3.2.2.4.)

It is customary to use the *empirical distribution* of data points as our approximation. Indeed, it is so conventional that many authors treat the two terms as synonymous – it is simply assumed that any probabilities required for an information measure are derived from the empirical distribution. As we shall see though, other approximations are possible.

The empirical probabilities are defined as follows: Suppose that our training set $\tilde{X}$ consists of $N$ points $\tilde{x}_1, \ldots, \tilde{x}_N$. Points in the training set are not necessarily unique; that is, we may have $\tilde{x}_i = \tilde{x}_j$ for $i \neq j$. For example, if our data points are English words and the training data consists of all the words in a given sample of text, many words will occur more than once. Let $c(x)$ be a 'count function' that returns the number of times the distinct data point $x$ occurs in the training data. We now define the empirical data distribution $\tilde{p}(x)$:

$$\tilde{p}(x) = \frac{c(x)}{N}$$

In other words, the empirical probability of a particular data point is the number of times it occurs in the training set, divided by the total size of the training set.

Notice that we can similarly define an empirical distribution of feature vectors:

$$\tilde{q}(y) = \frac{c(y)}{N}$$

Each data point $x$ will have a vector of feature values $y = F(x)$ assigned to it by our feature set; if we now let $c(y)$ be the number of times that a given feature vector $y$ appears in the training set, the empirical probabilities of feature vectors are defined in exactly the same way.

In classification tasks, it is often necessary to use more complicated distributions than the simple empirical probabilities. If we are principally interested in the conditional probability $\Pr(x \in \mathcal{C}_i | F(x) = y)$, where $F$ is a feature set and $\mathcal{C}_i$ is one of the possible classifications for the data point $x$, then the simple empirical probabilities $\tilde{p}(x)$ are not immediately of use to us; but they can be used to derive distributions which are. For instance, see the discussion of information gain in Section 3.3.3.2 and by Yang and Pedersen [YP97]. In the conditional case, information gain requires us to compute such probabilities as $\Pr(\mathcal{C}_i)$ (the probability that a randomly sampled data point will fall into the $i$th category) and the conditional probability $\Pr(\mathcal{C}_i | F_k(x) = y)$ (where $F_k(x)$ is the feature set obtained at the $k$th stage of a forward selection algorithm). Approximations to these probabilities can be derived from the simple empirical probabilities – assuming, of course, that we know the category of each data point in our training set.

There has been little investigation of approximate distributions other than the empirical distribution, with the significant exception of the work of Zaffalon and Hutter [ZH02]. They note that the empirical distribution does not carry information about the reliability of the training data; Bayesian techniques are employed to address this problem by adopting a suitable prior distribution, and using the posterior distribution as their approximation. Good results are obtained by combining this more sophisticated approximation with a filter based on mutual information.[8] An interesting possi-

[8]Mutual information is defi ned in Section 3.3.5.

ble topic for future research is the use of the Zaffalon-Hutter approximation with other filter measures.

Approximations defined in other ways are theoretically possible; indeed, we could use *any* computable distribution over the data space. One possible alternative to the empirical distribution is the trained model. Using the trained model as an approximation would give rise to a hybrid filter/wrapper method of feature selection. Essentially, we train the model with a feature set which is known to be less than optimal. Such a feature set could be the full feature set; a randomly chosen subset; or a subset chosen according to some other criterion, perhaps the one from the previous step in our feature selection process. We then use this sub-optimal model as the approximation to the 'true' distribution which is needed in order to employ a particular filter method.

Hybrid methods of this type have not been extensively investigated, although a similar idea of combining a filter and wrapper is used by Sebban and Nock [SN02]. (Indeed, a filter method might be combined with a wrapper in other ways; for instance, we could carry out forward selection using a quick filter measure, while periodically stopping and removing a few features using a slower wrapper measure.) A more thorough examination of filter-wrapper combinations is a possible topic for future research.

In practice though, the empirical distribution is generally an adequate approximation to the 'true' distribution. It has the advantages of simplicity, clarity, and ease of computation. Employing a more elaborate approximation would result in a significant decrease in speed, without necessarily producing a significant increase in accuracy.

## 3.3.2 Entropy

Entropy is in a sense the fundamental concept in information theory; more elaborate ideas such as information gain arise directly from it. The entropy of a probability distribution $p(u)$ is defined as:

$$H(p) = -\sum_u p(u) \log p(u)$$

It is conventional to take logs to base 2, but the choice of base is not crucial; changing from one base to another is equivalent to multiplying the entropies by a constant. See Cover and Thomas for further details [CT91]. For simplicity, we usually omit the log base from our notation. $0 \log 0$ is conventionally taken to be zero, on the grounds that $u \log u \to 0$ as $u \to 0$. Hence, adding terms with probability zero does not change the entropy.

Entropy may be thought of as the uncertainty inherent in a given probability distribution. More specifically, if we sample a point according to a given probability distribution, the entropy represents our degree of certainty as to which point will appear. For example, a distribution which assigns probability 1 to a single data point and 0 to all others has an entropy of zero. Conversely, the uniform distribution over $n$ data points – which assigns a probability of $1/n$ to each point – has an entropy of $\log n$. The more data points in a uniform distribution, the less certain we are which one will be sampled, and the greater the entropy. Intuitively, the outcome of a fair coin toss is in some sense ‘more certain’ than the roll of a fair die, because fewer outcomes are possible.

Another important quantity is the *conditional entropy*. For two random variables $U$ and $V$ with joint distribution $p(u,v)$, it is defined as:

$$\begin{aligned} H(V|U) &= \sum_u p(u) H(V|U=u) \\ &= -\sum_u \sum_v p(u,v) \log p(u|v) \\ &= -E_{p(u,v)} \log p(U|V) \end{aligned}$$

where as usual $E_p$ denotes an expectation over $p$. Again, see Cover and Thomas for further details [CT91]. Chapter 4 defines new methods for feature selection based on conditional entropy.

## 3.3.3 Relative Entropy and Information Gain

### 3.3.3.1 Relative Entropy

Information gain evaluation functions are based on the concept of *relative entropy*, which is an information-theoretic distance between probability distributions. Relative entropy is also known as cross-entropy, or as Kullback-Liebler (or K-L) distance (or divergence). For two probability mass functions $p(u)$ and $q(u)$, it is defined as:

$$D(p||q) = \sum_{u \in U} p(u) \log \frac{p(u)}{q(u)}$$

As usual, $U$ is the set of all possible $u$. By convention $0 \log \frac{0}{q} = 0$ and $p \log \frac{p}{0} = \infty$, based again on continuity arguments. It can be shown that relative entropy is always non-negative, and equals zero if and only if $p = q$. It is not symmetric (in that $D(p||q)$ may not equal $D(q||p)$) and does not satisfy the triangle inequality, so is not a true metric in the same sense as Euclidean distance; nevertheless, it is useful to think of it as a 'distance' between probability distributions. $D(p||q)$ can be thought of as measuring the error that arises through approximating $p(z)$ by $q(z)$; hence, $p(x)$ is generally taken to be the 'more informative' distribution. (See Cover and Thomas for proofs and discussion of the properties of relative entropy [CT91].)

Equipped with this distance and a suitable approximate distribution,[9] we can formulate a number of feature selection measures. We denote the approximate distribution over feature vectors which arises from a given feature set $F$ by $q_F(y)$. In addition, we must choose a suitable 'reference' distribution over feature vectors $y$.

Feature selection by *information gain* involves maximising the relative entropy from some less informative reference distribution to $q_F(y)$. The concept of entropy allows us to precisely define 'less informative': The greater the entropy of a distribution, the less information it contains. There are two natural choices of reference point for information gain. One of them gives rise to the commonly employed metric which we call *local information gain*; it is often referred to in the literature as simply 'information gain.' The other is discussed briefly in Section 3.3.3.3 below; it ultimately gives rise to the new class of entropy-based metrics presented in Chapter 4.

#### 3.3.3.2 Local Information Gain

Local Information Gain (LIG) is usually called simply 'information gain' in the literature, for instance by Yang and Pedersen [YP97]. LIG assumes that we are carrying out feature selection by forward selection (see Section 3.2.1), in which we start with

[9]Approximate distributions, which are conventionally derived from empirical distributions, are discussed in Section 3.3.1.

the empty feature set and iteratively add features. In LIG, our reference point is the distribution which arises from the *previous* feature set.

Suppose that our initial pool of available features $\mathcal{F}$ contains $N$ elements. At the $k$th step of our forward selection algorithm, we have a set $F_k$ containing $k$ features; and a set $\mathcal{F} \setminus F_k$, containing $N-k$ candidates for addition to $F_k$. Adding a particular feature $f_j$ gives rise to a new feature set:

$$F_k^{(j)} = F_k \cup \{f_j\}$$

The existing feature set $F_k$ and candidate $F_k^{(j)}$ give rise to empirical distributions over vectors of feature values, which we denote by $q_k(y)$ and $q_k^{(j)}(y)$ respectively. In LIG, we attempt to maximise the relative entropy from the existing distribution to the new one. Hence, at each step we seek:

$$\arg\max_j D(q_k(y) \,||\, q_k^{(j)}(y))$$

The above definition assumes that we are dealing with an unconditional model – one which simply seeks to assign a probability to each data point $x$, rather than attempting to classify it. A conditional model is of the form $\Pr(x \in C_i | F_k(x) = y)$, where each data point is in exactly one of finitely many categories $C_i$ and $F_k(x)$ is our existing feature set. As before, let $F_k^{(j)}$ denote a new feature set obtained by adding $f_j$ to $F_k$. In the conditional case, we similarly attempt to maximise the relative entropy:

$$\gamma_j(x) = D\left(\Pr(x \in C_i | F_k(x) = y) \,||\, \Pr(x \in C_i | F_k^{(j)}(x) = y')\right)$$

or in simpler notation,

$$\begin{aligned} \gamma_j(x) &= D\left(\Pr(C_i | F_k = y) \,||\, \Pr(C_i | F_k^{(j)} = y')\right) \\ &= D(P_k(C_i) \,||\, P_k^{(j)}(C_i)) \end{aligned}$$

Notice that the relative entropy $\gamma_j$ is dependent on a choice of data point $x$; this is because the conditional distribution $P_k$ depends on the vector of feature values:

$$F_k = \{f_{k1}, f_{k2}, \dots f_{kk}\}$$

where $f_{kn}$ is the $n$th feature in our existing feature set $F_k$. Similarly, $P_k^{(j)}$ depends on the value $F_k^{(j)} = \{f_{k1}, f_{k2}, \dots f_{kk}, f_j\}$ takes on a given data point $x$.

We would of course like to compare feature sets as a whole, not individual vectors of feature values. As Koller and Sahami note, simply considering the sum, maximum, or minimum of the relative entropies which occur on distinct data points in our training set is not appropriate, because some points are more likely to occur than others; an error in classifying a common data point is more troublesome than one in classifying a rare one in [KS96]. We therefore take the expectation with respect to the empirical distribution of possible feature vectors $\tilde{q}(y')$:

$$\Gamma_j = \sum_{y'} \tilde{q}(y') D\left(\Pr(\mathcal{C}_i | F_k(x) = y) \;||\; \Pr(\mathcal{C}_i | F_k^{(j)}(x) = y')\right)$$

where in each term, $y$ is the projection of $y'$ onto the reduced feature set $F_k$.[10]

We are thus computing a *conditional relative entropy*; see Cover and Thomas [CT91][p. 22]. As mentioned in Section 3.3.1, suitable approximations to probabilities such as $\Pr(\mathcal{C}_i | F_k(x) = y)$ can be derived from the empirical distribution, provided that the data points in our training set are labelled by category. We also note that $\Pr(x \in \mathcal{C}_i | F_k(x) = y)$ is an example of the *partition-conditional distributions* which will be defined in Chapter 4.

This conditional relative-entropy measure is unfortunately quite computationally expensive, being exponential in the number of features employed. This can be mitigated by the 'Markov blanket' methods discussed by Koller and Sahami, or by the accelerated feature selection technique presented in Chapter 4 [KS96]. Notice that instead of using the empirical distribution of possible feature vectors, we could take expectations with respect to the empirical distribution of data points $\tilde{p}(x)$ and compute the respective values of $F_k(x)$ and $F_k^{(j)}(x)$ for each $x$. The two approaches are equivalent although the latter is slightly more cumbersome.

LIG thus attempts to maximise the information gained by adding a new feature, using our current feature set as a benchmark. There is at least one significant drawback to this approach: The existing feature set $F_k$ has no global significance as a refer-

[10]The "gamma" notation here is analogous to the 'delta' notation used to defi ne the similar Koller-Sahami metric, which is discussed in Section 3.3.4 [KS96].

ence point. The largest possible step *away* from $F_k$ will not necessarily be the largest possible step *towards* some desirable endpoint. Koller and Sahami argue in greater detail that this lack of a global frame of reference is a significant flaw on the part of LIG [KS96].[11]

#### 3.3.3.3 Global Information Gain

One could also choose the *uniform* distribution $u(y)$ as our reference point. (If there are $N$ possible values for $y$, then $u(y) = 1/N$ for all $y$.) It can be shown (see [CT91]) that, of all possible distributions over a given set, the uniform distribution has the greatest entropy. The uniform distribution, then, has the greatest possible uncertainty; any other distribution can be thought of as providing more information than the uniform one. Choosing it as a benchmark therefore addresses the theoretical flaws of LIG which were described above; the uniform distribution is certainly a reference point of global importance.

The Kullback-Liebler divergence to the uniform distribution can be thought of as 'global information gain.' As discussed in Section 4.3.2, optimising the new entropy-based metrics presented in Chapter 4 can be thought of as maximising the global information gain; or equivalently, as minimising entropy. Feature selection algorithms based on minimising entropy have been employed by Toews and Arbel and Dash et al., but these are quite closely tailored to specific problems; the methods in Chapter 4 are far more general [TA03, DCSL02].

### 3.3.4 The Koller-Sahami Criterion

The Koller-Sahami (KS) evaluation metric first appeared in 1996; it has since received considerable attention, and been applied in a number of settings including the hybrid filter-wrapper method of Sebban and Nock and the construction of hidden Markov models for video structure discovery by Xie et al., and later experiments by Sahami et al. [KS96, SN02, XCDS02, Sah99, IGS01].

[11] Although some portions of Koller and Sahami's discussion of information-theoretic metrics are themselves flawed (see Section 3.3.4), its criticism of LIG is valid.

The KS metric can be thought of as ‘global information loss.’ It is defined in an analogous way to local information gain in Section 3.3.3.2, except that our reference point is derived using the set $\mathcal{F}$ of all available feature functions.[12] In the same way as above, we define a distribution:

$$P_\Omega = \Pr(C_i | \mathcal{F}(x) = y)$$

Similarly, the $j$th candidate feature set $F^{(j)}$ gives rise to a distribution $\Pr(C_i|F^{(j)} = y')$.

The idea behind the KS metric is that the distribution arising from the pool of all candidate features $\mathcal{F}$ is the most informative one available to us. Each feature will provide a non-negative amount of information about the data space; at the very worst, a feature function which takes the same value on all data points provides zero information. Note that an informative feature set – in the strict sense of information theory – is distinct from one which will give rise to good performance in a model. Simply put, not all information is useful.

Instead of maximising the divergence from an uninformative distribution, the KS criterion seeks to *minimise* the divergence from $P_\Omega$:

$$\delta_j = D\left(\Pr(C_i|\mathcal{F}(x) = y) \; || \; \Pr(C_i|F^{(j)} = y')\right)$$

We now take expectations over the empirical distribution of maximal feature vectors, $\tilde{q}(y)$:

$$\Delta_j = \sum_y \tilde{q}(y) D\left(\Pr(C_i|\mathcal{F}(x) = y) \; || \; \Pr(C_i|F_k^{(j)} = y')\right)$$

In this instance, the $y'$ in each term is the projection of $y$ onto the reduced feature set $F^{(j)}$. As with local information gain, computing the global information loss $\Delta_j$ is exponential in the number of features. KS present a method which uses so-called Markov blankets to reduce the computational expense of the KS metric; their method can equally well be applied to the family of information-gain measures [KS96]. As KS note, the Markov blanket unfortunately requires some naive assumptions about the data which may decrease performance.

[12]We could also defi ne a measure of ‘local information loss’ in which we carried out backward elimination of features and attempted to minimise the divergence from the previous feature subset, but this would suffer from the same theoretical weaknesses as local information gain.

**Assessment of the KS Metric:** The KS metric avoids the principal theoretical problem of local information gain; the set of all candidate features is a more significant reference point than the feature set obtained at the previous step of a forward selection search. KS claim that their experimental results demonstrate that the performance of the KS metric is superior to that of LIG [KS96]. However, in practice our chosen set of candidate features is not necessarily the only one possible. For instance, if we are categorising documents by key words or phrases, then we can hardly assess *all* available phrases in the English language using an information-theoretic metric. We may well have carried out some form of 'pre-selection' in order to obtain our collection of candidate features.

KS also suggest that their metric is theoretically optimal, due to its foundations in information theory and superiority to local information gain. The KS metric is certainly appealing, but 'optimal' seems to be too strong a word. It is far from clear whether it is better than other metrics based on information theory, such as absolute-entropy and mutual-information measures and the novel metrics presented in Chapter 4.

**KS Metric and Search Strategy:** KS incorrectly claim that the superiority of the KS metric over local information gain demonstrates that backward elimination of features is inherently better than forward selection. (See Section 3.2.1 for definitions of these search strategies.) In order to use LIG we must search by forward selection; but this is not true of the KS metric and backward elimination. Indeed, the experiments described in Chapter 5 successfully implement a variant of forward selection using the KS metric.

Seeking a final feature set which is as close as possible to $\mathcal{F}$, the set of all candidate features, does *not* mean that we must start at $\mathcal{F}$ and move away from it. We could equally well start far away from $\mathcal{F}$, for instance with the empty feature set, and attempt to move towards it. In the case of the KS metric (as with most others – LIG is an unusual exception), the metric for evaluating feature sets can be chosen independently of the method of generating them.

## 3.3.5 Mutual Information

The *mutual information* between two random variables $U$ and $V$, with a joint probability mass function $p(u,v)$ and respective marginal probability mass functions $p(u)$ and $p(v)$, is defined as:

$$I(U;V) = D(p(u,v)||p(u)p(v))$$

That is, mutual information is the relative entropy between the joint distribution and the product distribution. It can be shown that:

$$\begin{aligned} I(U;V) &= H(U) - H(U|V) \\ &= H(V) - H(V|U) \end{aligned}$$

where $H(U)$ and $H(U|V)$ respectively denote the entropy and conditional entropy, as defined in Section 3.3.2. (See Cover and Thomas for more details [CT91].) Notice that, unlike relative entropy, mutual information is symmetric in its arguments $U$ and $V$. Mutual information can be thought of as the information that $p(u)$ provides about $p(v)$ and vice versa.

Feature set evaluation measures can be defined using mutual information, in a similar way to those that employ relative entropy. The most commonly employed method is generally referred to simply as 'mutual information'; however, we will refer to it as *naive mutual information*, in order to distinguish it from the alternative metric of *joint mutual information*.

**Naive Mutual Information:** The naive mutual information (NMI) criterion defines a 'good' feature as one whose value has a high mutual information with the class variable. More formally, let $X$ be the set of all possible data points, denote individual data points by $x$, and suppose that each data point falls into exactly one category $\mathcal{C}_j$. Let $f_i$ denote our feature functions (where $i = 1,2,\ldots,n$), where each $f_i$ maps data points $x$ to values $y$. As usual, let $I(U;V)$ be the mutual information between two random variables $U$ and $V$ and $H(p(u))$ be the entropy of a probability distribution $p(u)$. We then have:

$$I(f_i\,;x \in \mathcal{C}_j) = H(\Pr(x \in \mathcal{C}_j)) - H(\Pr(f_i(x) = y | x \in \mathcal{C}_j))$$

If we simplify our notation by letting $x_j$ denote the event $x \in \mathcal{C}_j$ and writing $f_i = y$ instead of $f_i(x) = y$, we have:

$$I(f_i\,;x_j) = \left(-\sum_j \Pr(x_j)\log\Pr(x_j)\right) - \left(-\sum_j \Pr(x_j)\sum_y \Pr(f_i = y|x_j)\log\Pr(f_i = y|x_j)\right)$$

We seek to maximise the mutual information $I(f_i;x_j)$ between the feature and the class variable. This gives a score to each feature in isolation, and does not take account of how features are combined in a subset; naive mutual information is therefore a *hierarchical* measure (see Section 3.2.2.2). The name is analogous to that of the naive-bayes learner described in Section 2.3.3, which also assumes that the feature functions are independent of one another.

Feature selection by NMI is quite popular; see Dash and Liu, or Zaffalon and Hutter [DL97, YP97, ZH02]. Yang and Pedersen's survey of feature selection for text categorisation includes a comparison between the performance of information-gain and mutual-information metrics [YP97]. NMI is a fairly effective technique, although as noted by Tourassi et al. and in Chapter 6, it is often outperformed by more computationally intensive hierarchical metrics [TFMF01].

**Joint Mutual Information:** This is an alternative to NMI, first presented by Tourassi et al.[TFMF01]. It is non-hierarchical, and hence capable of considering interactions between features.

Joint Mutual Information (JMI) makes use of the chain rule for mutual information.[13] For a collection of random variables $\{U_1, U_2, \ldots, U_n, V\}$, the chain rule states:

$$I(U_1, U_2, \ldots, U_n; V) = \sum_{i=1}^{n} I(U_i\,; V | U_{i-1}, U_{i-2}, \ldots, U_1)$$

In the feature selection case, $V$ is the class variable arising from the event $x \in \mathcal{C}_j$, and the individual variables $U_i$ are our feature functions $f_i(x)$. We are interested in the

[13]See page 22 of Cover and Thomas for a proof of the chain rule [CT91].

distribution of vectors of feature values $F(x) = (f_1(x), \ldots, f_n(x))$. The JMI is therefore equal to:

$$I(F(x) = \vec{y}\,; x \in C_j) = \sum_{i=1}^{n} I(f_i = y_i\,; x \in C_j | f_{i-1} = y_{i-1}, f_{i-2} = y_{i-2}, \ldots f_1 = y_1)$$

Tourassi et al. note that JMI can be expected to outperform NMI, as it combines all the advantages of NMI with the ability to consider relationships between features. They also present reliable methods of approximating the JMI for large feature sets.

## 3.4 Related Topics

In this section we briefly outline a few areas which are closely related to feature selection. It is by no means an exhaustive list of related topics. As we have noted, feature selection is a very general technique applicable in a wide variety of domains; and conversely, many different problems can in principle be seen in terms of feature selection.

### 3.4.1 Dimensionality Reduction

The field of dimensionality reduction (DR) is very closely related to feature selection. The basic goal is the same: We attempt to reduce the number of variables – and so reduce the dimensionality of the data space – while losing as little information as possible. However, the term ‘dimensionality reduction’ conventionally implies a slightly different perspective from that of feature selection.

DR takes an essentially geometric approach to simplifying the description of data. Data points expressed using a set of $n$ variables are regarded as points in an $n$-dimensional space. The points composing a typical data set will often occupy a lower-dimensional surface or *manifold* within the $n$-dimensional space. DR typically seeks a suitable rotation of our coordinate axes which will allow the manifold to be projected into a lower-dimensional space with little or no loss of information. See Seung and Lee for a simple introduction to the use of DR in image recognition [SL00]; Kambhatla and Lee for a definition of the popular DR technique of Principal Component Analysis [KL97]; and Tenenbaum et al. for a discussion of some recent developments in DR [TdSL00].

If we regard our features as variables, then DR can be employed as a form of feature selection – and has been, for instance by Globerson and Tishby [GT03]. Conversely, DR can be used to bypass feature selection altogether. For example, if we are carrying out image recognition and have a space of $10^{12}$ possible combinations of pixel values, we could use DR directly to find a simplified representation of the pixel space – rather than first defining features and using DR or some other technique to select a 'good' feature set.

Feature selection is more flexible than DR, in that it allows a much wider variety of techniques for constructing a representation of the data. It may also be more useful in providing a qualitative understanding of the phenomenon we are studying. This is particularly true when the features have some inherent significance.

For instance, in document classification we may well be interested in which words or phrases are good indicators of the category of a given document. We could regard a document containing $n$ words as a point in $n$-dimensional space by placing our vocabulary in some fixed order, and seeking to rotate and project the representation of the document in this space; but interpreting these results to discover which key words are most useful would be very difficult at best. Conversely, feature selection allows us to easily make use of any prior knowledge of the domain by tailoring our features and selection strategy accordingly; this is considerably more difficult with geometric DR techniques.

Another consideration is that DR is only useful when the data points are confined to a relatively simple manifold. If they are evenly scattered in the $n$-dimensional data space, then classical DR will not be appropriate; but feature selection can still give useful results.

### 3.4.2 Minimum Description Length and Coding Theory

Another perspective on constructing efficient representations of data is provided by the *minimum description length* (MDL) criterion for selecting statistical models; Hansen and Yu provide a useful introduction to MDL [HY01]. As the name suggests, MDL instructs us to choose the model which provides the shortest possible description of data. Trying to find a small yet informative feature set can be seen as a specialised form of MDL.

The ‘length’ of a description in MDL is defined using measures of its information content drawn from information theory. Hence, MDL can be seen as seeking the *simplest* possible description of data. More specifically, MDL draws heavily upon *coding theory*, which deals with the problem of representing and transmitting data using a finite set of code symbols [CT91].

Coding theory can also be applied to feature selection. The key insight is that feature selection involves constructing a representation of the data, using a set of finitely many features. The relationship between feature selection and coding theory is particularly interesting, and will form the basis of the Expected Covering Entropy metrics introduced in Chapter 4 and experimentally evaluated in 7. Conversely, it can be useful to apply ideas from feature selection to problems traditionally defined in terms of coding theory; in Chapter 4, we establish that the classification technique of *error-correcting output coding* can be viewed as a special case of feature selection.

## 3.5 Summary

In this chapter we have outlined existing literature on feature selection. We concentrate on the conventional, general framework for feature selection, based on a division into *generation* and *evaluation* of candidate feature subsets, combined with an appropriate *halting criterion*. This includes the key distinction between generation by forward selection and by backward elimination; and that between filter and wrapper methods of evaluation. We continue with case studies of typical evaluation filters based on information theory; such measures are very popular in the literature, and highly relevant to the new theoretical work presented in Chapters 4 and experimentally investigated in Chapters 5, 6, and 7. Finally, we briefly consider the topics of dimensionality reduction and model selection by minimum description length, both of which are quite closely related to feature selection.

# Chapter 4

# Partitioning and Encoding

## 4.1 Introduction

In this chapter we present a new perspective on feature selection, based on the ideas of *partitioning* and *encoding*. As noted in Chapter 3, any feature selection process requries an evaluation function or *metric* which provides a quantitative measure of the desirability of a given feature subset. We begin by discussing the concept of partitioning, and using it to motivate a new metric which we call *expected partition entropy* or EPE. As we shall see, EPE can also be motivated by consideration of Kullback-Liebler divergence, in a similar way to the Local Information Gain and Koller-Sahami metrics discussed in Chapter 3. We then present the idea of encoding and use it to extend EPE, defining a class of metrics which we call *expected covering entropy* or ECE. Throughout the chapter, we will use terminology and notation from Chapters 2 and 3 where appropriate.

The concepts described in this chapter, and the metrics derived from them, are almost completely general; they can in principle be applied to any feature selection problem. In Chapters 5, 6, and 7, the new EPE and ECE metrics will be tested in practical experiments with a simplified form of part-of-speech tagging.

## 4.2 Feature Selection as Partitioning

A useful perspective on feature selection is provided by the observation that feature sets *partition* the data space $X$. That is, the features decompose the data space into disjoint, non-empty subsets whose union is all of $X$. This fact is most easily established by considering *equivalence relations*.

An equivalence relation $\sim$ on a set $A$ satisfies the following conditions:

1. $a \sim a \ \ \forall a \in A$
2. $a \sim b$ if and only if $b \sim a \ \ \forall a, b \in A$
3. If $a \sim b$ and $b \sim c$ then $a \sim c \ \ \forall a, b, c \in A$

Suppose that we have a feature set $F : X \mapsto Y^n$. Clearly, $x_1 \sim x_2$ if and only if $F(x_1) = F(x_2)$ defines an equivalence relation on $X$. Hence, the feature set divides $X$ into partitions, such that each data point $x$ is contained in one and only one partition. (See Chapter 12 of Armstrong for further details on equivalence relations [Arm97].)

There is one partition for each possible feature vector $y$. For the sake of brevity we will often refer to a given feature vector $y = F(x)$ as the *name* of the data point $x$. The members of the partition corresponding to a particular vector $y \in Y^n$ are the points $x$ satisfying $F(x) = y$. Data points $x$ in a given partition will all share the same name $y$; we can therefore regard $y$ as also being the name of the partition. As discussed in Section 2.2.1, there may be points in $Y^n$ which are not the name of any data point in $X$.

**Example 4.1:** Suppose that our data points are English words and we define binary features with the presence of particular substrings as their indicators. Let us define features $f_a$, $f_b$, and so on corresponding to each letter of the alphabet. We have $f_a(x) = 1$ on words $x$ which contain the letter a, $f_a(x) = 0$ otherwise; and so on for the rest of the alphabet.

Many combinations of these features can be guaranteed never to occur, as they correspond to words which contain only consonants. For instance, there will be no ‘partition’ corresponding to ‘$f_z = 1$, $f_x = 1$, all other features are zero’ since no English word will contain only z’s and x’s. □

A number of feature selection metrics consider the behaviour of vectors of feature values; these include the EPE and ECE metrics developed in this chapter, as well as the Local Information Gain and Koller-Sahami metrics described in Chapter 3. Such metrics can now be seen as examining the characteristics of partitions. This insight may prove helpful in understanding the behaviour of metrics which partition the data space. In a more concrete sense, it serves as motivation for the new EPE and ECE metrics defined in this chapter.

Because they directly consider and exploit the effect of partitioning, it was hoped that the new metrics would be effective tools for feature selection – particularly in the sense of giving rise to high accuracy on test data with feature sets much smaller than the pool of all available features. Furthermore, the new metrics can be used to measure the ways in which other metrics partition the data space, giving us additional information as to why *any* given metric succeeds or fails. As discussed in Chapters 6, 7, and 8, our experiments demonstrate that the new partitioning-based metrics can offer significant benefits in these respects.

Intuitively, we wish to find a feature set which partitions the data space in an informative way. It is natural to turn to information theory to define what might constitute an 'informative partition.' The partitioning idea then serves to motivate a new metric which we refer to as *expected partition entropy*.

## 4.3 Expected Partition Entropy

### 4.3.1 Motivation and Definition

In the standard classification problem, each data point $x \in X$ falls into exactly one category $C_j$. Suppose that we wish to construct a feature-based classifier which predicts the categories of previously unseen data points. A set of feature functions assigns a vector of feature values $y$ to each data point $x$; we refer to $y$ as the *name* of $x$. In attempting to classify data points, the model relies solely on the information it can derive from their names.

As we have seen, the feature set can be seen as partitioning the data space. Intuitively, we would like each name induced by the data set to be strongly associated with a particular category $C_j$. Also observe that we do not necessarily mind if two different data points $x_1$ and $x_2$ have the same name (vector of feature values) $y$, as long as they fall into the same category.

Information theory allows us to formalise this idea as follows: Consider the points in the partition defined by the vector of feature values $y$. We can establish a conditional distribution over categories:

$$P_y(j) = \Pr(x \in C_j | F(x) = y)$$

We refer to this as the *partition-conditional distribution* or PCD. In general we cannot determine the PCD exactly, but it can be reliably approximated by the empirical distribution or other methods; see Section 3.3.1 for details. Partition-conditional distributions have already been used to define the Local Information Gain (LIG) and Koller-Sahami (KS) metrics discussed in Sections 3.3.3.2 and 3.3.4, both of which make use of relative entropy (also known as Kullback-Liebler divergence). The relationship between relative entropy and the new Expected Partition Entropy metric will be discussed in Section 4.3.2 below. For the time being, we will take a different approach by simply considering the entropy of the PCD induced by a candidate feature set.

As discussed in Section 3.3.2, entropy can be thought of as a measure of the uncertainty inherent in a given probability distribution. The entropy of a distribution $p(u)$ is defined as:

$$H(p) = -\sum_u p(u) \log p(u)$$

Now consider the entropy of a partition-conditional distribution, $H(P_y(j))$. Given the vector of feature values $y$, we would like to be *as certain as possible* of the category $C_j$. In terms of information theory, this is equivalent to minimizing the entropy of the partition-conditional distribution. The quantity $H(P_y(j))$ attains its minimum value of zero when we have $P_y(j) = 1$ for one partition $C_j$, and $P_y(j) = 0$ for all others – that is, when we are absolutely certain of the category $C_j$ given the vector of feature values $y$.

We would like to combine the entropies for the different partitions (each of which corresponds to a particular name $y$) into a single quantity. We might naively take the sum or product of the individual entropies. However, it is more appropriate to take the expectation over the names $y$; the idea is that uncertainty on an uncommon name is less troubling than uncertainty on a common one. (This is the same approach to combining partition-conditional distributions as the one taken for the Koller-Sahami metric [KS96].)

The *expected partition entropy* or EPE of a candidate feature set $F$ is therefore defined as:

$$\begin{aligned}
\varepsilon &= \sum_y \Pr(F(x)=y) H\left(\Pr(x \in \mathcal{C}_j) | F(x)=y)\right) \\
&= -\sum_y \sum_j \Pr(F(x)=y) P_y(j) \log P_y(j)
\end{aligned}$$

where as usual $x$ is a data point drawn from a set $X$, $\mathcal{C}_j$ is one of countably many categories, and $F(x)$ is a set of feature functions which maps data points $x$ to vectors of feature values $Y \in Y$.

**Comments on the Definition of EPE:** Notice that the EPE is a *conditional entropy*, as defined in Section 3.3.2 or by Cover and Thomas [CT91]. It is the entropy of the conditional distribution $\Pr(x \in \mathcal{C}_j)|F(x) = y)$. EPE thus has a solid grounding in information theory; it is very much a natural way of quantifying the information about categories conveyed by a given feature subset. Hence, we will sometimes refer to EPE simply as the *entropy* of a feature subset, or as its *absolute entropy* if we wish to emphasize the distinction between EPE and metrics based on relative entropy. It is also worth noting that EPE is a completely general metric, which can be used for feature selection in any classification domain.

Intuitively, a feature set with high (that is, poor) EPE will have a broad spread of labels in a typical partition. Conversely, if our feature set has low EPE then knowing the name of a data point will typically give us a great deal of information as to its label. From the perspective of EPE, the best possible partition is one containing points which have only one label; two equally probable labels are less good; ten equally probable labels are worse still; and so on. Because we are taking an expectation, greater importance is placed upon the ‘goodness’ of particularly common partitions.

EPE is a type of (non-hierarchical) filter, as defined in Section 3.2.2.1. Hence, it is capable of considering the interactions between features, and it is independent of any particular model architecture. We can therefore expect EPE to give rise to somewhat lower accuracies than a wrapper method, but higher than a hierarchical filter (such as naive mutual information). In practice, EPE performs better than expected in comparison with wrappers; it outperforms a naive-bayes wrapper in particular circumstances and achieves similar performance to a maximum-entropy wrapper at far lower computational cost, as discussed in Chapter 6.

### 4.3.2 EPE and Relative Entropy

The *relative entropy* or Kullback-Liebler (K-L) divergence is an information-theoretic distance between probability distributions. For two probability mass functions $p(u)$ and $q(u)$ over a set $U$, it is defined as:

$$D(p||q) = \sum_{u \in U} p(u) \log \frac{p(u)}{q(u)}$$

Relative entropy can be thought of as measuring the error that arises from replacing $p(u)$ by $q(u)$; hence, $p(u)$ is conventionally taken to be the 'more informative' distribution. For additional details, see Section 3.3.2 or Cover and Thomas [CT91]. As discussed in Sections 3.3.3.2 and 3.3.4, the local information gain (LIG) and Koller-Sahami (KS) metrics make use of K-L divergence. LIG attempted to maximize the expected divergence to the PCDs obtained at the previous step of forward selection; the KS metric, to minimize the expected divergence from the PCDs derived from the collection of all available features. Both combine the relative entropies for PCDs into a single quantity by taking the expectation over the names $y$.

Although it has been motivated and defined in terms of straightforward entropy and conditional entropy, EPE can also be viewed in terms of relative entropy. Doing so suggests reasons why EPE may be superior to both LIG and the KS metric.

Suppose that instead of adopting another PCD as our reference point – as with both LIG and KS – we measure distances from the *uniform distribution*. The uniform distribution $U$ over $N$ points assigns a probability of $1/N$ to each point; it can be shown (see Cover and Thomas) that it is the distribution with the greatest possible entropy over a given set, and so the least possible amount of information [CT91]. We would like to maximize the divergence from the PCD $P_y(j)$ induced by our candidate feature set to $U(j)$, the uniform distribution over categories. Hence, we seek to maximize:

$$D(P_y(j)||U(j)) = \sum_j P_y(j) \log\left(\frac{P_y(j)}{U(j)}\right)$$

The uniform distribution is constant for all $j$. Denoting this constant by $k$, we have:

$$\begin{aligned}
D(P_y(j)||U(j)) &= \sum_j P_y(j) \log\left(\frac{P_y(j)}{k}\right) \\
&= \sum_j P_y(j) \left(\log(P_y(j)) - \log k\right) \\
&= \sum_j P_y(j) \log(P_y(j)) - \log k \sum_j P_y(j) \\
&= \sum_j P_y(j) \log(P_y(j)) - \log k \\
&= -\left(-\sum_j P_y(j) \log(P_y(j))\right) - \log k \\
&= -H(P_y(j)) - \log k
\end{aligned}$$

Hence, maximizing the relative entropy to the uniform distribution is equivalent to simply minimizing the entropy $H(P_y(j))$. Furthermore, the constant $k$ is equal to $1/N$, where $N$ is the number of possible categories; it is therefore the same for all partitions. Maximizing the expected divergence from the uniform distribution – in an analagous way to LIG or the KS metric – is therefore equivalent to optimising (that is, minimising) the EPE. EPE can thus be thought of as a measure of *global information gain*, in contrast with local information gain.

Koller and Sahami identify an important weakness of LIG: The PCD obtained at the previous step of a forward selection algorithm has little global significance as a reference point. They instead advocate measuring divergences from the pool of all available features. However, as noted in Chapter 1 and Section 3.3.4, the collection of available features itself is not necessarily an ideal reference point. We may well have carried out some form of ‘pre-selection’ in order to establish the pool, and it may not be feasible to work with the true set of all possible features – the number of possible features may be astronomically large or even infinite.

Moreover, our understanding of the partition-conditional distributions is limited by the quality of our training set. We cannot find the 'true' distributions governing the data, and instead must approximate them using the empirical distribution or some other probabilities derived from a set of labelled training data. Our training set will almost never be comparable in size to the set of all possible data points. Often, labelling training data will be difficult or unreliable, and the training set will be too small give us more than a rough idea of the behaviour of the data.

This presents a serious problem for the KS metric, since the mean divergence it computes is heavily dependent on the training data and any 'pre-selection' of features.[1] EPE has a very important advantage in this respect: We generally know exactly what the possible categories are, so the uniform distribution over categories is completely independent of the training data. As discussed in Chapter 6, these concerns were borne out by experiments. Feature sets which attained an optimal value for the KS metric were not optimal with respect to the EPE metric or their performance on held-out test data.

## 4.4 Feature Selection as Encoding

### 4.4.1 Encoding – Basic Ideas

We have noted that an FBPM can only obtain information about data points by observing their feature vectors. For instance, an FBPM utilising a feature set $F$ cannot distinguish between two data points $x_1$ and $x_2$ such that $F(x_1) = F(x_2)$. Intuitively, then, we would like our feature vectors to provide good *descriptions* of data points. In seeking to quantify the idea of a 'good description,' we can make use of ideas from coding theory.

We begin with the observation that our features can be thought of as *encoding* data points. Given a feature set $F$, each data point $x$ is assigned a 'code word' consisting of its vector of feature values $F(x)$, whose $i$th 'letter' is simply the value of the feature $f_i$. We will sometimes refer to the vector $F(x)$ as the *name* of $x$.

[1]The same is true for LIG, which also suffers from having a less general reference point than KS.

In traditional coding theory (see Cover and Thomas, chapters 5 and 13), we assign code words to 'data points' such as English words or sentences. A code word is a string of finitely many symbols, usually chosen from a finite alphabet.

Code words are selected with two principles in mind:

1. We would like to be able to reconstruct the original data point from its code word with as little expected loss of information as possible.

2. We would like the expected length of a code word to be as short as possible.

Information theory provides us with well-defined numerical measures of the 'expected loss of information.' Information loss may arise because our vocabulary has fewer code words than distinct data points; because the process of encoding the data points (or transmitting or storing the code) is noisy and leads to errors; or from some combination of the two.

In feature selection our task is slightly different, because we are usually not interested in reconstructing data points from their feature vectors. However, the goals of feature selection can be defined in a way that is clearly analagous to coding theory:

1. We would like our feature set to maximise the usefulness of an FBPM. 'Usefulness' may be defined quantitively, by the performance of the FBPM on unseen test data; or qualitatively, by its contribution to understanding the phenomenon we wish to model.

2. We would like our feature set to be as small as possible in order to maximise speed, reduce overfitting, and enable the features to be more easily understood.

In both coding theory and feature selection, we have a basic trade-off: Longer code words reduce information loss and large feature sets generally improve the accuracy of a model, but at the cost of reduced computational efficiency and – in the feature selection case – possible overfitting and the difficulty of understanding the behaviour of a large feature set. In both cases, we wish to provide the simplest possible description of our data space without sacrificing important information. This is in accordance with the ancient principle of Occam's Razor: 'Plurality should not be posited without necessity'; in other words, one should make no unnecessary assumptions.

General connections between coding theory and feature selection will be discussed in Section 4.4.3; they help to motivate the new class of *expected covering entropy* metrics discussed in Section 4.5. First, we examine the specific technique of *error-correcting output coding* (ECOC), which can be seen as a special case of feature selection; examination of ECOC also serves to illustrate the ideas of coding theory in greater depth.

## 4.4.2 Error-Correcting Output Coding

### 4.4.2.1 Definitions and Notation

The technique of *error-correcting output coding* or ECOC is a method of solving multi-way classification problems which has been successfully applied to a number of tasks in machine learning [Ber99]. As the name suggests, ECOC draws upon ideas from coding theory; and as we shall establish, ECOC can be viewed as a special case of feature selection.

Suppose that each data point $x$ falls into one (and only one) of $m$ categories:

$$\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$$

This is the standard setting for a conditional probabilistic model, as described in Section 2.3; the categories are also known as *labels*. We wish to construct a classifier which can predict the category of any point $x$ in our data space $X$.

In order to implement ECOC, we begin by assigning a unique $n$-bit vector to each label $\mathcal{C}_i$, where $n > \log_2 m$.

**Example 4.2:** Suppose that we wish to place Web pages in one of four categories: News, business, scientific/technical, and entertainment. We assign each category a bitvector as described in Table 4.1. □

One can view the $i$th vector as a unique code word for the $i$th label. We refer to the ordered set of bitvectors as a *code*, denoting the $i$th vector by $v_i$ and its $j$th digit by $v_{ij}$. In the above example, $v_1 = 00111001$ and $v_{23} = 0$. For obvious reasons, the set of bitvectors is also sometimes referred to as a *coding matrix*.

| **Label** | **Coding** |
|---|---|
| News | 00111001 |
| Business | 10010010 |
| Sci/Tech | 01001101 |
| Entertainment | 11100110 |

Table 4.1: Example of ECOC bitvectors for news categories.

Each data point has one and only one correct category; given a data point $x$, we denote the code word for its category by $v(x)$ and the $j$th digit of that code word by $v^{(j)}(x)$. If the category of $x$ is $C_i$, then $v(x) = v_i$. In our example, $v(\text{news.bbc.co.uk}) = v_1$ and $v^{(3)}(\text{news.bbc.co.uk}) = 1$ (being the third digit of the code word for the 'News' category). Notice that the bitvectors can be thought of as inducing a new set of 'superclasses' $V_1, V_2, \ldots, V_n$ on the data space $X$. A data point $x$ belongs to the superclass $V_j$ exactly when $v^j(x) = 1$.

The basic idea of ECOC is as follows: We construct a set of $m$ independent binary classifiers, one for each *column* of the code.[2] We denote the $j$th classifier by $c_j$. These binary classifiers are generally known as 'plug-in classifiers' or PiCs. Given a data point $x$, $c_j$ classifies $v^{(j)}(x)$ – the $j$th digit of the code word $v(x)$. In terms of the superclasses, the $j$th classifer is attempting to distinguish data points in the superclass $V_j$ from those in its complement $\overline{V_j}$. In our Web classification example above, the classifier $c_4$ attempts to distinguish between pages in the category 'News' or 'Business', and those in the category 'Sci/Tech' or 'Entertainment.'

Our classifiers can be viewed as functions – given a data point $x$, the $j$th PiC returns its predicted classification $c_j(x)$. The classification $c_j(x)$ may be chosen from the binary set $\{0,1\}$; in this instance, $c_j(x)$ is simply a guess at one of the two possible values of $v_j(x)$. Alternatively, $c_j(x)$ may be a real-valued probability measuring the classifier's confidence that $v^j(x) = 1$.

For any given data point $x$, we can form a classification vector $c(x)$ from the results

[2]As established by Berger, a Naive-Bayes model can be a suitable binary classifi er for ECOC models [Ber99].

of the individual PiCs:

$$c(x) = (c_1(x), c_2(x), \ldots, c_n(x))$$

The classification vector $c(x)$ is unlikely to be equal to any of the code words $v_i$ – particularly if the data point $x$ was not in the set of examples used to train the individual classifiers $c_j$. Intuitively though, if $c(x)$ is 'close to' some code word $v_i$, then $x$ is likely to fall into the $i$th category.

**Example 4.3:** Continuing our Web page classification example, suppose that $x$ denotes the page www.reuters.com, our classifiers are binary, and $c(x) = 10011001$. Also, recall that the code word for 'News' was $v_1 = 00111001$. Six of the eight bits in $c(x)$ match the corresponding bits in $v_1$. Therefore, our collection of classifiers is in some sense quite confident that $x$ denotes a news page (as we would hope).

Notice that the first and third bits of $c(x)$ are different from the corresponding bits of $v_1$; this implies that the first and third classifiers have misclassified $x$ – the first believes that it is in the category 'Business' or 'Entertainment' rather than 'News' or 'Sci/Tech', while the third believes it to be in 'Business' or 'Sci/Tech' rather than 'News' or 'Entertainment.'[3] ECOC is therefore quite robust to mistakes made by individual classifiers, as we discuss in greater detail below. $\square$

We will now formalise the notion of a classification vector $c(x)$ being 'close' to a code word $v_i$. Recall that both $c(x)$ and $v_i$ are $n$-dimensional vectors; the components of $v_i$ are chosen from the binary set $\{0,1\}$, while those of $c(x)$ are from either the binary set (if the plug-in classifiers $c_j$ are binary) or the unit interval (if the PiCs are probabilistic). We therefore choose a metric which defines distances between the appropriate vectors. If our classifiers are binary then we select a metric defined on bitvectors, such as Hamming distance (see below); if they are real-valued then one of the class of $L_p$ metrics on the unit interval (see Priestly) is appropriate [Pri97].

**Example 4.4:** The Hamming distance $\Delta(w, w')$ between two bitvectors $w$ and $w'$ is defined as the minimum number of bits that need to be changed in order to transform $w$ into $w'$; in other words, it is the number of places in which $w$ and $w'$ differ. In the case

[3]Of course, in practice it might be appropriate to assign more than one category to a web page – but we are examining the simpler problem in which each data point falls into exactly one category.

| **Label** | $v_i$ | $\Delta(c(x), v_i)$ |
|---|---|---|
| News | 00111001 | 2 |
| Business | 10010010 | 3 |
| Sci/Tech | 01001101 | 4 |
| Entertainment | 11100110 | 7 |

Table 4.2: Hamming distances between ECOC bitvectors.

of $c(x) = 10011001$, we have distances between $c(x)$ and each codeword $v_i$ as given in Table 4.2.

Therefore, according to our collection of binary classifiers, www.reuters.com is most likely to be in the category 'News' and least likely to be in 'Entertainment.' □

An ECOC classifier assigns a given data point $x$ to the category $\mathcal{C}_i$ whose codeword $v_i$ is closest to $c(x)$, the vector of outputs from the individual binary classifiers. If two categories are equally close, then we select one arbitrarily. Of course, 'closeness' is defined by our chosen metric. So long as the codewords $v_i$ are widely spaced with respect to the metric, ECOC is robust to errors by a small number of the binary classifiers.

#### 4.4.2.2 Comments on ECOC

**One Versus Rest:** Closely related to ECOC is the 'one versus rest' strategy for combining binary classifiers, which works as follows: For each of the $m$ categories $\mathcal{C}_i$, we train a binary classifier $c_i$ such that $c_i = 1$ if $x \in \mathcal{C}_i$, and $c_i = 0$ otherwise. This is a special case of ECOC in which our coding matrix is the $m \times m$ identity matrix. There are sound theoretical reasons to expect that a more general ECOC approach (using a coding matrix other than the identity matrix – it can be shown that a randomly generated coding matrix is appropriate) will outperform 'one versus rest,' as discussed by Berger [Ber99].

**Lower Bound on Code Word Length:** Notice that if we wanted our code words to be as short as possible, 2-bit vectors would suffice for a set of 4 categories; an example is given in Table 4.3.

| Label | Coding |
|---|---|
| News | 00 |
| Business | 10 |
| Sci/Tech | 01 |
| Entertainment | 11 |

Table 4.3: ECOC bitvectors of minimum length.

The lower bound of $\log_2 m$ for the length of a code word (where $m$ is the number of labels) is thus the minimum number of bits required to assign a unique code word to each category. Reducing the length of code words increases computational efficiency, but at the cost of greater vulnerability to errors by one or more of the plug-in classifiers.

#### 4.4.2.3 ECOC and FBPMs

An ECOC classifier is a special case of a feature-based probabilistic model (FBPM). Each PiC $c_i$ can be thought of as a feature function, taking data points $x$ and mapping them to a set $Y$ (where $Y$ is either the binary set or the unit interval). The number of features is fixed at $n$, where $n$ is the number of bits in the category code words $v_i$; but the features themselves can vary according to how the PiCs are selected and trained. Our pool of available features, then, is equivalent to a set of PiCs – such as the set of possible Naive-Bayes binary classifiers. The classification vector $c(x)$ is equivalent to a vector of feature values $F(x)$.

Recall that an FBPM maps vectors of feature values $F(x) \in Y^n$ to probabilities. In the classification case, it therefore takes the form:[4]

$$p(i|y) = \Pr(x \in \mathcal{C}_i | F(x) = y)$$

Given a data point $x$, the standard ECOC classifier simply returns a category $\mathcal{C}_i$. Such a model can be thought of as a conditional probability distribution $p(i|y) = \Pr(x \in \mathcal{C}_i|F(x) = y)$ in which, for any given $y$, $p(i|y) = 1$ for one particular value of $i$ and

[4]Notice that an ECOC classifi er – or indeed an FBPM – may also include a number of free parameters. The free parameters have been left implicit here, in order to simplify our notation.

$p(i|y) = 0$ otherwise. In other words, the ECOC is a conditional probability distribution which is absolutely certain of the category $\mathcal{C}_i$ of any given data point $x$. (It should be noted that ‘certain’ is not the same as ‘correct’; the standard ECOC is always precise, but it may be precisely wrong.)

The ‘absolutely certain’ FBPM arising from an ECOC can be easily extended to a more sophisticated probability distribution. Recall that under the ECOC scheme, the ‘most likely’ category for any given data point $x$ is the category $\mathcal{C}^*$ whose code word $v^*$ satisfies:

$$v^* = \arg\min_i d(v_i, c(x))$$

where $d$ is our chosen metric, and $c(x)$ is the classification vector. Recall also that $c(x)$ is equivalent to a feature vector.

Now, we are already computing the distances $d(v_i, c(x)$ for each $i$. Intuitively, the greater the distance between $c(x)$ and $v_i$, the less likely it is that $x$ belongs to category $\mathcal{C}_i$. We can easily define a probability distribution in which, for each possible value of $c(x)$, categories whose code words are more distant from $c(x)$ receive lower probabilities. For instance, we could have:

$$\psi(i|x) = \gamma \frac{1}{d(v_i, c(x))}$$

where $\gamma$ is a normalisation constant chosen so that the probabilities sum to 1.

**Example 4.5:** Returning to the Web page classification example, the distances of www.reuters.com from the code words for categories 1, 2, 3, and 4 were 2, 3, 4, and 7 respectively. The FBPM $\psi(i|x)$ defined above then assigns (un-normalised) probabilities of 1/2, 1/3, 1/4 and 1/7 to ‘News’, ‘Business,’ ‘Sci/Tech,’ and ‘Entertainment’ respectively; the normalisation constant is equal to 84/103. $\square$

To sum up, an ECOC classifier naturally gives rise to a family of FBPMs, in which the plug-in classifiers play the role of feature functions. This relationship arises from the fact that both ECOC and FBPMs are based on *encoding* data points.

### 4.4.3 Coding Theory and Feature Selection

In this section we discuss more general connections between coding theory and feature selection. We examine the key concepts of *redundancy* and *spacing* of code words; these concepts can also be profitably applied to vectors of feature values.

**Redundancy:** The notion of *redundancy* is very important in coding theory generally, and ECOC in particular. Simply put, a lengthy description of our data has the advantage that, if one or more digits are corrupted, we will still be likely to reconstruct our data point with reasonable accuracy. Redundancy means that an ECOC classifier with longer code words is robust to errors by one or more of its PiCs; an inaccuracy in one can be compensated for by the others. Of course, a high degree of redundancy in a code may be an inefficient use of available computational resources.

In terms of feature selection, the coding perspective provides some insight into why large feature sets tend to perform better than small ones: If our FBPM receives a lengthy description of each data point, then it is less likely to misclassify the point (or to assign it an incorrect probability, in the case of an unconditional model) because of the presence of a few irrelevant or misleading feature values.

For instance, suppose again that we are trying to classify Web pages by subject, and we have binary features whose indicators are the presence of certain key words. We run across a page on cooking, which mentions ‘recipes’ and ‘Jerusalem artichokes.’ Unfortunately, our classifier was trained on a number of pages which referred to ‘numerical recipes.’ The classifier (erroneously) associates occurrence of the word ‘recipe’ with the category ‘Science/Technology,’ and (understandably) associates the word ‘Jerusalem’ with ‘News/Current Events.’ If only these two features were active then our model would certainly misclassify the cooking page; but if many other features noticed the occurrence of food-related terms, then the classifier would be much more likely to reach the correct conclusion.

Notice that the ‘Jerusalem’ feature will in general be a very good one for distinguishing news reports from (say) cookery. An important challenge in feature selection is choosing appropriate *combinations* of features, in such a way as to compensate for the weaknesses of individual features.

**Spacing:** Another crucial concept in coding is that our code words should be as *widely spaced* as possible, with respect to Hamming distance or some other appropriate measure. In the ECOC setting, widely spaced code words again reduce the chances of misclassification; they can reasonably be expected to do the same in terms of feature selection. Obviously, if our code words are longer then it is easier to ensure that they are widely spaced.

The ideas of redundancy and spacing help to motivate an extension of Expected Partition Entropy, as discussed in Section 4.5.

## 4.5 Expected Covering Entropy

In this section we define an extension of EPE, which we refer to as *expected covering entropy* or ECE. The extension is motivated by ideas from coding theory, and from a wish to more explicitly consider the interactions between features.

Two of the key ideas of coding theory are *redundancy* and *spacing*, as detailed in Section 4.4.3. Essentially, we would like our code words to remain useful if errors occur in one or more places. In order to apply these concepts to feature selection, we note that EPE – and related metrics such as LIG and the KS metric – are motivated by the desire for each partition to be strongly associated with a particular category. More precisely, we would like knowing the vector of feature values $y = F(x)$ for a particular data point $x$ to give us as much information as possible about the category of $x$. The EPE metric addresses this question in a theoretically appealing fashion by making use of the conditional entropy.

However, as it stands EPE is vulnerable to misleading values taken by one or more features. For instance, if we are carrying out part-of-speech tagging for English words using binary features, we may find that the feature vector 10101001 is strongly associated with the `NOUN` tag; but that 10101101 always occurs with the `VERB` tag.[5] In order to distinguish between certain nouns and verbs, our classifier is totally dependent on the value of the sixth feature:

[5]Chapter 5 outlines an extensive investigation of a simplifi ed form of feature-based part-of-speech tagging, and discusses appropriate binary features in this setting.

$$f_6(x) : X \mapsto \{0,1\}$$

Because of the limitations of our training data, we may not have an entirely accurate picture of the behaviour of $f_6$. This places our model at some risk of incorrectly classifying a noun as a verb or vice versa. Intuitively, we would like our model to be robust to errors taken by one or more features. In the above example, we would like to introduce additional features to 'check' that the feature $f_6$ was arriving at the correct verdict – perhaps by replacing some of the other features, if necessary.

We formalise this idea by drawing upon coding theory, and attempting to ensure that the vectors of feature values are robust to 'errors' by one or more features, and that they are well-spaced with respect to Hamming distance or some other appropriate measure. Hamming distance is the standard measure for distance between bitvectors, such as vectors of values of binary features; for the sake of brevity, we will henceforth assume that Hamming distance is being used. Recall that the Hamming distance between two bitvectors is defined as the number of places in which they differ.

In the above example, the vectors 10101001 and 10101101 are very similar; this is reflected by the fact that the Hamming distance between them is only 1. Therefore, the partitions with respective names 10101001 and 10101101 can be thought of as being similar; they are 'close neighbours' in Hamming distance. It follows that our classifier may not be able to reliably distinguish between points in the two partitions. We therefore consider merging these two partitions into a single 'super-partition' or *region*. We are interested not so much in the information provided by the exact values of the names 10101001 or 10101101; we are more interested in the information obtained by knowing that the name is *either* 10101001 *or* 10101101.

The idea of merging partitions into regions can be generalised as follows. Let $y$ be a vector of feature values; $y$ is therefore the name of a partition, as discussed in Section 4.2. Let $\mathcal{P}(y)$ denote the set of points in the partition defined by $y$ – that is, $\mathcal{P}(y) = \{x : F(x) = y\}$. Finally, let $\delta(y_1, y_2)$ be the Hamming distance between a pair of bitvectors $y_1$ and $y_2$.

We define the $k$th-order region of the feature vector $y$ as the union of partitions whose names $y'$ are at a Hamming distance from $y$ of less than or equal to $k$:

$$\begin{aligned} R_k(y) &= \bigcup_{y':\delta(y,y' \leq k)} \mathcal{P}(y') \\ &= \{x : \delta(y, F(x)) \leq k\} \end{aligned}$$

The greater the value of $k$, the larger the regions. (Notice that $\delta(y,y) \equiv 0$, so $\mathcal{P}(y)$ will be contained in $R_k(y)$ for all $y$ and for any value of $k$.) Regions will certainly overlap, but in general two regions $R_k(y_1)$ and $R_k(y_2)$ need not be exactly the same, even if $y_1$ and $y_2$ themselves are separated by a Hamming distance of less than $k$. The regions form a *covering* of the data space: Each data point is contained in at least one region, and may be contained in more than one.

We can now construct region-conditional distributions in an analagous way to the partition-conditional distributions of EPE. The region-conditional distributions take the form:

$$\Pr(x \in \mathcal{C}_j | x \in R_k(y))$$

We can compute the entropies of the region-conditional distributions and take the expectation over vectors of feature values $y$, in exactly the same way as for EPE. We can therefore define the $k$th-order *expected covering entropy*:

$$\varepsilon_k = \sum_y \Pr(F(x) = y) H(\Pr(x \in \mathcal{C}_j | x \in R_k(y)))$$

Notice that if $k = 0$, ECE is equivalent to EPE. Hence, we will sometimes refer to EPE as the *zeroth-order entropy* of a feature set, and ECEs as entropies of first order, second order, and so on. ECE can be thought of as an extension to EPE, in which one or more features are permitted to take misleading values. The order $k$ of the entropy can be thought of as the maximum number of features which are permitted to be ‘wrong.’ Appropriate values for $k$ will depend on our feature set and data space. For instance, if we only have five features, then setting $k = 5$ renders ECE meaningless as each ‘region’ will contain the entire data space. In general, higher orders of ECE are more difficult tests for the ‘robustness’ of our feature set. As with a code, a feature set can be made highly robust to errors, but only at the cost of including massive redundancy.

Further theoretical properties of ECE will be discussed in Section 7.2, with reference to the part-of-speech tagging experiments outlined in Chapter 5; and in Section 8.5, regarding the Reuters document-classification experiments of Chapter 8. The latter is particularly interesting, as it provides a practical example of a situation in which ECE succeeds where the EPE metric fails to provide a useful criterion for feature selection.

## 4.6 Summary

In this chapter we have presented the concepts of partitioning and encoding, which provide two novel perspectives for feature selection. These concepts respectively motivate a new class of metrics using conditional entropy: We refer to these as Expected Partition Entropy (EPE) and Expected Covering Entropy (ECE). EPE can be motivated by considering either the conditional entropy or the expected relative entropy; ECE is an extension of EPE, which attempts to make our feature selection process robust to misleading values taken by one or more feature functions.

# Chapter 5

# Part-of-Speech Tagging: Experimental Setting

## 5.1 Introduction

In this chapter we describe the setting and parameters of experiments conducted to evaluate the relative performance of existing feature selection metrics described in Chapter 3, and new metrics introduced in Chapter 4. The setting is based on a modified form of part-of-speech tagging for English words, using labelled training data derived from the Penn Treebank [MSM95]. As we shall see, the setting is rich enough for feature selection to be an interesting problem, and for the differences between metrics to become apparent; but not so complex as to render feature selection prohibitively difficult or time-consuming.

It is worth noting that the experiments make very little attempt to employ any prior knowledge of the part-of-speech tagging domain. The metrics used characterise 'good' feature sets in quite general terms: The wrappers simply seek feature sets which give rise to high accuracy in predicting the tags of unseen words with a particular classifier, and the filters measure the 'goodness' of a feature set using feature frequencies or information-theoretic quantities. None of the metrics used is based on any special properties of the part-of-speech tagging problem; hence, the relative performance of the metrics can be expected to generalise to other domains.

After establishing some definitions and notation, we discuss the properties of the data and features. We continue by briefly reviewing the classifiers and feature selection metrics used; the classifiers were introduced in Chapter 2, existing metrics in Chapter 3, and new metrics in Chapter 4. We then introduce a new class of *accelerated* feature selection search algorithms, and detail the exact parameters for our selection of feature subsets. The subsets obtained are assessed according to classification accuracy on test data in Chapter 6; and using the EPE and ECE metrics in Chapter 7.

## 5.2 Data Points and Features

### 5.2.1 Basic Definitions

The general setting chosen for the experiments is a simple form of unknown-word part-of-speech tagging for English words. The training and test data were derived from sets of pre-tagged text files taken from the Penn Treebank WSJ corpus [MSM95]. Each word constitutes one data point, with the part-of-speech tag as its label. Non-words consisting only of digits or punctuation marks were excluded from the training data; capitalisation was ignored, with all words being treated as entirely lower-case; however proper names were included. The training data consisted of 992 157 data points, with 38 438 distinct words and 41 different tags.

Features are defined by substrings of five letters or less, including beginning-of-word and end-of-word symbols. A given feature takes the value 1 if its substring is present, in which case the feature is said to be *active*; otherwise the feature takes the value 0 and is said to be *inactive*. In the terminology of Chapter 2, we are using *binary features* with the presence of particular substrings as their *indicators*. Models using these features are implementing a simplified form of traditional part-of-speech tagging since, as we discuss further in Section 5.2.3, they do not consider the surrounding context in which a word occurs.

As usual when dealing with binary features, we sometimes abuse terminology by using the word ‘feature’ for both a feature function and its indicator. However, it is important to keep in mind that indicators and feature functions are two different things. The former are events – in this case, particular substrings being present – while the latter are functions mapping the set of possible words to the binary set $\{0,1\}$. In order to reflect this, we write strings and substrings in the form `red`; the feature function with the presence of `red` as its indicator is written $f[\texttt{red}](x)$, or more simply as `[red]`.

For purposes of actually incorporating our features into models, things are somewhat more complicated. Strictly speaking, the features included in our model architectures are active on predicate/label pairs (in which the predicates are substrings), not simply on predicates. Hence, $f[\texttt{red}](x)$ is really shorthand for a class of features, one for each possible part-of-speech tag: $f[\texttt{red},\texttt{NOUN}](x)$, $f[\texttt{red},\texttt{VERB}](x)$, and so on. However, all of our feature selection algorithms treat each equivalence class of features as a unit; we either include or exclude the entire class $f[\texttt{red}](x)$. We therefore use simplified notation and terminology in which a 'feature' is a function of the form:

$$f[\texttt{red}](x) = \begin{cases} 1 & \text{if } x \text{ contains the substring } \texttt{red} \\ 0 & \text{otherwise} \end{cases}$$

**Example 5.1:** Consider the English word `writing`, which can be either a noun or a verb. Let `1` and `9` be the universal beginning-or-word and end-of-word markers. The string `1writing9` contains the following 34 substrings of five symbols or fewer: `1, w, r, i, t, n, g, 9, 1w, wr, ri, it, ti, in, ng, g9, 1wr, wri, rit, iti, tin, ing, ng9, 1wri, writ, riti, itin, ting, ing9, 9writ, writi, ritin, iting, ting9`.

Each substring gives rise to a binary feature: `[ting9]`, `[iting]`, and so on. Notice that given feature cannot be active 'more than once'; the `[i]` feature takes the value 1 on this data point, despite the fact that the letter i occurs twice. In other words, our features take the value 1 if their substring occurs *at least once* on the word, and 0 otherwise. □

### 5.2.2 Properties of Features

We can at once make several observations regarding the properties of our features. First of all, the set of all possible features is very large. Given that any string of letters could in principle occur in a document – if only as an abbreviation – we immediately have $26^5 + 26^4 + 26^3 + 26^2 + 26$ or more than twelve million possible features. This figure will be multiplied still further when beginning- and end-of-word symbols for a given substring are included. In order to reduce our pool of available features to a more manageable size, we consider only those substrings which occur in our training set. As we shall see, this reduces the size of our pool of possible features to about 7000 for a typical 5000-word training set, rising to about 35 000 for a set containing roughly 800 000 words.

Notice that, even if we use only the substrings which occur in our training set, we still have a great many irrelevant and redundant features. For instance, in Example 1 above `[ting9]` is almost redundant if our feature set also contains `[ing9]`, as the former feature is unlikely to give us much additional help in predicting part-of-speech tags. Furthermore, the features `[1]` and `[9]` are entirely irrelevant, since *every* word is guaranteed to have a beginning and end. They are part of a broader class of irrelevant features; ones such as `[ri]` and `[ng]` are unlikely to give us any useful information about the part-of-speech tag of the word. Indeed, they could well lead to overtraining; if our training set happens to include a large number of nouns containing the substring `ng`, our learner could erroneously associate this substring with nouns. The task of weeding out such irrelevant features is an important challenge for any feature selection algorithm.

Using all 34 possible features to describe the data point `1writing9` seems excessive, to say the least. In the case of a common word such as this, it might be advantageous to choose features that give us a good chance of identifying the word uniquely. It seems reasonable to suppose that in a typical training or test set, `1writing9` is likely to be the only point on which the features `[1writ], [riti]` and `[ting9]` are active. For most practical purposes, these three features provide as much information about the data point `1writing9` as the full set of 34 features. However, we should not reject the remaining 31 features out of hand; for instance, the feature `[ing9]` – that is, the one corresponding to the suffix -ing – may well prove useful for distinguishing between nouns and verbs.

Striking an appropriate balance between ‘specialised’ features which are good for identifying particular common data points, and ‘generalist’ features which can identify uncommon and unseen data points, is an important task for any feature selection process. The above discussion also illustrates the importance of choosing good *combinations* of features, not just features which perform well in isolation.

**Example 5.2:** In our chosen setting, words of three letters or less can be identified with no ambiguity by a single feature: For instance, `1and9`, `1for9`, and `1a9`. In principle, this allows a learner to efficiently recognise and classify short words, many of which are very common. □

It should be noted that, even if our feature set allows us to uniquely identify a particular word, this does not guarantee that the word will be correctly classified. One possible problem is the presence of misleading features; for instance, while `[1and9]` uniquely identifies a data point as being the conjunction ‘and,’ our classifier may also note the features `[1an]` and `[nd]`, which could be associated with different part-of-speech tags. Not all classifiers will be sufficiently sophisticated to assign the correct tag to `1and9` in this case. This danger can be reduced somewhat by careful feature selection.

### 5.2.3 Context-Sensitive Features

The accuracy of our classifiers could be increased by introducing features which considered the context of a given word. For instance, we might find that ‘writing’ was more likely to be a noun if it was immediately preceded by the word ‘new.’ It would be simple enough to introduce a feature `[1new9-]` which was active exactly when a word was immediately preceded by ‘new’. Alternatively, we could define a feature `[1new9-1writing9]` which was active on the word-pair ‘new writing’.

The introduction of context-sensitive features would be particularly useful in dealing with words which have more than one possible tag. Suppose that the word ‘writing’ is classified as a verb in 80% of the instances in our training set, and as a noun in the other 20%. A classifier will give each data point what it deems to be the most probable part-of-speech tag, given the set of features active upon it. If it always classifies ‘writing’ as a verb – which is intuitively the ‘best it can do’ without considering context – then it will still be wrong about 20% of the time.

However, the use of context-sensitive features would significantly complicate the task of feature selection, and it was not attempted in our experiments. The introduction of context-sensitive features in this setting is a possible topic for future research.

## 5.3 Classifiers

The classifiers employed in this chapter are types of *feature-based probabilistic model* or FBPM; FBPMs are defined in Chapter 2. We suppose that each data point $x$ falls into exactly one of countably many classes $C_i$. As we have already indicated, the classifiers take the form $\Pr(x \in C_i | F(x) = y)$; that is, they assign a probability distribution over categories to a data point, given the value of our feature set on the data point.

The classifiers arise from *model architectures*, which are schemes for incorporating a variable set of features into a probabilistic model. We use a collection of labelled training data to choose what we hope will be a good feature set. Having chosen a feature set, a classifier can then be used to predict the label of previously unseen data points; its accuracy can be assessed by running it on a held-out set of labelled test data. We employ two different types of classifier in our experiments: These are the Naive-Bayes and Maximum-Entropy model architectures. The theoretical properties and previous applications of these models are discussed in Sections 2.3.3 and 2.3.4. In Sections 5.3.1 and 5.3.2 we briefly review the model architectures, with particular attention to their implementation in our chosen experimental setting.

### 5.3.1 Naive-Bayes

The naive-bayes (NB) model architecture gives rise to a family of simple feature-based classifiers, as detailed in Sections 2.3.3. At this point, it is worth noting that naive-bayes models do not contain any free parameters; and that they assume that features take values independently of one another.

The naive-bayes models live up to their name in our chosen setting of part-of-speech tagging with features defined by substrings. The presence of a given substring will certainly be dependent on the presence or absence of other substrings; indeed, modelling such dependencies may be crucial for successful classification. While NB classifiers can be moderately successful in this setting, they are drastically outperformed by the more sophisticated maximum-entropy models which we discuss in Section 5.3.2. On the other hand, NB models do have the advantage of simplicity; they can be trained and evaluated significantly more quickly than maximum-entropy models.

The code used to implement the naive-bayes classifier in this chapter was written by Miles Osborne.

### 5.3.2 Maximum Entropy

Our second model architecture is the maximum-entropy (ME) scheme, which was introduced in 2.3.3. In contrast to NB models, ME classifiers can model dependencies between features; and they include a set of real-valued free parameters or *weights*, which can be used to assign different relative importances to the members of a feature set.

The same set of labelled training data used to select a feature set is also employed to adjust the weights to optimal or near-optimal values. We take a ‘black box’ approach to the weights; they are adjusted by a training algorithm which remains fixed, and we instead try to optimise performance of the classifier by varying its feature set. The code used to implement the maximum-entropy models in this chapter was written by Rob Malouf [Mal02].

### 5.3.3 Aside – Classifi cation with Maximal Feature Set

Classification was carried out with the set of all available features, including a separate feature for each distinct word, in order to to provide a reference point for our results. This maximal feature set contained some 83 000 features. The mean percentage accuracies obtained with tenfold cross-validation were 84.96 and 86.11 for the NB and ME classifiers, respectively. If we include data points consisting only of digits and punctuation marks – which are very common and easy to classify – then the accuracies increase to 90.02 for NB and 89.65 for ME, which is in line with typical baseline results for part-of-speech tagging.

Selecting smaller feature sets with the techniques described in this chapter led to lower classification accuracy. However, moderately high accuracies of approximately 70% were achieved with as few as one-tenth the maximum number of features. Furthermore, the objective of our feature selection was not really to develop new methods of part-of-speech tagging, but rather to provide a suitable arena in which the *relative* performance of different feature selection metrics could be assessed. In the latter respect the experiments were very successful, as clear differences between the metrics became apparent and interesting properties of the new conditional-entropy metrics could be observed.

## 5.4 Feature Selection Metrics

As discussed in Chapter 3, any feature selection process requires an evaluation function or *metric* which provides a quantitative measure of the desirability of a given feature set. By attempting to optimise the value of our metric, we hope to obtain a feature set which gives rise to a more effective model. In this section, we briefly describe the metrics employed in this chapter, with particular attention to their implementation in our chosen experimental setting. Section 3.2.2 includes a more detailed consideration of the role of a feature selection metric, and the properties of existing metrics. Chapter 4 gives motivation for and definitions of the new feature selection metrics and discusses their theoretical properties.

Table 5.1 summarises the metrics used in this chapter, including abbreviations for their names and a rough indication of their complexity. Selection of features was also carried out entirely at random in order to provide a baseline for our results. It should be noted that the two simplest measures, frequency-based cutoff and naive mutual information, are *hierarchical* metrics; that is, they place the features in a fixed order of desirability, effectively ignoring any possible interactions between them. (See Chapter 3 for further details.) The other metrics used are non-hierarchical.

As Table 5.1 indicates, several of the metrics used in this chapter are *information filters* – that is, they make use of measures derived from information theory such as entropy, relative entropy, or mutual information.[1] Information measures generally re-

[1]See Chapter 3 or Cover and Thomas for formal defi nitions of these quantities [CT91].

| Name | Full Name | Type | New | Complexity |
|---|---|---|---|---|
| RND | Random selection | Not applicable | No | Very low |
| FBC | Frequency-based cutoff | Distance filter | No | Very low |
| NMI | Naive mutual-information | Information filter | No | Low |
| KS | Koller-Sahami metric | Information filter | No | Moderate |
| NB | Naive-Bayes | Wrapper | No | Moderate |
| ME | Maximum-entropy | Wrapper | No | High |
| EPE | Expected Partition Entropy | Information filter | Yes | Moderate |
| ECE | Expected Covering Entropy | Information filter | Yes | Very high |

Table 5.1: Summary of Metrics Used in Experiments

quire estimates of the probability distributions for data points and their associated feature values. In all cases, the estimate used is the empirical distribution:[2] If a data point $\tilde{x}$ appears $c(\tilde{x})$ times in a training set with $N$ elements, then its empirical probability is simply $c(\tilde{x})/N$.

A quantitive assessment of the computational cost of each metric, in terms of the time required to select a fixed number of features with our chosen experimental parameters, appears in Section 5.6.3.

## 5.4.1 Existing Metrics

**Frequency-based Cutoff:** Keep the features which occur most frequently in the training set, and discard the others.

**Naive Mutual Information:** Compute the mutual information of each feature with the class variable. The greater the mutual information, the more desirable the feature. Let $I(U;V)$ denote the mutual information between two random variables $U$ and $V$. In our chosen setting, we have:

$$I(f_i; x \in C_j) = H(\Pr(x \in C_j)) - H(\Pr(f_i(x) = y | x \in C_j))$$

[2]The empirical distribution and other possible approximations for use in information measures are discussed further in Chapter 3.

If we simplify our notation by letting $x_j$ denote the event $x \in \mathcal{C}_j$ and writing $f_i = y$ instead of $f_i(x) = y$, we have:

$$I(f_i;x_j) = \left( -\sum_j \Pr(x_j) \log \Pr(x_j) \right) - \left( -\sum_j \Pr(x_j) \sum_y \Pr(f_i = y|x_j) \log \Pr(f_i = y|x_j) \right)$$

Notice that the first term in the above expression is constant for all features; and that in the second term, $y$ can take only two possible values – 0 and 1. The naive mutual-information is therefore quite simple to compute.

**Koller-Sahami Metric:** Find the partition-conditional label distribution for the pool of all available features $\mathcal{F}$:

$$p_{\max}(\mathcal{C}_i|y') = \Pr(x \in \mathcal{C}_i)|\mathcal{F}(x) = y')$$

Then find the partition-conditional label distribution for the candidate feature set $F$:

$$p_F(\mathcal{C}_i|y) = \Pr(x \in \mathcal{C}_i)|\mathcal{F}(x) = y)$$

where $y$ is the projection of $y'$ onto the the reduced feature set $F$. Minimise the expected Kullback-Liebler divergence between the two:

$$\sum_{y'} \Pr(y') D\left(p_{\max}(\mathcal{C}_i|y')||p_F(\mathcal{C}_i|y)\right)$$

**Naive-Bayes Wrapper:** Assessment using the Naive-Bayes learner with the training set, candidate feature, set and a held-out test set. The higher the accuracy of the learner, the better the feature set.

**Maximum-Entropy Wrapper:** Exactly the same as the Naive-Bayes wrapper, except that the maximum-entropy learner is used for assessment.

### 5.4.2 New Metrics

**Expected Partition Entropy:** Minimise the expected entropy of the partition-conditional label distributions arising from the candidate feature set:

$$\sum_y \Pr(y) H(p_F(\mathcal{C}_i|y))$$

where as before, $p_F(\mathcal{C}_i|y) = \Pr(x \in \mathcal{C}_i)|\mathcal{F}(x) = y)$.

**Expected Covering Entropy:** As above, but instead of partitions we use *regions*, which are defined in terms of Hamming distance. (Recall that the Hamming distance between two bitvectors is the number of places in which they differ.) Each partition is defined by a vector of feature values $y$; the region $R_k(y)$ of a given partition is made up of the partitions whose defining vectors are at a Hamming distance $\delta$ of less than or equal to some fixed value $k$:

$$R_k(y) = \{x : F(x) = \hat{y} | \delta(y, \hat{y}) \leq k\}$$

Reasonable values for the maximum Hamming distance $k$ are 1, 2, or 3; if $k = 0$ then ECE is equivalent to EPE.

## 5.5 Search Strategies

### 5.5.1 Background

We seek to evaluate the performance of each feature selection metric by finding feature sets which receive good scores with respect to the metric, and evaluating their performance. A pool of available features with $n$ elements will have $2^n$ distinct subsets. Given that between 7000 and 35000 features are available in our chosen setting, the number of possible subsets is astronomically high, and exhaustively evaluating each distinct subset is effectively impossible.

Practical feature selection methods use a variety of heuristics for searching the possible subsets more quickly, while still having a reasonable chance of finding a ‘good’ feature subset. As noted in Chapter 3, a typical feature selection procedure has four distinct elements:

1. **Generation procedure:** A means of generating candidate feature subsets.

2. **Evaluation function:** A function which produces a numerical ‘score’ for each candidate feature subset.

3. **Stopping criterion:** A means of deciding when to terminate the search.

4. **Validation procedure:** Any feature selection method must be *validated* by assessing the performance of the FBPM which arises from it.

Our evaluation functions are the ones described in Section 5.4. Our principal validation procedure is finding the accuracy on held-out test data of naive-bayes and maximum-entropy models with our chosen feature sets, as described in Chapter 6. We will also explore the use of our new metrics to assess feature sets obtained using existing metrics, as we describe in Chapter 7. We seek to assess the performance of our metrics on twenty different subset sizes for each training set. This determines our stopping criterion: We will halt when all twenty subsets are obtained or when our chosen metric reaches its optimum possible value, whichever comes first.

A suitable generation procedure now needs to be chosen. Recall that generation procedures include a *starting point* somewhere in the space of available subsets, and a *search strategy* for moving through the space. In our experiments we employ a novel generation procedure which we call *accelerated forward selection*.

### 5.5.2 Accelerated Forward Selection

In a standard forward selection algorithm, we begin with the empty feature set and add features one at a time. At each step we assess all the candidate features available for inclusion in our model and add the one which gives rise to the highest-scoring feature set, with respect to our chosen metric.[3] Unfortunately, the non-hierarchical metrics used in this chapter – that is, all of them apart from frequency-based cutoff and naive mutual information – are too slow for traditional forward selection to be practical with the data sets used in this chapter.

We therefore speed things up by introducing an element of randomness to our feature selection process. The basic idea is simple: Start with the empty feature set, but instead of adding one feature at a time assess a number of randomly selected blocks of features, and adopt the best one for use in our model. This can result in significantly faster feature selection, as the following example illustrates.

[3]Forward selection and its relation to other search techniques are described in greater detail in Section 3.2.1.

**Example 5.3:** Suppose we have a pool of 1000 available features and wish to obtain a subset of 100 features. If we were using forward selection, we would have to call our evaluation metric for $1000+999+998+\ldots+901$ or 95050 distinct subsets. On the other hand, if we evaluate 100 randomly chosen blocks of 10 features each, choose the one whose addition gives rise to the best feature set, and repeat 10 times, we only have to call the evaluation function 1000 times. All else being equal, the feature selection process has been sped up almost a hundredfold, and we can still reasonably hope to obtain a ‘good’ subset with respect to our chosen metric.

Note that at each of 10 steps, we are evaluating 100 randomly chosen blocks of 10 features each. A total of 100 000 features are therefore considered for inclusion in our model – inevitably with some repetition. Thus, each of our 1000 available features is almost certain to be considered for inclusion at least once. □

It is important to note that accelerated selection offers a means of employing feature selection metrics which would otherwise be prohibitively slow. For instance, Koller and Sahami note that the KS metric is extremely complicated; they propose an elaborate means of approximating it through so-called Markov blankets, one which requires us to make a number of undesirable assumptions [KS96]. Accelerated feature selection allows us to precisely calculate the KS metric – and others of similar complexity, such as wrapper approaches and EPE – at the cost of significantly reducing the scope of our search.

Accelerated forward selection has not to our knowledge been used before, although it is somewhat similar in spirit to the random recombination of parameters in genetic algorithms. (The use of genetic feature selection algorithms is discussed further in Section 3.2.1.) Indeed, one could define an accelerated genetic algorithm which assessed multiple feature subsets obtained by addition or removal of randomly selected blocks. However this would suffer from the usual drawback of genetic algorithms – namely, that of requiring the assessment of a very large number of feature subsets. Accelerated genetic algorithms were therefore rejected in favour of a ‘higher-resolution’ use of accelerated forward selection – that is, applying the available computational resources to assessing more blocks and reducing the size of block added at each step.

An accelerated approach will in general produce less useful feature sets than a traditional forward selection algorithm, as it is carrying out a less extensive search of the space of available subsets. However, while the results of using a given metric with accelerated forward selection will probably be *quantitatively* worse than the same metric applied to traditional forward selection, one would not expect a *qualitative* difference in the relative usefulness of different metrics. In simpler terms, if metric *A* is better than metric *B* at traditional forward selection – in the sense of giving rise to a model with greater accuracy on held-out test data – then this should still hold true in the accelerated setting.

The hypotheses of quantitative improvement and no qualitative change are borne out by the experiments described in Section 6.9. If the size of block added is reduced, then in an important sense the accelerated selection process becomes closer to traditional forward selection – after all, the classic forward selection algorithm is equivalent to simply having a 'block size' of 1. In practice, reducing the block size while keeping the metric fixed has little effect on performance, and the relative success of different metrics with the same block size remains unchanged. It is also worth noting that use of an accelerated algorithm is more than sufficient to attain an optimal value for the Koller-Sahami metric, and to improve metrics such as EPE to near-optimal values.

**Hierarchical Metrics:** It should be noted that accelerated feature selection is inappropriate for hierarchical metrics. If features are ranked in a fixed order of desirability, independently of how they are combined in a feature subset, then there is no point in evaluating randomly selected blocks of features. With a hierarchical metric, we know what the highest-scoring subset of *n* features is; it is simply the subset containing the *n* features with the highest individual scores. We therefore employ conventional forward selection and backward elimination with the hierarchical FBC and NMI metrics. Interestingly, the hierarchical metrics are significantly outperformed by more complex metrics which require the use of accelerated selection. It would seem that in this setting, the advantage of adopting a more sophisticated evaluation function more than offsets the disadvantage of being able to assess comparatively few feature subsets.

**Other Accelerated Techniques:** Several obvious variations on accelerated forward selection are possible. One of them is accelerated backward elimination, in which we start with the set of all available features and iteratively remove randomly selected blocks. Another is to allow our algorithm to 'backtrack' by periodically removing a block of features instead of adding one (or vice versa for accelerated backward elimination). In our experiments, however, we restrict ourselves to accelerated forward selection without backtracking. Because of the large random element already present in the search process, it seems very implausible that the choice of starting point or introduction of backtracking will have a drastic effect.

## 5.6 Parameters for Part-of-Speech Tagging Experiments

### 5.6.1 Training and Test Data

Our set of labelled data for these experiments was derived from 25 sections of tagged text from the Penn Treebank WSJ corpus, each approximately 35 000 words in length. [MSM95] (Recall that for our purposes, each word is one data point.) We conducted experiments with two different sizes of training set. In the larger of the two, we merged 23 of the files into a large training set of approximately 800 000 words, while holding out two files as test sets. One test set was used for feature selection by wrappers, which themselves require a test set; the other was used for evaluation. This setting was used for evaluation of the FBC, NMI, KS, NB, and EPE metrics.

The maximum-entropy wrapper was too slow for use in feature selection with the large training set. We therefore conducted a second set of experiments in which the training set was a sequence of 5000 words from one of the files, and sequences of 1000 words from two other files were used as test sets. This reduced training set was small enough to allow implementation of the ME wrapper. For comparison, feature selection was also carried out in this setting using EPE and the NB wrapper. The same test sets were used for both small and large training sets.

As noted in Section 5.2, the experiments employed binary features with the presence of substrings of 5 letters or less (including beginning- or end-of-word symbols) as their indicators. For each training set, the collection of features appearing in the training set was adopted as the pool of candidate features for inclusion in the model. This resulted in approximately 7200 possible features for a 5000-member training set, and approximately 36 000 for an 800 000-member training set. The maximum size of subsets examined were 7000 and 35 000, respectively.

### 5.6.2 Accelerated Forward Selection

The experiments used accelerated forward selection, returning results for twenty different subset sizes. The selection process was halted when it reached the maximum subset size or the optimum possible value for the metric. (In practice the latter did not occur except for the Koller-Sahami metric.) In each instance, 100 feature selection steps were carried out, each time adding a number of features equal to 1% of the desired final subset size. At each step 10 randomly chosen blocks were assessed the one which gave rise to the highest-scoring feature set was added to the feature set. Every five steps, the feature set was output for later assessment, for a total of twenty outputs. The parameters are summarised in Tables 5.2 and 5.3. See Table 5.1 for a summary of the metrics' properties and a key to the abbreviations for their names.

| **Parameter** | **Value** |
|---|---|
| Size of block to add at each step | 350 |
| Number of blocks assessed at each step | 10 |
| Number of steps | 100 |
| Size of final feature set | 35000 |
| Frequency of output | 5 |
| Total outputs | 20 |
| Metrics not used | ME, ECE |

Table 5.2: Parameters for Large Training Sets

Even with the use of accelerated techniques, the ECE metrics were too slow for practical feature selection in this domain; the same was true of the ME wrapper for the larger feature set. For the part-of-speech tagging problem, these slow metrics were reserved for evaluation rather than selection of feature sets. Evaluation by the maximum-entropy and naive-bayes learners is described in Chapter 6, and assessment by ECE

| **Parameter** | **Value** |
|---|---|
| Size of block to add at each step | 70 |
| Number of blocks assessed at each step | 10 |
| Number of steps | 100 |
| Size of final feature set | 7000 |
| Frequency of output | 5 |
| Total outputs | 20 |
| Metrics not used | RND, FBC, NMI, KS, ECE |

Table 5.3: Parameters for Small Training Sets

in 7. Successful feature selection by ECE in the Reuters document-classification domain is presented in Chapter 8.

Tenfold cross-validation was carried out by repeating the feature selection process 10 times for each of the remaining metrics, each time with a different split between the training and test sets, and finding the mean of the results. This enabled us to control for the effect of different test sets, as well as the random element present in accelerated forward selection. In the case of completely random selection, it seemed prudent to increase the number of runs in order to ensure an accurate baseline; we therefore carried out feature selection twice for each of the 10 training/test splits, for a total of 20 different results to be averaged instead of the usual 10.

### 5.6.3 Computational Cost of Metrics

Table 5.4 quantifies the computational cost of the metrics used in the part-of-speech tagging domain. It records the time in hours required to select a given number of features using a Pentium 4 PC, giving mean values across the ten training/test splits. The parameters used for feature selection are those given in Tables 5.2 and 5.3; see Table 5.1 for a summary of the metrics' properties and a key to abbreviations of their names.

Recall that the FBC and NMI metrics are *hierarchical* measures; they rank features in a fixed order of desirability, irrespective of how they are combined in a feature set.

Therefore, in feature selection by FBC or NMI we only have to compute the metric value over the pool of all available features; in order to select a subset of *n* features, we simply choose the *n* highest-ranked members of the pool. In our chosen setting, selection by hierarchical metrics could be accomplished in at most a few minutes.

For non-hierarchical metrics such as KS, EPE, and the NB and ME wrappers, the metric must be recalculated for each candidate feature set. Feature selection with these metrics was therefore much slower. However, as discussed in Chapter 6, in many instances the non-hierarchical metrics gave rise to considerably greater accuracy on test data than the hierarchical ones.

| **Metric** | **Time to select features (hours)** | |
|---|---|---|
| | **7000 features** | **30000 features** |
| FBC | N/A | Negligible |
| NMI | N/A | Negligible |
| EPE | 6.3 | 28.7 |
| KS | N/A | 29.0 |
| NB | 8.4 | 41.1 |
| ME | 35.8 | N/A |

Table 5.4: Computational cost of metrics.

The difference in speed between the EPE and KS metrics was comparatively minor, and is probably not of general significance; in theory, the two metrics are of very similar computational complexity. It can be seen though that both the EPE and KS filters were considerably faster than the naive-bayes (NB) wrapper, which in turn was faster than the maximum-entropy (ME) wrapper.

## 5.7 Summary

In this chapter we have outlined the setting and parameters used to carry out feature selection in a part-of-speech tagging domain. More specifically, the task considered in these experiments was a simplified version of part-of-speech tagging for English words, in which features were defined by the presence or absence of particular substrings within each word, and words were considered in isolation from their contexts. Training data was derived from the Penn Treebank WSJ corpus. We began by presenting a novel feature selection strategy called *accelerated forward selection*. This strategy enabled significantly faster feature selection, at the expense of carrying out a less extensive search of the space of subsets.

Accelerated forward selection was carried out using the new EPE metric, and for comparison with the existing FBC, NMI, KS, NB and ME metrics. Selection of features at random was also conducted in order to provide a baseline for other results. In most cases, accelerated forward selection was fast enough for us to use large training sets of about 800 000 words, which contained about 37 000 features. However, the ME metric was prohibitively slow in this setting; we therefore carried out feature selection using the ME, NB, and EPE metrics with smaller training sets of 5000 words, which each contained approximately 7200 features. Feature selection using the extension of the EPE metric to ECE was too slow to be practical in this setting; successful experiments with ECE are presented in Chapter 8.

The properties of the feature sets obtained for the part-of-speech tagging task will be assessed in Chapter 6, which will examine the classification accuracy they obtain on test data; and in Chapter 7, which will examine the behaviour of the new conditional-entropy metrics on feature sets selected by a variety of methods.

# Chapter 6

# Part-of-Speech Tagging: Accuracy on Test Data

## 6.1 Introduction

This chapter is primarily devoted to comparing EPE to the baseline provided by random selection, and to existing feature selection metrics: FBC, NMI, KS, NB and ME. (A brief explanation of these abbreviations is contained in Table 5.1; detailed definitions of the metrics are given in Chapters 3 and 4.) We also compare the ME and NB wrappers to one another, and briefly consider the effect of varying our search parameters. The results in this section are largely presented in graphical form; more complete tables of results are contained in Appendix A.

Recall that EPE and NB were used for both large and small training sets; random selection, FBC, NMI and KS for large ones only; and ME for small ones only. Assessment was carried out using both the ME and NB learners. Accuracy of the ME models was consistently better than that of the NB ones; it was also less subject to change according to the feature selection metric used. This was to be expected, as the ME model architecture is considerably more sophisticated. However, interesting variations occurred within the results for each method of assessment, according to the metric used for feature selection.

## 6.2 EPE and Random Selection

Figure 6.1 compares the mean accuracies attained by EPE with random feature selection. In the ME setting, EPE is clearly better than random selection; with the NB learner, the advantage of EPE is less pronounced but still noticeable for most sizes of feature set.

Random selection performs particularly well in combination with the NB learner for relatively small and large feature sets. Indeed, it is on average better than EPE for the final subset size of 35 000. This may reflect the fact that overtraining will not occur with random selection, as well as the relative crudeness of the NB learner. In general these results should be treated with some caution because of the large random element present in both feature selection processes. In particular, the mean results for feature selection by EPE for the NB learner for between about 31 000 and 35 000 features appear to have been dragged down by some particularly bad 'worst-case' results; these results may not be indicative of generally poor performance of EPE for NB on very large feature sets.

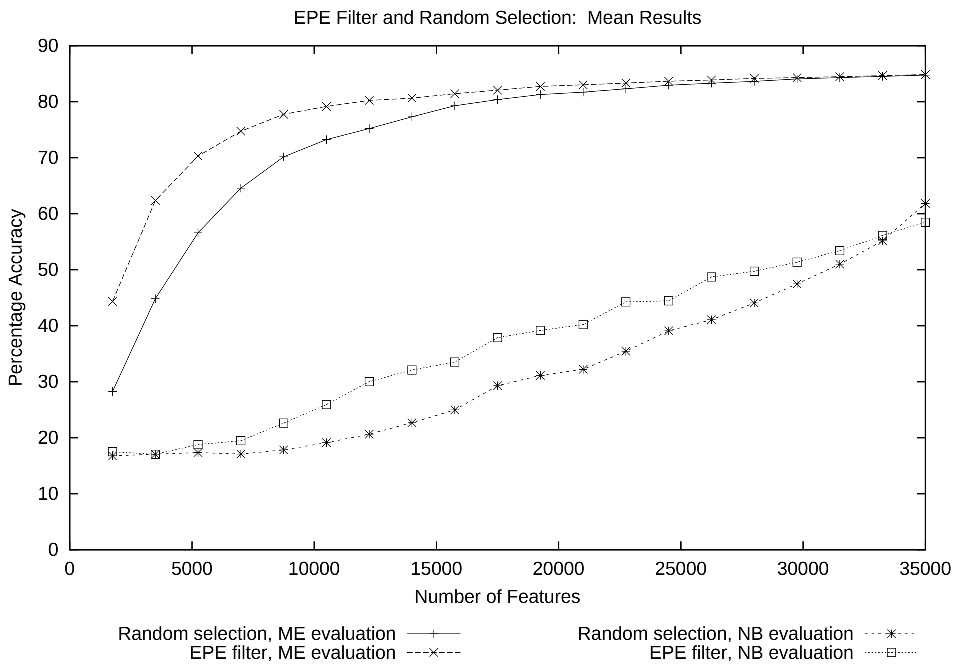

Figure 6.1: Mean accuracy for 10 runs with EPE metric and random selection.

## 6.3 EPE and Frequency-Based Cutoff

Figure 6.2 displays the mean accuracies for the EPE and FBC filters. When implemented using the ME learner, feature sets selected using EPE have a clear advantage. With the NB classifier, the results are somewhat less conclusive, with FBC sometimes performing better than EPE and sometimes not. It may be that FBC is particularly well-suited to the NB learner because, in different ways, they both ignore any possible interactions between the features.

However, EPE is still reasonably effective in the NB setting; it seems that considering interactions between features in our feature selection process can still be an effective strategy, even when the learner does not attempt to model these interactions directly. It is also worth noting that the best results of EPE are usually much better than the best results for FBC, as illustrated by Figure 6.3. This suggests that a more extensive search of the possible feature subsets with the EPE metric would significantly outperform FBC, even in combination with the NB learner.

Another occurrence worth noting is the drastic improvement in FBC results with the NB learner between 33 000 and 35 000 features. It seems that for the NB learner, discarding the least common 2000 or so features is a particularly effective strategy. This has interesting implications for the ideal size of training set; if including very uncommon features is a significant drawback, then perhaps using a small training set for feature selection will lead to an increase in accuracy as well as speed. This will be considered further in Section 6.9.

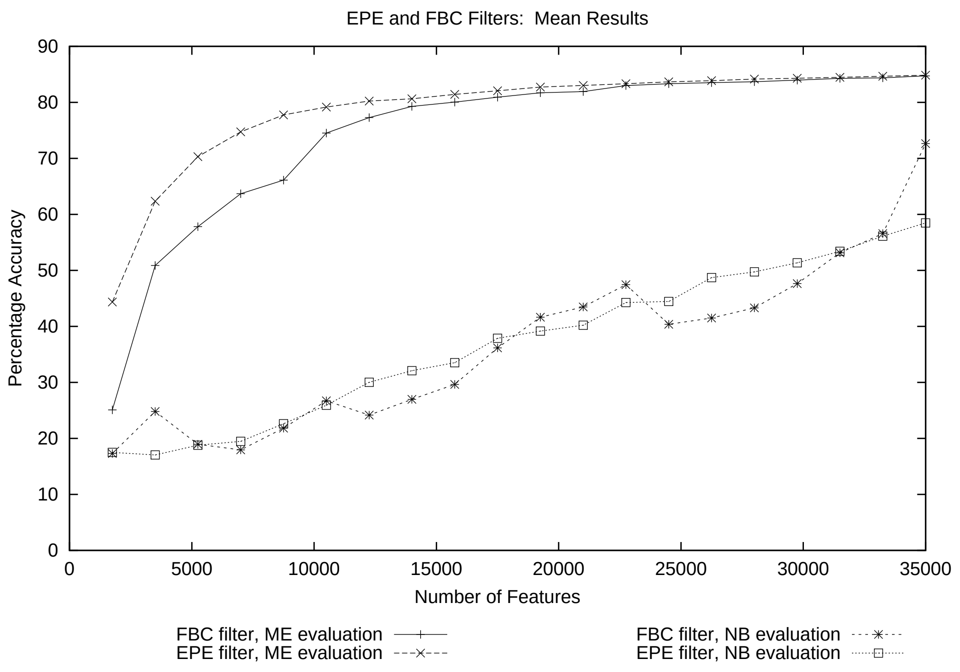

Figure 6.2: Mean accuracy for 10 runs with EPE and FBC metrics.

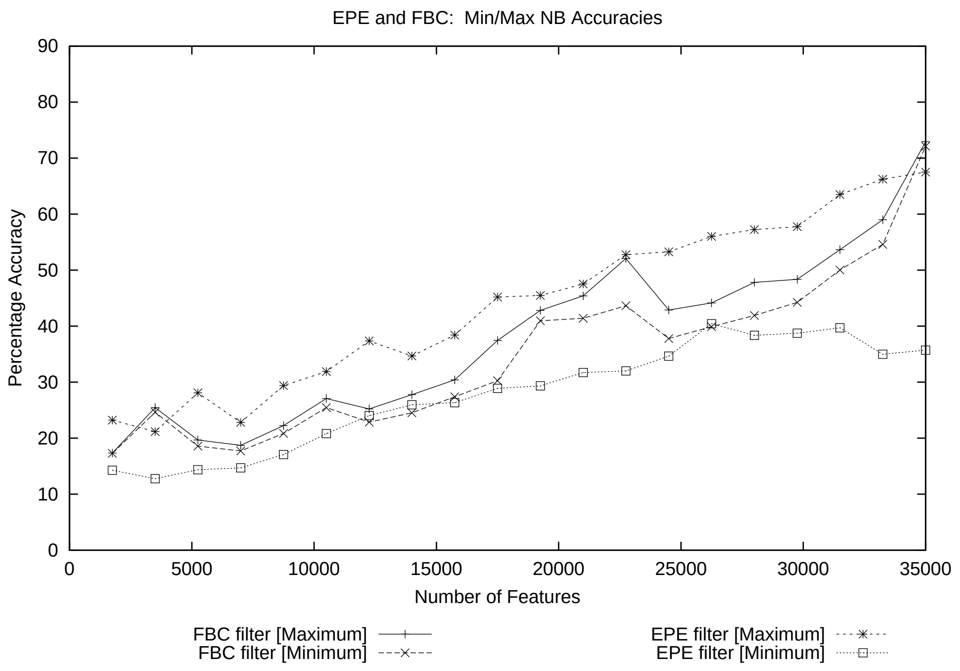

Figure 6.3: Min/max accuracy for 10 runs with EPE and FBC metrics.

## 6.4 EPE and Naive Mutual Information

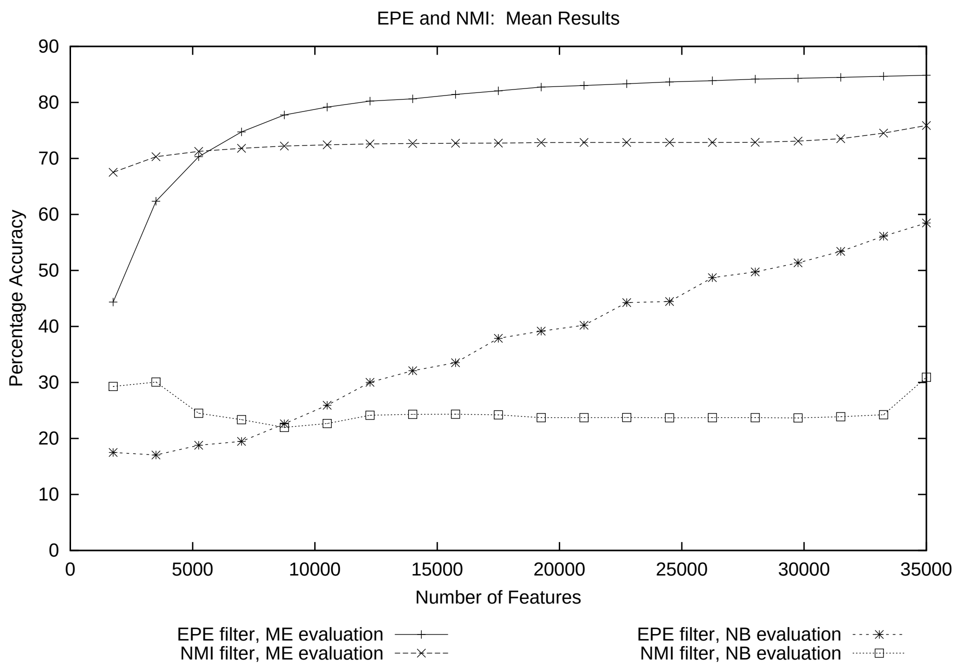

Figure 6.4: Mean accuracy for 10 runs with EPE and NMI metrics.

Figure 6.4 gives the mean accuracies for feature sets obtained using the EPE and NMI metrics. EPE clearly outperforms NMI for more than about 5000 features with the ME learner, and about 8000 features with the NB learner. Feature selection by EPE results in a steady rise in accuracy as the subset size increases, whereas the accuracy arising from NMI tends to remain static or actually decline. This is a very important result; it demonstrates that EPE can significantly outperform the very popular feature selection technique of NMI, despite the fact that NMI – being a hierarchical metric – can be optimised exactly, whereas with EPE we are forced to rely on a randomised search which in general will not attain the optimal possible EPE values.

In fact, for more than about 10 000 features NMI does worse than random selection. This may be because NMI positively encourages redundant features. Feature functions which provide virtually the same information as ones already in the feature subset will nevertheless be favoured by NMI, as long as they have a high mutual information with the class variable.

The performance of the NMI metric combined with the NB learner for small sets of up to 3500 features is particularly good, being unequalled by EPE until it reaches approximately 15 000 features. With the ME learner, NMI has a less dramatic but still noticeable advantage for sets of less than about 5000 features. However, it should be noted that there is significant variation in the results, illustrated by Figure 6.5.

Nevertheless, it seems that NMI generally has a slight advantage over EPE for selection of small feature sets, particularly in combination with the NB classifier. We hypothesise that the generally poorer performance of EPE with small feature sets may be due to overtraining. As with FBC, the advantage of NMI for small feature sets with the NB learner may be particularly strong because both NMI and NB ignore any possible interactions between features; NMI selects features which are 'good in isolation,' so it seems reasonable that the best such features are particularly well suited to an NB model.

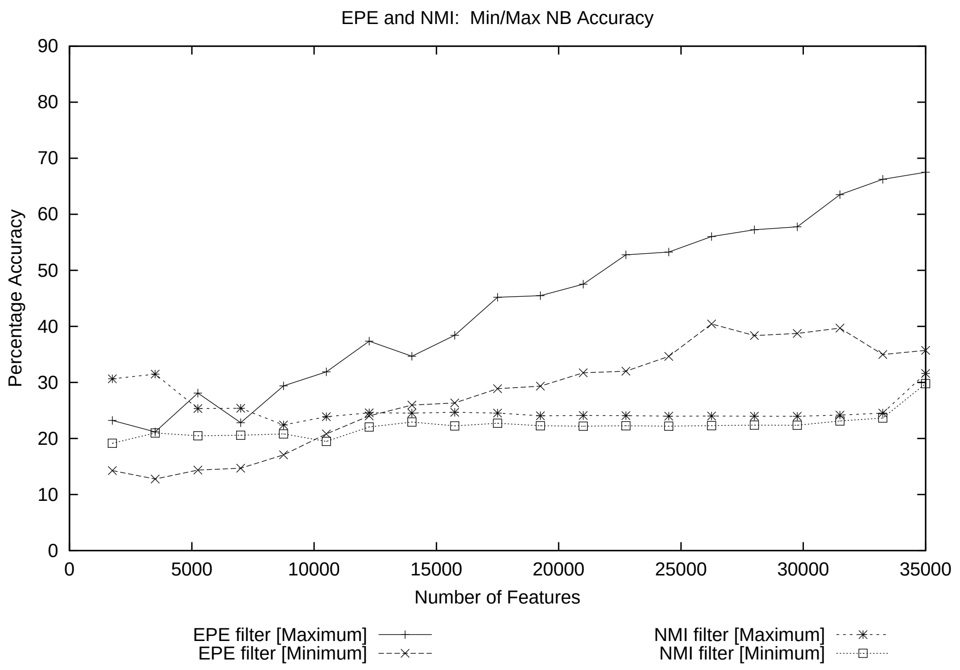

Figure 6.5: EPE and NMI: Min/max NB accuracy for 10 runs.

Figure 6.5 gives the minimum and maximum accuracies over ten runs for the NB learner. (Results for the ME learner were very consistent, with no substantial departures from the mean.) The much larger variation in results for EPE is unsurprising, as feature selection by NMI was completely deterministic; the only scope for variation arose from the different splits between training and test data. Beyond about 13 000 features, the worst results of EPE were still an improvement over the best ones of NMI.

## 6.5 EPE and the Koller-Sahami Metric

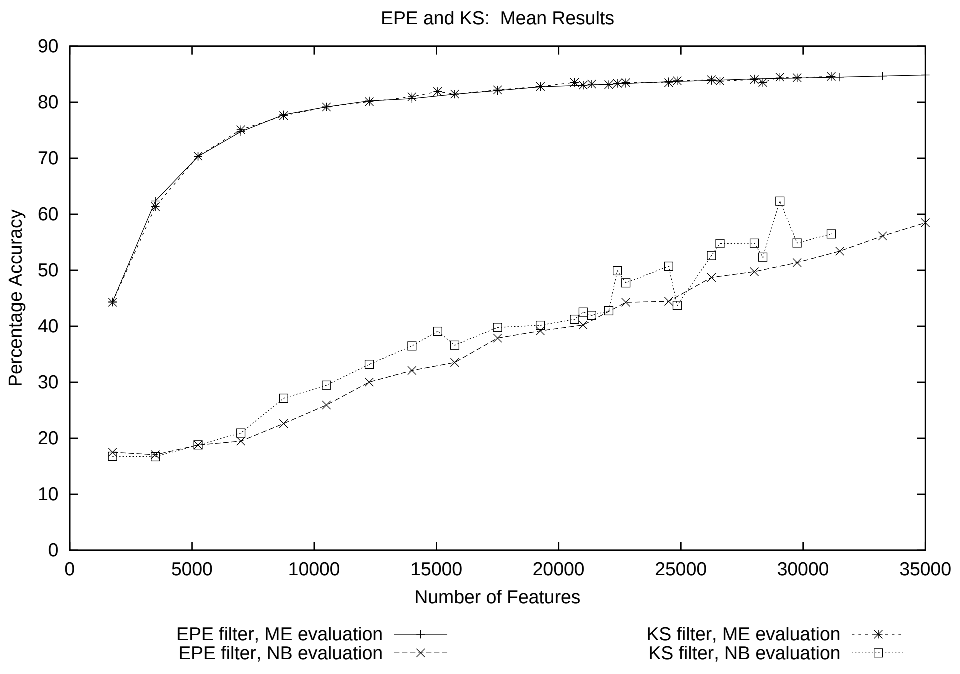

Figure 6.6: Mean accuracy for 10 runs with EPE and KS metrics.

Figure 6.6 compares mean results for the EPE and KS metrics. When carrying out accelerated forward selection, the KS evaluation function reached its optimal value for feature sets as small as 15 000 features and always reached it before attaining 32 000 features. The feature selection process was halted when the KS metric reached its optimal value; the additional points for the KS metric denote these final feature sets. The first of these terminations was at 15050 features; six occurred at or before 24 850 features; and the maximum size attained was 31 150. Table 6.1 gives the subset sizes at which they occurred, together with the accuracy of the maximum-entropy and naive-bayes learners on the final subsets.

As a result of the early halts to feature selection, the mean figures for the KS metric used to produce Figure 6.6 were over fewer and fewer feature sets as the number of features increased. This helps account for the rather erratic results for KS with the NB learner and large feature sets.

At first glance, the results of EPE and KS with the ME learner are virtually indis-

| Size of Final Subset | Maximum-Entropy | Naive-Bayes |
|---|---|---|
| 15050 | 81.90 | 39.10 |
| 20650 | 83.52 | 41.24 |
| 21350 | 83.24 | 41.94 |
| 22050 | 83.15 | 42.75 |
| 22400 | 83.35 | 49.91 |
| 24850 | 83.84 | 43.71 |
| 26600 | 83.76 | 54.76 |
| 28000 | 84.32 | 50.64 |
| 29050 | 84.48 | 62.34 |
| 31150 | 84.59 | 56.48 |

Table 6.1: Final subsets for KS with large training set

tinguishable, and KS is noticeably better for the NB learner. However, the results for KS with more than 25 000 or so features should be treated with considerable caution, as they are the averages of four runs or fewer and each run contains a significant element of randomness. In short, the apparently superior performance of KS with NB for large feature sets may be a coincidence. This is illustrated by Figures 6.7 and 6.8, which give the minimum and maximum accuracies over ten runs for the two evaluation methods.[1] Even with this note of caution, though, it appears that the KS metric gives rise to slightly higher accuracies on the NB learner.

However, the KS metric suffers from a very damaging drawback in comparison to EPE. KS tends to reach an ‘optimal’ value of zero for comparatively small feature sets, which are by no means optimal with respect to their performance on held-out test data. It may be possible to obtain considerable improvement by adding more features, particularly if a KS score of zero is reached for a relatively small feature set. Indeed, a feature subset optimal with respect to KS is not necessarily the best subset *of its size*; better results for a given subset size were sometimes obtained by EPE. This result clearly demonstrates an important general advantage of EPE over the KS metric.

[1]The ranges have been reduced slightly in Figure 6.7 so that the results for more than 5000 features can be clearly seen.

Recall that the KS metric seeks to minimise the expected Kullback-Liebler divergence from the partition-conditional distribution arising from the pool of available features to the one from our chosen feature subset.[2] In practice, a feature set with zero divergence (in terms of the KS metric) will in general be less than optimal in terms of its performance on held-out test data. In contrast, EPE can be thought of as having the uniform distribution as its reference point. It seems that the objective of EPE, which is both more general and more difficult to attain, gives rise to a more reliable method of feature selection. It appears that the concerns raised in Chapters 3 and 4 about the pool of available features not being a sufficiently general reference point are well justified in this setting.

[2]The pool of available features and the set of *all possible* features may be very different. In these experiments for instance, we have carried out a 'pre-selection' of 35 000 features from a set of over 12 million possible ones.

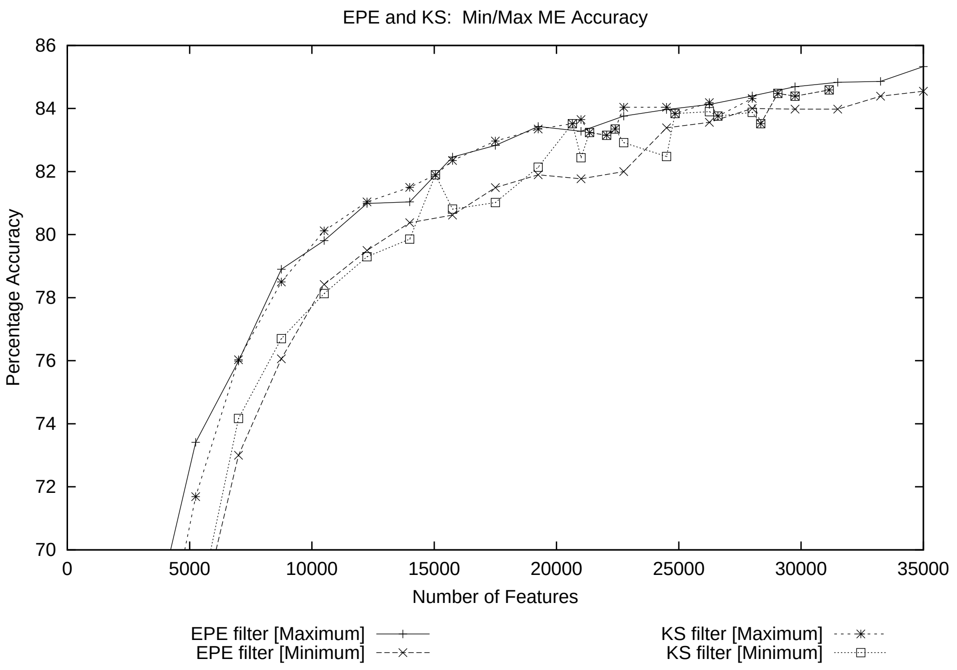

Figure 6.7: EPE and KS: Min/max ME accuracy for 10 runs.

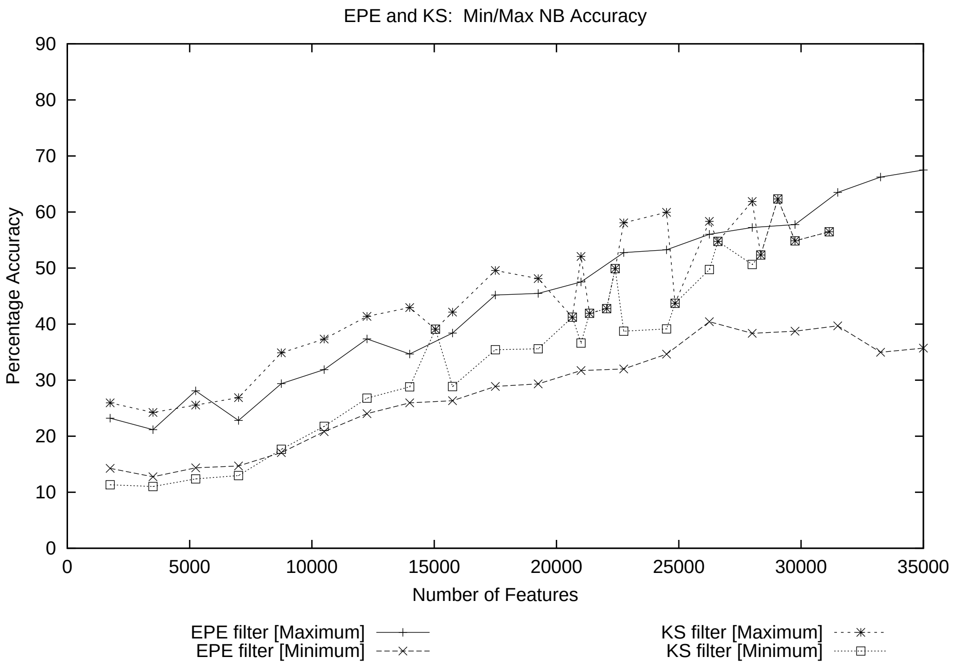

Figure 6.8: EPE and KS: Min/max NB accuracy for 10 runs.

## 6.6 EPE and the Naive-Bayes Wrapper

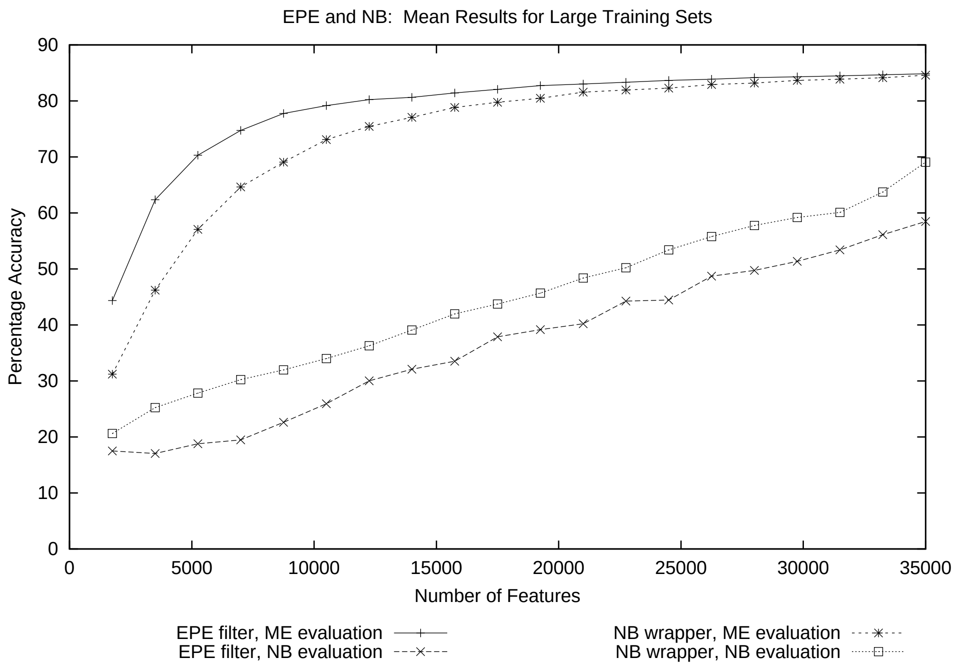

Figure 6.9: Mean EPE/NB accuracy for 10 runs with large training set.

Recall that the EPE and NB metrics were used with both small and large training sets; the respective mean results are given in Tables 6.9 and 6.10. It can immediately be seen that for both sizes of training set, EPE performs better in combination with the ME classifier, whereas the the NB wrapper outperforms EPE in selecting features for the NB classifier. It is not particularly surprising that NB outperforms EPE with the NB learner, since accuracy of an NB learner on held-out test data is precisely the criterion that the NB wrapper uses for feature selection; in a sense, the NB wrapper is optimised for such a task.

Much more interesting is the fact that the advantage of the NB wrapper does not carry over into the ME setting. It is our hypothesis that the optimisation of the NB wrapper for its own learner comes at the expense of comparatively poor performance on other learners. This may be particularly pronounced with ME and NB classifiers because the former takes account of possible dependencies between features, while the latter does not. It seems plausible that the two model architectures require qualitatively

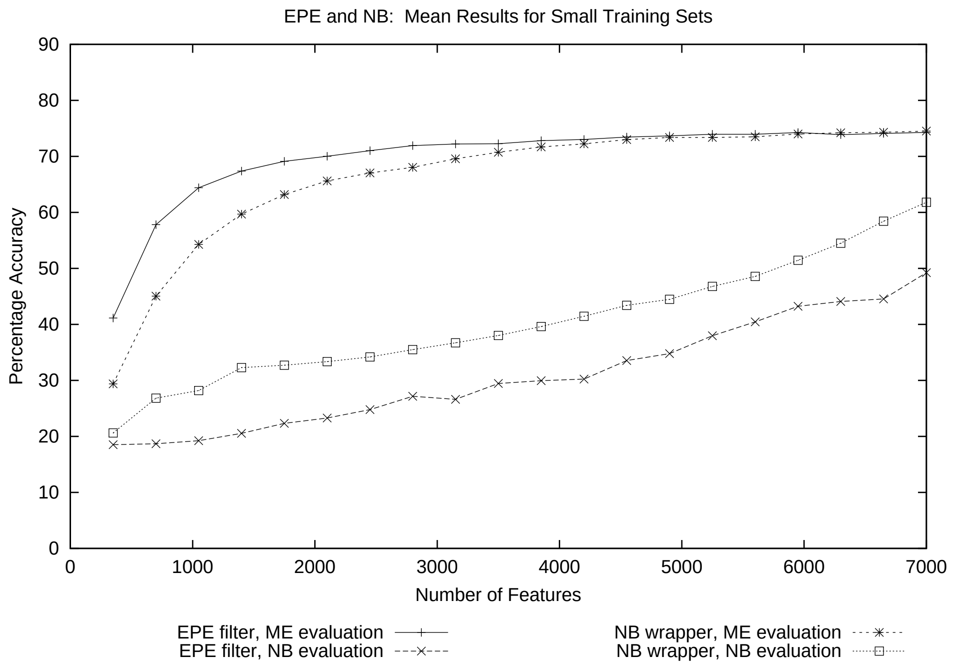

Figure 6.10: Mean EPE/NB accuracy for 10 runs with small training set.

different feature sets in order to attain good performance: NB requires features which are ‘good in isolation,’ whereas ME can better exploit relationships between features.

In short, it appears that ‘like should select for like.’ This hypothesis is supported by the comparisons of EPE with the ME wrapper and of the two wrapper methods with each other, as discussed in Sections 6.7 and 6.8.

## 6.7 EPE and the Maximum-Entropy Wrapper

In Figure 6.11 we present mean results for EPE and the ME wrapper. Because of the comparative slowness of the ME wrapper, it was used for feature selection only with small training sets. The two metrics gave rise to very similar results on both the ME and NB learners. Little additional commentary is needed at this point; however, it should be noted that the EPE metric was very much faster in this setting.

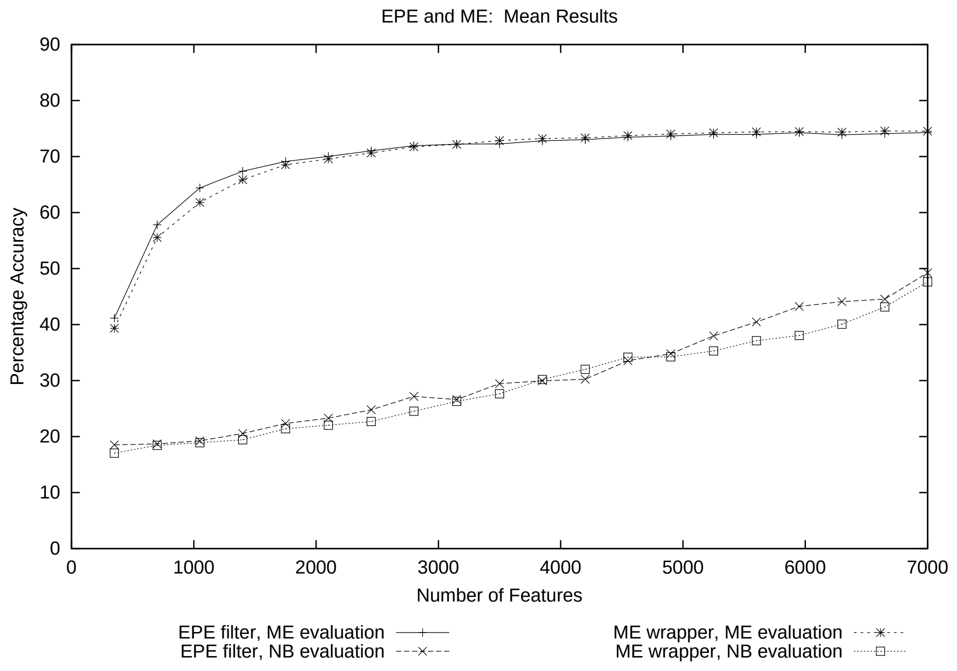

Figure 6.11: Mean EPE/ME accuracy for 10 runs with small training set.

## 6.8 Comparison of ME and NB Wrappers

Figure 6.12 gives mean results for the ME and NB wrappers. We can immediately see that the ME wrapper outperforms the NB wrapper when assessed using the ME learner; conversely, the NB wrapper performs better in assessment with the NB learner. This supports the hypothesis that 'like should select for like,' which was introduced in Section 6.6.

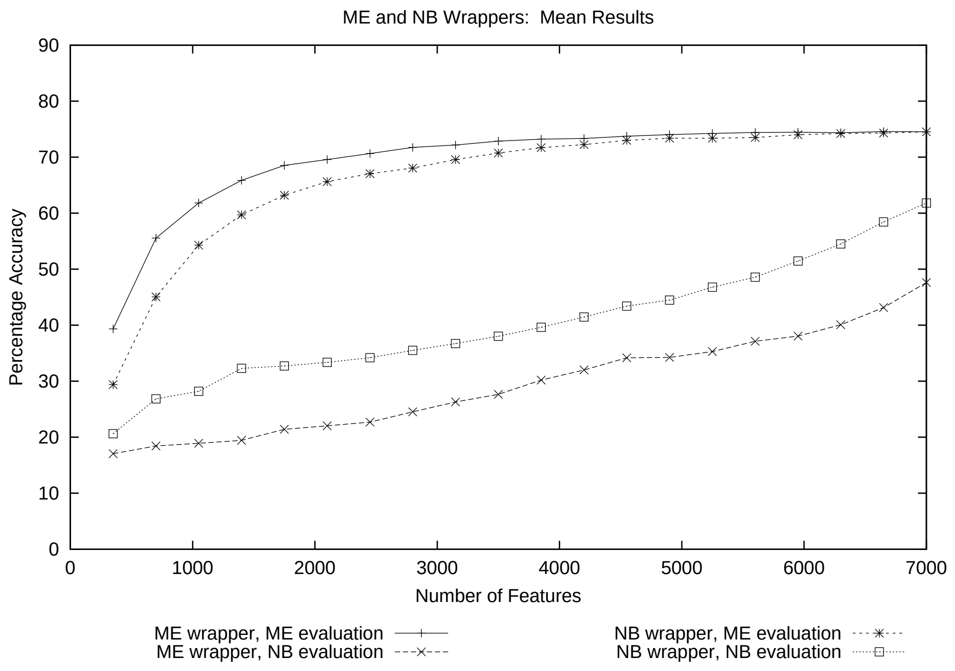

Figure 6.12: Mean ME/NB accuracy for 10 runs with small training set.

## 6.9 Aside – Search Parameters and Training Sets

It is interesting to compare the results obtained on equivalent subset sizes with the small and large training sets; they are shown in Figures 6.13 and 6.14 for ME and NB assessment, respectively. The accuracy obtained with the small training set is dramatically greater in the NB case. At their final subset size of 7000, feature sets obtained using the small training sets had accuracies more than 30% greater than those of their counterparts obtained with large training sets. The difference in accuracy under ME assessment was less marked but still clearly noticeable.

It is not immediately obvious whether the faster improvement in performance with the small training set was due to the training set itself, the different search parameters, or a combination of both. (Recall that the same test sets were used for both sizes of training set.) We therefore investigated the effect of different search parameters with the same training set. This was done by repeating feature selection with the small training sets for the EPE metric, with the number of features added at each step increased from 70 to 350. The exact parameters used are given in Table 6.2; as usual, we carried out tenfold cross-validation. The results are given in Figure 6.15.

| **Parameter** | **Value** |
|---|---|
| Size of block to add at each step | 350 |
| Number of blocks assessed at each step | 10 |
| Number of steps | 20 |
| Size of final feature set | 7000 |
| Frequency of output | 2 |
| Total outputs | 10 |
| Metrics used | EPE |

Table 6.2: Large-Block Parameters for Search Strategy Comparison

It is interesting, and perhaps surprising, to see that the block size has very little effect on accuracy. It would seem that the more rapid improvement in performance with small training sets was primarily due to the properties of the training sets themselves. The clearest difference between the small and large training sets is that while very

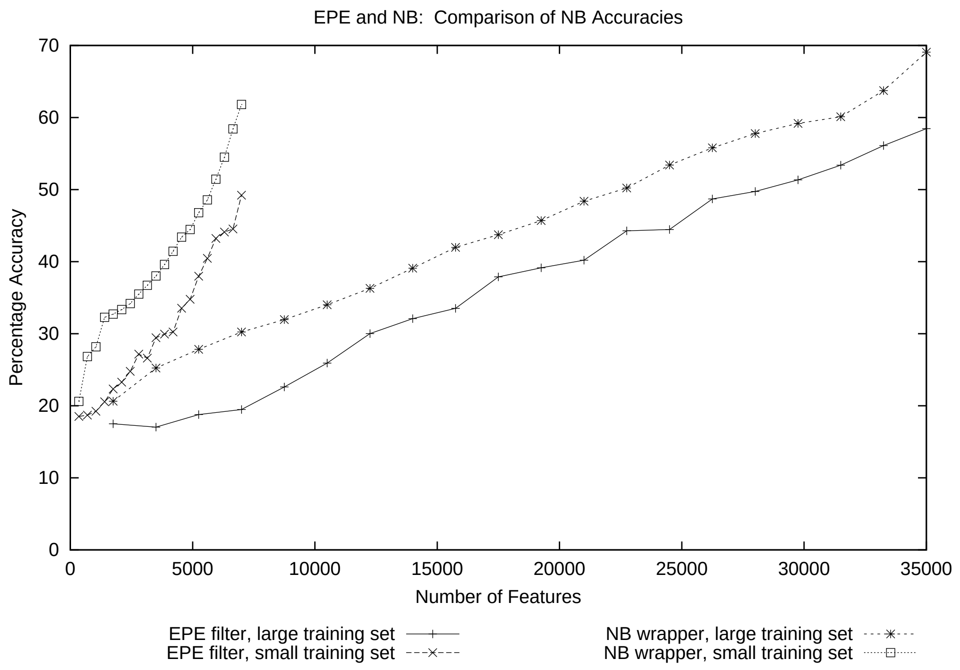

Figure 6.13: NB results for small and large training sets over 10 runs.

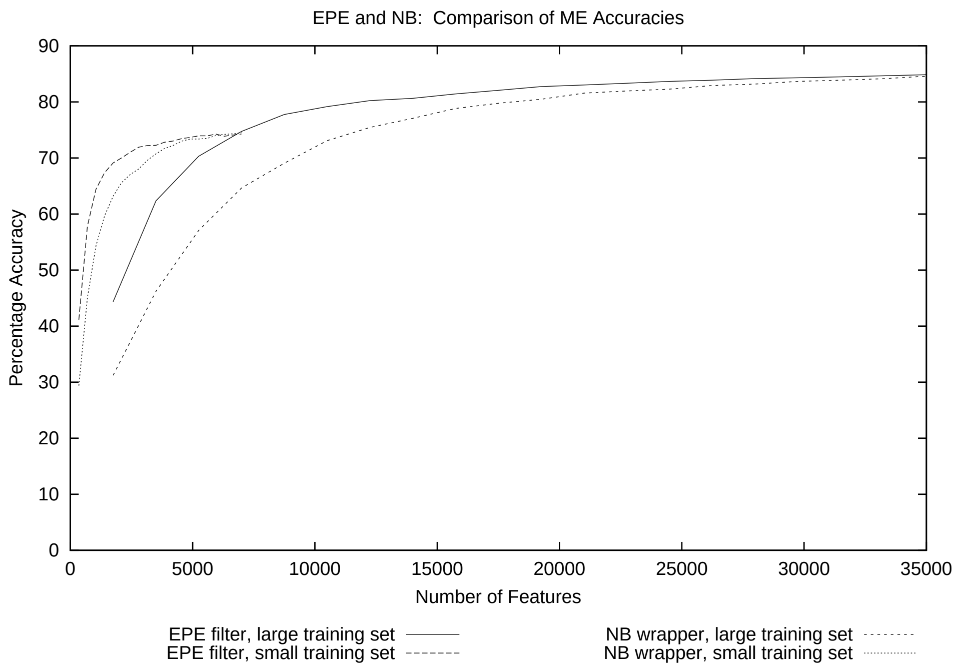

Figure 6.14: ME results for small and large training sets over 10 runs.

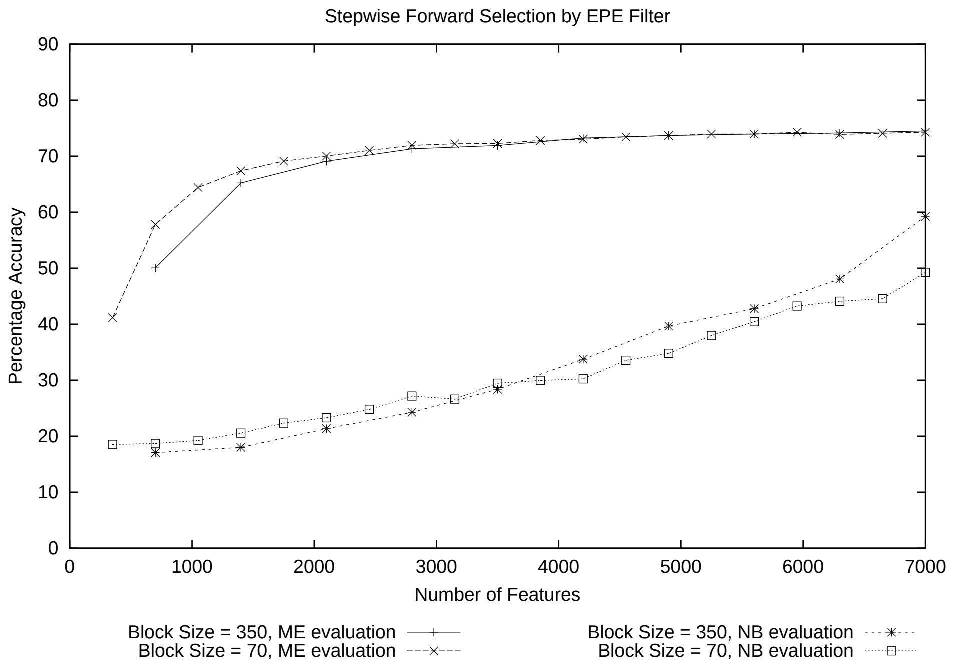

Figure 6.15: Effect of block size on accelerated forward selection by EPE.

common features are likely to be contained in both, the large training sets will contain many more uncommon features. In using a small training set we are favouring common features; effectively, we are carrying out a form of 'pre-selection' by frequency-based cutoff (FBC).

As Section 6.3 notes, FBC is a reasonably good feature selection metric, particularly for the NB learner. It therefore seems plausible that combining FBC and EPE by reducing the size of the training set leads to a more rapid improvement in results than EPE alone. However, the larger training sets (and correspondingly larger feature sets) do allow higher maximum accuracies to be obtained; for instance, the accuracy of the ME model levels off at about 85% for large training sets, compared to about 70% for small ones.

In summary, it seems that uncommon features should be handled with care, but can be useful if chosen well. It would be interesting to experiment with more sophisticated ways of combining FBC and EPE, or indeed FBC and other feature selection metrics. For instance, one could start with the 2000 or so most common features out of 35 000 and select additional features by EPE. Investigation of such techniques is a possible topic for future research.

## 6.10 Summary

In this section we have investigated the classification accuracy arising from feature sets derived with the new EPE metric; and existing metrics including FBC, NMI, and KS filters and ME and NB wrappers. Accuracy was assessed by incorporating the feature sets into NB and ME classifiers, and evaluating the performance of the models on test data. Tenfold cross-validation was carried out in each case, and the results presented are average values for ten different runs of accelerated forward selection, with different splits between training and test set. The behaviour of the metrics was consistent in each repetition of accelerated forward selection, and relative differences between the metrics are clearly visible from the mean accuracies. Hence, our results can be regarded as a very reliable illustration of metric behaviour.

The primary goal of the experiments was to investigate the properties of EPE, but the behaviour of other metrics was also examined. The NB wrapper was found to be superior to EPE in selecting features for the NB classifier, but not for the ME model. This result suggested a more general hypothesis: That a wrapper approach to feature selection produces feature sets which are optimised for use with the learner which provided the wrapper metric, but this good performance will not necessarily carry over to other model architectures. This conjecture that ‘like should select for like’ was supported by a direct comparison of the ME and NB metrics. As a slight diversion, the effect of varying the parameters for accelerated forward selection was also investigated. It was found that the size of ‘step’ was not crucial, but that varying the size of training set could significantly alter the results.

The new EPE metric performs very well in comparison with the other measures examined. Broadly speaking, it is faster than ME; more reliable than KS (which, unlike EPE, suffers from premature attainment of its optimal value); leads to greater accuracy than FBC or NMI (except for very small feature sets); and produced greater accuracy than NB on the ME learner. The advantages of EPE were clearly visible in each case.

# Chapter 7

# Part-of-Speech Tagging: Assessment by Conditional-Entropy Metrics

## 7.1 Introduction

In this chapter, we investigate the behaviour in the part-of-speech tagging domain of the new conditional-entropy metrics: Expected Partition Entropy (EPE) and its extension Expected Covering Entropy (ECE). Recall that EPE is sometimes called *zeroth-order entropy*, while the ECEs are referred to as entropies of first order and higher. The results in this chapter do not entail using the conditional-entropy metrics to select new feature sets; rather, we compute the EPE and ECE values for existing feature sets selected using EPE and other metrics.

We concentrate on feature subsets obtained by accelerated forward selection using the NMI, EPE, and NB metrics and the large training sets. These three metrics were chosen in order provide a reasonably representative sample of the measures used: NMI is a hierarchical metric, EPE is a non-hierarchical information-based filter, and NB is a wrapper. We begin by briefly examining the properties of ECE in our experimental setting, and then present results for the entropy-based metrics in graphical form. Tables containing more complete results, including those for other metrics and training sets, are given in Appendix B.

It was our conjecture that reducing the entropy of our feature set will lead to better accuracy on held-out test data; and conversely, that feature sets giving rise to better accuracy have low entropy. The first part of the conjecture is strongly supported by the success of EPE in comparison to other metrics for accelerated forward selection, as discussed in Chapter 6. We now examine the second part: That regardless of how they were obtained, feature sets giving rise to more accurate models will have lower entropy.

Assessing the entropy of feature sets which had already been obtained required significantly fewer computational resources than selecting new feature sets. Evaluation by entropy thus allowed the use of the ECE family of metrics.

**Aside – Evaluation by Entropy for Other Metrics:** Entropies were also computed for feature sets obtained using the other metrics: FBC and KS for large training sets, and NB, EPE and ME for small training sets. Complete tables of entropy results are contained in Appendix B. The results for other metrics were essentially consistent with the conclusions drawn below; in this section we have concentrated on only three metrics for the sake of clarity.

## 7.2 Properties of ECE

The new Expected Covering Entropy (ECE) metrics are considerably more difficult to compute than EPE.[1] The basic reason is that ECE requires us to consider the Hamming distances between all distinct pairs of vectors of feature values which appear in the training set. Using the terminology of Chapter 4, we will sometimes refer to a vector of feature values as a *name*.

Recall that ECE requires us to specify a maximum Hamming distance $k$. Each *region* is defined by a vector of feature values $y$. The region $R_k(y)$ consists of exactly the data points whose names are at a Hamming distance from $y$ of less than or equal to $k$. The regions form a *covering* of the data space, in that each data point is in at least one region and may be in more than one. The regions of ECE play the same role as the partitions of EPE. If $k = 0$, then ECE is equivalent to EPE. Therefore, we sometimes

[1]See Chapter 3 for formal defi nitions of these metrics.

refer to EPE as the *zeroth-order entropy* of a feature subset; ECE with $k = 1$ is the first-order entropy, and so on.

**Example 7.1:** Suppose that we have a set of binary features:

$$F = \langle f_1(x), f_2(x), f_3(x), f_4(x)\rangle$$

Each of the $f_i$ maps the set of possible data points $X$ to the binary set $\{0,1\}$. The feature set $F$ can be thought of as a vector-valued function:

$$F : X \mapsto \{0,1\}^4$$

So, for two particular data points $x_1$ and $x_2$ we might have names $y_1 = F(x_1) = 1001$ and $y_2 = F(x_2) = 0101$. Recall that the Hamming distance $\delta$ between two bitvectors is defined as the number of places in which they differ; hence, $\delta(y_1,y_2) = 2$. If $k \geq 2$ then $x_1 \in R_k(y_2)$ and $x_2 \in R_k(y_2)$, because $\delta(F(x_1,F(x_2)) \leq k$.

Conversely, $x_1$ is not in $R_1(y_2)$ and $x_2$ is not in $R_1(y_1)$. However, if we had a third data point $x_3$ such that $y_3 = F(x_3) = 1101$, then $x_1$ and $x_2$ would be contained in $R_1(y_3)$ because $\delta(y_1,y_3) = \delta(y_2,y_3) = 1$. Notice that if $y_1 = y_2$ then $\delta(y_1,y_2) = 0$. So by definition, every data point $x$ is contained in $R_k(F(x))$ for any value of $k$.

For a more concrete example of how this works in our chosen setting, suppose that the indicators for our four features are the substrings `1un`, `re`, `bar`, and `ing9`, where as usual 1 and 9 respectively denote the beginning and end of a word. Let the data points $x_1$ and $x_2$ be the words `undoing` and `redoing`. As before, we have $y_1 = F(x_1) = 1001$ and $y_2 = F(x_2) = 0101$, with $\delta(y_1,y_2) = 2$. The maximum Hamming distance $k$ can be thought of as the greatest allowable difference between two points which are considered to be 'close together' with respect to our chosen feature set. If $k \geq 2$, then `undoing` and `redoing` are 'similar enough' to be regarded as neighbours; otherwise they are not. $\square$

For a set of $N$ distinct names, finding the ECE requires the computation of $N^2 - N$ Hamming distances. The number of distinct names can itself be very large; if we have a set of $n$ features, each taking $r$ values, then $N$ is at most $r^n$. The difficulty of evaluating the ECE therefore grows very quickly with the size of the feature set. It proved impractical to carry out feature selection by ECE for sets of thousands or tens of thousands of features. Instead, we investigate its behaviour on feature sets obtained using other metrics.

We can in general expect the ECE of a given feature set to be greater than its EPE, as the following example illustrates:

**Example 7.2:** Suppose that our feature set contains the following features: `[foo]`, `[1g]`, `[in]`, `[n9]`, `[gr]`. The data points `green` and `grin` then have names 01011 and 01111, respectively. For the purposes of this example, we simplify things by assuming that the two data points have the same empirical probability $p_0$, `green` is the only data point with the name 01011, and `grin` is the only point named 01111.

The Hamming distance between the names is 1; `green` and `grin` are thus considered 'near neighbours' for the purposes of first-order entropy. However, the distributions of part-of-speech tags for `green` and `grin` will be very different. Suppose that we have $\Pr(\texttt{green}, \texttt{ADJ}) = 0.95$, $\Pr(\texttt{green}, \texttt{NOUN}) = 0.05$, $\Pr(\texttt{grin}, \texttt{VERB}) = 0.6$, and $\Pr(\texttt{grin}, \texttt{NOUN}) = 0.4$.

Recall that the entropy of a probability distribution $p(u)$ is defined as:

$$H(p) = -\sum_u p(u) \log p(u)$$

Given the simplifying assumptions above, entropy of the partition with name 01011 is:

$$-(0.6 \log 0.6 + 0.4 \log 0.4) = 0.466$$

to 3 decimal places, where logs are taken to base $e$. Similarly, the partition with name 01111 has an entropy of 0.199. The contribution of these two partitions to the zeroth-order entropy or EPE is thus:

$$p_0(0.466 + 0.199) = 0.665 \times p_0$$

Now let us consider the first-order entropy. As usual, we denote the $k$th-order region of a name $y$ by $R_k(y)$. Because the Hamming distance between 01011 and 01111 is 1, 01011 falls into $R_1(01111)$ and 01111 is contained in $R_1(01011)$. For simplicity, assume that $R_1(01111)$ and $R_1(01011)$ contain only the data points `green` and `grin`.[2]

[2]In general the two regions could well contain different sets of points.

A point in either of the two regions then has the tag NOUN with probability 0.225, ADJ with probability 0.475, and VERB with probability 0.3. The entropy of the region $R_1(01111)$ is therefore:

$$-(0.225\log 0.225 + 0.475\log 0.475 + 0.3\log 0.3) = 1.050$$

By symmetry, the region $R_1(01011)$ also has an entropy of 1.050. The contribution of the two regions to the first-order entropy is therefore:

$$p_0(1.050 + 1.050) = 2.100 \times p_0$$

So in regarding these two points as equivalent, we have increased our uncertainty. More specifically, if we do not permit ourselves to rely on the single feature [in] to distinguish these two data points, our uncertainty as to the labels of green and grin goes up significantly. $\square$

In order to investigate ECE, we concentrate on feature sets derived using the NMI, NB and EPE metrics for large training sets. We present the first-, second-, and third-order entropy for each of our chosen feature sets; for comparison, we also include the zeroth-order entropy or EPE. Computing the ECE on large training sets containing some 800 000 words proved impractical; instead, we found the ECE for the first 5000 words of each training set. Even this simplified measure produced interesting results, as we discuss below.

## 7.3 Results

ECE was used to assess the results obtained by the NMI and EPE filters and NB wrapper in combination with large training sets. For each combination of metric and training set size there were 20 different feature subset sizes and 10 different runs of accelerated forward selection, for a total of 200 outputs. For each metric ECE was calculated on the first 5000 data points in each training set for all 200 outputs, and the mean values were found for each subset size. The zeroth- through third-order entropies of feature sets obtained using NB and EPE are displayed in Figures 7.1 and 7.2; results for NMI are in Figure 7.3, with the third-order entropies omitted as they were virtually identical to the second-order ones. Entropies of the same order for the three different metrics are compared in Figures 7.4, 7.5, and 7.6.

All entropies use logarithms taken to base $e$. Note that the graphs use base-10 log scales to display the entropy, as entropies initially declined quite rapidly and then very slowly as the size of feature set increased.

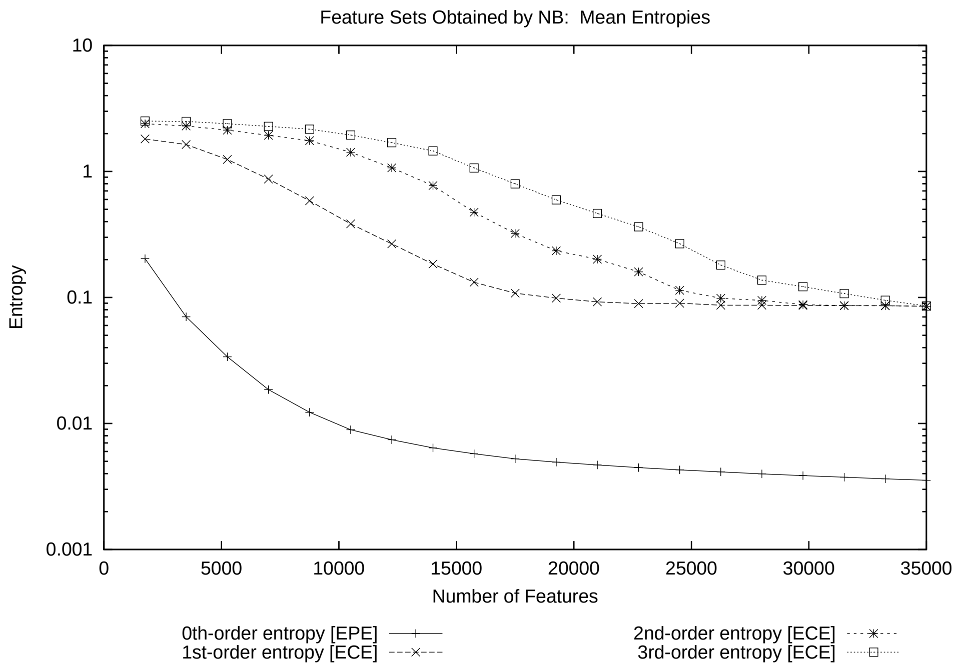

Figure 7.1: Entropies of order 0 through 3 for NB selection.

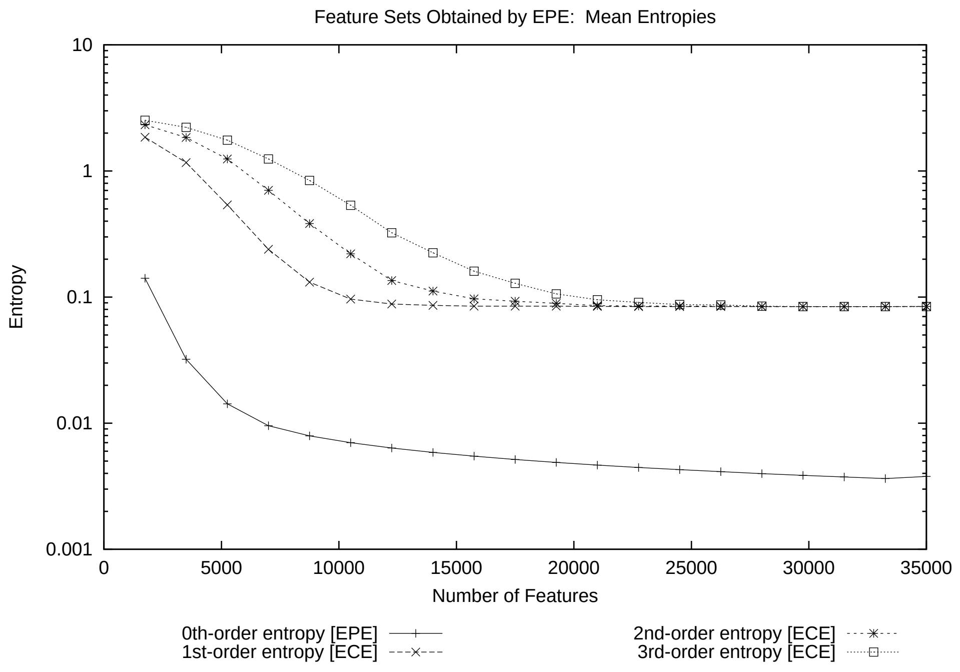

Figure 7.2: Entropies of order 0 through 3 for EPE selection.

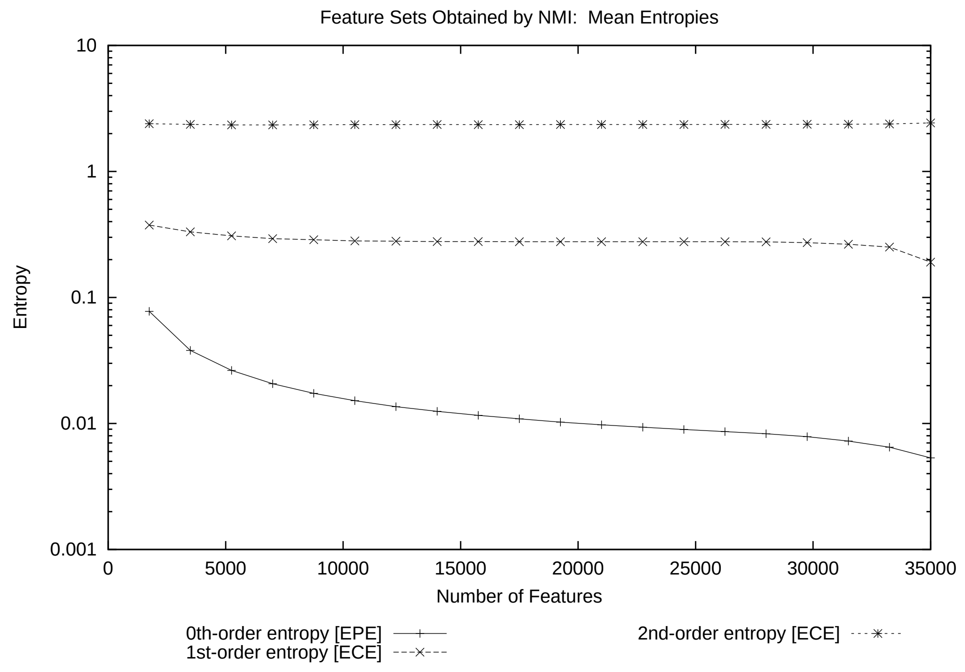

Figure 7.3: Entropies of order 0 through 2 for NMI selection.

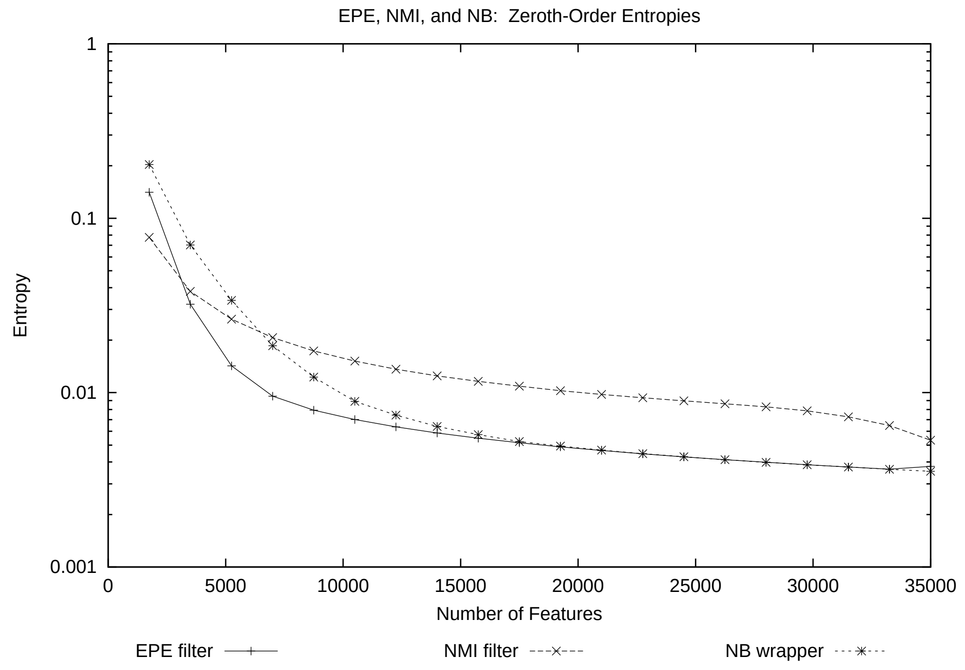

Figure 7.4: Zeroth-order entropies for EPE, NMI, and NB.

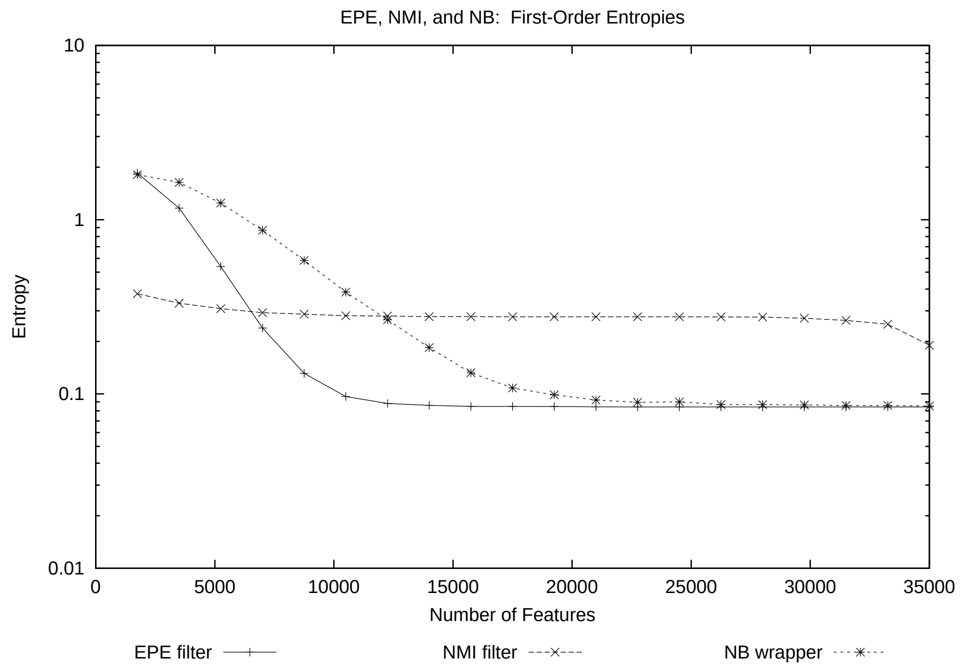

Figure 7.5: First-order entropies for EPE, NMI, and NB.

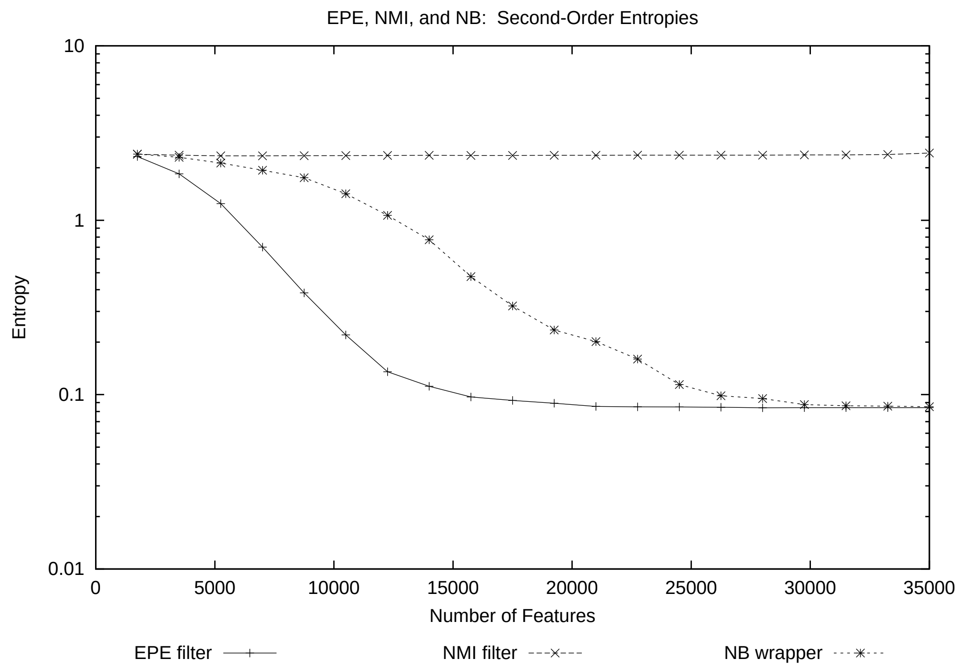

Figure 7.6: Second-order entropies for EPE, NMI, and NB.

## 7.4 Discussion

### 7.4.1 Interpretation of Subset Entropy

Entropy is fundamentally a measure of *uncertainty*. The greater the entropy of a probability distribution over a set of events, the less certain we are of which event will occur. This is the case for the entropies of region-conditional distributions of labels; hence, the entropy metrics can be seen as the expected uncertainty of labels given our feature set. (So for our purposes, a low entropy metric is desirable.)

Ideally, we would like each name to be strongly associated with a particular label.[3] Strong association of distinct names with labels is equivalent to having a low zeroth-order entropy. For entropies of first order, we wish to keep a strong association of names and labels even if one feature value is changed. For second order, we wish the association to be kept even if two feature values change, and so on.

Recall that, as discussed in Chapter 4, the names can be seen as *encoding* our data points. From the perspective of error-correcting output coding, we would like our code names to be widely spaced with respect to Hamming distance. Then, even if some digits in the code name are changed, we still have a good change of being able to decode the name correctly. In the feature selection setting, attaining a wide spacing of names means that our feature set is robust to misleading values taken by one or more features. The order of entropy can thus be thought of as a maximum permitted error rate.

With this theoretical motivation in mind, we can now draw some conclusions from the results in Section 7.3. In this particular setting the zeroth-order entropy continues to decline as features are added but does not approach zero; instead it generally stays above about 0.004. The entropies of first order and higher remain above about 0.1; they attain this lower bound quite rapidly in the case of selection by EPE, somewhat more slowly for NB, and not at all for NMI.

It is not surprising that the entropies stay well above zero. As we have noted, there is an irreducible element of uncertainty in the labels of data points with a given name; this might be true in any event, but it will certainly be the case in our chosen

[3]Recall that in this context, a ‘name’ is shorthand for a vector of feature values.

experimental setting. Because our features do not take account of context, and many words can have more than one label, a degree of uncertainty is inevitable. Entropy measures this uncertainty, and so there is a lower bound on the entropy-based metrics.

It is interesting that the first, second, and third-order entropies all attain the same lower bound when selection is carried out by EPE or NB. If entropies of more than one order have the same value, then increasing the permitted error rate – from one feature to two, or even from one to three – does not result in additional uncertainty as to the label. This suggests that, in the cases of forward selection by EPE and NB, we have also achieved the objective of ECE by ensuring that the code names are widely spaced.

The situation with the NMI metric is very different. In this case, increasing the permitted error rate from 0 to 1 results in a dramatic increase in label uncertainty, as does increasing it from 1 to 2. Increasing it from 2 to 3 has no significant effect; this may simply indicate that things are so bad with a permitted error rate of 2 that the uncertainty is unlikely to become any worse.

In summary, low values of the ECE metrics are associated with good performance with respect to the EPE and NB metrics. As we shall see, low ECE is also associated with good performance of the ME learner.

### 7.4.2 Relation to Accuracy of ME learner

The results in Figures 7.1 through 7.6 indicate a significant relationship between high accuracy on held-out test data and low entropy. The correlation appears stronger for results on the ME learner than for those with the NB classifier. This is unsurprising, as the entropy-based measures – and in particular ECE – attempt to explicitly consider interactions between features. The ME classifier is much more capable of modelling such interactions than its NB counterpart. For ease of reference, we reproduce the ME accuracies for NMI, EPE, and NB in Figure 7.7.

The first- and second-order entropies appear to be particularly good indicators of ME performance, in that they ‘level off’ at about the same point that ME performance does. It is particuarly interesting to see that in the case of NMI, the entropies start off better than those for EPE and NB but are rapidly overtaken; and that the first-order entropy of feature sets selected by NMI improves slightly between about 32 000 and

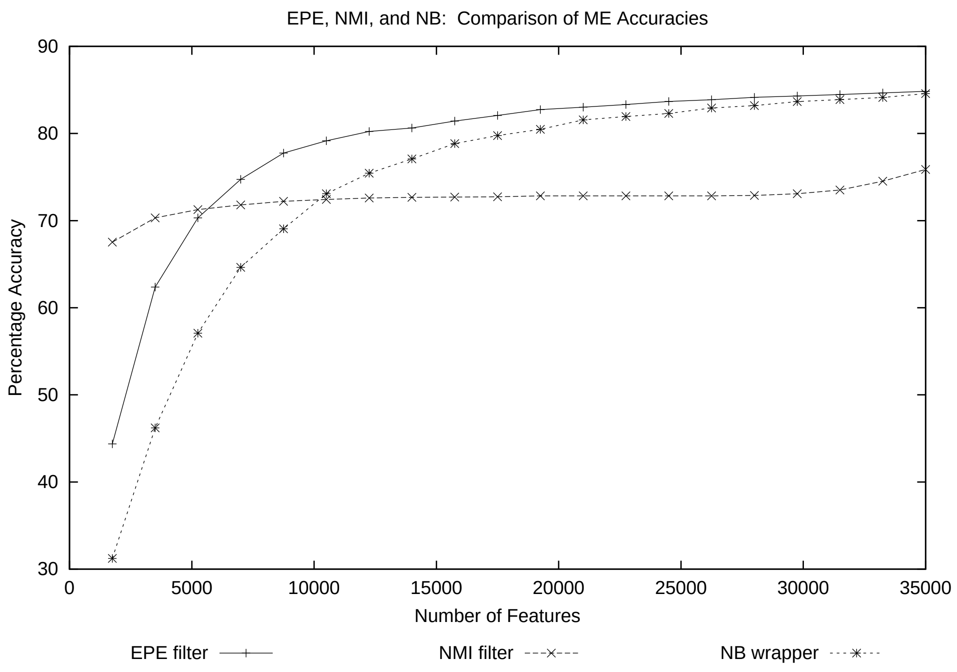

Figure 7.7: ME accuracies for EPE, NMI, and NB.

35 000 features, just as the ME accuracy does.

## 7.5 Summary

In this chapter, the behaviour of the conditional-entropy EPE and ECE on feature sets obtained by accelerated forward selection using the EPE, NMI, and NB metrics was investigated. Mean results were found for ten different repetitions of accelerated forward selection, in order to account for the element of randomness in our search strategy. A strong correlation was found between low conditional entropy and good performance on held-out test data; this correlation was particularly strong in the case of implementation with the ME classifier.

The results in this chapter show that the values of the conditional-entropy metrics on a given feature set are a good indicator of the feature set's performance on test data, regardless of how that feature set was initially obtained. This strongly supports the ideas of EPE and ECE, and also provides some insight into how existing wrapper and filter methods optimise a feature set. In particular, it seems that any successful feature selection metric seeks low partition-conditional entropies as dictated by EPE. Furthermore, the convergence of first-, second-, and third-order ECEs for feature sets which perform well in classification supports the theoretical ideas behind ECE, suggesting that achieving widely spaced vectors of feature values is a valid and important goal.

# Chapter 8

# Document Classification Experiments

## 8.1 Introduction

In this chapter, we present experiments with feature-based probabilistic models for classification of news articles from the Reuters corpus [Reu]. The task considered provides a different perspective on feature selection from the experiments with part-of-speech (POS) tagging discussed in Chapters 5 and 6, and allows further comparison with the work of Koller and Sahami (KS). [KS96] The KS metric is discussed further in Section 3.3.4, and it is closely related to the new EPE metric introduced in Chapter 4. Hence, it is interesting to evaluate EPE and compare its performance with that of the KS metric in the document classification setting.

In particular, KS report that feature selection for document classification can give rise to a feature-based probabilistic model (FBPM) which classifies unseen test data with greater accuracy than one which simply uses the pool of all available features (also referred to as the *maximal feature set*). These increases in accuracy stand in contrast to the POS-tagging experiments discussed in earlier chapters, in which the maximal feature set gave rise to greater accuracy than any of the subsets resulting from feature selection. The POS-tagging experiments were still very useful for demonstrating the differences between feature selection metrics, and the slight reduction in accuracy with the smaller feature sets was offset by gains in speed and efficiency. Nevertheless, it is a general objective of feature selection to create probabilistic models which are more accurate, as well as simpler, than those which employ the set of all available features. In addition to the general interest of the comparison between EPE and KS, the experiments described in this chapter were intended to demonstrate an increase in classification accuracy obtained using EPE.

In practice, it became apparent that both EPE and the KS metric suffer from limitations in the document-classification setting. If either metric is used to select subsets of more than about 40 features (from a maximal feature set of over 2000), afterwards they prove to be equivalent to selecting features at random.

In response to the problems faced by the EPE and KS metrics, we make use of the extension of EPE to *extended covering entropy* (ECE) defined in Section 4.5. The definition of ECE was motivated by the concept of *encoding* of data points by vectors of feature values, and in particular by the idea of spacing between vectors, as discussed in Section 4.4.3. ECE proves far more robust in this setting than EPE or the KS metric, providing a meaningful criterion for the selection of arbitrarily large subsets of the maximal feature set.

## 8.2 Data Points and Features

The experiments in this chapter are based on classification of news articles from the Reuters corpus. [Reu] Each article in the corpus consists of a main body of text accompanied by a number of tags, including zero or more topics. Other tags such as ‘places’ and ‘exchanges’ were ignored; for the purposes of these experiments we focus solely on the body text (which we refer to as a *document*) and the topic.

Two separate three-way classification problems were considered. In the first, the training data was derived from articles with the topics `coffee`, `iron-steel`, and `livestock`, while in the second the topics were `gold`, `reserves`, and `gross domestic product`. Only articles with a single relevant topic were included in the training data: An article with the topics `coffee` and `tea` would be included in the training set for the first task, as would one with the single topic `coffee`, but one with the topics `coffee` and `livestock` would not. There were no articles relevant to both classification tasks – for instance, no news stories in the corpus had both `coffee` and `gold` as their topics.

Each body/topic pair constituted one data point. In the terminology of Chapter 2, the body text describes the *predicate* while the topic is its *label*, and we define *binary features* with the presence of words in the body text as their *indicators*. For example, let the feature function with the word `stock` as its indicator be denoted by `[stock]`. Suppose that a document is denoted by $x$. We then have:

$$[\mathtt{stock}](x) = \begin{cases} 1 & \text{if document } x \text{ contains the word `stock'} \\ 0 & \text{otherwise} \end{cases}$$

Features for other words are defined similarly. As discussed in Chapter 2, a given set of features can be incorporated into an FBPM, which can in turn be used to attempt to predict the labels of previously unseen data points. In selecting the words which would define the initial pool of features available for feature selection (also referred to as the maximal feature set), the approach of KS was followed as closely as possible. Words for the maximal feature set were obtained as follows:

- A 'word' was defined as a string of two or more letters delimited by non-word characters. This had the effect of breaking hyphenated pairs of words into their two components, and rendering a possessive such as 'London's' equivalent to 'London.'
- Case was ignored: 'Coffee' and 'coffee' were regarded as equivalent.
- Words which occurred fewer than three times in the set of relevant articles were ignored. This approach was taken by KS; in their work, there is some ambiguity as to whether a word included in the feature set must occur in three different articles, or could occur (for instance) three times in one article but not in any others. [KS96] The latter interpretation was chosen for the experiments described in this chapter – this would allow the inclusion of words which, say, occurred in only two documents but were still of some general use for classification.

Clearly, a number of somewhat arbitrary decisions had to be made in order to define a suitable maximal feature set. Various permutations of the above choices were considered in an attempt to reproduce the feature set used by KS. Unfortunately, none of them resulted in feature sets of the same size as those reported by KS; all the sets obtained were significantly larger or smaller. It is unclear why this may have occurred, unless KS used a very large test set and the features in their pool were derived only from the training set – they do not give any details of the training/test split employed.

It is somewhat disappointing that the maximal feature sets used by KS could not be reproduced exactly. Nevertheless, the setting employed in this chapter is very similar to the one used by KS, and the experiments described in Section 8.3 below are a valid investigation of the KS and EPE metrics in their own right.

## 8.3 Feature Selection Procedure

### 8.3.1 Training and Test Sets

The files of news articles which made up the Reuters corpus were concatenated into a single file. Articles whose topic was empty or not relevant to either of the classification tasks were discarded; for relevant articles, the features which satisfied the above definition were recorded, together with their topics. This resulted in sets of 313 labelled data points for the first classification task (`coffee/iron-steel/livestock`), and 344 for the second (`gold/reserves/gdp`). The total numbers of features present were 2368 and 2435 for the first and second tasks, respectively.

These sets of data points were split into ten subsets of roughly equal size, where the first subset contained the first, eleventh, twenty-first... from the initial file, and so on. Tenfold cross-validation was carried out, with one of the ten subsets held out as a test set and the other nine making up the training data. The pool of features used for feature selection was derived solely from the training set; therefore, the maximal feature set varied slightly with the training/test split.

### 8.3.2 Search Strategy

In order to select features, we used the method of *accelerated forward selection* presented in Section 5.5.2; at each step, a number of blocks of features were considered for addition to our existing subset, and the block which gave rise to the best value of our chosen feature selection metric was adopted.

In their 1996 experiments, KS reported that feature selection using their metric was computationally intractable, and instead employed an approximation to it. [KS96] However, with modern facilities it has proved feasible to compute both the EPE and KS metrics exactly. (In fact KS may have been too pessimistic, even given their limitations on available computing power; see Section 8.4 below for further discussion.) As detailed in Chapter 6, good results were obtained in part-of-speech tagging experiments by using the EPE and KS metrics with the technique of accelerated forward selection; the search attained optimal values of the KS metric, and near-optimal values of EPE.

In the document-classification experiments, the EPE and KS metrics were again computed exactly. Accelerated forward selection was carried out with the parameters given in Table 8.1. KS used backward elimination in their 1996 experiments. [KS96] However, as discussed in Chapter 3, this was motivated by an erroneous belief that their metric *required* the use of backward elimination. In general there is no inherent reason to prefer one method over the other; but in this setting forward selection proved to be far more efficient, because the metrics reached their optimal values for very small feature subsets. Indeed, for reasons discussed in Section 8.4, it can be shown that the EPE and KS metrics do not provide meaningful criteria for backward elimination in the setting of this chapter.

| **Parameter** | **Value** |
|---|---|
| Blocks evaluated at each step | 1000 |
| Features in each block | 5 |
| Total steps | 400 |

Table 8.1: Parameters for initial accelerated forward selection.

The parameters in Table 8.1 would have resulted in a final subset of 2000 features, chosen from the maximal subset of approximately 2400 features, although in practice feature selection was terminated earlier when the metrics reached their optimal value. Notice that at each step, a total of 5000 features were considered for inclusion in the model, inevitably with some repetition. Thus, the random element in selection was highly unlikely to cause a particularly good feature to be ‘missed out,’ particularly over the course of 400 steps. Such optimism proved more than justified from the perspective of optimising EPE and the KS metric.

## 8.4 Behaviour of the EPE and KS Metrics

### 8.4.1 Optimal Metric Values

Both the EPE and KS metrics rapidly attained their optimal value of zero in this setting. For every training/test split and for both metrics, a zero value was reached at between 35 and 45 features chosen from the maximal set of about 2400. In other words, the metrics were minimised at between 7 and 9 steps of accelerated feature selection with the parameters given in Table 8.1.

This rapid convergence was not particularly surprising for either EPE or KS. Its implications for selection of larger feature sets will be discussed in Section 8.4.4. First, we will briefly examine why the metrics attained their optimal value so quickly.

It should be noted that in the cases of both EPE and KS, the value of the metric is dependent on our training set. Both the metrics are defined with reference to probability distributions over the data; but in practice we do not have access to the 'true' distribution, and must instead approximate it using a set of training data. These *approximate distributions* are discussed further in Section 3.3.1. Therefore, the feature subsets which attained zero values of EPE or KS are only 'optimal' with respect to the available training data; given a different training set, the same feature subsets might well be assigned different metric values.

### 8.4.2 Minimisation of EPE

Recall that EPE can be viewed as measuring the entropies of *partition-conditional distributions* (PCDs). Using the notation introducted in Chapter 2, let data points be denoted by $x$, and their vectors of feature values by $F(x)$, with respect to a given feature set $F$. The PCD for a vector of feature values $y$ is defined as the distribution over labels of the data points $x$ such that $F(x) = y$. (For the sake of brevity, we sometimes refer to a vector of feature values as a *name*.) In assessing feature subsets by EPE, we seek to minimize the expected value of the entropies of the PCDs.[1]

The minimum possible value for entropy is zero, and this value is attained by a distribution which assigns the probability 1 to a single point and 0 to all others. (See Cover and Thomas for further details [CT91].) In the context of feature selection, the PCD for a given name $y$ will have zero entropy exactly when the data points $x$ satisfying $F(x) = y$ all share the same label.[2] In particular, if only one data point $x$ satisfies $F(x) = y$ – that is, $x$ is the only data point with the name $y$ – then the PCD for $y$ will automatically have zero entropy.

Recall that a data point is a predicate/label pair; for instance, a word and its part-of-speech (POS) tag, or a document and its category. In the POS-tagging setting of Chapter 3.3.1, we were very likely to encounter data points with the same predicate and different labels. This was simply because many words have more than one possible POS tag; for instance, 'fall' could be a noun or a verb.

Conversely, in the document-classification setting of this chapter it is likely that there will be a one-to-one correspondence between names and training data points. Our feature values are determined by the presence or absence of particular words, and we are very unlikely to find two documents with different topics which contain *exactly* the same set of words. It is plausible that we could find two such documents in a large training set, particularly as we are excluding features based on very rare words, but this did not occur in the training data used for these experiments.

Notice that with 35 binary features, we have at most $2^{35}$ or about $3.4 \times 10^{10}$ possible names, while the training set only contains about 300 data points. Assigning a different name to each point in our training set should therefore be quite straightforward; this will be sufficient for all PCDs to have an entropy of zero, and hence for zero EPE to be attained. So, it is not surprising that we were easily able to find a set of 35 or 40 features which assigned a unique name to each data point in the training set.

[1]The definition of EPE is considered in much greater depth in Chapter 4; partition-conditional distributions are considered in greater detail in Sections 4.3.1 and 8.5.

[2]In this instance we are only concerned with data points which are assigned non-zero probability by our approximate distribution $p(x)$. (Approximate distributions are defined in Section 3.3.1.) The label of a data point $x$ such that $p(x) = 0$ will not affect the PCD for $y = F(x)$.

### 8.4.3 Minimisation of the KS Metric

In this document-classification setting, the situation for the KS metric is very similar to that for EPE. As we have noted, no two documents in the training set shared *exactly* the same set of words. Hence, the maximal feature set assigned a different vector of feature values to every point in our training set.

Recall that in selecting features with the KS metric, we seek to minimize the expected value of an information-theoretic divergence (known as Kullback-Liebler or KL divergence) from the PCDs induced by the maximal feature set to those induced by our candidate subset. (See Chapter 3 or Koller and Sahami's 1996 paper for details [KS96].) Because the maximal set assigns a different name (vector of feature values) to each training point, zero expected divergence will be reached if our candidate feature set also assigns a unique name to each training point. Again, this proved quite easy to achieve with the training set employed in this chapter.

It is worth noting that, for a very similar task, KS did not report the attainment of optimal values for their metric. This appears to be because they were not computing the KS metric exactly, but instead used an approximation to it. The approximation would not necessarily have revealed whether the KS metric had reached zero. KS were in a sense too pessimistic; as the results in this chapter demonstrate, even a very limited search through the space of feature subsets is sufficient for their metric to reach its optimal value of zero.

### 8.4.4 Optimal Metric Values and Continued Feature Selection

Attaining the optimal value for our metrics on such small feature sets is undesirable, to say the least. One would not expect a set of only 35 features to attain good performance in classifying unseen test data; and indeed, this proves to be the case in our chosen setting. Furthermore, for reasons we now consider, the EPE and KS metrics are no better than random selection for choosing additional features once their optimal values have been attained.

If a given feature set has attained the minimal value of zero for EPE or KS, the metric value will remain at zero if we add new features to the set. This is essentially because our estimate of the probability of a given data point (that is, a predicate/label pair) is provided by the empirical distribution, and as such is independent of our chosen feature set. Recall that the empirical distribution $\tilde{p}(x)$ is defined by $\tilde{p}(x) = c(x)/N$, where $c(x)$ is the number of times that the data point $x$ occurs in the training set $\tilde{X}$, and $N$ is the total number of data points in $\tilde{X}$. Hence, data points which do not appear in our training set receive zero probability.[3]

As discussed in Chapter 4 and Section 8.4.2, a feature set has the effect of *partitioning* the set of possible data points. If each training data point is in a different partition, then adding more features will not change matters: Each training data point will still have a partition to itself. As noted above, assigning a different vector of feature values to each training data point is sufficient to attain zero values of EPE and the KS metric. In the case of the experiments in this chapter, this is exactly how zero values were achieved.[4]

Therefore, if we add *any* feature to a set with zero EPE, the EPE will remain at zero; and matters are similar for the KS metric. Intuitively, the criteria set by the EPE and KS metrics are 'too easy' in the document-classification setting of this chapter. Many feature subsets, of any given size greater than about 35, are optimal with respect to these two metrics. The 'optimal' subsets may well vary widely in their usefulness for classifying test data, and the metrics provide no guidance whatsoever in choosing between them. If we are to select feature subsets of a larger size, then a new means of deciding which features to include in our set is required.

## 8.5 Expected Covering Entropy: Preliminaries

In response to the limitations of the EPE and KS metrics discussed in Section 8.4, we select features using the *Expected Covering Entropy* (ECE) metric introduced in Sec-

[3]The estimate of probabilities is known as an *approximate distribution*, and is considered further in Section 3.3.1. Other approximate distributions are possible; for instance, we could use a smoothed version of the empirical distribution, with the aim of reducing overfi tting to the training data. However, it is common practice to use the empirical distribution for the sake of simplicity, and this was the approach taken in this chapter.

[4]The situation is somewhat more complicated if some partitions contain more than one training data point; but it can still be shown that adding features to a set with zero EPE will not cause the EPE to increase, and similarly for the KS metric.

tion 4.5. Section 8.5.1 briefly reviews the definition of ECE and provides an example to illustrate its potential advantages in the setting of this chapter. We then record the parameters used for feature selection using ECE and evaluate the results, in terms of accuracy of an FBPM on unseen test data and the behaviour of the ECE metric itself.

## 8.5.1 Background

ECE is based on regarding vectors of feature values as equivalent if they are 'close together' with respect to some appropriate distance measure. In the case of vectors of binary values such as the ones employed in this chapter, the conventional measure is Hamming distance; the Hamming distance $\delta(a,b)$ between two bitvectors $a$ and $b$ is simply the number of places in which they differ. With non-binary features other measures, such as the Euclidean distance, could be used instead.

Let us fix a distance $k$ such that two names $y_1$ and $y_2$ are regarded as equivalent if $\delta(y_1,y_2) \leq k$. The partition corresponding to a given name $y$ is then replaced by a 'fuzzy partition' or *region*, denoted by $R_y^{(k)}$ and defined as follows:

$$R_y^{(k)} = \{x : \delta(y, F(x)) \leq k\}$$

The regions form a *covering* of the data space; each data point $x$ is in at least one region, and may be in more than one. We can now define *region-conditional distributions* or RCDs in exactly the same way as the partition-conditional distributions discussed in Sections 4.2 and 8.4.2. The RCD for a region $R_y^{(k)}$ is the distribution over labels of points $x$ such that $\delta(y, F(x)) \leq k$. The *Expected Covering Entropy* or ECE is defined as the expectation of the entropies of the RCDs; $k$ is referred to as the *order* of the ECE. Notice that if $k = 0$, ECE is equivalent to EPE.

**Example 8.1:** Suppose that we have fifty labelled training points corresponding to Reuters news articles, of which twenty-five have the topic `coffee` and twenty-five the topic `iron-steel`; and suppose that we wish to select a subset of six binary features from a much larger maximal feature set. Hence, each candidate feature subset gives rise to at most $2^6 = 64$ distinct vectors of feature values. Now assume that we have two candidate subsets: $F_1$, which assigns a different vector of feature values to each point in the training data; and $F_2$, which satisfies $F_2(x) = 111000$ for all the training points $x$ whose topic is `coffee`, and $F_2(x) = 000111$ for all those with the topic `iron-steel`.

Both $F_1$ and $F_2$ will have zero EPE with respect to the empirical distribution of our training data, since knowing the name assigned by either $F_1$ or $F_2$ to any point in the training set gives us absolute certainty as to its label. However, the ECE of orders 1 and 2 will be zero for $F_2$, and non-zero for $F_1$. In other words, if we ignore the values of *any* two features in $F_2$, knowing $F_2(x)$ is still sufficient to tell us the label of any point $x$ in the training set; but this is not the case for $F_1$.

Notice that the first three features in $F_2$ are active on points in the training set exactly when they have the topic `coffee`, and similarly for the last three and the topic `iron-steel`. We can therefore be quite confident that the features in $F_2$ will generalise to unseen data points. Conversely, many of the features in $F_1$ will be active on both `coffee` and `iron-steel` articles from the training set, and will not necessarily be good general indicators of a data point's topic. □

### 8.5.2 Choosing the Order of ECE

We now need a means of deciding which order of ECE to use. As discussed in Sections 4.5 and 8.5, the order is the maximum number of feature values by which two names can differ if they are to be regarded as equivalent. The higher the order, the more widely spaced the names must be in order to optimise the ECE and the more 'fuzzy' the covering of the data space that is considered. If the order is too low, then the metric may be too easily minimised – as we have seen with EPE, which is equivalent to ECE of order zero. On the other hand, if the order is too high then achieving sufficiently wide spacing may be too difficult, resulting in the rejection of useful feature sets. For instance, the feature set $F_2$ in the example in Section 8.5.1 would receive the highest possible (that is, worst) ECE for any order of three or greater.

In carrying out forward selection, we set the order using a scheme which we refer to as *ascending ECE*. The idea is to begin with an order of zero; if the ECE of a given order reaches its optimal value, then we increase the order by one. Notice that if the ECE metric reaches its optimal value of zero for a given order $k$, then the ECE is also equal to zero for all orders less than $k$.

As we add more features, the mean spacing between names of points in the training set can be expected to increase, and hence the order to which the ECE can be optimised will increase as well. This seemed particularly likely to be the case in the setting of this chapter, given that ECE of order zero could be optimised with as few as 35 features. In practice, while selecting 1500 features from a maximal set of about 2400 using this scheme, the order of ECE increased from zero to between 20 and 30. (The behaviour of ascending ECE will be discussed further in Section 8.6.)

**Aside – Descending ECE:** If we wished to carry out backward elimination using the ECE metric, then a similar idea can be used to determine the order. Recall that in backward elimination, we start with the maximal feature set and progressively remove features. We may find that the maximal feature set has optimal ECE up to a high order; this will certainly be the case in the setting of this chapter. We can then set the initial order to the highest value at which the maximal feature set does not have optimum ECE.

As the feature set becomes smaller, the high initial order of the ECE may give rise to 'too much fuzziness,' such that all candidate feature sets receive the same high value of ECE. As a very simple example, if we have only twenty binary features, then setting the order of ECE to 20 is pointless; the distance between names cannot exceed 20 in this instance, so all names will be regarded as equivalent and all feature sets will receive the same ECE value. If the ECE levels off in this fashion, then it would be reasonable to reduce the order.

### 8.5.3 Search Parameters

As previously noted, feature selection by ECE in the part-of-speech tagging setting described in Chapter 5, was prohibitively slow. The difficulty of computing ECE increases roughly with the square of the size of our training set. At most, ECE requires us to compute the distance $\delta$ between the vectors of feature values assigned to each pair of points in the training set; in a set with $N$ data points, there are $(N^2 - N)/2$ pairs to consider. (We already know that $\delta(a,a) = 0$ and $\delta(a,b) = \delta(b,a)$ for all vectors $a$

and $b$.)

In practice we may be able to reduce the number of computations further by exploiting the triangle inequality: $\delta(a,b)+\delta(b,c) \geq \delta(a,c)$, so if $\delta(a,c)-\delta(a,b) \geq k$ then $\delta(b,c) \geq k$. Hence, if $k$ is the order of the entropy and we know that $\delta(a,c)-\delta(a,b) \geq k$, we know that $c$ will not be in the same region as $a$. This simplification was not attempted in the experiments described in this chapter, but could potentially be useful. It is also worth noting that feature selection of any kind lends itself very well to parallel processing; a large number of candidate feature sets can be assessed in parallel, and the best one adopted.

Even computing all $(N^2-N)/2$ distances proved feasible in our setting. Given that our training sets contained at most 344 data points, there were at most 58,996 pairs of vectors of feature values to be considered. Accelerated forward selection was carried out using the parameters in Table 8.2.

| **Parameter** | **Value** |
|---|---|
| Blocks evaluated at each step | 50 |
| Features in each block | 10 |
| Total steps | 150 |

Table 8.2: Parameters for accelerated forward selection by ECE.

The final feature set therefore contained 1500 features, chosen from the maximal feature set of about 2400. Feature selection was rather slow; selecting a full set of 1500 features required about 32 hours on a Pentium 4 PC. Notice that only 500 features were evaluated at each step, rather than 5000 as with feature selection by EPE. This resulted in slightly less rapid improvement in the metric; EPE (that is, ECE of order zero) reached zero at about 80 features out of 2400 in these experiments, as opposed to 40 with the more extensive search described in Section 8.3. Nonetheless, the parameters in Table 8.5.3 were sufficient to produce interesting results in terms of both ECE and accuracy on test data, as described in Sections 8.6.1 and 8.6.2.

## 8.6 Expected Covering Entropy: Results

### 8.6.1 Behaviour of the ECE Metric

Feature selection began with the ECE at order zero; if the ECE reached its minimal value, then the order was increased by one. In the `coffee/iron-steel/livestock` classification task, the greatest order of ECE used for feature selection was 22 for 8 of the 10 training/test splits; for the other two, the maximum orders were 27 and 28. In the `gold/reserves/gdp` task, a maximum order of 31 was attained for all 10 training/test splits. So, in the latter task, the values of up to 31 features were ignored in computing the expected entropy for large feature sets.

The maximum order for which the ECE reached its optimal value of zero was one less than the highest order used for feature selection. For instance, in the `gold/reserves/gdp` case, an expected entropy of zero was attained at order 30 for each training/test split. Figure 8.1 shows the mean size of feature set at which each order of ECE was minimised. For instance, in the `gold/reserves/gdp` task, ECE of order 0 was minimised with an average of 62 features, and of order 30 with an average of 1015 features. Full tables of results are given in Appendix C.

Notice that little or no increase in the order of ECE occurred with subsets of more than about 800 features for `coffee/iron-steel/livestock` data, and of more than about 1000 features for `gold/reserves/gdp`. After the order levelled off at a particular value, feature selection continued with the objective of minimising the ECE for this fixed order. However, the expected entropy itself soon levelled off at a non-zero value, which remained unchanged even as several hundred features were added.

By definition of the ascending ECE scheme, the expected entropy could never reach zero; if it did, then the order would simply be increased until we attained a non-zero ECE value. (It is impossible for a feature set to have zero ECE for all orders; as the order approaches the number of features, the ECE will always attain a non-zero value.) Therefore, the feature selection process never became equivalent to selecting features at random. However, it appears that improving the ascending ECE metric becomes very difficult for large feature sets. The implications of the behaviour of ECE will be discussed with reference to accuracy on test data in Section 8.6.2.

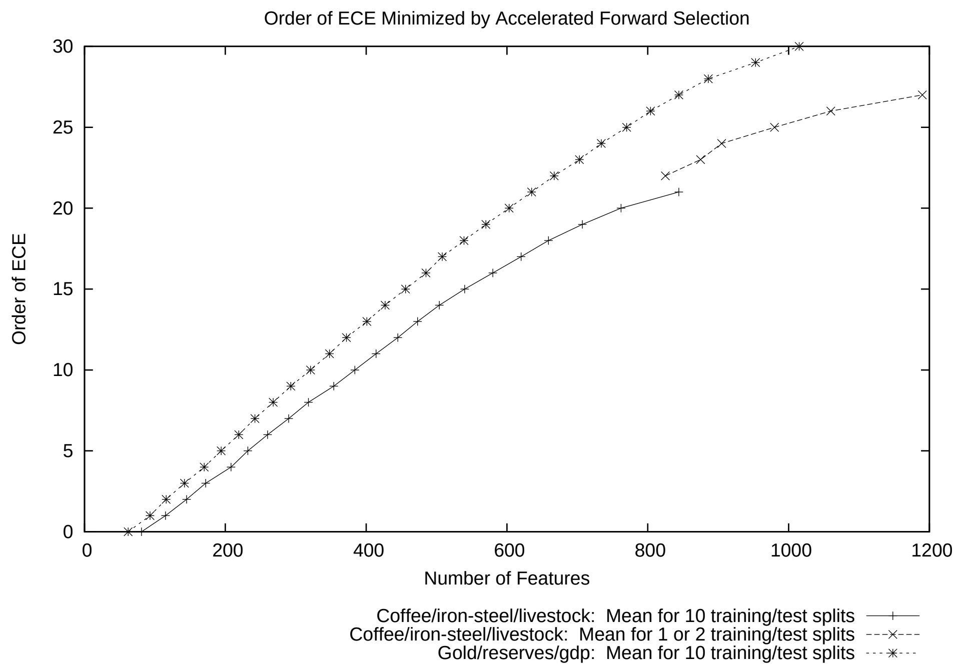

Figure 8.1: Order of ECE minimised, by task and mean size of feature set.

### 8.6.2 Accuracy on Test Data

Evaluation was conducted using the naive-bayes (NB) learner described in Section 2.3.3. As previously noted, tenfold cross-validation was carried out. Figure 8.2 shows the mean accuracy of classification by the NB learner on test data, over each of the ten training/test splits, and for both classification tasks. The relationship between the order of ECE and the accuracy on test data, and connections with KS' experimental work, will be considered later in this section; first, we present the mean results together with some general commentary.

The mean accuracy of classification using the maximal set of approximately 2400 features was 90.09% for the `coffee/iron-steel/livestock` task, and 89.60% for `gold/reserves/gdp`. In the former task, the greatest mean accuracy attained using feature selection by ascending ECE was 92.32% with 800 features; in the latter, the greatest mean accuracy was 93.92% with 200 features. Hence, the gains in mean accuracy over that with the maximal feature set were respectively 2.23% and 4.32%.

The improvement in accuracy in comparison with the maximal feature set is par-

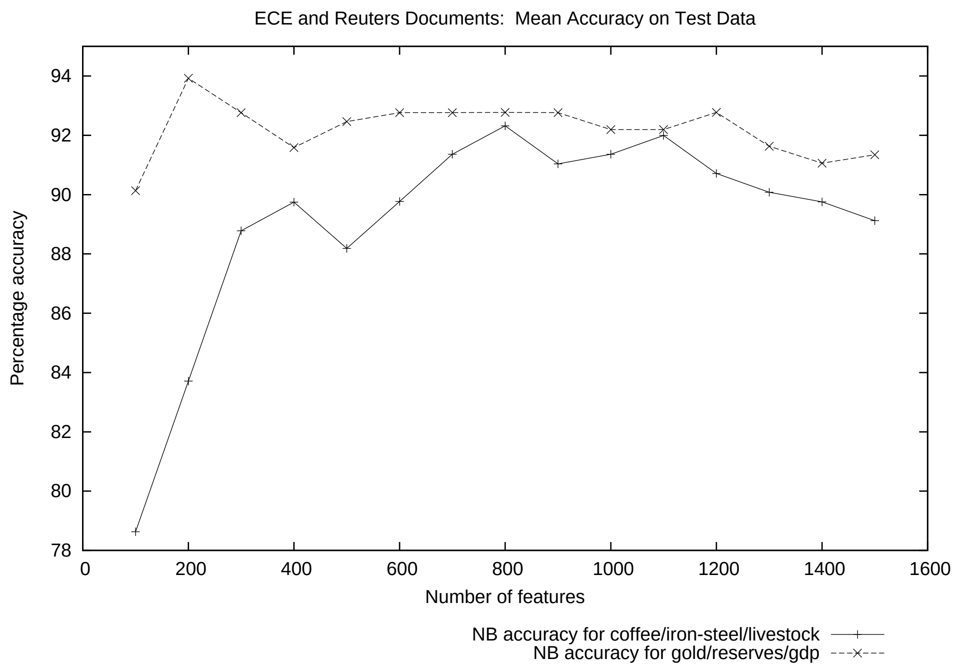

Figure 8.2: Mean accuracy on test data for feature selection by ECE.

ticularly striking given that, because of the difficulty of computing the ECE metric, relatively few candidate feature subsets were assessed. In spite of the limited nature of our search, superior accuracy to the maximal feature set was consistently attained for subsets of between 700 and 1200 features with `coffee/iron-steel/livestock` data, and between 200 and 1500 with `gold/reserves/gdp`. It is also notable that the best accuracy attained in the second task was with a feature subset less than one-tenth the size of the maximal feature set.

**Comparison with KS Results:** The results in this chapter are not directly comparable to those in KS' 1996 paper. [KS96] This is partly because it was not practical to reproduce the exact maximal feature sets and training/test splits employed by KS. Much more importantly, as discussed in Section 8.4.4, the KS metric itself fails to provide a meaningful criterion for selecting sets of more than about 40 features from the maximal sets defined in Section 8.3.

Given how easily the KS metric reached an optimal value in the setting of this chapter, it seems almost certain that similarly rapid optimisation would occur in the setting used by KS. Certainly, one would expect the KS metric to attain an optimal value on any set of more than 600 features; this would render it equivalent to random selection for reducing a set of about 1600 features to about 600 by backward elimination, as described by KS.

However, it is important to note that KS did not compute their metric exactly, but instead employed an approximation to it. KS report some instances of subsets of about 600 features giving rise to greater classification accuracy than their maximal feature sets. Discovering why these improvements came about would require further investigation of the properties of the KS approximation, which it was not feasible to conduct as part of this thesis.

**Order of ECE and Accuracy on Test Data:** It is interesting to consider what relationship, if any, there is between increase in the order of ECE and improvement in classification accuracy on test data. Recall that if the order of ECE levels off at a particular value, the addition of more features does not give rise to drastic increases in spacing between names; so, from the perspective of the ECE metric, the additional features do not add much in the way of useful redundancy. In light of this observation, and the results with POS-tagging experiments discussed in Chapter 7, it seemed plausible that a levelling-off of the order of ECE would be accompanied by a lack of improvement in classification accuracy.

This hypothesis was only partly borne out by the document-classification experiments. In the `coffee/iron-steel/livestock` classification task, mean accuracy does peak at 800 features, which is also the point at which the order of ECE stopped increasing for 8 of the 10 training/test splits; there is then a second, smaller peak at 1100 features, just before the order stopped increasing altogether at 1200 features. However, in the `gold/reserves/gdp` task, maximum accuracy was obtained with 200 features, well before the order of ECE levelled off at just over 1000 features.

The experimental results do support a weaker hypothesis that maximum accuracy will be obtained before the order of ECE levels off. Regrettably, time did not permit a thorough investigation of the relationship between order of ECE and classification accuracy, or assessment of subsets of between 1500 and 2400 features selected using ascending ECE. Such investigations would be an interesting topic for future research.

## 8.7 Summary

This chapter has presented experiments with classification of Reuters news stories by topic, very similar to the ones carried out by Koller and Sahami for the evaluation of their information-theoretic metric. [KS96] Unlike KS, we computed the KS metric exactly, rather than employing an approximation to it. It transpired that both the KS metric, and the Expected Partition Entropy (EPE) metric presented in Chapter 4, suffered from limitations in the document-classification setting. Specifically, for subsets of more than about 40 features chosen from a maximal set of about 2400, both metrics were equivalent to selecting features at random.

In response to the difficulties faced by the EPE and KS metrics, we conducted a second set of experiments using the extension of EPE to the Expected Covering Entropy (ECE) metric. In order to do so, we defined a scheme called *ascending ECE*, which provides a means of setting the parameter of the ECE metric known as the *order* to an appropriate value. This second set of experiments proved far more successful; we obtained subsets as little as one-tenth the size of the maximal feature set, which nonetheless gave rise to greater classification accuracy on test data than the maximal set.

# Chapter 9

# Discussion

## 9.1 Introduction

This chapter presents the conclusions of the thesis. It begins with a discussion of new theoretical ideas for feature selection, including some extensions to the existing general framework for the feature selection task; a critique of feature selection metrics based on information theory, particularly the Koller-Sahami evaluation function; and motivation for and discussion of a novel family of metrics based on conditional entropy. We then consider two interesting side issues which arose from our experimental evaluation of entropy-based metrics: Accelerated search algorithms and the 'like-for-like' hypothesis for wrapper evaluation functions. Finally, we discuss the experimental evaluation of the new conditional-entropy metrics. On both theoretical and empirical grounds, the new metrics appear very promising as a method of feature selection.

Throughout the chapter, possible extensions and further research will be suggested where appropriate. At this point, it is important to note that most of the techniques in this thesis, and the questions which they raise, are completely general. The theoretical developments, accelerated search algorithms, examination of wrapper metrics, and novel absolute-entropy metrics can in principle be applied to any problem in feature selection; such problems include not only part-of-speech tagging and document classification but bioinformatics, image recognition, medical diagnosis, and even detection of land mines.

General applicability is a significant strength of the work presented in this thesis. Virtually all of the theories and techniques considered would benefit from having their effectiveness investigated in other domains. One obvious possibility is to extend our experiments with simplified part-of-speech tagging to a more sophisticated scheme which makes use of context-sensitive features.

## 9.2 Theoretical Developments

In Chapter 2, we presented preliminary definitions and ideas, including some suggested extensions to the standard theoretical framework for feature selection. Throughout the thesis, we stress the importance of keeping a clear distinction between the colloquial use of the word *feature* as simply meaning an identifiable characteristic of a data point; and a *feature function*, which maps the set of data points to another set with the goal of providing a simpler – or otherwise more effective – description of the data.[1]

Equally important is the distinction between *evaluation functions*[2] which provide a numerical measure of the desirability of a given feature subset; and *generation procedures*. Even a modestly sized pool of possible features will have a very large number of subsets; in general, exhaustively evaluating all the possible subsets will be prohibitively slow. As the name suggests, a generation procedure is a strategy for finding 'good' subsets with respect to our chosen evaluation function without carrying out an exhaustive search. With a few specialised exceptions, our choice of generation procedure does not tie us to a particular evaluation function and vice versa.

Extensions to the general framework for feature selection included the definitions of a *non-informative value*, an output which indicates that a particular feature function obtains no significant information from a particular data point; and a *model architecture*, a scheme for incorporating variable sets of feature functions into a family of probabilistic models. A more significant development is the proposed division of evaluation functions into *hierarchical* and *non-hierarchical* metrics. A hierarchical metric such as naive mutual information ranks the members of a pool of possible features in

[1]Unless specifi cally noted otherwise, we use the word 'feature' as an abbreviation for 'feature function.'

[2]Evaluation functions are also known as *feature selection metrics*, or simply as *metrics*.

a fixed order of desirability. Conversely, a non-hierarchical metric such as the Koller-Sahami (KS) measure or a wrapper scheme assigns a score to a feature subset as a whole, rather than its component features; such metrics implicitly or explicitly consider possible dependencies and interactions between features.

Chapter 3 surveys the existing literature on feature selection, with particular attention to the Koller-Sahami metric and other evaluation functions based on information theory. Information-theoretic measures are fundamentally based on relationships between probability distributions. However, we do not have access to the ‘true’ distribution which governs our data, and so must approximate it in some way. The empirical distribution is usually used as an approximation; other possibilities have generally not been considered in the literature, with the exception of Zaffalon and Hutter’s proposal of a new class of approximate distributions for a particular feature selection task [ZH02]. Applying Zaffalon-Hutter approximations to other feature selection domains, along with some new methods which we suggest, would be an interesting topic for future research.

A rigorous consideration of information-theoretic metrics leads to some interesting observations. It is worth noting that the KS metric is not inherently tied to the generation procedure known as backward elimination, as its originators claim. Indeed, we later successfully implement a variant of the alternative forward selection procedure using the KS metric. Furthermore, there are theoretical reasons to suspect that, contrary to the assertions of its originators, the KS metric is not an optimal method of feature selection. In particular, the KS metric can be thought of as measuring the Kullback-Liebler information-theoretic distance from the set of features available for inclusion in our model; we raise the question of whether this is a sufficiently general reference point. As we discuss in Section 9.5, these concerns were supported by experimental data.

In Chapter 4 we continue by presenting a new class of feature selection metrics based on *conditional entropy*. The new metrics can be motivated in at least two different ways. One is a similar motivation to that of the KS metric, in terms of information-theoretic distances from a chosen reference point; in this context, minimising the absolute entropy is equivalent to maximising the Kullback-Liebler divergence from the uniform distribution. An alternative motivation is provided by the related concepts of *partioning* and *encoding*. A feature set can be thought of as dividing the data space into a number of compartments or *partitions*; alternatively, it can be seen as assigning a code word of feature values to each data point. This insight enables us to apply ideas from the fields of discretization (also known as quantization) and coding theory to feature selection.

Theoretical considerations give rise to a family of conditional-entropy metrics. Zeroth-order entropy or *expected partition entropy* (EPE) can be thought of as an alternative to the KS metric, with the uniform distribution as its reference point; alternatively, optimising the EPE can be seen as trying to find the feature set which provides the most informative partition of our data space. We then draw upon ideas from coding theory to extend EPE to entropies of first order and higher, which we refer to as *expected covering entropies* or ECE. Coding theory seeks to encode data in a form robust to transmission errors; similarly, optimising the ECE gives rise to feature sets which remain informative if one or more features take misleading values. The order of the ECE can be thought of as the number of features which are permitted to be 'wrong.'

The solid theoretical foundation of EPE and ECE is appealing in itself, and also ensures their applicability to a wide variety of feature selection domains.

## 9.3 Accelerated Search Algorithms

Although non-hierarchical metrics are generally more sophisticated than hierarchical ones and can be expected to give better results, they suffer from the drawback of being much slower. In order to implement feature selection with non-hierarchical metrics for large sets of training data and available features, a new variant of forward selection known as *accelerated forward selection* was developed. The basic idea is simple: Instead of adding features one at a time, at each step we evaluate several randomly chosen blocks of features and add the best one. This leads to significantly faster feature selection.

Accelerated forward selection proved successful at improving the values of feature selection metrics, which in turn led to better accuracy on held-out test data. The KS metric reached its optimal value of zero under accelerated forward selection – in many cases quite quickly – and other metrics attained near-optimal values. In most cases, we assessed ten blocks at each step in the part-of-speech tagging domain; each block was equal in size to 1% of the pool of available features. However, small-scale experiments in which the block size was increased to 5% of the pool of available features did not lead to any decline in performance. Similarly, out of necessity we evaluated comparatively few candidate subsets while selecting features in the document-classification domain by ECE, but significant improvements in accuracy over that obtained with the set of all available features were obtained nonetheless.

It would be interesting to investigate this matter further. In simple terms, how quick and careless a search can we get away with? If the block size was very large or very few sets were assessed at each step, one would expect a decline in performance; but it would be interesting to see when and how rapidly this decline occurred.

Other variants of accelerated feature selection are possible: For instance accelerated backward elimination, feature selection with backtracking, or selection by genetic algorithm. Because of the large element of randomness present in accelerated selection, the choice between (for instance) forward and backward elimination was thought unlikely to have a significant effect. Nevertheless, it might be worthwhile to carry out experiments to confirm this hypothesis.

## 9.4 Specialisation of Wrapper Metrics

An interesting observation arose from the results of our feature selection experiments. We used two different types of classifer: These were the naive-bayes (NB) and maximum-entropy (ME) model architectures. Classifiers were used to assess the usefulness of feature sets; but they were also used as *wrappers* for feature selection in their own right. Selecting features using the NB wrapper gave rise to greater accuracy on the NB classifier than the ME wrapper; but the ME wrapper outperformed the NB wrapper in selecting features for the ME classifier. Similarly, the EPE filter outperformed NB in selecting features for an ME model, but not for an NB model.

This supports the hypothesis that ‘like should select for like.’ In other words, a wrapper metric – which is based on a particular model architecture – is very good at selecting features for classifiers using the same model architecture. But a wrapper is highly specialised; its superior performance in ‘selecting features for itself’ comes at the cost of lower accuracy in other domains. More extensive testing of the like-for-like hypothesis, using different learning tasks and other model architectures besides the NB and ME schemes, is a possible topic for future research.

## 9.5 Entropy-Based Metrics in Practice

The theoretical appeal of the conditional-entropy metrics suggested that they would be effective techniques for feature selection. More specifically, we hypothesised that improving the absolute entropy of feature sets would lead to better performance on held-out test data; and conversely, that feature sets with better performance on held-out test data – regardless of how they were obtained – would have lower values of the metrics. Experiments were carried out in two different domains: One involved a simplified form of part-of-speech tagging, the other classification of Reuters news stories.

### 9.5.1 Part-of-Speech Tagging

In the part-of-speech tagging setting, we began by investigating the effectiveness of accelerated forward selection by EPE in comparison with other metrics: Frequency-based cutoff (FBC), naive mutual information (NMI), the Koller-Sahami metric (KS), a naive-bayes wrapper (NB), and a maximum-entropy wrapper (ME). The feature sets obtained were assessed by using them to implement NB and ME classifiers on held-out test data. ECE proved too slow for practical feature selection in this domain, even using accelerated techniques. However, we were able to evaluate feature sets obtained using EPE and the other metrics by both EPE and ECE, in order to determine whether ECE was a good general indicator of classification accuracy on test data.

**Feature Selection by EPE:** The general setting for the first set of experiments was a simplified form of part-of-speech tagging for English words, using training data derived from the Penn Treebank WSJ corpus. [MSM95] EPE was found to perform well in comparison with all of the existing metrics employed. In short, it was significantly faster than ME while providing comparable accuracy on both classifiers; more reliable than KS; led to greater accuracy in most cases than FBC or NMI; and produced greater accuracy than NB on the ME learner.

Some of the above comments deserve a little further explanation. As we have noted, the KS metric tended to reach its optimal value of zero quite quickly; but this was not necessarily an advantage. If a given feature subset has a KS score of zero, then the KS metric gives us no guidance as to how we can improve it further; but there may be considerable room for improvement in terms of its performance on held-out test data, and the feature set may not even the best possible set of its size. It seems that the theoretical concerns about KS not having a sufficiently general reference point were well justified in this setting. EPE did not suffer from this problem in the POS-tagging experiments; instead it exhibited gradual improvement without attaining its optimal value of zero.

FBC and NMI generally fared worse than EPE, but they were noticeably better at selecting relatively small feature sets – particularly for the NB classifier. This suggests that they could be usefully combined with the EPE metric, or indeed with other feature selection metrics: For instance, we could constrain our feature set to include a small number of features which scored best according to FBC or NMI, and select additional features by a different method.

**Evaluation by EPE and ECE:** When the new metrics were used for evaluation of pre-selected feature sets, it was found that they were strongly associated with good performance on held-out test data. First and second-order ECE appeared to give particularly good indications of the accuracy arising from a given feature set. The success of ECE seems to indicate its theoretical motivation – that our vectors of feature values should be widely spaced, and so robust to misleading values taken by one or more features – is sound. The value of ECE was further confirmed by experiments in the Reuters document-classification domain.

### 9.5.2 Document Classification

We hypothesised that ECE would be particuarly useful in environments with very sparse training data. If we only have one data point in a typical partition, then EPE will be of very limited use; given the information available in the training data, it will (perhaps erroneously) be absolutely certain of the label associated with a given partition, and so will rapidly attain its ideal value of zero – but this optimisation may not reflect good performance on held-out test data. It was expected that extending the partitions of EPE to the larger *regions* of ECE would get around this problem, because the regions would contain more data points; in effect, optimising ECE is a more difficult challenge than optimising EPE.

This is exactly the situation which occurred in the Reuters document-classification experiments: EPE and the KS metric rapidly attained their minimum values, as each point in the training set was assigned a different vector of feature values. In the document-classification domain, it was possible to directly use the ECE metrics for feature selection. As expected, there was a very wide variation in size and classification accuracy between feature sets with zero values of EPE and the KS metric; but ECE proved an effective tool for discriminating between them. Feature selection by ECE produced feature sets somewhat larger than the smallest ones for which minimal values of EPE and the KS metric were attained, and considerably smaller than the pool of all available features, which gave rise to better accuracy than either in classifying test data.

A particularly interesting question which arose from our experiments is the relationship between the *order* of ECE, and classification accuracy on test data. Recall that the order of ECE is the maximum distance between two vectors of feature values which are regarded as equivalent; the greater the order, the more difficult it is to optimise the metric. If ECE reaches its optimal value of zero for a particular order, it is also optimal for all lower orders. It seems that optimisation of ECE to a high order will give rise to high accuracy in classification of test data; and conversely, that feature subsets giving high accuracy are likely to have optimal or near-optimal ECE to a high order. However, the exact nature of this relationship remains an open question, and would be a suitable topic for future research.

It would also be interesting to further consider the idea of seeking widely spaced vectors of feature values. This could perhaps be investigated in other ways besides ECE. For instance, we could compute the mean Hamming distance between pairs of vectors and attempt to maximise its mean value for those with different labels, while minimising it for those with the same one. Moreover, in this thesis we have only considered a few of the ideas and techniques contained in quantization and coding theory. Many such concepts may be applicable to feature selection; this thesis has laid the groundwork for further investigation of the connections between quantization and coding theory on the one hand, and feature selection on the other.

# Appendix A

# Tables of Results: Part-of-Speech Tagging and Accuracy

## A.1 Large Training Sets

In this section we give the minimum, maximum, and mean percentage accuracy over ten runs of stepwise forward feature selection in the part-of-speech tagging domain, for large training sets of about 800 000 data points, with a maximum of 35 000 features. Additional detail on the experimental setting and parameters is given in Chapter 5. Both the maximum-entropy and naive-bayes learners were used for assessment, with a different split between held-out test data and training data on each run. Graphs and discussion of the results are presented in Chapter 6.

### A.1.1 Random Selection

| **Subset size** | **Maximum-Entropy** | | | **Naive-Bayes** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 18.10 | 35.98 | 26.77 | 13.98 | 22.31 | 16.72 |
| 3500 | 29.20 | 51.92 | 44.19 | 13.60 | 23.26 | 17.22 |
| 5250 | 47.29 | 60.95 | 55.64 | 13.09 | 23.69 | 17.41 |
| 7000 | 56.49 | 68.37 | 63.48 | 12.95 | 21.54 | 17.00 |
| 8750 | 60.86 | 73.08 | 69.25 | 14.46 | 23.42 | 17.86 |
| 10500 | 68.70 | 77.79 | 73.15 | 14.40 | 24.73 | 19.47 |
| 12250 | 70.97 | 78.83 | 75.27 | 15.95 | 25.69 | 20.90 |
| 14000 | 73.50 | 79.54 | 77.42 | 17.57 | 28.66 | 22.58 |
| 15750 | 77.80 | 80.53 | 79.31 | 19.62 | 29.04 | 24.25 |
| 17500 | 78.78 | 81.26 | 80.51 | 23.41 | 33.24 | 28.05 |
| 19250 | 80.18 | 82.10 | 81.21 | 24.65 | 34.54 | 30.49 |
| 21000 | 80.06 | 82.72 | 81.79 | 23.94 | 36.61 | 31.53 |
| 22750 | 81.60 | 83.30 | 82.42 | 26.54 | 40.94 | 34.47 |
| 24500 | 81.17 | 83.80 | 82.77 | 31.75 | 45.81 | 36.95 |
| 26250 | 81.34 | 83.78 | 83.34 | 33.18 | 44.56 | 38.69 |
| 28000 | 82.85 | 84.10 | 83.62 | 35.89 | 51.22 | 43.85 |
| 29750 | 82.85 | 84.66 | 84.14 | 37.30 | 52.95 | 46.74 |
| 31500 | 83.90 | 84.77 | 84.39 | 41.80 | 58.51 | 51.07 |
| 33250 | 84.05 | 84.90 | 84.54 | 42.24 | 67.63 | 54.60 |
| 35000 | 84.38 | 85.06 | 84.73 | 48.09 | 70.84 | 61.62 |

Table A.1: Random selection with large training set

### A.1.2 Frequency-Based Cutoff

| **Subset size** | **Maximum-Entropy** | | | **Naive-Bayes** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 24.95 | 25.24 | 25.11 | 17.28 | 17.33 | 17.31 |
| 3500 | 50.67 | 51.18 | 50.87 | 24.62 | 25.45 | 24.80 |
| 5250 | 56.25 | 58.50 | 57.80 | 18.56 | 19.68 | 18.95 |
| 7000 | 63.55 | 63.87 | 63.69 | 17.71 | 18.72 | 17.95 |
| 8750 | 65.72 | 66.31 | 66.11 | 20.84 | 22.24 | 21.83 |
| 10500 | 74.34 | 74.62 | 74.52 | 25.46 | 27.08 | 26.71 |
| 12250 | 76.31 | 78.20 | 77.30 | 22.86 | 25.22 | 24.17 |
| 14000 | 78.85 | 79.82 | 79.29 | 24.52 | 27.76 | 26.97 |
| 15750 | 79.62 | 80.41 | 80.04 | 27.38 | 30.41 | 29.63 |
| 17500 | 80.46 | 81.70 | 80.93 | 30.26 | 37.46 | 36.16 |
| 19250 | 81.21 | 82.39 | 81.72 | 40.95 | 42.81 | 41.64 |
| 21000 | 81.58 | 82.23 | 81.93 | 41.40 | 45.42 | 43.47 |
| 22750 | 82.14 | 83.55 | 83.00 | 43.64 | 52.08 | 47.46 |
| 24500 | 82.53 | 84.06 | 83.34 | 37.80 | 42.87 | 40.38 |
| 26250 | 82.93 | 84.01 | 83.53 | 39.90 | 44.15 | 41.50 |
| 28000 | 83.30 | 84.20 | 83.69 | 41.90 | 47.81 | 43.32 |
| 29750 | 83.56 | 84.28 | 83.97 | 44.25 | 48.36 | 47.65 |
| 31500 | 83.87 | 84.69 | 84.28 | 50.01 | 53.64 | 53.14 |
| 33250 | 83.83 | 84.86 | 84.37 | 54.61 | 59.00 | 56.58 |
| 35000 | 84.09 | 84.92 | 84.74 | 72.11 | 72.94 | 72.65 |

Table A.2: FBC with large training set

### A.1.3 Naive Mutual Information

| **Subset size** | **Maximum-Entropy** | | | **Naive-Bayes** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 67.27 | 67.77 | 67.53 | 19.13 | 30.64 | 29.29 |
| 3500 | 70.21 | 70.43 | 70.32 | 20.97 | 31.47 | 30.08 |
| 5250 | 71.04 | 71.46 | 71.26 | 20.47 | 25.32 | 24.51 |
| 7000 | 71.56 | 72.01 | 71.80 | 20.57 | 25.39 | 23.35 |
| 8750 | 72.02 | 72.45 | 72.21 | 20.81 | 22.39 | 21.99 |
| 10500 | 72.26 | 72.65 | 72.43 | 19.49 | 23.89 | 22.67 |
| 12250 | 72.43 | 72.86 | 72.60 | 22.05 | 24.57 | 24.15 |
| 14000 | 72.52 | 72.89 | 72.67 | 22.93 | 24.52 | 24.31 |
| 15750 | 72.58 | 72.92 | 72.71 | 22.24 | 24.67 | 24.33 |
| 17500 | 72.61 | 72.96 | 72.74 | 22.72 | 24.53 | 24.23 |
| 19250 | 72.70 | 73.09 | 72.85 | 22.26 | 24.06 | 23.72 |
| 21000 | 72.70 | 73.09 | 72.85 | 22.20 | 24.10 | 23.72 |
| 22750 | 72.70 | 73.09 | 72.85 | 22.26 | 24.08 | 23.73 |
| 24500 | 72.70 | 73.09 | 72.85 | 22.20 | 23.98 | 23.68 |
| 26250 | 72.70 | 73.10 | 72.85 | 22.29 | 23.99 | 23.72 |
| 28000 | 72.74 | 73.14 | 72.89 | 22.39 | 23.98 | 23.72 |
| 29750 | 72.91 | 73.29 | 73.09 | 22.35 | 23.94 | 23.68 |
| 31500 | 73.34 | 74.03 | 73.52 | 23.12 | 24.17 | 23.89 |
| 33250 | 74.29 | 75.07 | 74.52 | 23.64 | 24.52 | 24.24 |
| 35000 | 75.70 | 76.49 | 75.88 | 29.78 | 31.60 | 30.93 |

Table A.3: NMI with large training set

### A.1.4 Koller-Sahami Metric

In all ten runs with large training sets, the Koller-Sahami (KS) metric attained its optimal value of zero before the feature set reached its maximum size of 35 000. The feature selection process was halted when a KS score of zero was attained, as any further addition of features could not improve the KS score of the feature set.

Table A.4 reproduces Table 6.1. It gives the subset sizes at which selection by the KS metric was halted, together with the accuracy of the maximum-entropy and naive-bayes learners on the final subsets.

| **Size of Final Subset** | **Maximum-Entropy** | **Naive-Bayes** |
|---|---|---|
| 15050 | 81.90 | 39.10 |
| 20650 | 83.52 | 41.24 |
| 21350 | 83.24 | 41.94 |
| 22050 | 83.15 | 42.75 |
| 22400 | 83.35 | 49.91 |
| 24850 | 83.84 | 43.71 |
| 26600 | 83.76 | 54.76 |
| 28000 | 84.32 | 50.64 |
| 29050 | 84.48 | 62.34 |
| 31150 | 84.59 | 56.48 |

Table A.4: Final subsets for KS with large training set

Table A.5 gives results for the ‘standard’ subset sizes obtained using the KS filter. Feature selection by the KS filter was halted before the maximum subset size of 35 000 was achieved, as the KS metric attained its optimal value of zero; this phenomenon is discussed further in Section 6.5. As the subset size becomes larger, the minimum, maximum, and mean values are for fewer and fewer runs; no run obtained feature sets with a size greater than 31150.

| **Subset size** | **Maximum-Entropy** | | | **Naive-Bayes** | | | **Total Runs** |
|---|---|---|---|---|---|---|---|
| | *Min* | *Max* | *Mean* | *Min* | *Max* | *Mean* | |
| 1750 | 41.69 | 47.09 | 44.28 | 11.34 | 25.95 | 16.79 | 10 |
| 3500 | 56.32 | 65.08 | 61.37 | 11.03 | 24.24 | 16.68 | |
| 5250 | 67.74 | 71.69 | 70.36 | 12.39 | 25.54 | 18.81 | |
| 7000 | 74.17 | 76.03 | 75.11 | 12.97 | 26.89 | 20.94 | |
| 8750 | 76.70 | 78.50 | 77.59 | 17.66 | 34.88 | 27.15 | |
| 10500 | 78.13 | 80.12 | 79.14 | 21.75 | 37.33 | 29.48 | |
| 12250 | 79.30 | 81.04 | 80.10 | 26.78 | 41.39 | 33.18 | |
| 14000 | 79.86 | 81.50 | 80.98 | 28.80 | 42.96 | 36.47 | |
| 15750 | 80.81 | 82.35 | 81.45 | 28.85 | 42.14 | 36.62 | 9 |
| 17500 | 81.02 | 82.97 | 82.19 | 35.42 | 49.58 | 39.80 | |
| 19250 | 82.14 | 83.35 | 82.80 | 35.60 | 48.13 | 40.17 | |
| 21000 | 82.44 | 83.65 | 83.05 | 36.62 | 52.08 | 42.52 | 8 |
| 22750 | 82.92 | 84.04 | 83.50 | 38.74 | 58.08 | 47.75 | 5 |
| 24500 | 82.48 | 84.04 | 83.47 | 39.15 | 59.96 | 50.71 | 4 |
| 26250 | 83.90 | 84.19 | 84.00 | 49.74 | 58.34 | 52.61 | 3 |
| 28000 | 83.88 | 84.32 | 84.04 | 50.64 | 61.89 | 54.83 | 2 |
| 29750 | 84.39 | 84.39 | 84.39 | 54.85 | 54.85 | 54.85 | 1 |

Table A.5: KS metric with large training set

### A.1.5 Naive-Bayes Wrapper

| **Subset size** | **Maximum-Entropy** | | | **Naive-Bayes** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 21.75 | 39.23 | 31.22 | 17.77 | 23.91 | 20.63 |
| 3500 | 35.79 | 56.58 | 46.21 | 19.99 | 31.17 | 25.23 |
| 5250 | 49.78 | 64.20 | 57.07 | 20.62 | 36.97 | 27.83 |
| 7000 | 55.71 | 69.91 | 64.63 | 21.15 | 42.27 | 30.25 |
| 8750 | 60.71 | 74.75 | 69.06 | 25.64 | 46.16 | 31.97 |
| 10500 | 69.95 | 76.83 | 73.09 | 24.74 | 46.83 | 34.01 |
| 12250 | 71.68 | 77.86 | 75.45 | 28.85 | 48.12 | 36.30 |
| 14000 | 73.91 | 78.94 | 77.08 | 31.86 | 50.61 | 39.09 |
| 15750 | 76.45 | 80.17 | 78.84 | 36.19 | 51.06 | 41.98 |
| 17500 | 76.48 | 81.13 | 79.76 | 37.15 | 51.34 | 43.75 |
| 19250 | 77.65 | 81.64 | 80.48 | 39.09 | 52.77 | 45.69 |
| 21000 | 80.47 | 82.38 | 81.56 | 42.08 | 55.12 | 48.39 |
| 22750 | 79.99 | 83.13 | 81.95 | 46.83 | 56.73 | 50.21 |
| 24500 | 81.29 | 83.04 | 82.30 | 48.71 | 58.64 | 53.40 |
| 26250 | 82.22 | 83.42 | 82.92 | 50.20 | 62.13 | 55.79 |
| 28000 | 82.60 | 83.75 | 83.20 | 50.92 | 63.28 | 57.77 |
| 29750 | 82.73 | 84.20 | 83.66 | 52.83 | 64.24 | 59.19 |
| 31500 | 82.93 | 84.62 | 83.89 | 53.03 | 66.39 | 60.11 |
| 33250 | 83.53 | 84.71 | 84.13 | 56.73 | 70.13 | 63.73 |
| 35000 | 84.19 | 85.16 | 84.59 | 57.17 | 76.67 | 69.07 |

Table A.6: NB wrapper with large training set

## A.1.6 Expected Partition Entropy

| **Subset size** | **Maximum-Entropy** | | | **Naive-Bayes** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 39.82 | 54.94 | 44.37 | 14.25 | 23.22 | 17.49 |
| 3500 | 58.97 | 67.60 | 62.37 | 12.77 | 21.16 | 17.04 |
| 5250 | 67.29 | 73.41 | 70.32 | 14.36 | 28.09 | 18.78 |
| 7000 | 73.00 | 75.99 | 74.75 | 14.69 | 22.82 | 19.49 |
| 8750 | 76.06 | 78.90 | 77.76 | 17.06 | 29.39 | 22.62 |
| 10500 | 78.42 | 79.81 | 79.17 | 20.80 | 31.88 | 25.93 |
| 12250 | 79.50 | 80.99 | 80.23 | 24.02 | 37.35 | 30.02 |
| 14000 | 80.38 | 81.04 | 80.64 | 25.95 | 34.66 | 32.09 |
| 15750 | 80.62 | 82.46 | 81.43 | 26.34 | 38.40 | 33.51 |
| 17500 | 81.49 | 82.83 | 82.08 | 28.88 | 45.19 | 37.89 |
| 19250 | 81.90 | 83.43 | 82.74 | 29.34 | 45.47 | 39.16 |
| 21000 | 81.77 | 83.28 | 83.02 | 31.72 | 47.53 | 40.21 |
| 22750 | 82.00 | 83.76 | 83.33 | 31.99 | 52.75 | 44.27 |
| 24500 | 83.39 | 83.97 | 83.67 | 34.64 | 53.26 | 44.46 |
| 26250 | 83.56 | 84.13 | 83.88 | 40.44 | 56.03 | 48.71 |
| 28000 | 84.00 | 84.40 | 84.16 | 38.36 | 57.23 | 49.74 |
| 29750 | 83.98 | 84.69 | 84.30 | 38.74 | 57.75 | 51.35 |
| 31500 | 83.98 | 84.83 | 84.48 | 39.68 | 63.49 | 53.40 |
| 33250 | 84.39 | 84.86 | 84.66 | 34.97 | 66.23 | 56.12 |
| 35000 | 84.55 | 85.33 | 84.85 | 35.72 | 67.50 | 58.46 |

Table A.7: EPE with large training set

## A.2 Small Training Sets

This section contains the minimum, maximum, and mean percentage accuracy over ten runs of stepwise forward feature selection for smaller training sets of 5000 data points, with a maximum of 7000 features. This setting allowed use of the maximum-entropy wrapper method; we compare it with the naive-bayes wrapper and EPE filter. Both the maximum-entropy and naive-bayes learners were used for assessment, with a different split between held-out test data and training data on each run.

### A.2.1 Naive-Bayes Wrapper

| **Subset size** | **Maximum-Entropy** | | | **Naive-Bayes** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 350 | 23.25 | 40.80 | 29.39 | 18.14 | 25.86 | 20.61 |
| 700 | 37.55 | 48.87 | 45.04 | 19.66 | 32.76 | 26.84 |
| 1050 | 44.47 | 59.26 | 54.28 | 20.06 | 33.20 | 28.19 |
| 1400 | 51.28 | 64.70 | 59.70 | 26.72 | 38.86 | 32.28 |
| 1750 | 54.54 | 67.65 | 63.19 | 25.68 | 40.19 | 32.72 |
| 2100 | 58.37 | 70.20 | 65.62 | 26.40 | 39.45 | 33.35 |
| 2450 | 62.31 | 69.80 | 67.06 | 30.77 | 39.33 | 34.19 |
| 2800 | 66.12 | 70.84 | 68.05 | 30.81 | 40.83 | 35.50 |
| 3150 | 67.04 | 71.39 | 69.56 | 30.93 | 43.38 | 36.72 |
| 3500 | 69.49 | 71.56 | 70.73 | 31.63 | 44.17 | 38.03 |
| 3850 | 69.88 | 73.85 | 71.70 | 29.75 | 47.90 | 39.61 |
| 4200 | 71.43 | 73.44 | 72.24 | 30.93 | 47.26 | 41.45 |
| 4550 | 71.97 | 74.90 | 73.00 | 30.42 | 48.82 | 43.41 |
| 4900 | 72.44 | 74.60 | 73.38 | 31.95 | 50.49 | 44.47 |
| 5250 | 72.36 | 74.31 | 73.39 | 37.62 | 51.68 | 46.79 |
| 5600 | 72.53 | 74.77 | 73.49 | 37.44 | 53.55 | 48.57 |
| 5950 | 72.52 | 75.35 | 74.00 | 45.43 | 58.09 | 51.45 |
| 6300 | 73.49 | 75.08 | 74.21 | 48.16 | 63.61 | 54.49 |
| 6650 | 73.14 | 75.29 | 74.30 | 45.87 | 67.78 | 58.43 |
| 7000 | 73.72 | 75.74 | 74.51 | 47.14 | 70.03 | 61.81 |

Table A.8: NB wrapper with small training set

### A.2.2 Maximum-Entropy Wrapper

| **Subset size** | **Maximum-Entropy** | | | **Naive-Bayes** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 350 | 34.64 | 43.80 | 39.34 | 8.28 | 23.70 | 17.05 |
| 700 | 48.26 | 60.80 | 55.55 | 15.33 | 24.49 | 18.44 |
| 1050 | 55.97 | 65.53 | 61.82 | 13.52 | 26.18 | 18.90 |
| 1400 | 60.38 | 68.68 | 65.86 | 14.43 | 25.32 | 19.43 |
| 1750 | 65.89 | 69.85 | 68.52 | 13.38 | 32.59 | 21.41 |
| 2100 | 67.64 | 70.54 | 69.58 | 13.41 | 33.76 | 22.03 |
| 2450 | 69.06 | 71.36 | 70.64 | 16.14 | 33.47 | 22.69 |
| 2800 | 71.11 | 72.41 | 71.73 | 18.87 | 33.77 | 24.53 |
| 3150 | 70.97 | 72.92 | 72.17 | 20.48 | 34.57 | 26.29 |
| 3500 | 71.96 | 73.82 | 72.85 | 19.61 | 34.87 | 27.65 |
| 3850 | 72.58 | 74.58 | 73.22 | 20.20 | 36.42 | 30.18 |
| 4200 | 72.57 | 73.88 | 73.34 | 23.72 | 40.17 | 32.00 |
| 4550 | 73.28 | 74.35 | 73.75 | 24.50 | 40.85 | 34.16 |
| 4900 | 73.48 | 74.67 | 74.03 | 25.04 | 42.40 | 34.23 |
| 5250 | 73.71 | 74.63 | 74.23 | 27.37 | 43.73 | 35.28 |
| 5600 | 73.72 | 75.09 | 74.41 | 29.03 | 46.28 | 37.13 |
| 5950 | 73.39 | 75.25 | 74.46 | 32.71 | 49.30 | 38.05 |
| 6300 | 73.31 | 75.04 | 74.37 | 32.76 | 50.80 | 40.08 |
| 6650 | 73.64 | 75.10 | 74.56 | 31.07 | 53.33 | 43.15 |
| 7000 | 73.46 | 75.07 | 74.53 | 34.98 | 68.81 | 47.62 |

Table A.9: ME wrapper with small training set

### A.2.3 Expected Partition Entropy

| **Subset size** | **Maximum-Entropy** | | | **Naive-Bayes** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 350 | 37.56 | 46.14 | 41.15 | 14.83 | 24.59 | 18.53 |
| 700 | 53.33 | 63.03 | 57.81 | 13.74 | 22.86 | 18.70 |
| 1050 | 58.92 | 67.05 | 64.40 | 15.64 | 26.57 | 19.25 |
| 1400 | 65.47 | 69.14 | 67.37 | 15.44 | 29.71 | 20.56 |
| 1750 | 67.10 | 70.78 | 69.12 | 16.15 | 34.43 | 22.34 |
| 2100 | 68.80 | 70.79 | 70.02 | 17.60 | 34.23 | 23.28 |
| 2450 | 69.97 | 72.60 | 71.03 | 19.54 | 29.56 | 24.78 |
| 2800 | 70.87 | 72.98 | 71.94 | 23.00 | 34.09 | 27.16 |
| 3150 | 71.16 | 73.06 | 72.20 | 21.42 | 33.93 | 26.62 |
| 3500 | 71.19 | 73.29 | 72.27 | 22.79 | 36.38 | 29.45 |
| 3850 | 72.20 | 73.46 | 72.81 | 22.98 | 35.11 | 29.95 |
| 4200 | 71.62 | 74.45 | 73.01 | 26.46 | 35.17 | 30.25 |
| 4550 | 72.67 | 74.33 | 73.44 | 28.41 | 40.55 | 33.55 |
| 4900 | 72.87 | 74.25 | 73.67 | 28.27 | 43.13 | 34.78 |
| 5250 | 72.98 | 74.94 | 73.95 | 28.36 | 44.52 | 37.98 |
| 5600 | 72.51 | 74.95 | 73.96 | 30.86 | 46.74 | 40.46 |
| 5950 | 73.47 | 75.25 | 74.25 | 29.72 | 51.42 | 43.23 |
| 6300 | 72.58 | 75.02 | 73.89 | 34.48 | 51.94 | 44.11 |
| 6650 | 72.61 | 75.11 | 74.09 | 25.73 | 53.46 | 44.54 |
| 7000 | 72.92 | 75.03 | 74.29 | 25.38 | 62.53 | 49.23 |

Table A.10: EPE with small training set

# Appendix B

# Tables of Results: PoS Tagging and Conditional-Entropy Metrics

This appendix contains results of using the Expected Partition Entropy/Expected Covering Entropy (EPE/ECE) family of conditional-entropy metrics to evaluate feature sets obtained by stepwise forward selection in the part-of-speech tagging domain. EPE and ECE are formally defined in Chapter 4; experiments in which the feature sets were obtained are presented in 5; and Chapter 7 contains graphs and discussion of the results in this appendix. Recall that EPE is also known as zeroth-order entropy; entropies of first order and higher are ECEs.

Chapter 7 discusses the implications of selection by entropy; for the sake of clarity, it concentrates on the examples of NMI, NB, and EPE for large training sets. Here we present more complete results, which are essentially consistent with the conclusions drawn in Chapter 7.

Entropies of zeroth, first, second, and third order were found for feature sets selected using the naive mutual information (NMI), Koller-Sahami (KS), naive-bayes (NB), maximum-entropy (ME), and EPE metrics. The NMI and KS metrics had been used for feature selection only with large training sets of approximately 800 000 data points which contained a maximum of about 37 000 features; ME small training sets of 5000 data points, each of which contained at most about 7000 features; and EPE and NB for both small and large feature sets. Calculating the ECEs was rather time-consuming, and so the feature sets produced using FBC and random selection were not assessed using entropy.

Note that in order to speed up the computation of ECE, it was calculated only for the first 5000 words in the large training sets. In calculating all entropies, logarithms were taken to base *e*.

## B.1 Large Training Sets

### B.1.1 Naive Mutual Information

| **Subset size** | **Zeroth-Order Entropy** | | | **First-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 0.0038 | 0.6807 | 0.0776 | 0.3657 | 0.4034 | 0.3756 |
| 3500 | 0.0033 | 0.2986 | 0.0380 | 0.3257 | 0.3550 | 0.3314 |
| 5250 | 0.0031 | 0.1882 | 0.0264 | 0.3015 | 0.3270 | 0.3085 |
| 7000 | 0.0030 | 0.1356 | 0.0207 | 0.2826 | 0.3160 | 0.2926 |
| 8750 | 0.0029 | 0.1055 | 0.0173 | 0.2758 | 0.3053 | 0.2873 |
| 10500 | 0.0028 | 0.0867 | 0.0152 | 0.2714 | 0.2960 | 0.2812 |
| 12250 | 0.0027 | 0.0731 | 0.0136 | 0.2726 | 0.2944 | 0.2798 |
| 14000 | 0.0027 | 0.0635 | 0.0125 | 0.2682 | 0.2932 | 0.2773 |
| 15750 | 0.0027 | 0.0563 | 0.0116 | 0.2694 | 0.2929 | 0.2776 |
| 17500 | 0.0026 | 0.0504 | 0.0109 | 0.2694 | 0.2939 | 0.2770 |
| 19250 | 0.0026 | 0.0456 | 0.0103 | 0.2690 | 0.2941 | 0.2771 |
| 21000 | 0.0026 | 0.0418 | 0.0098 | 0.2692 | 0.2943 | 0.2772 |
| 22750 | 0.0026 | 0.0386 | 0.0093 | 0.2692 | 0.2943 | 0.2772 |
| 24500 | 0.0026 | 0.0358 | 0.0090 | 0.2692 | 0.2941 | 0.2772 |
| 26250 | 0.0025 | 0.0333 | 0.0086 | 0.2692 | 0.2896 | 0.2768 |
| 28000 | 0.0025 | 0.0310 | 0.0083 | 0.2682 | 0.2898 | 0.2763 |
| 29750 | 0.0025 | 0.0285 | 0.0079 | 0.2618 | 0.2837 | 0.2720 |
| 31500 | 0.0024 | 0.0251 | 0.0073 | 0.2595 | 0.2741 | 0.2643 |
| 33250 | 0.0022 | 0.0216 | 0.0065 | 0.2473 | 0.2551 | 0.2509 |
| 35000 | 0.0019 | 0.0167 | 0.0053 | 0.1814 | 0.1951 | 0.1905 |

Table B.1: NMI with large training set: Zeroth and first-order entropies

| **Subset size** | **Second-Order Entropy** | | | **Third-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 2.3618 | 2.4734 | 2.3896 | 2.4753 | 2.5436 | 2.4925 |
| 3500 | 2.3009 | 2.4644 | 2.3608 | 2.4474 | 2.5374 | 2.4749 |
| 5250 | 2.3133 | 2.4512 | 2.3395 | 2.4602 | 2.5333 | 2.4723 |
| 7000 | 2.3032 | 2.4690 | 2.3390 | 2.4593 | 2.5369 | 2.4757 |
| 8750 | 2.2969 | 2.4657 | 2.3470 | 2.4601 | 2.5301 | 2.4809 |
| 10500 | 2.3163 | 2.4568 | 2.3499 | 2.4663 | 2.5218 | 2.4817 |
| 12250 | 2.3146 | 2.4530 | 2.3534 | 2.4684 | 2.5222 | 2.4851 |
| 14000 | 2.3220 | 2.4599 | 2.3562 | 2.4684 | 2.5255 | 2.4874 |
| 15750 | 2.3211 | 2.4602 | 2.3536 | 2.4677 | 2.5271 | 2.4868 |
| 17500 | 2.3211 | 2.4615 | 2.3535 | 2.4682 | 2.5293 | 2.4869 |
| 19250 | 2.3245 | 2.4638 | 2.3588 | 2.4732 | 2.5316 | 2.4879 |
| 21000 | 2.3245 | 2.4647 | 2.3588 | 2.4734 | 2.5323 | 2.4881 |
| 22750 | 2.3253 | 2.4647 | 2.3596 | 2.4737 | 2.5321 | 2.4885 |
| 24500 | 2.3263 | 2.4648 | 2.3599 | 2.4744 | 2.5324 | 2.4891 |
| 26250 | 2.3263 | 2.4734 | 2.3609 | 2.4746 | 2.5380 | 2.4901 |
| 28000 | 2.3271 | 2.4745 | 2.3617 | 2.4760 | 2.5375 | 2.4908 |
| 29750 | 2.3372 | 2.4799 | 2.3689 | 2.4820 | 2.5352 | 2.4950 |
| 31500 | 2.3433 | 2.5233 | 2.3716 | 2.4838 | 2.5645 | 2.4974 |
| 33250 | 2.3507 | 2.5236 | 2.3813 | 2.4861 | 2.5689 | 2.5047 |
| 35000 | 2.3774 | 2.6042 | 2.4280 | 2.4961 | 2.6477 | 2.5365 |

Table B.2: NMI with large training set: Second and third-order entropies

### B.1.2 Koller-Sahami Metric

Note that feature selection by the Koller-Sahami (KS) metric was terminated when the metric attained its optimal value; this often occurred for as few as 15 000 features. Thus, the results for more than 15 000 features given below are the average of fewer than 10 runs, and should be treated with some caution. See Section 6.5 for discussion of the early halting of feature selection by KS. In addition to the 'standard' feature subset sizes, Tables B.3 and B.4 contain figures for the 'non-standard' subset sizes at which selection by KS was halted.

| **Subset size** | **Zeroth-Order Entropy** | | | **First-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 0.0102 | 1.3704 | 0.1668 | 1.7115 | 1.9833 | 1.8323 |
| 3500 | 0.0040 | 0.2629 | 0.0394 | 0.9229 | 1.4969 | 1.1699 |
| 5250 | 0.0022 | 0.1069 | 0.0175 | 0.3733 | 0.7961 | 0.5789 |
| 7000 | 0.0018 | 0.0638 | 0.0115 | 0.1701 | 0.3900 | 0.2299 |
| 8750 | 0.0018 | 0.0485 | 0.0096 | 0.0969 | 0.1578 | 0.1192 |
| 10500 | 0.0018 | 0.0401 | 0.0085 | 0.0842 | 0.1041 | 0.0923 |
| 12250 | 0.0017 | 0.0344 | 0.0078 | 0.0833 | 0.1008 | 0.0889 |
| 14000 | 0.0017 | 0.0301 | 0.0072 | 0.0832 | 0.1009 | 0.0876 |
| 15050 | 0.0267 | 0.0267 | 0.0267 | 0.1009 | 0.1009 | 0.1009 |
| 15750 | 0.0017 | 0.0136 | 0.0045 | 0.0832 | 0.0863 | 0.0852 |
| 17500 | 0.0017 | 0.0128 | 0.0044 | 0.0832 | 0.0862 | 0.0846 |
| 19250 | 0.0017 | 0.0122 | 0.0043 | 0.0831 | 0.0862 | 0.0839 |
| 20650 | 0.0029 | 0.0029 | 0.0029 | 0.0832 | 0.0832 | 0.0832 |
| 21000 | 0.0017 | 0.0116 | 0.0043 | 0.0831 | 0.0861 | 0.0839 |
| 21350 | 0.0017 | 0.0017 | 0.0017 | 0.0845 | 0.0845 | 0.0845 |
| 22050 | 0.0034 | 0.0034 | 0.0034 | 0.0831 | 0.0831 | 0.0831 |
| 22400 | 0.0062 | 0.0062 | 0.0062 | 0.0832 | 0.0832 | 0.0832 |
| 22750 | 0.0020 | 0.0111 | 0.0045 | 0.0831 | 0.0832 | 0.0832 |
| 24500 | 0.0019 | 0.0106 | 0.0044 | 0.0831 | 0.0832 | 0.0832 |
| 24850 | 0.0025 | 0.0025 | 0.0025 | 0.0832 | 0.0832 | 0.0832 |
| 26250 | 0.0019 | 0.0101 | 0.0047 | 0.0831 | 0.0832 | 0.0832 |
| 26600 | 0.0021 | 0.0021 | 0.0021 | 0.0831 | 0.0831 | 0.0831 |
| 28000 | 0.0019 | 0.0097 | 0.0054 | 0.0832 | 0.0832 | 0.0832 |
| 28350 | 0.0093 | 0.0093 | 0.0093 | 0.0832 | 0.0832 | 0.0832 |
| 29050 | 0.0019 | 0.0019 | 0.0019 | 0.0832 | 0.0832 | 0.0832 |
| 29750 | 0.0043 | 0.0043 | 0.0043 | 0.0832 | 0.0832 | 0.0832 |
| 31150 | 0.0043 | 0.0043 | 0.0043 | 0.0832 | 0.0832 | 0.0832 |

Table B.3: KS: Zeroth and first-order entropies

| **Subset size** | **Second-Order Entropy** | | | **Third-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 2.1539 | 2.4236 | 2.3019 | 2.3790 | 2.6183 | 2.5162 |
| 3500 | 1.7011 | 2.1499 | 1.8810 | 2.1331 | 2.4546 | 2.2499 |
| 5250 | 0.9802 | 1.7690 | 1.3519 | 1.6177 | 2.2065 | 1.8841 |
| 7000 | 0.4004 | 1.1253 | 0.6832 | 0.8799 | 1.7528 | 1.2483 |
| 8750 | 0.1964 | 0.4626 | 0.3116 | 0.4174 | 1.0296 | 0.7443 |
| 10500 | 0.1307 | 0.2598 | 0.1800 | 0.2796 | 0.6227 | 0.4003 |
| 12250 | 0.1068 | 0.1746 | 0.1349 | 0.1964 | 0.3939 | 0.2790 |
| 14000 | 0.0929 | 0.1252 | 0.1069 | 0.1415 | 0.2836 | 0.1999 |
| 15050 | 0.1107 | 0.1107 | 0.1107 | 0.2089 | 0.2089 | 0.2089 |
| 15750 | 0.0862 | 0.1141 | 0.0980 | 0.1115 | 0.1731 | 0.1475 |
| 17500 | 0.0848 | 0.0973 | 0.0903 | 0.0929 | 0.1426 | 0.1215 |
| 19250 | 0.0831 | 0.0940 | 0.0852 | 0.0864 | 0.1206 | 0.0999 |
| 20650 | 0.0832 | 0.0832 | 0.0832 | 0.0862 | 0.0862 | 0.0862 |
| 21000 | 0.0830 | 0.0909 | 0.0848 | 0.0830 | 0.1138 | 0.0922 |
| 21350 | 0.0882 | 0.0882 | 0.0882 | 0.0965 | 0.0965 | 0.0965 |
| 22050 | 0.0831 | 0.0831 | 0.0831 | 0.0830 | 0.0830 | 0.0830 |
| 22400 | 0.0831 | 0.0831 | 0.0831 | 0.0858 | 0.0858 | 0.0858 |
| 22750 | 0.0826 | 0.0832 | 0.0830 | 0.0841 | 0.0867 | 0.0855 |
| 24500 | 0.0831 | 0.0832 | 0.0831 | 0.0841 | 0.0859 | 0.0848 |
| 24850 | 0.0831 | 0.0831 | 0.0831 | 0.0841 | 0.0841 | 0.0841 |
| 26250 | 0.0831 | 0.0832 | 0.0832 | 0.0832 | 0.0859 | 0.0847 |
| 26600 | 0.0831 | 0.0831 | 0.0831 | 0.0846 | 0.0846 | 0.0846 |
| 28000 | 0.0831 | 0.0832 | 0.0832 | 0.0830 | 0.0853 | 0.0838 |
| 28350 | 0.0832 | 0.0832 | 0.0832 | 0.0830 | 0.0830 | 0.0830 |
| 29050 | 0.0832 | 0.0832 | 0.0832 | 0.0832 | 0.0832 | 0.0832 |
| 29750 | 0.0831 | 0.0831 | 0.0831 | 0.0853 | 0.0853 | 0.0853 |
| 31150 | 0.0832 | 0.0832 | 0.0832 | 0.0831 | 0.0831 | 0.0831 |

Table B.4: KS: Second and third-order entropies

## B.1.3 Naive Bayes Wrapper

| **Subset size** | **Zeroth-Order Entropy** | | | **First-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 0.0108 | 1.8003 | 0.2034 | 1.3999 | 2.3648 | 1.8125 |
| 3500 | 0.0073 | 0.5579 | 0.0702 | 1.1224 | 2.2803 | 1.6378 |
| 5250 | 0.0041 | 0.2454 | 0.0338 | 0.8372 | 2.0537 | 1.2452 |
| 7000 | 0.0023 | 0.1219 | 0.0186 | 0.6397 | 1.5037 | 0.8700 |
| 8750 | 0.0018 | 0.0755 | 0.0123 | 0.3689 | 0.9395 | 0.5844 |
| 10500 | 0.0016 | 0.0509 | 0.0089 | 0.2683 | 0.5943 | 0.3840 |
| 12250 | 0.0015 | 0.0409 | 0.0074 | 0.1851 | 0.3977 | 0.2666 |
| 14000 | 0.0014 | 0.0338 | 0.0064 | 0.1434 | 0.2391 | 0.1845 |
| 15750 | 0.0013 | 0.0286 | 0.0057 | 0.1056 | 0.1678 | 0.1321 |
| 17500 | 0.0013 | 0.0243 | 0.0052 | 0.0987 | 0.1217 | 0.1082 |
| 19250 | 0.0013 | 0.0220 | 0.0049 | 0.0909 | 0.1135 | 0.0988 |
| 21000 | 0.0012 | 0.0201 | 0.0047 | 0.0836 | 0.1101 | 0.0923 |
| 22750 | 0.0012 | 0.0185 | 0.0045 | 0.0836 | 0.1031 | 0.0895 |
| 24500 | 0.0012 | 0.0172 | 0.0043 | 0.0836 | 0.1058 | 0.0901 |
| 26250 | 0.0012 | 0.0160 | 0.0041 | 0.0835 | 0.0926 | 0.0868 |
| 28000 | 0.0012 | 0.0150 | 0.0040 | 0.0832 | 0.0926 | 0.0867 |
| 29750 | 0.0012 | 0.0141 | 0.0039 | 0.0832 | 0.0926 | 0.0864 |
| 31500 | 0.0012 | 0.0134 | 0.0037 | 0.0832 | 0.0926 | 0.0859 |
| 33250 | 0.0012 | 0.0127 | 0.0036 | 0.0832 | 0.0926 | 0.0859 |
| 35000 | 0.0012 | 0.0120 | 0.0035 | 0.0832 | 0.0926 | 0.0852 |

Table B.5: NB with large training set: Zeroth and first-order entropies

| **Subset size** | **Second-Order Entropy** | | | **Third-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 2.2915 | 2.5472 | 2.3951 | 2.4300 | 2.6172 | 2.5148 |
| 3500 | 2.0249 | 2.4907 | 2.2977 | 2.3395 | 2.6457 | 2.4937 |
| 5250 | 1.6833 | 2.3944 | 2.1286 | 2.0932 | 2.5951 | 2.3944 |
| 7000 | 1.5262 | 2.2835 | 1.9339 | 1.9721 | 2.5571 | 2.2807 |
| 8750 | 1.1875 | 2.0748 | 1.7544 | 1.6754 | 2.3579 | 2.1628 |
| 10500 | 0.9582 | 1.8040 | 1.4195 | 1.4588 | 2.1175 | 1.9428 |
| 12250 | 0.6788 | 1.4997 | 1.0674 | 1.2758 | 1.9383 | 1.6910 |
| 14000 | 0.4100 | 1.0815 | 0.7729 | 1.0429 | 1.6705 | 1.4540 |
| 15750 | 0.2949 | 0.7432 | 0.4749 | 0.7814 | 1.3058 | 1.0645 |
| 17500 | 0.1858 | 0.5560 | 0.3224 | 0.5732 | 1.1290 | 0.7978 |
| 19250 | 0.1392 | 0.5226 | 0.2346 | 0.3960 | 0.8967 | 0.5950 |
| 21000 | 0.1171 | 0.4668 | 0.2014 | 0.2862 | 0.6948 | 0.4638 |
| 22750 | 0.0952 | 0.3957 | 0.1598 | 0.1699 | 0.5906 | 0.3635 |
| 24500 | 0.0843 | 0.1723 | 0.1141 | 0.1237 | 0.4734 | 0.2670 |
| 26250 | 0.0831 | 0.1319 | 0.0985 | 0.0913 | 0.3125 | 0.1807 |
| 28000 | 0.0850 | 0.1201 | 0.0949 | 0.0892 | 0.2177 | 0.1374 |
| 29750 | 0.0847 | 0.0926 | 0.0877 | 0.0892 | 0.1835 | 0.1219 |
| 31500 | 0.0832 | 0.0926 | 0.0862 | 0.0851 | 0.1361 | 0.1074 |
| 33250 | 0.0832 | 0.0926 | 0.0859 | 0.0831 | 0.1238 | 0.0951 |
| 35000 | 0.0832 | 0.0926 | 0.0852 | 0.0831 | 0.0926 | 0.0855 |

Table B.6: NB with large training set: Second and third-order entropies

### B.1.4 Expected Partition Entropy

| **Subset size** | **Zeroth-Order Entropy** | | | **First-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 0.0067 | 1.2463 | 0.1409 | 1.4021 | 2.0487 | 1.8505 |
| 3500 | 0.0027 | 0.2508 | 0.0321 | 0.9668 | 1.4435 | 1.1651 |
| 5250 | 0.0016 | 0.1017 | 0.0142 | 0.3748 | 0.8052 | 0.5387 |
| 7000 | 0.0014 | 0.0621 | 0.0095 | 0.1660 | 0.4798 | 0.2391 |
| 8750 | 0.0013 | 0.0484 | 0.0079 | 0.0987 | 0.1603 | 0.1313 |
| 10500 | 0.0013 | 0.0401 | 0.0070 | 0.0860 | 0.1121 | 0.0965 |
| 12250 | 0.0013 | 0.0344 | 0.0064 | 0.0831 | 0.0995 | 0.0882 |
| 14000 | 0.0013 | 0.0301 | 0.0059 | 0.0826 | 0.0934 | 0.0860 |
| 15750 | 0.0013 | 0.0267 | 0.0055 | 0.0827 | 0.0926 | 0.0848 |
| 17500 | 0.0013 | 0.0240 | 0.0052 | 0.0831 | 0.0926 | 0.0848 |
| 19250 | 0.0013 | 0.0219 | 0.0049 | 0.0832 | 0.0926 | 0.0846 |
| 21000 | 0.0012 | 0.0200 | 0.0047 | 0.0830 | 0.0926 | 0.0843 |
| 22750 | 0.0012 | 0.0185 | 0.0045 | 0.0831 | 0.0926 | 0.0842 |
| 24500 | 0.0012 | 0.0172 | 0.0043 | 0.0832 | 0.0926 | 0.0841 |
| 26250 | 0.0012 | 0.0160 | 0.0041 | 0.0832 | 0.0926 | 0.0841 |
| 28000 | 0.0012 | 0.0150 | 0.0040 | 0.0832 | 0.0926 | 0.0841 |
| 29750 | 0.0012 | 0.0141 | 0.0039 | 0.0832 | 0.0926 | 0.0841 |
| 31500 | 0.0012 | 0.0134 | 0.0037 | 0.0832 | 0.0926 | 0.0841 |
| 33250 | 0.0012 | 0.0127 | 0.0036 | 0.0832 | 0.0926 | 0.0841 |
| 35000 | 0.0012 | 0.0120 | 0.0038 | 0.0832 | 0.0926 | 0.0842 |

Table B.7: EPE with large training set: Zeroth and first-order entropies

| **Subset size** | **Second-Order Entropy** | | | **Third-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 1750 | 2.2266 | 2.4961 | 2.3209 | 2.3919 | 2.6533 | 2.5218 |
| 3500 | 1.6924 | 2.1015 | 1.8448 | 2.0943 | 2.4192 | 2.2160 |
| 5250 | 1.0169 | 1.6043 | 1.2454 | 1.5282 | 1.9897 | 1.7555 |
| 7000 | 0.5239 | 1.2313 | 0.7017 | 1.0101 | 1.8104 | 1.2463 |
| 8750 | 0.2522 | 0.6020 | 0.3833 | 0.5485 | 1.1207 | 0.8405 |
| 10500 | 0.1760 | 0.2750 | 0.2201 | 0.3675 | 0.6398 | 0.5342 |
| 12250 | 0.0941 | 0.1725 | 0.1352 | 0.2237 | 0.4838 | 0.3239 |
| 14000 | 0.0888 | 0.1446 | 0.1116 | 0.1586 | 0.3303 | 0.2248 |
| 15750 | 0.0866 | 0.1362 | 0.0970 | 0.1018 | 0.2505 | 0.1603 |
| 17500 | 0.0836 | 0.1294 | 0.0926 | 0.0930 | 0.2165 | 0.1286 |
| 19250 | 0.0831 | 0.1232 | 0.0892 | 0.0861 | 0.1730 | 0.1059 |
| 21000 | 0.0830 | 0.0926 | 0.0855 | 0.0829 | 0.1294 | 0.0952 |
| 22750 | 0.0830 | 0.0926 | 0.0850 | 0.0829 | 0.1125 | 0.0907 |
| 24500 | 0.0830 | 0.0926 | 0.0849 | 0.0825 | 0.0940 | 0.0871 |
| 26250 | 0.0830 | 0.0926 | 0.0847 | 0.0830 | 0.0936 | 0.0865 |
| 28000 | 0.0830 | 0.0926 | 0.0841 | 0.0830 | 0.0931 | 0.0847 |
| 29750 | 0.0832 | 0.0926 | 0.0841 | 0.0830 | 0.0931 | 0.0841 |
| 31500 | 0.0832 | 0.0926 | 0.0841 | 0.0830 | 0.0926 | 0.0841 |
| 33250 | 0.0832 | 0.0926 | 0.0841 | 0.0830 | 0.0926 | 0.0841 |
| 35000 | 0.0832 | 0.0926 | 0.0842 | 0.0830 | 0.0926 | 0.0842 |

Table B.8: EPE with large training set: Second and third-order entropies

## B.2 Small Training Sets

### B.2.1 Naive-Bayes Wrapper

| **Subset size** | **Zeroth-Order Entropy** | | | **First-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 350 | 0.00999 | 1.80181 | 0.20269 | 1.18245 | 2.39227 | 1.75819 |
| 700 | 0.00503 | 0.55546 | 0.06646 | 0.84441 | 1.74106 | 1.21650 |
| 1050 | 0.00256 | 0.25666 | 0.03137 | 0.55755 | 1.26301 | 0.80442 |
| 1400 | 0.00133 | 0.12859 | 0.01617 | 0.37125 | 0.91046 | 0.53351 |
| 1750 | 0.00084 | 0.08367 | 0.01046 | 0.19170 | 0.49920 | 0.30209 |
| 2100 | 0.00065 | 0.04379 | 0.00591 | 0.12880 | 0.28596 | 0.19331 |
| 2450 | 0.00060 | 0.02802 | 0.00418 | 0.08388 | 0.27271 | 0.15228 |
| 2800 | 0.00052 | 0.01620 | 0.00291 | 0.07448 | 0.21590 | 0.11641 |
| 3150 | 0.00042 | 0.01388 | 0.00258 | 0.06242 | 0.13891 | 0.09524 |
| 3500 | 0.00041 | 0.00959 | 0.00210 | 0.04914 | 0.10346 | 0.06836 |
| 3850 | 0.00041 | 0.00816 | 0.00192 | 0.04796 | 0.09427 | 0.06570 |
| 4200 | 0.00040 | 0.00748 | 0.00183 | 0.04796 | 0.08411 | 0.06179 |
| 4550 | 0.00040 | 0.00690 | 0.00176 | 0.04287 | 0.08411 | 0.06106 |
| 4900 | 0.00040 | 0.00641 | 0.00169 | 0.04287 | 0.08411 | 0.06105 |
| 5250 | 0.00039 | 0.00598 | 0.00163 | 0.04275 | 0.08411 | 0.06158 |
| 5600 | 0.00039 | 0.00561 | 0.00158 | 0.04275 | 0.08411 | 0.06103 |
| 5950 | 0.00039 | 0.00528 | 0.00153 | 0.04275 | 0.08411 | 0.06103 |
| 6300 | 0.00039 | 0.00499 | 0.00149 | 0.04275 | 0.08411 | 0.06103 |
| 6650 | 0.00039 | 0.00472 | 0.00145 | 0.04275 | 0.08411 | 0.06103 |
| 7000 | 0.00038 | 0.00449 | 0.00141 | 0.04275 | 0.08411 | 0.06103 |

Table B.9: NB with small training set: Zeroth and first-order entropies

| **Subset size** | **Second-Order Entropy** | | | **Third-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 350 | 2.33643 | 2.60023 | 2.44137 | 2.40541 | 2.71950 | 2.54463 |
| 700 | 2.07167 | 2.60959 | 2.25634 | 2.36345 | 2.71304 | 2.49786 |
| 1050 | 1.74912 | 2.30093 | 2.00706 | 2.17489 | 2.53670 | 2.37010 |
| 1400 | 1.50563 | 2.07225 | 1.77784 | 2.05291 | 2.43055 | 2.24068 |
| 1750 | 1.07139 | 1.73184 | 1.36310 | 1.64476 | 2.27620 | 1.98932 |
| 2100 | 0.51472 | 1.59859 | 1.05667 | 1.30659 | 1.94411 | 1.71848 |
| 2450 | 0.22868 | 1.17215 | 0.74160 | 0.71033 | 1.80518 | 1.39276 |
| 2800 | 0.23164 | 0.92522 | 0.53379 | 0.77775 | 1.50323 | 1.15664 |
| 3150 | 0.18174 | 0.70569 | 0.39732 | 0.56071 | 1.28395 | 0.92539 |
| 3500 | 0.07334 | 0.59829 | 0.25650 | 0.31076 | 1.12019 | 0.68688 |
| 3850 | 0.06578 | 0.56065 | 0.17408 | 0.20093 | 1.10281 | 0.48635 |
| 4200 | 0.06381 | 0.16242 | 0.10314 | 0.19230 | 0.48432 | 0.31443 |
| 4550 | 0.06018 | 0.12767 | 0.08925 | 0.10387 | 0.45543 | 0.24817 |
| 4900 | 0.05286 | 0.12767 | 0.08153 | 0.07123 | 0.44098 | 0.18299 |
| 5250 | 0.05121 | 0.12028 | 0.07146 | 0.06815 | 0.22099 | 0.11874 |
| 5600 | 0.04736 | 0.09667 | 0.06656 | 0.06815 | 0.12028 | 0.08862 |
| 5950 | 0.04736 | 0.09667 | 0.06656 | 0.06018 | 0.10387 | 0.07777 |
| 6300 | 0.04275 | 0.09667 | 0.06555 | 0.05270 | 0.09667 | 0.07031 |
| 6650 | 0.04275 | 0.08411 | 0.06103 | 0.04145 | 0.09667 | 0.06755 |
| 7000 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.08411 | 0.06316 |

Table B.10: NB with small training set: Second and third-order entropies

### B.2.2 Maximum-Entropy Wrapper

| **Subset size** | **Zeroth-Order Entropy** | | | **First-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 350 | 0.00844 | 1.06906 | 0.12375 | 1.32562 | 2.09197 | 1.75880 |
| 700 | 0.00274 | 0.20720 | 0.02763 | 0.87124 | 1.57415 | 1.22989 |
| 1050 | 0.00136 | 0.07198 | 0.01091 | 0.52899 | 1.24708 | 0.78299 |
| 1400 | 0.00084 | 0.03785 | 0.00586 | 0.23695 | 0.76229 | 0.42385 |
| 1750 | 0.00060 | 0.02552 | 0.00408 | 0.16585 | 0.34762 | 0.24718 |
| 2100 | 0.00046 | 0.01781 | 0.00311 | 0.11866 | 0.29732 | 0.16645 |
| 2450 | 0.00044 | 0.01431 | 0.00266 | 0.06563 | 0.23987 | 0.11844 |
| 2800 | 0.00043 | 0.01187 | 0.00236 | 0.05659 | 0.14551 | 0.08249 |
| 3150 | 0.00041 | 0.01045 | 0.00220 | 0.04796 | 0.12585 | 0.07327 |
| 3500 | 0.00040 | 0.00940 | 0.00207 | 0.04306 | 0.11237 | 0.06925 |
| 3850 | 0.00040 | 0.00855 | 0.00196 | 0.04306 | 0.08671 | 0.06382 |
| 4200 | 0.00040 | 0.00784 | 0.00187 | 0.04306 | 0.08411 | 0.06108 |
| 4550 | 0.00040 | 0.00690 | 0.00176 | 0.04287 | 0.08411 | 0.06106 |
| 4900 | 0.00040 | 0.00641 | 0.00169 | 0.04287 | 0.08411 | 0.06104 |
| 5250 | 0.00039 | 0.00598 | 0.00163 | 0.04275 | 0.08411 | 0.06103 |
| 5600 | 0.00039 | 0.00561 | 0.00158 | 0.04275 | 0.08411 | 0.06103 |
| 5950 | 0.00039 | 0.00528 | 0.00153 | 0.04275 | 0.08411 | 0.06103 |
| 6300 | 0.00039 | 0.00499 | 0.00149 | 0.04275 | 0.08411 | 0.06103 |
| 6650 | 0.00039 | 0.00472 | 0.00145 | 0.04275 | 0.08411 | 0.06103 |
| 7000 | 0.00038 | 0.00449 | 0.00141 | 0.04275 | 0.08411 | 0.06103 |

Table B.11: ME with small training set: Zeroth and first-order entropies

| **Subset size** | **Second-Order Entropy** | | | **Third-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 350 | 2.13045 | 2.46302 | 2.32540 | 2.41287 | 2.61951 | 2.52242 |
| 700 | 1.65177 | 2.28672 | 2.01163 | 2.09289 | 2.59921 | 2.36409 |
| 1050 | 1.24143 | 2.15681 | 1.71657 | 1.76964 | 2.48543 | 2.17805 |
| 1400 | 0.82022 | 1.94443 | 1.26849 | 1.40216 | 2.35299 | 1.86815 |
| 1750 | 0.62222 | 1.30959 | 0.91216 | 1.10899 | 1.94842 | 1.58134 |
| 2100 | 0.28718 | 1.08471 | 0.61372 | 0.75247 | 1.78582 | 1.21127 |
| 2450 | 0.23776 | 0.55638 | 0.38333 | 0.62462 | 1.20261 | 0.90026 |
| 2800 | 0.13525 | 0.37781 | 0.23225 | 0.36836 | 0.76324 | 0.60545 |
| 3150 | 0.11118 | 0.23638 | 0.16398 | 0.25000 | 0.67098 | 0.49360 |
| 3500 | 0.06742 | 0.21353 | 0.11741 | 0.16759 | 0.62021 | 0.36203 |
| 3850 | 0.04306 | 0.14621 | 0.08475 | 0.10415 | 0.42975 | 0.23932 |
| 4200 | 0.04306 | 0.09838 | 0.07341 | 0.08815 | 0.36218 | 0.17164 |
| 4550 | 0.04287 | 0.08411 | 0.06830 | 0.07167 | 0.22063 | 0.13351 |
| 4900 | 0.04287 | 0.08411 | 0.06869 | 0.05286 | 0.17047 | 0.09375 |
| 5250 | 0.04275 | 0.08411 | 0.06103 | 0.05286 | 0.12938 | 0.07485 |
| 5600 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.09555 | 0.06751 |
| 5950 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.08974 | 0.06364 |
| 6300 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.08974 | 0.06239 |
| 6650 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.08411 | 0.06103 |
| 7000 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.08411 | 0.06103 |

Table B.12: ME with small training set: Second and third-order entropies

### B.2.3 Expected Partition Entropy

| **Subset size** | **Zeroth-Order Entropy** | | | **First-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 350 | 0.00689 | 0.87186 | 0.10271 | 1.08095 | 1.92845 | 1.71987 |
| 700 | 0.00146 | 0.09949 | 0.01451 | 0.59409 | 1.44773 | 0.95618 |
| 1050 | 0.00078 | 0.03394 | 0.00539 | 0.16272 | 0.73279 | 0.39121 |
| 1400 | 0.00061 | 0.02270 | 0.00360 | 0.10530 | 0.34139 | 0.16390 |
| 1750 | 0.00043 | 0.01795 | 0.00304 | 0.07700 | 0.16466 | 0.10772 |
| 2100 | 0.00041 | 0.01496 | 0.00271 | 0.05146 | 0.11090 | 0.08539 |
| 2450 | 0.00041 | 0.01282 | 0.00247 | 0.05146 | 0.08411 | 0.07094 |
| 2800 | 0.00041 | 0.01122 | 0.00229 | 0.04414 | 0.08411 | 0.06353 |
| 3150 | 0.00041 | 0.00997 | 0.00214 | 0.04414 | 0.08411 | 0.06280 |
| 3500 | 0.00040 | 0.00898 | 0.00202 | 0.04414 | 0.08411 | 0.06218 |
| 3850 | 0.00040 | 0.00816 | 0.00192 | 0.04298 | 0.08411 | 0.06130 |
| 4200 | 0.00040 | 0.00748 | 0.00183 | 0.04275 | 0.08411 | 0.06128 |
| 4550 | 0.00040 | 0.00690 | 0.00176 | 0.04275 | 0.08411 | 0.06129 |
| 4900 | 0.00040 | 0.00641 | 0.00169 | 0.04275 | 0.08411 | 0.06103 |
| 5250 | 0.00039 | 0.00598 | 0.00163 | 0.04275 | 0.08411 | 0.06103 |
| 5600 | 0.00039 | 0.00561 | 0.00158 | 0.04275 | 0.08411 | 0.06103 |
| 5950 | 0.00039 | 0.00528 | 0.00153 | 0.04275 | 0.08411 | 0.06103 |
| 6300 | 0.00039 | 0.00499 | 0.00149 | 0.04275 | 0.08411 | 0.06103 |
| 6650 | 0.00039 | 0.00472 | 0.00145 | 0.04275 | 0.08411 | 0.06103 |
| 7000 | 0.00038 | 0.00449 | 0.00141 | 0.04275 | 0.08411 | 0.06103 |

Table B.13: EPE with small training set: Zeroth and first-order entropies

| **Subset size** | **Second-Order Entropy** | | | **Third-Order Entropy** | | |
|---|---|---|---|---|---|---|
| | *Minimum* | *Maximum* | *Mean* | *Minimum* | *Maximum* | *Mean* |
| 350 | 2.16026 | 2.49560 | 2.33069 | 2.38906 | 2.71052 | 2.55185 |
| 700 | 1.33751 | 2.10093 | 1.71875 | 1.91439 | 2.41208 | 2.18651 |
| 1050 | 0.56193 | 1.71432 | 1.13308 | 1.14531 | 2.27878 | 1.73270 |
| 1400 | 0.31358 | 0.93414 | 0.54084 | 0.70094 | 1.60337 | 1.12966 |
| 1750 | 0.22895 | 0.55852 | 0.30651 | 0.48029 | 1.16343 | 0.74407 |
| 2100 | 0.15700 | 0.28071 | 0.20454 | 0.34346 | 0.74723 | 0.53890 |
| 2450 | 0.10788 | 0.19958 | 0.15271 | 0.30350 | 0.59131 | 0.41250 |
| 2800 | 0.07799 | 0.16561 | 0.11570 | 0.19677 | 0.41086 | 0.31162 |
| 3150 | 0.05156 | 0.14921 | 0.09362 | 0.13617 | 0.28012 | 0.22522 |
| 3500 | 0.04406 | 0.14063 | 0.08265 | 0.10642 | 0.27163 | 0.19200 |
| 3850 | 0.04291 | 0.14063 | 0.07652 | 0.07900 | 0.22372 | 0.14563 |
| 4200 | 0.04275 | 0.13667 | 0.07200 | 0.07099 | 0.19015 | 0.12961 |
| 4550 | 0.04275 | 0.08411 | 0.06229 | 0.04598 | 0.19050 | 0.10611 |
| 4900 | 0.04275 | 0.08411 | 0.06181 | 0.04268 | 0.19050 | 0.08844 |
| 5250 | 0.04275 | 0.08411 | 0.06082 | 0.04275 | 0.14935 | 0.07664 |
| 5600 | 0.04275 | 0.08411 | 0.06100 | 0.04275 | 0.12028 | 0.06761 |
| 5950 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.12028 | 0.06681 |
| 6300 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.08411 | 0.06103 |
| 6650 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.08411 | 0.06103 |
| 7000 | 0.04275 | 0.08411 | 0.06103 | 0.04275 | 0.08411 | 0.06103 |

Table B.14: EPE with small training set: Second and third-order entropies

# Appendix C

# Tables of Results: Reuters Document Classification

The following tables refer to the experiments with classification of Reuters news stories by topic presented in Chapter 8; see that chapter for additional details.

## C.1 Order of the ECE Metric

Tables C.1 and C.2 give the mean size of subset at which ECE of each order reached its minimal value of zero, and the number of training sets for which this minimization occurred, for the `coffee/iron-steel/livestock` and `gold/reserves/gdp` classification tasks. As discussed in Sections 4.5 and 8.5, the order is the maximum number of feature values by which two vectors of feature values can differ, if they are to be regarded as equivalent for the purposes of calculating the expected entropy.

The final order of ECE used was one greater than the highest order at which optimisation occurred. So for the first classification task, the final order was 22 for eight training/test splits, 27 for one split, and 28 for one split. For the second, the final order was 31 in all cases.

| **ECE Order** | **Mean Size for Optimisation** | **Number of Sets** |
|---|---|---|
| 0 | 81 | 10 |
| 1 | 115 | 10 |
| 2 | 145 | 10 |
| 3 | 172 | 10 |
| 4 | 208 | 10 |
| 5 | 232 | 10 |
| 6 | 260 | 10 |
| 7 | 290 | 10 |
| 8 | 318 | 10 |
| 9 | 354 | 10 |
| 10 | 384 | 10 |
| 11 | 414 | 10 |
| 12 | 445 | 10 |
| 13 | 473 | 10 |
| 14 | 504 | 10 |
| 15 | 540 | 10 |
| 16 | 580 | 10 |
| 17 | 620 | 10 |
| 18 | 659 | 10 |
| 19 | 707 | 10 |
| 20 | 762 | 10 |
| 21 | 844 | 10 |
| 22 | 825 | 2 |
| 23 | 875 | 2 |
| 24 | 905 | 2 |
| 25 | 980 | 2 |
| 26 | 1060 | 2 |
| 27 | 1190 | 1 |

Table C.1: Behaviour of the ECE metric with `coffee/iron-steel/livestock` training data.

| **ECE Order** | **Mean Size for Optimisation** | **Number of Sets** |
|---|---|---|
| 0 | 62 | 10 |
| 1 | 93 | 10 |
| 2 | 116 | 10 |
| 3 | 142 | 10 |
| 4 | 170 | 10 |
| 5 | 194 | 10 |
| 6 | 219 | 10 |
| 7 | 242 | 10 |
| 8 | 268 | 10 |
| 9 | 293 | 10 |
| 10 | 321 | 10 |
| 11 | 348 | 10 |
| 12 | 372 | 10 |
| 13 | 401 | 10 |
| 14 | 427 | 10 |
| 15 | 456 | 10 |
| 16 | 485 | 10 |
| 17 | 508 | 10 |
| 18 | 539 | 10 |
| 19 | 570 | 10 |
| 20 | 603 | 10 |
| 21 | 635 | 10 |
| 22 | 667 | 10 |
| 23 | 703 | 10 |
| 24 | 734 | 10 |
| 25 | 770 | 10 |
| 26 | 804 | 10 |
| 27 | 844 | 10 |
| 28 | 886 | 10 |
| 29 | 953 | 10 |
| 30 | 1015 | 10 |

Table C.2: Behaviour of the ECE metric with `gold/reserves/gdp` training data.

## C.2 Accuracy on Test Data

Evaluation was conducted with the naive-bayes learner described in Section 2.3.3, and tenfold cross-validation was carried out. Tables C.3 and C.4 give the mean accuracy over each of the ten training/test splits for each size of feature subset. The mean accuracy for the maximal sets of approximately 2400 features was 90.09% for the `coffee/iron-steel/livestock` task and 89.60% for `gold/reserves/gdp`.

| **Number of Features** | **Mean Pecentage Accuracy** |
|---|---|
| 100 | 78.63 |
| 200 | 83.71 |
| 300 | 88.78 |
| 400 | 89.75 |
| 500 | 88.19 |
| 600 | 89.77 |
| 700 | 91.36 |
| 800 | 92.32 |
| 900 | 91.04 |
| 1000 | 91.36 |
| 1100 | 92.00 |
| 1200 | 90.72 |
| 1300 | 90.08 |
| 1400 | 89.76 |
| 1500 | 89.12 |

Table C.3: Classification accuracy with `coffee/iron-steel/livestock` training data and feature subsets selected by ECE.

| **Number of Features** | **Mean Accuracy** |
|---|---|
| 100 | 90.13 |
| 200 | 93.92 |
| 300 | 92.76 |
| 400 | 91.59 |
| 500 | 92.46 |
| 600 | 92.76 |
| 700 | 92.76 |
| 800 | 92.77 |
| 900 | 92.76 |
| 1000 | 92.19 |
| 1100 | 92.19 |
| 1200 | 92.77 |
| 1300 | 91.63 |
| 1400 | 91.06 |
| 1500 | 91.34 |

Table C.4: Classification accuracy with `gold/reserves/gdp` training data and feature subsets selected by ECE.

# Bibliography

[AB96] D.W. Aha and R.L. Bankert. A comparative evaluation of sequential feature selection algorithms. In D. Fisher and J.H. Lenz, editors, *Artificial Intelligence and Statistics*. Springer-Verlag, 1996.

[Acu03] Edgar Acuna. A comparison of filters and wrappers methods for feature selection methods in supervised classification. In *Interface 2003: Security and Infrastructure Protection, Salt Lake City, Utah*, 2003.

[AD91] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, pages 547–552, Anaheim, California, 1991. AAAI Press.

[Arm97] M.A. Armstrong. *Groups and Symmetry*. Springer-Verlag, 1997.

[AZ00] D. Antonic and M. Zagar. Method for determining classification significant features from acoustic signature of mine-like buried objects. In *Landmine Detection Workshop, 15th World Conference on Non-Destructive Testing*, Rome, Italy, October 2000.

[BDPDP96] A. Berger, S. Della Pietra, and V. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[Ber99] A. Berger. Error-correcting output coding for text classification. In *IJCAI'99: Workshop on machine learning for information filtering*, 1999. Stockholm, Sweden.

[BL97] Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[CT91] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley Interscience, 1991.

[DCSL02] M. Dash, K. Choi, P. Scheuermann, and H. Liu. Feature selection for clustering – a filter solution. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*, pages 115–122, December 2002.

[DL97] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3), 1997.

[DPDPL97] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on pattern analysis and machine intelligence*, 19(4):380–393, April 1997.

[For03] George Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289–1305, March 2003.

[GE03] Isabelle Guyon and Andre Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

[Gol89] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[GT03] Amir Globerson and Naftali Tishby. Sufficient dimensionality reduction. *Journal of Machine Learning Research*, 3:1289–1305, Mar 2003.

[GW86] G. Grimmett and D. Welsh. *Probability: An Introduction*. Oxford University Press, 1986.

[HY01] M.H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96:746–774, 2001.

[IGS01] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: Categorizing hidden web databases. In *SIGMOD Conference*, 2001.

[ILS01] I. Inza, P. Larranaga, and B. Sierra. Feature subset selection by bayesian networks: a comparison with genetic and sequential algorithms. *International Journal of Approximate Reasoning*, 27(2):143–164, 2001.

[JKP94] George John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129. Morgan Kaufmann, July 1994.

[KJ97] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[KJ98] R. Kohavi and G.H. John. The wrapper approach. In H. Liu and H. Motoda, editors, *Feature Selection for Knowledge Discovery and Data Mining*, pages 33–50. Kluwer Academic Publishers, 1998.

[KL97] Kambhatla and Lee. Dimension reduction by local principal component analysis. *Neural Computation*, 9:1493–1516, 1997.

[KS95] Ron Kohavi and Dan Sommerfield. Feature subset selection using the wrapper model: Overfitting and dynamic search space topology. In *The First International Conference on Knowledge Discovery and Data Mining*, pages 192–197. AAAI Press, Menlo Park, California, August 1995. Journal version in AIJ, available at http://citeseer.nj.nec.com/13663.html.

[KS96] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996.

[Lan94] Pat Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance*, New Orleans, USA, 1994. AAAI Press.

[Lew98] David Lewis. Naive bayes at forty: The independence assumption in information retrieval. In *Proc. 10th European Conference on Machine Learning ECML-98*, pages 4–15, 1998.

[LS94] Pat Langley and Stephanie Sage. Oblivious decision trees and abstract cases. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*. AAAI Press, 1994.

[LS98] H. Liu and R. Setiono. Some issues on scalable feature selection. In *4th World Congress of Expert Systems: Application of Advanced Info. Technologies*, 1998.

[LS6a] H. Liu and R. Setiono. Feature selection and classification - a probabilistic wrapper approach. In *Proceedings of the Ninth International Conference on Industrial and Engineering Applications of AI and ES*, 1996a.

[LY02] H. Liu and L. Yu. Feature selection for data mining. Survey draft available at http://www.public.asu.edu/ huanliu/, 2002.

[Mal02] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pages 49–55, 2002.

[MN98] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*. 1998.

[MSM95] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Penn treebank release 2. Linguistic Data Consortium (LDC), University of Pennsylvania, 1995. CD-ROM catalogue number LDC95T7.

[Pri97] H.A. Priestley. *Introduction to Integration*. Oxford University Press, 1997.

[Rat96] Adwait Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, 1996.

[Rat98] Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, 1998.

[Reu] Reuters. Reuters text categorization corpus 21578. Available from http://www.daviddlewis.com/resources/testcollections/reuters21578/. See also http://about.reuters.com/researchandstandards/corpus/available.asp.

[Ros96] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computers, Speech and Language*, (10):187–228, 1996.

[Sah99] M. Sahami. *Using Machine Learning to Improve Information Access*. PhD thesis, Computer Science Department, Stanford University, 1999.

[SH98] D. H. Schuschel and C.N. Hsu. A weight analysis-based wrapper approach to neural nets feature selection. In *Proceedings of the 10th IEEE International Conference on Tools with AI (ICTAI-98)*, pages 89–96, 1998.

[Ska94] David B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *International Conference on Machine Learning*, pages 293–301, 1994.

[SL00] H.S. Seung and D.D. Lee. The manifold ways of perception. *Science*, 290:2319–2323, 2000.

[SN02] Marc Sebban and Richard Nock. A hybrid filter/wrapper approach of feature selection using information theory. *Pattern Recognition*, 35(4):835–846, 2002.

[SYBL02] Z. Sun, X. Yuan, G. Bebis, and S. Louis. Genetic feature subset selection for gender classification: A comparison study. In *Sixth IEEE Workshop on Applications of Computer Vision*, December 2002.

[TA03] M. Toews and T. Arbel. Entropy-of-likelihood feature selection for image correspondence. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 1041 –1047, October 2003.

[TdSL00] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science,* 290:2319–2323, 2000.

[TFMF01] G. D. Tourassi, E. D. Frederick, M. K. Markey, and C.E. Floyd, Jr. Application of the mutual information criterion for feature selection in computer-aided diagnosis. *Medical Physics*, 28:2394–2402, 2001.

[VD93] H. Vafaie and K. DeJong. Robust feature selection algorithms. In *Proceedings of the Fifth Conference on Tools for Artificial Intelligence, Boston, MA*, pages 356–363, 1993.

[VD95] H. Vafaie and K. DeJong. Genetic algorithms as a tool for restructuring feature space representation. In *Proc. of the International Conference on Tools with Artificial Intelligence*. IEEE Computer Soc. Press, 1995.

[XCDS02] L. Xie, S.F. Chang, A. Divakaran, and H. Sun. Learning hierarchical hidden markov models for video structure discovery. Technical Report 2002-006, ADVENT Group, Columbia Univ., December 2002.

[YH98] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13:44–49, 1998.

[YP97] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[ZH02] M. Zaffalon and M. Hutter. Robust feature selection by mutual information distributions. In A. Darwiche and N. Friedman, editors, *UAI-2002: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 577–584, 2002.