An Exploration of Feature Selection as a Tool for Optimizing Musical Genre Classification

Rebecca Fiebrink

Music Technology Area
Department of Theory
Schulich School of Music
McGill University, Montreal

Submitted June 2006

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Arts

Your file Votre référence
ISBN: 978-0-494-28557-2
Our file Notre référence
ISBN: 978-0-494-28557-2

Acknowledgments

My advisor, Professor Ichiro Fujinaga, has been instrumental both in guiding the work performed in this thesis and in overseeing the writing of this document. I gratefully acknowledge the enormous amount of his support, advice, and insight that made this work possible.

The work discussed in Chapter 7 was performed during an internship at Sun Microsystems. I am thankful for the opportunity to have worked on the Search Inside the Music Project, and for the encouragement and inspiration provided by Sun's wonderful researchers in Burlington, Massachusetts. I especially appreciate the contributions of my supervisor, Paul Lamere, who supplied invaluable ideas, code, and feedback throughout my internship. He also provided the figures and tables appearing in Chapter 7.

I gratefully acknowledge the support of the Social Sciences and Humanities Research Council, which funded portions of the work presented in this thesis. I am also very grateful for the generous financial backing of the Max Stern Fellowship in Music, which has funded me throughout my graduate study. I sincerely thank the faculty, staff, and students of the Music Technology Area for making my time at McGill a challenging, enriching, and enjoyable experience. In particular, the knowledge and camaraderie of the students of the Distributed Digital Music Archives and Libraries lab have been indispensable. I would also like to thank Professor Hank Knox, Professor Don McLean, Helene Drouin, and Mary-Beth Campbell for their assistance, not the least of which has been their contributions to ensuring my ongoing financial support.

Among the other individuals who have provided vital support and encouragement are my friends, colleagues, professors, and mentors from The Ohio State University; Tim Horton, who fueled many hours of my work through his local franchise's delicious honey crullers; Ariane Alexander, whose unremitting helpfulness extended to translating my abstract into French; and Mark, Dianne, and Stephanie Fiebrink and the rest of my family. I am indebted to my family for having always placed a high priority on education, for contributing their financial resources, and for supporting me personally; their involvement has been most crucial to my success.

Abstract

The computer classification of musical audio can form the basis for systems that allow new ways of interacting with digital music collections. Existing music classification systems suffer, however, from inaccuracy as well as poor scalability. Feature selection is a machine-learning tool that can potentially improve both accuracy and scalability of classification. Unfortunately, there is no consensus on which feature selection algorithms are most appropriate or on how to evaluate the effectiveness of feature selection. Based on relevant literature in music information retrieval (MIR) and machine learning and on empirical testing, the thesis specifies an appropriate evaluation method for feature selection, employs this method to compare existing feature selection algorithms, and evaluates an appropriate feature selection algorithm on the problem of musical genre classification. The outcomes include an increased understanding of the potential for feature selection to benefit MIR and a new technique for optimizing one type of classification-based system.

Sommaire

La classification automatique de l'audio musical peut constituer la base de systèmes pouvant ouvrir la porte à de nouvelles façons d'interagir avec les collections de musique numérique. Les systèmes actuels n'ont pas l'extensibilité et la précision adaptées. La sélection des caractéristiques est un outil d'apprentissage automatique qui a le potentiel d'améliorer l'extensibilité et la précision de la classification. Malheureusement, il n'y a pas de consensus quant aux algorithmes les plus convenables ou quant à l'évaluation de l'efficacité de la sélection des caractéristiques. La thèse détermine une méthode appropriée d'évaluation pour la sélection des caractéristiques qui est basée sur la litérature en recherche d'information musicale (RIM), l'apprentissage automatique et des tests empiriques. Cette méthode est utilisée pour comparer les algorithmes qui existent, et pour évaluer un algorithme approprié pour le problème de la classification des genres de musique. Les résultats incluent une meilleure compréhension du potentiel de la sélection des caractéristiques pour la RIM et une façon nouvelle pour optimiser un type de système de classification.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Overview

The computer classification of musical audio is an important task in music information retrieval. Classification is a standard machine-learning task that typically involves predicting an output (for example, the name of an appropriate musical genre) from an input (for example, an audio file stored on a computer). Classification can form the basis for systems that allow us to interact with music collections in new ways; some recent projects employing classification involve instrument recognition, playlist generation, and music visualization.

Unsurprisingly, music classification is a hard task. For one thing, classification uses several measurements ("features") of the audio signal to predict the output, but it is not obvious what measurements will be most relevant to complex musical concepts such as genre. For another, processing audio to obtain these measurements and running the classification itself can require vast amounts of time and computer memory. Feature selection is an existing machine-learning tool that could potentially address both of these problems, making music classification more accurate and more efficient. It works by selecting those available features that are most relevant to the classification problem; after feature selection has been applied, only the selected features need to be extracted from the audio signal, stored, and used in classification.

The application of feature selection to a particular problem is, unfortunately, complicated by the fact that there is no consensus on which of the many existing feature selection algorithms are better, faster, or otherwise most appropriate. Furthermore, existing research on feature selection does not offer clear guidelines on how to evaluate whether one selection method is better than another, or even how to evaluate whether feature selection actually offers any improvement in classification accuracy. This thesis therefore investigates how one might prudently evaluate feature selection's ability to improve classification, chooses appropriate feature selection algorithms to apply to music classification,

and evaluates whether feature selection legitimately offers any benefits to music classification.

In the following work, the classification of musical genre is used as an example music classification problem. Genre classification is a popular task, ground truth (that is, the "true" genre names as specified by the record label or distributor) is typically available, and genre classifiers can be of practical use, for example in systems for playlist generation and music collection visualization. Many of the conclusions regarding the efficacy of feature selection for genre classification also apply to other musical audio classification problems.

1.2 Organization of the thesis

Chapter 2 provides an explanation of classification as it is used in machine learning, an overview of the use of classification in music, and a discussion of feature selection as it has previously been used and evaluated. This background is used to motivate the empirical work in Chapters 4 through 6 that constitutes the bulk of this thesis, and Chapter 3 provides an overview of the software and other tools used to conduct that work.

Chapter 4 discusses the problem of choosing an appropriate methodology for evaluating whether feature selection is effective for a particular problem. It reviews the relevant literature on evaluation strategies, and it presents new work to assess the behavior of several approaches to feature selection evaluation. Chapter 5 discusses the results of employing the evaluation methodology proposed in Chapter 4 to compare several feature selection algorithms on standard machine-learning datasets, with the goal of choosing selection algorithms appropriate for music classification. Chapter 6 discusses the results of applying feature selection to genre classification, using the evaluation methodology of Chapter 4, and it also elaborates on the potential benefits and shortcomings of feature selection for classification of genre and other musical problems.

Chapter 7 augments the work on evaluating feature selection for genre classification with a discussion of a new approach to optimization that is derived from feature selection. It presents work showing that this new approach is

potentially more effective than traditional feature selection for improving a real, classification-based digital music management system. Chapter 8 concludes the thesis by reiterating the implications of this work for genre classification and related tasks in music information retrieval.

2 BACKGROUND

This chapter begins with a general discussion of classification and an overview of specific terminology relating to classification as it is used throughout the remainder of this document. An overview of music classification follows, with a focus on audio genre classification, and it is accompanied by a discussion of particular challenges inherent to the task of audio classification. Feature selection is presented as a method that addresses some of these challenges, and wrapper methods for feature selection are explained along with an overview of several common wrapper algorithms. A brief explanation of how feature selection has been evaluated in some research is provided in the context of recent critiques of evaluation methods. Finally, in light of past work in classification and feature selection, a set of tasks is put forth that must be completed in order to reasonably assess whether and how feature selection might be useful to musical genre classification.

2.1 Classification

2.1.1 What is classification?

Duda et al. (2001, 1) describe pattern recognition as "the act of taking in raw data and making an action based on the 'category' of the pattern." Classification is a necessary component of pattern recognition systems, in which properties of data are used to determine an appropriate category for the data. A simple example of a classification problem is supplied by Witten and Frank (2005, 10): one might wish to examine properties of weather such as the outlook (e.g., sunny), temperature, humidity, and wind conditions, and then make an appropriate assessment of whether it is a good day to go out and play.

A computer program for classifying the weather would take as an input each of the weather properties, or "features," and provide an output classification of "Play" or "Don't play" (Figure 1). Generally, the classifier will "learn" how to distinguish between good and bad days for going outside to play through a

training process, in which it is provided many example sets of weather features and the correct classification of "Play" or "Don't play." The goal of computer classification systems such as this is to "learn" about the relationship between the input features and the output classes, so that a classifier can be given new features (e.g., today's weather) and classify them appropriately. Integral to this goal is learning to generalize well: even if today's exact weather conditions have never been seen before, the classifier should be able to make a reasonable classification.

Computers can be used to perform classification tasks that are too complex or time-consuming for people to perform. Computer classification systems can also unearth new and interesting patterns in large quantities of data, as is performed in data mining (Witten and Frank 2005). Common application areas for pattern recognition and classification currently include speech recognition, fingerprint identification, optical character recognition, and DNA sequence identification (Duda et al. 2001, 1).

Research into applying computers to pattern recognition tasks has been performed for several decades (see Duda and Hart 1973), and work developing and analyzing classification and pattern recognition systems continues to be performed today under the umbrella of machine learning. Examples of current research include constructing classification methods for which it is possible to statistically reason about bounds on generalization performance (e.g., SVM: see Cortes and Vapnik 1995), combining multiple classifiers to work together on a single problem (e.g., AdaBoost: see Freund and Schapire 1997), or improving the quality of results obtained by existing classification methods via optimization techniques such as feature selection (discussed below).

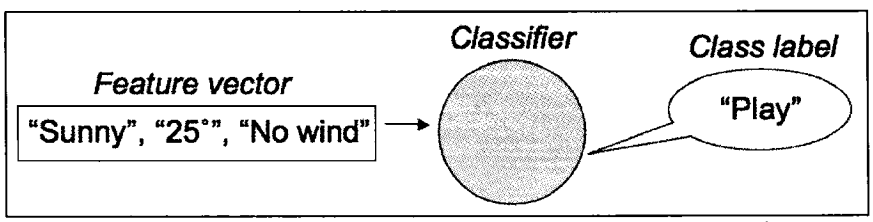

Figure 1: A classifier outputs a class label for every feature vector input.

2.1.2 Basic concepts

In order to understand existing or new work that deals with applying and evaluating classification methods, it is necessary to be familiar with some basic concepts related to machine learning and the evaluation of machine learning techniques. I present the following core concepts to aid the reader who lacks a background in machine learning and to clarify the specific definitions of these concepts as I employ them in later sections:

- *Class*: A class is a category learned by a classifier. In the weather example above, the two possible classes are the labels "Play" and "Don't play." In musical genre classification, classes might be the labels "Rock," "Pop," "Classical," etc. In classification problems discussed in this document, there are a finite number of discrete classes possible for any classification problem.
- *Instance*: Witten and Frank (2005, 45) define an instance as "an individual, independent example of the concept to be learned." In the weather example, an instance is some moment in time (e.g., yesterday afternoon) for which there is an appropriate class of "Play" or "Don't Play." In musical genre classification, where one might wish to classify an MP3 file with an appropriate genre class label, an instance would be a representation of the contents of an MP3 file.
- *Feature*: Each instance is represented by a set of features. In the weather classification problem, each weather property (outlook, temperature, etc.) is a feature. In musical genre classification, features might describe the instrumentation present, song length, pitch classes present, or lower-level measurements of the audio signal (discussed below). Features can have nominal values (e.g., the outlook can be "sunny," "cloudy," etc.) or numeric values (e.g., the temperature might be 20 degrees, -10.34 degrees, or another real value). The input to a classifier is a vector of features, $\mathbf{x}$, for each instance, where $\mathbf{x} = \langle x_1, x_2, \ldots, x_d \rangle$ for d features. Features are also referred to as "attributes" or "variables" in some literature.

- *Supervised learning*: Supervised learning is a type of machine learning wherein a learning algorithm (e.g., a classifier) learns the relationship between the input features and output classes from a set of training data. This data must be labelled with the true class values (e.g., for genre classification, the data might be labelled with the "official" genre labels from the record company). The classifier is able to classify new instances only after this training stage. All classification discussed in this thesis is supervised learning.
- *Training set*: The training set is a subset of all available labelled instances, and it consists of the feature values and true class values for each of these instances. In the training stage of supervised learning, the classifier learns to generalize about the relationship between input features and output classes using the training set instances only.
- *Testing set*: The testing set is a subset of all available labelled instances, and it is used to test how well the training stage has taught the classifier to generalize about the relationship between features and classes. Typically, the testing set consists of all the available labelled instances that are *not* in the training set.
- *Test-set accuracy*: The test-set accuracy is a widely-used measure of classifier performance. To test a classifier, the feature values for each instance from the testing set are input to the classifier, and the class label assigned to the instance by the classifier is compared to the known true class label. The test-set accuracy is defined as the percentage of classifier-assigned class labels that match the true class labels. It is important to use a testing set that is separate from the training set in order to obtain the most realistic assessment of classifier prediction performance, because the classification performance on instances used in the training stage is typically not a good indicator of the performance one can expect on new instances (Witten and Frank 2005, 144–5). Other measures of the quality of learning of a classifier exist, but classification

accuracy on new data is quite straightforward to compute and easy to understand.

- *Stratification*: Stratification refers to the practice of partitioning the available labelled data into training and testing sets in a manner that preserves class proportions. For example, if 80% of the instances in an entire dataset belong to Class A, and 20% of the instances belong to Class B, stratified training and test sets are created such that they are also comprised of 80% Class A and 20% Class B. Otherwise, if stratification is not used, instances are assigned to the testing and training sets using a random process that is blind to class values. This allows phenomena such as all instances from one class appearing in only the testing set; in this case, the classifier will likely not perform well on instances from this class, and testing accuracy will drop (Witten and Frank 2005, 149).
- *Cross-validation*: Cross validation (CV) involves splitting the dataset into several mutually exclusive segments of equal size, called folds. In *n*-fold CV, *n* folds are used. Figure 2 illustrates 3-fold CV, using the typical practice where one fold is used for the testing set and the rest are used for the training set for each iteration. The CV accuracy is the classification accuracy on the test fold averaged across all *n* iterations. Stratification may be used to preserve class proportions among the instances in each fold. Leave-one-out (LOO) CV is a special case of *n*-fold CV where *n* is equal to the number of instances in the dataset.
- *Sampling*: Sampling (or "repeated hold-out") is an alternative to CV in which repeated trials of training followed by testing are performed using independently chosen training and testing sets. The testing and training sets for a given trial are always mutually exclusive, but testing sets from different trials may have instances in common (unlike in cross-validation). The size of the test set and the number of trials are not specified, whereas they are constrained with respect to *n* for cross-validation.

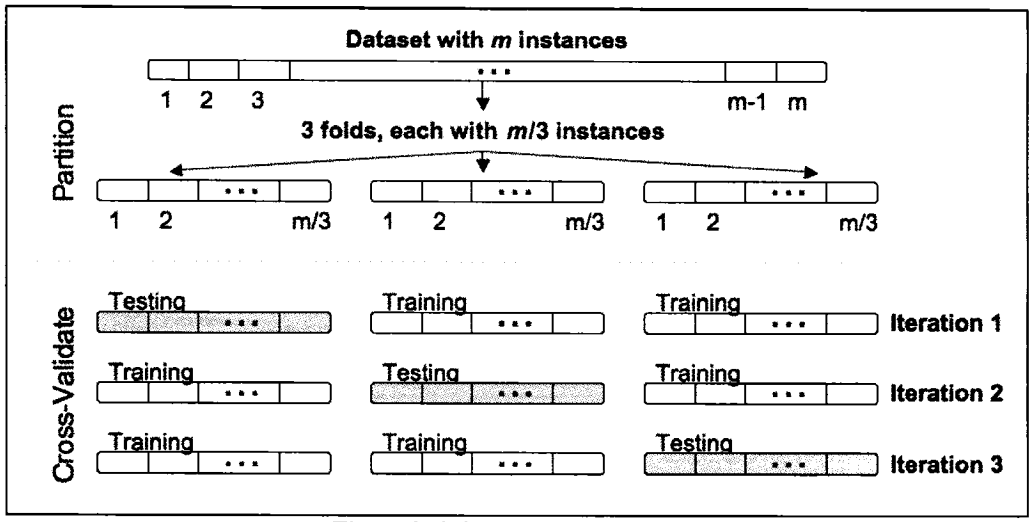

Figure 2: 3-fold cross-validation.

2.1.3 Classifiers

There are many ways to approach the problem of "learning" how to classify new instances from training examples. For this reason, many classification algorithms exist. Algorithms differ in the relationship of the numbers of instances and features to the time needed for training and testing, in the assumptions that they make regarding the distribution of the training instances, in their sensitivity to noise in the training instances, in the types of features (e.g., nominal or numeric) that they allow, and in many other ways. A few commonly used classifiers in music research include *k*-nearest-neighbor (Cover and Hart 1967), support vector machines (SVM) (Cortes and Vapnik 1995), neural networks (Cowen and Sharp 1988), and decision trees (e.g., CART: see Breiman et al. 1984). The classification algorithms relevant to this work will be explained in greater depth in later sections, but further discussion of classification algorithms lies outside the scope of this thesis. Readers interested in learning more about classifiers used in MIR are advised to consult resources by Duda et al. (2001) and Witten and Frank (2005).

The "No Free Lunch" (NFL) theorems of Wolpert and Macready lay out theoretical limitations on the ability to make claims regarding the relative performance of learning algorithms. According to the NFL theorems, there is no

classifier or learning algorithm that is superior overall to any other. That is, "[i]f the goal is to obtain good generalization performance, there are no context-independent or usage-independent reasons to favor one learning or classification method over another" (Duda et al. 2001, 454; see also Wolpert 1995). The NFL theorems have interesting implications for feature selection as well, which are discussed in Chapter 4. In the context of this section, though, it is most important to note that a researcher desiring to build a classification system for a particular problem (e.g., classifying the genre for a collection of audio files, using some set of features representing the audio files) is faced with a choice among a variety of potentially viable classification algorithms. The choice of a particular algorithm can be informed by such factors as a precedent of successful application to similar problems, the fit of the time and space requirements of the algorithm to the numbers of features and instances of the data to be classified, and the number of algorithm parameters for which reasonable settings must be found.

One classification algorithm used extensively in this thesis as well as in much other applied and theoretical work in machine learning (e.g., Aha 1992; Alpaydin 1997; Wettschereck et al. 1997; Fujinaga 1998; Fujinaga et al. 1998; Fraser and Fujinaga 1999) is *k*-nearest-neighbor, or kNN. Russell and Norvig (2003) credit the origin of nearest-neighbor models to Fix and Hodges (1951) or earlier. kNN is one of a class of "instance-based learning" methods, because it classifies instances "solely with respect to previously presented instances" and assumes that "similar instances will have similar classifications" (Aha 1992, 270). kNN is a "lazy learning" algorithm, characterized by deferring processing of inputs (i.e., training data) until a new instance must be classified, utilizing the entire collection of stored training data in order to classify the new instance, and discarding the byproducts of all processing performed for classifying the new instance after classification is done (Aha 1997).

In the training stage, a kNN classifier simply stores the feature values of each training instance as well as its class. A new instance is classified by assigning it the same class label as the majority of the *k* most similar training instances, where common values for *k* include 1, 5, and 10. The similarity

between two instances is commonly computed using the Euclidean distance between their representations in feature space (it is assumed that features have numeric values). That is, for d features, instances **n** and **m** will each be represented as feature vectors of length d, and the distance D between them will be computed as

$$\mathrm{D}(\mathbf{n},\mathbf{m}) = \left(\sum_{i=1}^{d} (n_i - m_i)^2\right)^{1/2}.$$

The k nearest neighbors to instance **n** are those instances in the training set with the lowest distance to **n**. Figure 3 below illustrates the classification of a new point using k=1 and k=3 for a simple classification problem with two features.

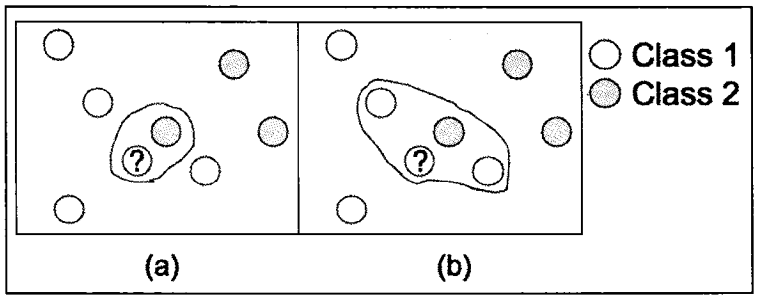

Figure 3: A kNN classifier. In (a), k=1, and the new instance marked with a '?' is labelled Class 2. In (b), k=3, and the new instance is labelled Class 1.

The implementation of kNN used in this work classifies instances exactly as above, but using linear normalization of the feature values before classification is performed. That is, each instance $\mathbf{n} = \langle n_1, n_2, \ldots, n_d \rangle$ is mapped to $\mathbf{n}_{\mathbf{new}} = \langle \mathrm{scale}(n_1), \mathrm{scale}(n_2), \ldots \mathrm{scale}(n_d) \rangle$, where $\mathrm{scale}(n_i)$ maps n_i to the range [0,1] by

$$\mathrm{scale}(n_i) = \frac{n_i - \min_i}{\max_i - \min_i},$$

where $\min_i$ is the minimum value of feature i and $\max_i$ is the maximum value of feature i over all training instances. This type of normalization is commonly used when the original features span different ranges of values (Aha 1992).

Benefits of kNN include its simplicity, the fact that there are few explicit parameters to tune in the specification of the algorithm described above (in fact, there is only the value of k to set), and the fact that it often works well in practice (Russell and Norvig 2003, 735). Another benefit of using kNN for this study is

that the effects of feature selection and feature weighting are quite easy to understand, as discussed below. Furthermore, classification using kNN scales linearly in the number of features and in the number of instances, which makes it practical to use for repeated iterations of training and testing (such as in the context of this thesis).

The design, analysis, and use of classification algorithms continues to be a lively area of study, and this thesis does not attempt to cover more than the very basic information necessary to understand the following work. Readers interested in learning more about classification algorithms should refer to popular introductory texts such as Duda et al. (2001) and Hastie et al. (2001).

2.2 Music classification

The use of classification in music information retrieval (MIR) systems predates the first MIR conference in 2001, and classification continues to pervade MIR research today. Classification has been used in systems for identifying the instruments present in an audio signal (Martin and Kim 1998; Fujinaga 1998), automatically transcribing audio signals to score representations (Poliner and Ellis 2005), discriminating between speech and musical content (Casagrande et al. 2005), and many other tasks. The classification of musical audio by genre is used as an example music classification problem for the work in this thesis, with the knowledge that many of the findings regarding the process and outcomes of applying feature selection are relevant to many other classification problems in MIR.

2.2.1 Audio genre classification

2.2.1.1 Previous work

The content-based classification of musical audio by genre is one of the most popular applications of classification in MIR. Here, "content-based" refers to classification on the basis of the audio content alone, ignoring non-audio data such as annotated artist and album information. A great variety of approaches to classification, including many classifiers and feature sets, have been applied to the

genre classification problem, as the following section illustrates. Readers interested in learning more about these classifiers should consult any machine learning text (e.g., Duda et al. 2001). Section 2.2.2.2 provides further background on the choice of features for musical genre classification, but due to the wide variety of features employed, the reader is advised to consult the following works directly for more detailed information.

The first content-based genre classification system was designed by Tzanetakis et al. (2001). The authors calculated features for audio from radio, compact disks, and the Internet. These features were used to train a classifier on broad musical genre categories ("Classical," "Country," etc.), specific classical music categories ("Orchestra," "Piano," etc.), and also categories of music versus speech. The features extracted from the audio included "musical surface features" related to texture, timbre, and instrumentation, "rhythm features" representing the rhythmic structure, and Mel-frequency cepstral coefficients (MFCCs), a feature popular in speech classification (Hunt et al. 1980). This work used a simple Bayesian classifier to approximate each genre class as a multidimensional Gaussian distribution in the feature space (see Duda et al. 2001). Tzanetakis and Cook (2002) later expanded on this study to compare results of different classifiers (including kNN) and to investigate the effects of adding a pitch-based set of features to those mentioned above.

Since this work, there has been much other research on content-based genre classification, the goal of which is typically to increase classification accuracy by varying the classifier, the type of features used, and/or the entity being classified (e.g., using instances representing individual audio frames, not whole songs, and combining frame-level classifications to produce a single label per song). Xu et al. (2003) used a multi-layer approach to classification into four genres, using SVM classifiers. Features were quite similar to Tzanetakis' "musical surface features" and included MFCCs and linear predictive coding (LPC) coefficients. Li and Tzanetakis (2003) also investigated the use of SVM, as well as linear discriminant analysis (LDA; see Duda et al. 2001). This work also

further investigated the relative usefulness of the feature sets used in Tzanetakis' earlier work (Tzanetakis et al. 2001; Tzanetakis and Cook 2002).

Li et al. (2003) explored the use of a new, wavelet-based feature called Daubechies Wavelet Coefficient Histograms (DWCHs), while also evaluating SVM, kNN, and Gaussian Mixture Model (GMM) classifiers. West and Cox (2004) demonstrated an approach for classifying spectral features using CART and LDA classifiers, where each song is represented as a "bag" of frames that are each classified individually. West and Cox's later work (2005) modified this model to classify not individual frames, but musical "events" corresponding roughly to note onsets. Bergstra et al. (forthcoming) also use a mid-level representation (that is, between individual frames and entire songs) for classification, in a system employing AdaBoost and a large variety of features.

In the last two years, several research groups (including many of those mentioned in the previous paragraphs) submitted their genre classification systems for large-scale evaluation on common datasets, so that meaningful comparisons could be drawn among the wide variety of systems proposed. These common datasets were also quite larger (on the order of several hundred or thousands of songs) than those used in most of the studies above, so these tasks better simulated real-world performance of the systems. In 2004, an Audio Description Contest was held in conjunction with the International Conference on Music Information Retrieval (ISMIR), and researchers from five organizations competed in the genre classification task. In 2005, the ISMIR 2004 contest was expanded into ten contest categories under the name "MIREX" (Music Information Retrieval Evaluation eXchange) (Downie et al. 2005). The genre classification category was the most popular, receiving thirteen submissions.

2.2.1.2 Performance of audio genre classification systems

The performance of the systems in independent, published evaluations and in the genre classification contests suggests that machine learning of musical genre is feasible, but it is not a trivial problem. Tzanetakis' first system (2001) obtained 62% classification accuracy, evaluated using 10-fold CV on fifty music

files from ten genres. Published work since 2001 typically demonstrates classification rates higher than this, though not necessarily on the same dataset (and often using quite different numbers and types of genre classes). The highest recently reported accuracy rates tend to be around 80%; for example, Bergstra et al. (forthcoming) demonstrate 83% classification accuracy on Tzanetakis' dataset, and West and Lamere (forthcoming) demonstrate 83% accuracy on their own 6-genre dataset. In MIREX 2005 (Downie), the best algorithms' raw classification accuracy scores were 69.5% on the Magnatune dataset (Magnatune 2006) and 86.9% on the USPOP dataset (Ellis), both obtained by Bergstra et al. (2005) using a multi-resolution AdaBoost system similar to that described by Bergstra and others (forthcoming).

These results appear especially good when compared to human ability to classify genre. While there are no human studies using data that allows direct comparison to the above studies or the MIREX contest results, one study showed that humans were able to correctly identify the genre for 70% of three-second samples selected from a group of ten genres (whose labels were known to the subjects) (Perrot and Gjerdigen 1999).

The fact that automatic genre classification systems show better accuracy than human listeners, based on this study, does not imply that the classification abilities of current systems are sufficient. For one thing, humans employed in the manual annotation of genre, for example for use in systems such as Allmusic Guide (All Media Guide), are likely to be well-trained and achieve well over 70% classification accuracy. This type of labeling may be a more significant benchmark against which to compare automatic systems.

In addition to accuracy, the scalability performance of genre classification systems is a pertinent factor in their usefulness. Scalability refers to the impact on time, space, and accuracy of a system when the number of songs and/or features increases. Genre classification systems must become more scalable if they are to be used to classify real-world datasets. For example, the MIREX 2005 genre contest datasets contained around 1000 songs each, and the winning algorithm ran for 6.5 hours to classify the songs in one dataset. In contrast, the iTunes database

currently has over two million songs, and a 60GB iPod alone can hold 15,000 songs (Apple Computer Inc.). It is an open problem how to perform accurate genre classification on collections of this size while keeping time and space requirements feasible.

2.2.1.3 Motivations for work in audio genre classification

Audio genre classification is certainly an interesting toy problem for assessing the robustness of a given machine learning algorithm to a novel application domain, or for demonstrating that features such as MFCCs are useful for describing musical data and might be successfully employed in a variety of MIR applications. However, there are other reasons for continuing to pursue better means of automatically classifying audio signals based on musical genre. McKay and Fujinaga (2006) argue that genre is a compelling means of categorizing music, and end users of music distribution and recommendation systems are likely to employ genre labels in browsing and searching for new music. Music distribution and recommendation systems that use classifiers to accurately and consistently label audio files by genre could potentially be much more efficient than those requiring human annotators for this task. Furthermore, McKay and Fujinaga argue that genre classification systems could be used to gain musicological insights; for example, tests using classifiers could show that particular musical features are sufficient to discriminate among certain genres.

Another motivation behind work in genre classification is the close relationship between musical genre and the concept of musical similarity. Several projects in MIR deal with the task of automatically generating playlists of songs that sound the most similar to a query song (e.g., West and Lamere forthcoming; Logan 2002). The concept of musical similarity can also be used to create two- or three-dimensional visualizations of music collections, in which similar songs appear close to each other (e.g., West and Lamere forthcoming; Mörchen et al. 2005). Much work in musical similarity systems such as these incorporates the assumption that pieces of music that share a genre are more likely to be perceptually similar than pieces of music from different genres. This assumption

is reflected in evaluation practices for musical similarity systems; for example, one might evaluate a playlist generator based on how many songs in the generated playlists are of the same genre as the query songs (e.g., Logan 2002). In this approach, easy-to-obtain genre labels are used as a substitute for music similarity ground truth, because ground truth similarity data would require extensive user testing. Additionally, the assumption that genre is related to similarity can be used in building similarity systems: one might assume that a classifier that performs well at classifying genre has learned something relevant to music similarity and could be therefore be integrated into similarity calculations. West and Lamere (forthcoming) take this approach: they train a genre classifier using genre metadata, classify songs in a collection using this classifier, and use the outputs of classification as inputs to an inter-song similarity computation. (This system is described in greater detail in Chapter 7).

2.2.2 Challenges in music classification

Many characteristics of musical classification set it apart from classification of other media such as images and text. These characteristics present some unique challenges that must be addressed for music classification to be successful and feasible. The scarcity of musical data and the wide array of features that are available to use for music classification are two challenges that are particularly relevant to this thesis.

2.2.2.1 Scarcity of data

A significant challenge facing researchers in MIR is the difficulty of obtaining musical audio collections for training and evaluating classification systems. Building one's own collection by hand can be quite costly, both in terms of time and money spent legitimately acquiring CDs or MP3s. However, the large size of music collections makes it infeasible to transfer entire collections online, and copyright protections impair the ability of researchers to share their collections with each other legally through any means. Using existing collections such as Magnatune (2006), which allows free use of its music for academic purposes, circumvents some of the above problems, while also allowing

researchers to compare their results on the same data. However, Magnatune only has a few thousand songs, and it contains a peculiar variety of music styles that is not representative of popular music in general. Furthermore, machine learning research suffers from overuse of the same datasets for a variety of reasons (Salzberg 1997). However, because genre classification accuracy has already been computed on the Magnatune data for many state-of-the-art classification systems in the context of MIREX 2005, and because of the relative ease of obtaining the Magnatune collection, the Magnatune data was used in the genre classification experiments in this thesis.

2.2.2.2 Choice of features

A wide array of features has been used successfully in genre classification and other audio classification tasks. These features tend to be derived from simple time- or frequency-domain measurements of the audio signal and are often borrowed from audio features that perform well in speech classification, though features designed particularly for musical classification and analysis have also sometimes been used. Researchers building systems for musical genre classification must choose among these features, and they must also specify parameters regarding how these features will be computed (e.g., the size of analysis frames or the number of MFCCs to extract).

A thorough overview of many features used in musical genre classification appears in Bergstra et al. (forthcoming). Other features are outlined in Aucouturier and Pachet (2003). Examples of commonly-used features include spectral measurements such as fast-Fourier transform (FFT) coefficients, spectral centroid, spectral flux, and spectral rolloff (these last three features are used in Tzanetakis and Cook 2002 but have been employed in numerous other studies). Mel-frequency cepstral coefficients (MFCCs) are a feature originally used for speech classification (Hunt et al. 1980) that has been used in many of the above studies in genre classification, as well as in other types of audio classification. Time-domain measurements such as zero-crossing rate have also been used (e.g., Tzanetakis and Cook 2002).

Some studies have used features that attempt to describe properties of the audio signal that are particularly relevant for music. For example, Tzanetakis et al. (2001) and Tzanetakis and Cook (2002) construct beat histogram features that capture information about beat strength and the relationships between the periodicities present in music. Tzanetakis and Cook (2002) also employ pitch histograms, a feature that captures information about the pitch content of musical audio. West and Cox (2004, 2005) use an octave-scale spectral contrast feature that captures similar information as MFCCs but that is tailored for musical audio, created by Jiang et al. (2002).

It is notable that no existing audio genre classification system employs many high-level features explicitly related to instrumentation, dynamics, or metric, melodic, or harmonic structure, and most employ none at all. McKay (2004) demonstrates that a genre classification system using such features can achieve very high classification rates (86% correct using nine genres). However, McKay's work deals with classification of music in symbolic format (i.e., MIDI), where such features are explicitly available or easily computed. While his work suggests that these types of features could be very beneficial to improving performance of audio genre classification systems, there do not exist reliable and efficient-to-compute means of obtaining these features from an audio signal. Research continues to be performed on systems for transcription (e.g., Poliner and Ellis 2005), meter detection (e.g., Eck and Casagrande 2005), and other types of analysis that could one day provide more meaningful features for genre classification systems, but the current state-of-the-art does not offer many viable tools for this purpose.

Computation for all commonly-used features involves first making decisions regarding parameters such as the size of the analysis frames, how many frames will be computed for each song, and where in the song frames will be computed. For multi-dimensional features, it must be decided how many of the available dimensions (e.g., how many MFCCs) will be extracted and used in classification. Additionally, a non-trivial problem is how to construct feature vectors from the frame-level measurements. One might compute the average and

standard deviation for each value (e.g., each MFCC) over all frames in a piece, thereby representing a piece by a feature vector composed of the global means and standard deviations. Or, one might model a piece as a "bag" of frames, each with an associated feature value (West and Cox 2004). Each frame of the training data could be used a training instance, and the class labels assigned to each frame of a testing song could then be combined to assign an overall label to that song.

The challenge presented by the choice of features is ubiquitous: any work in music classification must begin by choosing a set of features and specifying how they will be computed from the audio files. The effect of the features used on classification accuracy is quite large; Li et al. (2003) found that the features employed in genre classification had an even greater influence over classification accuracy than did the choice of classifier. However, neither musical intuition nor the existing literature prescribes an unequivocally "best" set of features to use for genre classification. It is hard to say, for example, whether a feature such as zero-crossing rate will be at all related to musical genre. However, it is not advisable to attempt to extract and use all features that might be related to the classification task. For one thing, this can result in an explosion of time and space necessary for extracting and storing features. More crucially, as discussed below, the inclusion of redundant, irrelevant, and simply more numerous features in a classification system is likely to degrade the quality of the classification results.

2.3 Feature selection

2.3.1 Overview

Feature selection denotes a family of techniques that aim to improve the accuracy and practicality of classification systems. Feature selection algorithms select a subset of all available features that can be used in classification, such that classification using this subset can outperform classification using the entire available feature set in terms of classification accuracy and/or efficiency. Feature selection is only concerned with choosing a set of features from those available, not with constructing new features (Kohavi and John 1997).

2.3.1.1 Goals of feature selection

The inclusion of irrelevant features has been shown to hurt classifier accuracy for a number of classification algorithms. For other algorithms, the presence of relevant but highly-correlated features can decrease accuracy (Kohavi and John 1997). This implies that the simple strategy of constructing a feature subset of each of the individually most predictive features is not necessarily appropriate. kNN is one classifier that is negatively affected by redundant, irrelevant, interacting, and noisy features (Wettschereck et al. 1997). Furthermore, the practice of using a large number of features for classification, regardless of their relevance or redundancy, presents problems in itself. The training and testing time required by classification algorithms may scale poorly as the number of features increases. The time to compute features and the storage space needed to store a dataset predictably increase with the number of features. More importantly, for any classification algorithm to be able to generalize well from the training data, the number of training instances must increase exponentially as the number of features grows linearly (Duda et al. 2001, 169–70). This is called the "curse of dimensionality," and it has several potentially damaging ramifications. First of all, it limits the number of features that can be practically employed in domains where there is little training data available, such as often occurs in music classification. Second, adding enough training data to counterbalance the addition of new features may have a very detrimental impact on the training and testing time of a classifier.

Selecting a small, relevant, non-redundant set of features from all available features therefore has the potential to improve the classification accuracy, training and testing time, and storage space needed for a dataset. Selecting fewer features also significantly relaxes the demand for a large amount of training data. One is therefore presented with the problem of how to select among a potentially very large array of features, when it is unclear even to a domain expert which features are likely to be best for a given classification problem. Automatic feature selection algorithms present a tool for solving this problem.

2.3.1.2 Wrapper methods

The family of feature selection methods examined in this thesis falls into the category of "wrapper" methods for feature subset selection, which are named thus because the feature subset selection algorithm "exists as a wrapper around the induction algorithm" (John et al. 1994). Wrapper methods are essentially search algorithms that search for a good subset of features, using an induction algorithm (i.e., a classifier) as a "black box" for evaluating the quality of candidate feature subsets (Kohavi and John 1997). This measure of quality is typically computed as the *n*-fold CV accuracy of the classifier on the data, using only the specified subset of features and ignoring all others in both the training and testing stages (John et al. 1994). Wrapper methods thus take into account the performance of a given classifier on a dataset, and their goal is to find an optimal feature subset for classifying the data *with that classifier*.

2.3.1.3 Related methods

John et al. (1994) distinguish wrapper feature selection methods from those methods that select a feature subset without using classification to evaluate subsets. These methods, called "filter" methods, "attempt to assess the merits of the features from the data, ignoring the induction algorithm" (Kohavi and John 1997). "Relief" is an example of a filter method; it works by determining the relevance of each available feature to the target concept (i.e., the class values to be learned). It therefore selects all features whose variations are highly predictive of the class label (Kohavi and John 1997; see Kira and Rendell 1992 for a more detailed discussion of Relief). Other examples of filter methods include FOCUS (Almuallim and Dietterich 1991) and decision-tree-based selection (Cardie 1993).

Filter methods offer the advantage over wrapper methods that there is no need to train and test a classifier hundreds or thousands of times to evaluate each candidate subset. However, certain filter methods still scale poorly (e.g., FOCUS requires an exhaustive search through all possible feature subsets). Another possible benefit of filter methods is that they provide a "generic" selection of features suitable for any classifier (Guyon and Elisseeff 2003), which might be

desirable if one is interested in performing feature selection as a preprocessing step before comparing the performance of several classifiers on a problem. However, Kohavi and John (1997) have shown that classifier-independent reasons for favoring certain features over others (e.g., the relevance criterion used by Relief) do not necessarily lead to selecting subsets that perform optimally in practice. If one knows in advance that a particular classifier will be used for the problem, wrapper methods can yield superior performance by explicitly searching for a subset that performs optimally with the given classifier (Kohavi and John 1997).

Wrapper and filter methods for feature selection fall under the umbrella of a larger group of dimensionality reduction techniques. Feature selection reduces the dimensionality of the feature space by removing some of the available features. Other techniques reduce dimensionality by remapping the data into a lower-dimensional feature space. Principal components analysis (PCA) is one such technique, which works by choosing a new coordinate system for the feature space. The first axis is placed in the direction of greatest variance, the second axis in the orthogonal direction of next greatest variance, and so on. Dimensionality reduction can be achieved by keeping only the first n dimensions (the n dimensions that capture the most variance) (Witten and Frank 2005). Dimensionality reduction methods can potentially offer the same benefits as feature selection, in that the features ultimately used in classification are "better" for the classification problem (e.g., less redundant), the classifier uses fewer features (and therefore runs faster and needs less training data to escape the curse of dimensionality), and the storage space needed for the data is smaller. Methods such as PCA can involve shorter computation time than feature selection, and their results are classifier-independent. However, because PCA remaps the data using a transformation on all of the original features, the time for feature extraction and the space needed to store the original features are not reduced. Additionally, PCA cannot produce results tailored to optimize classification for a given classifier.

Another method closely related to feature selection is feature weighting (Witten and Frank 2005, 237–8). In feature weighting, each of the available features is assigned a weight proportionate to its "importance" to the classification. In a kNN classifier, this results in each dimension of the feature space being weighted by a real value in the distance computation. That is, each feature i receives a weight w_i, and the distance computation between feature vectors for instances **m** and **n** becomes

$$\mathrm{D}(\mathbf{n},\mathbf{m}) = \left(\sum_{i=1}^{d} w_i(n_i - m_i)^2\right)^{1/2}$$

(Wettschereck et al. 1997). Figure 4 shows the effect of feature weighting on a kNN classifier. Wrapper feature weighting can be implemented similarly to wrapper feature selection: each set of feature weights is evaluated by training and testing a classifier using those weights, and the performance of weight sets guides the search to new sets of weights. Wrapper feature selection can be viewed as a special case of feature weighting, in which the weights are constrained to the values 0 and 1.

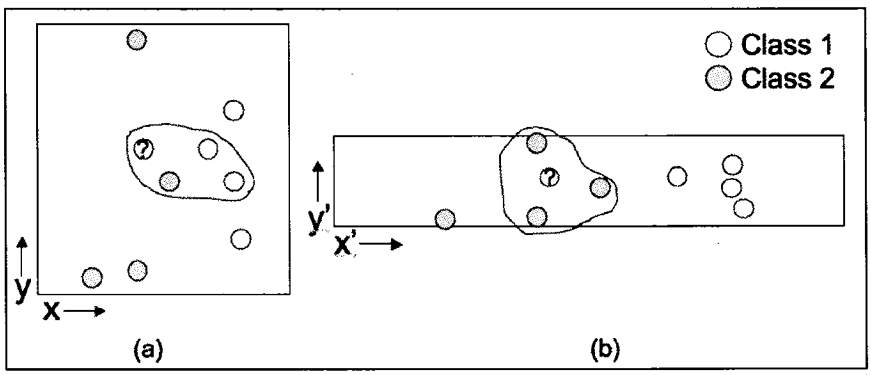

Figure 4: Effect of feature weighting on kNN. In (a), if k=3, the unknown instance will be labelled Class 1. In (b), where feature x has been weighted by $w_x > 1$ and feature y has been weighted by $w_y < 1$, the instance will be classified with Class 2.

2.3.2 Existing wrapper methods

When the number of available features is few, it may be feasible to search exhaustively through the set of all possible feature subsets, evaluate each subset using a classifier, and pick the subset that results in the best classification

accuracy. However, the number of possible subsets is 2^d for d available features, so exhaustive search quickly becomes impossible for many available features. One alternative is to sample s subsets randomly from the set of all possible feature subsets and choose the best of all these, where s is chosen such that a variety of subsets are evaluated while ensuring that the computation involved is feasible given the available time and resources. (This approach is referred to hereafter as "Monte Carlo" selection.)

However, in practice, wrapper feature selection is usually accomplished by using classification accuracy on given subsets to direct the search to new subsets. Different feature selection algorithms differ primarily in the algorithm used to perform this iterative search through the space of all subsets. Algorithms that are commonly used for wrapper feature selection include forward selection, backward selection, genetic algorithms, and random mutation hill climbing. This section briefly describes each of these methods.

2.3.2.1 Forward selection

Forward selection (or "sequential forward selection") is a heuristic, greedy (i.e., approximating a globally optimal decision by a series of locally optimal decisions) search method commonly used for feature selection (John et al. 1994; Reunanen 2003). In the first iteration of forward selection, there are d unique candidate subsets, each consisting of one of the d available features. In the second iteration of forward selection, each of the d–1 candidate subsets consists of the best performing feature from the first iteration, plus one of the remaining features. The algorithm continues such that each iteration evaluates the best subset from the last iteration with each of the yet-unselected features added to it individually. The last iteration (the dth iteration) consists of one "subset" containing all of the available features. In each iteration, the algorithm remembers the subset with the best classification accuracy, and at termination, the chosen subset is the one with the highest accuracy. The chosen subset may therefore have anywhere from 1 to d features. Forward selection always visits $d(d+1)/2$ feature subsets, so its running time is fixed for a given dataset.

Several variants of forward selection have also been proposed, such as "sequential forward floating selection" (Pudil et al. 1994). Some literature (e.g., Pudil et al. 1994; Kudo and Sklansky 2000) has suggested that sequential forward floating selection is more effective than forward selection, but Reunanen (2003) refutes this claim based on criticisms of the evaluation method used in this literature (see Sections 2.3.3, 4.3.1, and 5.1.1).

2.3.2.2 Backward elimination

Backward elimination is a greedy algorithm quite similar to forward selection (John et al. 1994). In the first iteration, the set of all available features is evaluated. In the second iteration, d unique candidate subsets are evaluated, each consisting of all *but* one of the d available features. In the third iteration, each of the d–1 candidate subsets consists of the best performing subset from the second iteration, *minus* one of the remaining features. Each subsequent iteration removes one feature, until the dth (last) iteration consists of one subset, which contains a single feature. As in forward selection, the feature subset chosen by the algorithm in the end is that which resulted in the highest classification accuracy observed over the entire search.

Backward elimination has the same runtime performance as forward selection. There is no clear agreement on whether forward selection or backward elimination is generally more appropriate (Guyon and Elisseeff 2003).

2.3.2.3 Random mutation hill climbing

Random mutation hill climbing (RMHC) is a stochastic hill-climbing algorithm outlined by Forrest and Mitchell (1993) and used for feature selection by Skalak (1994). RMHC begins by randomly generating a feature subset, where each of the d available features is randomly chosen to be included or excluded from the subset. This subset is then evaluated, and it is called the "Best Evaluated" subset. In the next iteration, one feature is chosen at random from the d available features. If this feature is included in the "best evaluated" subset, a new subset is created that is identical to "Best Evaluated" except that this feature is excluded. Similarly, if this chosen feature is excluded from "Best Evaluated,"

the new subset is created to be identical to "Best Evaluated" except that this feature is included. The new subset is evaluated, and if it performs better than "Best Evaluated," then this new subset becomes "Best Evaluated." The algorithm continues in this manner, mutating the "Best Evaluated" subset by one randomly-selected feature in each iteration, until a predetermined number of iterations have completed.

RMHC always visits n subsets over n iterations, and n can be set to be any value. This means that the time needed to run RMHC can be adjusted for a given dataset and resource pool. However, one drawback of RMHC is that, unlike forward selection and backward elimination, multiple subsets cannot be evaluated in parallel. Each subset must be evaluated before it is known what the next subset to be evaluated will be. In forward selection and backward elimination, on the other hand, each iteration consists of several independent subsets, each of which can be evaluated in parallel given the appropriate hardware. As a result, given the capability to perform parallel computations, RMHC is able to evaluate fewer subsets in a given amount of time.

2.3.2.4 Genetic algorithms

Genetic algorithms (GAs) are another stochastic method used for feature selection. GAs mimic evolutionary processes to arrive at solutions to optimization problems for which maxima are hard to find deterministically. GAs have been employed in a variety of problems since their conception in 1975 (Holland).

Each potential problem solution is represented in a GA as a chromosome, typically consisting of a vector of binary or real values, called genes. A GA begins by randomly initializing a population of many chromosomes. Each of these chromosomes is evaluated to assess the quality of the solution it represents, and this quality is stored as the fitness value of the chromosome. The next stage of the GA involves evolving the population by combining parent chromosomes to produce children chromosomes, in a process called "crossover" (Figure 5). Typically, the evolution process favors chromosomes with higher fitness values, so that they are more likely to be chosen to produce offspring via crossover.

Generally, crossover between two chromosomes consists of randomly choosing a locus, or index into the parent vectors; the children are produced by swapping the parent chromosome portions on either side of this locus. During the crossover process, there is a predetermined probability of mutation occurring, wherein one or more genes of the child chromosome are changed to random values. After a new generation of child chromosomes has been created, these children become the new population, and the fitness of each child is evaluated. The process of evolution continues until some convergence criterion is met, such as the fitness of the fittest individual not improving for several consecutive generations. Figure 6 illustrates a typical GA.

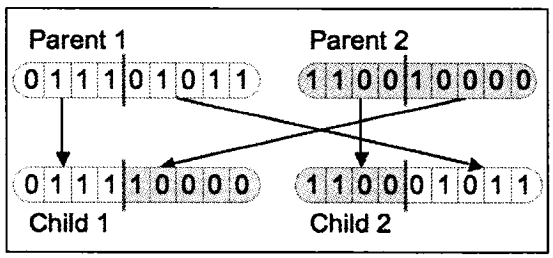

Figure 5: The crossover process, using chromosomes of 9 binary values and a locus between the fourth and fifth genes.

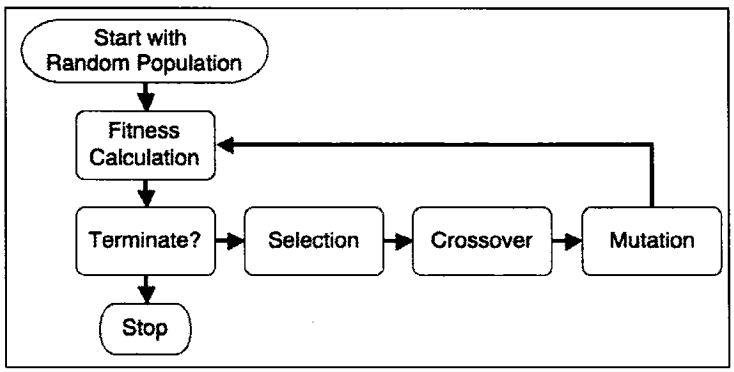

Figure 6: A typical genetic algorithm.

In GA feature selection, each chromosome represents a feature subset (Siedlecki and Sklansky 1989). A feature set is represented as a chromosome of d binary-valued genes, each denoting the inclusion or exclusion of a feature from the subset. The fitness of a chromosome is the classification accuracy obtained by training and testing a classifier using the indicated feature subset, just as in other

wrapper feature selection methods. By modifying this approach slightly and allowing chromosomes to be vectors of real values, GAs can be used for feature weighting. In this case, each of the d values indicates the weight of the associated feature in the feature set. Punch et al. (1993) demonstrated that using GAs for selection and then applying GAs for weighting yielded better results than performing selection alone.

GAs differ strikingly from the above search methods in that there are many parameters that must be set. There exist several established methods for using the fitness of individual chromosomes in a population to influence which chromosomes will produce children via crossover, and one of these methods must be chosen. The mutation rate and initial population size must be set. Each of these values has implications for the running time and quality of final solution, but there is no "correct" setting of these parameters, and there is no exact way to reason about tradeoffs between runtime and solution quality.

GAs are very easy to implement in parallel. Portions of each generation can undergo fitness evaluation on separate processors; this is called "master-slave parallelism" by Cantú-Paz (2000) and "micrograined parallelism" by Punch et al. (1993). Additionally, there exist many interesting ways of expanding on the evolutionary metaphor by evolving isolated or semi-isolated subpopulations and managing migration among these "islands" (Cantú-Paz 2000); the potential for such techniques to improve upon performance of simpler GAs for feature selection has been as of yet unexplored.

2.3.3 Evaluating wrapper feature selection methods

There exist many papers that purport to demonstrate that wrapper methods for feature selection are likely to improve accuracy for a particular problem or a variety of problems (e.g., Siedlecki and Sklansky 1989; Skalak 1994; Kohavi and John 1997). This seems to be a reasonable claim; after all, many classifiers are weakened by the presence of irrelevant or redundant features. Indeed, all these papers demonstrate that the feature selection method(s) employed found subsets of the available features whose classification accuracy (measured using test-set or

cross-validation accuracy) outperformed classification using all features, using the same evaluation measure. Similar findings have been reported for feature weighting (Punch et al. 1993). Additionally, at least one study has shown that certain methods are able to find the optimally- or near-optimally-performing feature subsets in much less time than is required for exhaustive search (Siedlecki and Sklansky 1989).

There also exist many papers that compare a small or large set of feature selection algorithms against each other, to make claims that one method or another is more appropriate for a particular problem or set of problems (e.g., Kudo and Sklansky 2000; Pudil et al. 1994). In some of this work, the efficacy of one algorithm ("A") is compared to another algorithm ("B") by comparing the cross-validation accuracy of the feature subset selected by "A" with that of the feature subset selected by "B." It seems reasonable to make at least casual (non-statistically-rigorous) claims that "A" is somehow "better" than "B" if it repeatedly and consistently chooses feature subsets that have higher cross-validation accuracy scores. The work by Pudil et al. (1994), for example, uses this method to "show" that sequential forward floating selection is favorable to forward selection.

However, recent work calls this approach to evaluating feature selection—and by extension the many studies employing this evaluation approach—into question. Reunanen in particular (2003, 2004) has noted fundamental problems with certain work using the above methodologies. He observes that, when comparing algorithms, it is insufficient to point to the fact that one algorithm finds feature subsets with higher cross-validation accuracy than another algorithm, even if it finds them consistently. The big problem is that such algorithms may not consistently find subsets that result in better performance on *unseen* data; in fact, the performance on unseen data could be worse. Similarly, Reunanen (2004) later observes that, when evaluating whether feature selection is an effective tool, it is insufficient to point to the fact that an algorithm finds feature subsets with higher cross-validation accuracy than is obtained using all features, even if it finds them

consistently. Again, such subsets may lead to decreased performance on unseen data, relative to using all features.

The problem of evaluating approaches to assessing feature selection is a tricky one. There is no canonical means of "properly" evaluating feature selection, though several researchers have presented their own approaches in varying degrees of detail. Relevant work on this issue, its implications, and further work toward defining an approach to evaluating feature selection are discussed in greater depth in Chapter 4.

The above problems imply that the findings of published work employing problematic evaluation methodology, or work where not enough details regarding evaluation methodology are supplied to deduce whether it falls into the traps described by Reunanen, cannot necessarily be trusted. The current body of literature on feature selection does not present clear answers to what feature selection methods might be best to use for a problem such as music classification, how to evaluate feature selection's efficacy for a problem, or whether it is even likely to be beneficial.

2.3.4 Use of feature selection in music research

Feature selection has not been widely used in music classification systems. Many researchers have recognized the limited usefulness of intuition in reasoning about which features are most useful for a particular classification problem, and several of the genre classification studies above have addressed the question of feature quality by comparing classification accuracy using several manually-selected sets of features. For example, Tzanetakis and Cook (2002) trained and tested the classifier using only pitch histogram features, only beat histogram features, only timbral-textural features, only MFCCs, and all features together. Their results allow one to draw conclusions regarding the relative importance of these feature groups (e.g., MFCCs alone allow much better performance than the pitch histogram alone) and demonstrate that combining all features generally results in classification accuracy that is as good as or better than that obtained using any single group. Studies such as this, however, are not adequate for

determining which individual features are most useful for classification, or for finding a feature subset that allows optimal classification accuracy.

Essid et al. (2004) employed a filter feature selection method, "Inertia Ratio Maximization using Feature Space Projection" (IRMFSP) to select features for a musical instrument identification task. They compared several methods of applying IRMFSP for the purposes of pair-wise instrument classification, which allowed different features to be selected for use in discriminating between each possible pair of instruments. That is, one classifier could use one feature set to choose between "Piano" and "Bassoon" and another classifier could use another feature set to choose between "Bassoon" and "Oboe." Classification accuracy was computed for each instrument, and accuracy varied between 55% and 95% for most instruments.

Fujinaga (1998) employed GAs for feature selection and weighting in an instrument recognition task. GA selection found an optimal subset of just seven of the original 352 features. When GA weighting was performed using these seven features, the system was able to classify instruments with accuracy similar to human listeners (50%).

Fiebrink et al. (2005) investigated the application of GAs for feature selection for a variety of timbre recognition problems, including beat-box identification and snare-hit identification. They found that GAs were indeed able to find feature subsets that resulted in higher cross-validation accuracy than using all available features.

However, both Fujinaga (1998) and Fiebrink et al. (2005) used the evaluation methodology of Kudo and Sklansky (2000), which has been criticized by Reunanen (2004) as likely to overestimate the benefits of feature selection. To date, there has been no research in MIR that carefully investigates whether feature selection can address some of the challenges of music classification, using an evaluation methodology that is not susceptible to such criticisms.

2.4 Necessary work

Many features might be used as inputs to music classification systems. However, the space needed to store features, and the time needed to extract features, train a classifier, and use a classifier to classify new instances all grow as the number of features used increases. Additionally, classification accuracy can be impaired by the presence of noisy, irrelevant, or redundant features. Furthermore, the number of training instances needed to train a classifier capable of generalizing grows exponentially with the number of features present. However, it is not at all intuitive which low-level features are most suitable for classification problems such as musical genre. Therefore, there appears to be great potential for feature selection to improve genre classification accuracy.

Recent findings suggest, however, that it is not clear that existing literature promoting some feature selection methods over others can be trusted, due to a prevalence of poor evaluation methodology. Nor is it clear that one can believe that feature selection is as likely to improve classification accuracy in practice as has been suggested, for the same reason. Unfortunately, while key problems in previously employed methodologies for assessing feature selection's efficacy have been exposed, the literature does not offer a comprehensive, well-reasoned alternative approach to evaluation.

In order to most effectively assess whether feature selection is a useful tool for musical classification, it is first necessary to specify a good method for evaluating feature selection. Then, it is desirable to employ this method to compare a set of feature selection algorithms in a reasonable way, and choose one or more methods to apply to a music classification problem, such as musical genre. Only then is it possible to assess the efficacy of these methods on musical genre and come to meaningful conclusions regarding how and whether feature selection might be used to improve music classification. The next several chapters of this thesis describe the work that was performed to accomplish each of these steps.

3 TOOLS

3.1 Introduction

The work in Chapters 4 through 6 involves empirical testing that requires a substantial software infrastructure, involving components for feature extraction, classification, distributed computing, and feature selection. Some of the software components used are existing (mostly open-source) third-party software, and others (including the global infrastructure facilitating collaboration between different components) were written specifically for this project. This chapter presents the software tools used in this thesis, for the purposes of clarifying how the tests in the next three chapters were performed and of delimiting what is original work and what is the work of others.

The tests in this thesis also involved the use of several datasets from a public repository, and this chapter concludes with a mention of the additional work performed to facilitate the selection of appropriate datasets from the repository.

3.2 Feature extraction

jAudio (McEnnis et al. 2005) was used to extract features from the audio files. jAudio is an open-source feature extractor written in Java that was designed specifically for use in MIR. It includes a collection of many standard features that have been previously used by MIR researchers for audio classification and other tasks. Only standard, built-in features were used, without modification, in the classification experiments performed in this thesis (see Chapter 6 for a description of these features).

3.3 Classification

Weka (Witten and Frank 2005) is an open-source, Java workbench that includes a collection of many standard machine learning and data mining algorithms. The kNN classifier that is used as the primary classifier in this thesis

is implemented with the Weka IBk classifier, and the C4.5 decision tree classifier that is used occasionally is implemented with the Weka J48 classifier. Default implementations and parameter settings for both classifiers are used throughout this thesis, unless otherwise indicated.

3.4 Distributed computing

Because of the large number of classification evaluations required for many of the feature selection tests used in this thesis, the Data-to-Knowledge (D2K) Toolkit (Automated Learning Group) was used to distribute these computations over a number of machines whenever possible. D2K is an environment for parallel and distributed data mining developed by the National Center for Supercomputing Applications, and it has been used previously with a music-specific toolkit (called M2K, not used in this thesis) as the platform for the MIREX 2005 evaluations (Downie et al. 2005).

A D2K program (or "itinerary") is built from a set of interconnected, independent components, called "modules." Each module performs a self-contained computational program, and it may input objects from other modules and output objects to other modules. A user designs an itinerary using a graphical interface to choose modules and then connect their inputs and outputs together. Each module may also have parameters (called "properties") that the user can specify in the itinerary or at runtime. When an itinerary is run, modules without inputs execute immediately, and modules with inputs execute as soon as all their desired input objects have been supplied by previously executed modules.

A simple itinerary for evaluating a Weka Naïve Bayes classifier using D2K appears in Figure 7. This itinerary is included as an example in the D2K distribution. The *Input File Name* module has one property, allowing the specification of the name of the file containing the dataset (stored in Weka ARFF format). It outputs the name of the file, which is input to the *WEKA_ReadARFF* module. This module reads in the data and passes it to the *WEKA_CVClassifierEvaluator*. The *WEKA_NaiveBayesModelProducer* produces a Weka Naïve Bayes classifier object, and it passes this object to the

WEKA_CVClassifierEvaluator module. This module allows the specification of the number of CV folds as a property, and it evaluates the Naïve Bayes classifier on the dataset using the specified number of CV folds. (The evaluation results are printed out to the screen when the itinerary executes.)

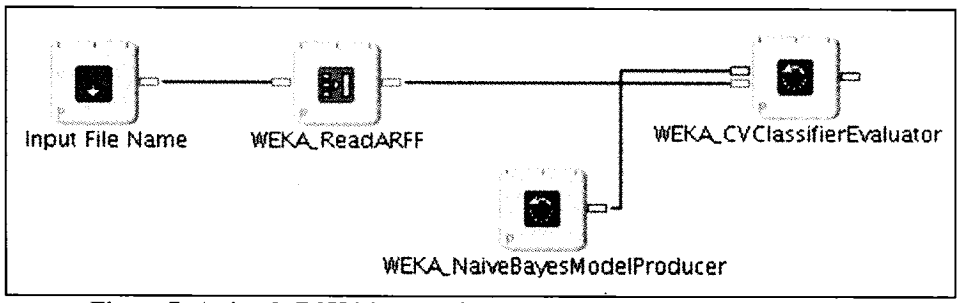

Figure 7: A simple D2K itinerary for evaluating a Naïve Bayes classifier.

D2K is written in Java, allowing cross-platform compatibility. While D2K is not open-source, it is possible to develop new D2K modules in Java and integrate them into the D2K environment. Several new modules were written for the tests in this thesis.

D2K allows simple control over parallel and distributed execution, in that each module may be specified to run on one or many of the available processors. Additionally, reentrant modules have the property that they may be run on several processors simultaneously. Reentrant modules are therefore appropriate for evaluating many different feature subsets simultaneously, as is possible in selection algorithms such as GAs or forward selection in which a large number of subsets are evaluated independently during each iteration of the algorithm. In practice, communication overhead between processors was low compared to the computation time needed to evaluate feature subsets, so the computation time decreased nearly linearly with the number of processors used. Figure 8 illustrates this speed-up for performing GA selection on an example dataset.

3.5 Genetic algorithms

An existing generic GA implementation, JGAP (Rotstan and Meffert), was modified for use as a feature selection algorithm for the tests in this thesis. JGAP

is an open-source GA package written in Java. It provides a high degree of flexibility in specifying how fitness is calculated for a chromosome; here, each chromosome was evaluated by using CV to evaluate a Weka classifier wrapped in a D2K module, and the CV accuracy was assigned to be the fitness for the chromosome. JGAP also allows acceptable degrees of control over the initial population size, mutation rate, selection mechanism, and other GA behaviors.

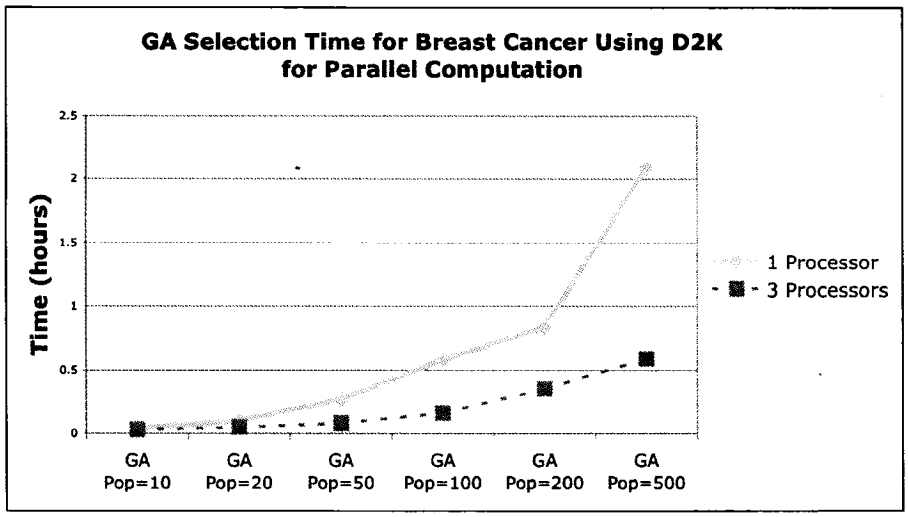

Figure 8: For GA selection and a large initial population, using D2K to increase the number of processors from 1 to 3 improves speed by nearly a factor of 3.

3.6 Other feature selection algorithms

Other feature selection algorithms were implemented from scratch in Java. All feature selection algorithms, including GAs, were designed to extend an abstract feature selection class, allowing each feature selection algorithm to be managed using a uniform set of methods.

3.7 Overall system

A software system was constructed to integrate all the above components, allowing complete tests of feature selection to be run with a single command (or with a one-line shell script). All parameters of the feature selection evaluation are specifiable at run-time, including:

- Location of the dataset

- Location and names of the output files generated
- Location of the D2K itinerary to use for evaluating feature subsets
- How to split the data into outer training and testing sets (i.e., the relative size of these sets, and whether to use stratification; see Chapter 4 for further information)
- How to perform repeated testing (i.e., whether to perform outer CV, and if so, the number of folds, or whether to perform repeated sampling, and if so, the number of iterations)
- How to evaluate each feature subset on the training data (i.e., the number of inner CV folds, and whether to use stratification; see Chapter 4 for more information)
- The feature selection method and its parameters

Figure 9 shows the high-level integration of the feature selection algorithm and D2K. Figure 10 shows the actual D2K itinerary used in this system.

The overall system includes backup, restart, and logging functionality. The final version of the system consists of 24 Java classes (over 7500 lines of code). This includes the additions to JGAP for implementing fitness evaluation using D2K, as well as the new D2K modules. The overall system can be used for future work applying feature selection to any dataset encoded in Weka ARFF format.

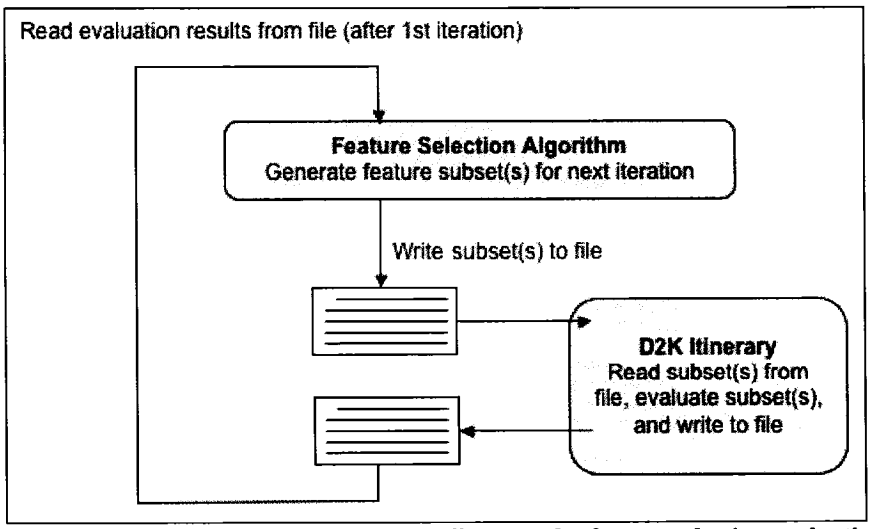

Figure 9: Integration of D2K into the overall system for feature selection evaluation.

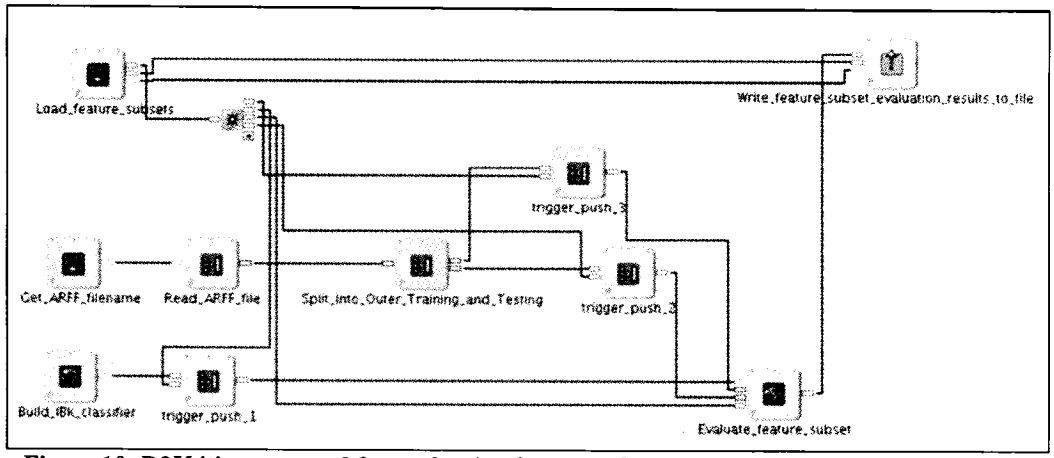

Figure 10: D2K itinerary used for evaluating feature subsets. The *Evaluate_feature_subset* module is reentrant.

3.8 Datasets

Several popular non-musical datasets from the UCI Repository (Newman et al. 1998) were used for the work in this thesis. The UCI Repository contains over 120 datasets with varying numbers of instances, features, and classes, created from a variety of domains. These datasets appear in other work in feature selection (e.g., Kohavi et al. 1997; Reunanen 2003, 2004; Skalak 1994; Wettschereck et al. 1997), and they are available for anyone to use for the purpose of comparing results. Additionally, they are a convenient resource when multiple tests must be run using a variety of datasets, such as is the case for the work in Chapters 4 and 5.

In order to make sense of the wide variety of datasets available in the UCI Repository, and to facilitate the selection of appropriate datasets for use in the tests in this thesis, a table summarizing all datasets in the repository was created[1]. The table includes relevant properties such as the number of instances, the number of features, the number of classes, the type of class (e.g., nominal or numeric), whether any instances are missing class labels, whether any instances are missing feature values, the number of each type of feature (i.e., nominal, binary, or numeric), and the number of instances in the majority class. The table also provides the source of the data, a text description of the dataset, and additional

[1] This table appears online at http://www.music.mcgill.ca/~rebecca/thesis/UCI_table.htm.

notes about the dataset. This table offers an easier way to browse and search the repository than the UCI description webpage[2]. In addition to serving as a resource for dataset selection in this thesis, this table may benefit others doing work that involves the use of UCI datasets.

[2] The UCI description page appears online at http://www.ics.uci.edu/~mlearn/MLSummary.html.

4 CHOOSING A PRINCIPLED EVALUATION METHOD FOR FEATURE SELECTION

4.1 Overview

Reunanen's work (2003, 2004) suggests that wrapper feature selection methods have not always been properly evaluated in prior work. One can therefore not necessarily trust existing work claiming that feature selection is a generally useful tool for improving classification accuracy (see Reunanen 2004) or work showing that one feature selection method is more likely than another to result in higher classification accuracy (see Reunanen 2003).

Alongside his exposition of potential flaws in methodologies that have often been used to evaluate feature selection, Reunanen has explicated several characteristics of what he believes to be a better methodology. However, one is confronted with the task of setting many outstanding parameters of the experimental design if one wishes to measure the relative performance of different algorithms for a particular domain, or if one wishes to assess whether feature selection is beneficial at all. Unfortunately, neither Reunanen nor any other researcher has laid out clear guidelines that are specific, comprehensive, and well-reasoned enough to form a sufficient foundation for someone wishing to perform feature selection evaluation in a way that the choices of all such parameters are well-understood to be methodologically sound.

This chapter examines relevant literature regarding model selection, feature selection, and evaluation methods. It identifies problematic gaps in the literature and describes attempts to address those gaps with new, original empirical testing. Then, based on this testing and the literature, it proposes a judicious method of evaluation for feature selection for employment in this project and in other projects in music and machine learning.

4.2 Goals

An evaluation of feature selection (such as is performed in this thesis) might address two distinct questions: First, is feature selection a useful tool for improving classification accuracy on a particular problem? A more specific formulation of this question is, does feature selection algorithm A improve classification accuracy on dataset X? Second, what are the relative performance characteristics of various feature selection algorithms that one might employ? A more specific formulation of this question is, what are the relative performance characteristics of algorithms A, B, and C, as they might be applied to classification of dataset X? In answering these questions, one might be interested in both efficacy and speed characteristics of various feature selection algorithms. Additionally, because one is usually more concerned with the ability of the final model to generalize well than with its ability to classify only known (training) data well, assessment of efficacy involves examining performance measures related to "out-of-sample performance prediction" or "generalization prediction" (Guyon and Elisseeff 2003, 1173).

The goal of this section is to outline a reasonably good evaluation methodology that will allow one to answer the questions above. A principled methodology will allow one to make sound judgments about an algorithm's performance, and it will steer one away from the trap of overestimating the performance of an algorithm due to overfitting (see below). Without such a principled methodology, any experiments one performs to evaluate different feature selection methods to apply to the genre classification problem are of questionable helpfulness, and experiments one performs to examine feature selection's efficacy on the genre classification problem are of questionable worth. Specific priorities one should consider in choosing an evaluation methodology include:

- Speed: One must be able to perform evaluations in a reasonable amount of time.
- Consistency: The outcomes of comparison experiments should be consistent; for example, if a comparison is repeated with different data

drawn from the same distribution, the results of the experiment should not change appreciably. Otherwise, multiple experiments must be performed to gain an insight into the general relative performance of competing algorithms.

- Accuracy: Evaluation experiments should rate "good" algorithms well and "bad" algorithms poorly, or in the case of experiments comparing algorithms, it should rate "better" algorithms better than "worse" algorithms.[3] The machine learning literature employs the terms *bias* and *variance* rather than accuracy and consistency, respectively (Duda et al. 2001, 465–71). A good evaluation methodology that minimizes variance and bias will be both consistent and accurate. However, as Kohavi (1995) has discussed, when one is interested in making meaningful comparisons, low variance is more critical than low bias.
- Understandability: One would like the outcomes of the evaluations to be understandable to others in machine learning. The outcomes should be easy to replicate and easy to compare with other results.
- Applicability: The evaluation method should be applicable to a broad range of datasets and algorithms (e.g., not only two-class problems or wrapper-based selection methods).

It must be made clear that the goals of the following work do not include making the claim that one feature selection method is always (or even generally) superior to another, nor searching for an evaluation methodology that would allow one to make such a claim. Wolpert and Macready (1997) have laid out a set of "no free lunch" (NFL) theorems for optimization holding that "... for any algorithm,

[3] I surround "good" and "bad" with quotation marks because one can reasonably argue that the evaluation experiment itself is the best objective indicator of an algorithm's performance, or that *no* algorithm is inherently superior to any other (see Wolpert and Macready 1997). However, it is trivial to assign arbitrary scores to algorithms in a way that is consistent but useless in informing one of their likely future performances; this would undoubtedly be a poor evaluation methodology. Conversely, I am most concerned that the evaluation outcomes are relevant to the real performance of these algorithms on similar data.

any elevated performance over one class of problems is offset by performance over another class;" that is to say, there is no algorithm that is better overall than any other method (Wolpert and Macready 1997; Duda et al. 2001). Chapters 4 and 5 here concern themselves with evaluating whether feature selection *can* be a beneficial tool, and if so, which algorithms are likely to work well in practice (considering both accuracy and implementation issues). Chapter 6 explores the use of feature selection on improving accuracy for a specific learning approach to a specific domain, that of classifying musical genre with a given set of available features using a particular set of classifiers. The NFL theorems do not necessarily preclude the ability to generalize about the performance of various algorithms within a constrained scope, where one has prior knowledge about the nature of the data and classification task. Furthermore, the applicability of these theorems to the set of problems one actually encounters as a machine learning practitioner is not known; for example, Salzberg writes, "experimental science is concerned with data that occurs in the real world, and it is not clear that these theoretical limitations [i.e., the NFL theorems] are relevant" (Salzberg 1997, 326).

4.3 Evaluation in the literature

4.3.1 Approaches employing cross-validation performance for evaluation and comparison

Some evaluations of the efficacy of wrapper methods for feature selection in the literature are quite straightforward. Kudo and Sklansky (2000), for example, evaluate several wrapper feature selection algorithms. In their study, each feature selection algorithm evaluates feature subsets using leave-one-out cross-validation (LOO CV). They assess the relative quality of the algorithms by comparing the LOO CV score of each algorithm's chosen feature subset.

CV is an established and popular method for evaluating classification accuracy in a way that addresses the generalization of the classifier to new data (as opposed to considering only accuracy on the training data, which does not address generalization ability). However, researchers have recently noted that repeatedly applying CV to the same dataset, like Kudo and Sklansky do, is a

misuse of this tool. Moore and Lee (1994) write, "... a naïve intensive use of cross validation, perhaps over many thousands of models, may produce a deceptively good lowest-error model, in a manner similar to overfitting of data." Furthermore, Jensen and Cohen (2000) describe the phenomenon of oversearching as being closely related to overfitting, and this concept is relevant to wrapper selection: one effect of searching among many feature subsets is that, even if they are all equally good, the probability that the best feature subset found will have considerably better CV performance increases simply with the number of candidate feature subsets evaluated. Using CV performance to compare an algorithm that assesses many feature subsets (such as sequential forward floating selection, e.g. in Pudil et al. 1994) to an algorithm that evaluates fewer subsets (such as forward selection) will therefore automatically favor the algorithm that visits more subsets in its search even if those subsets are not of higher quality (this is also discussed in Reunanen 2003). According to Reunanen, validating with a test set that is not shown to the feature selection algorithm during the search for a good or optimal feature subset is a much better way to evaluate whether one algorithm is actually outperforming another (Reunanen 2003) or that feature selection is indeed increasing accuracy at all (Reunanen 2004).

4.3.2 Reunanen's evaluation methodology

Reunanen (2003, 2004) demonstrates the use of an evaluation methodology that employs such a test set to validate the performance of feature selection algorithms: he creates a stratified testing set and training set (which is referred to in this thesis as the "outer testing" and "outer training" sets for reasons explained below), runs the wrapper-based selection algorithm on the outer training set, then evaluates the final model using the outer test set. The feature selection algorithm searches for a subset with optimal LOO CV performance on this outer training set, in the same manner as in Kudo and Sklansky; the feature set chosen by the algorithm in the end is the feature set that results in the highest LOO CV accuracy of a classifier using the given features. To evaluate an algorithm's performance, the classifier is trained on the outer training set, using

the selected features, and the accuracy with which it classifies the previously unseen data in the outer testing set is recorded. To determine whether feature selection has improved accuracy, one can compare this accuracy to the accuracy of a classifier trained on the outer training set, using all features to classify the outer testing set. To compare different feature selection algorithms, one can compare the classification accuracy obtained on the outer testing set by classifiers using the feature subsets chosen by the different algorithms.

4.3.3 Loughrey and Cunningham's evaluation methodology

Loughrey and Cunningham (2004) use a similar evaluation methodology that involves nested CV. Like Reunanen, they show the feature selection algorithm only a portion of the available data, then evaluate the performance of the algorithm by classifying the held-out portion of the data using the selected feature subset. Unlike Reunanen, they repeat this process multiple times, using a CV approach to split the entire dataset into "outer" folds and iteratively use each fold as the outer testing set. They refer to this level of CV as the "outer" CV, and they refer to the CV used by the feature selection algorithm on the training data to guide the search for good feature subsets as the "inner" CV.

Reunanen's and Loughrey and Cunningham's methods address the fundamental problem of evaluating generalization using very similar approaches. Loughrey and Cunningham's method has an advantage over Reunanen's in that it involves computing several scores (specifically, one score per outer fold) for a given selection method and dataset, rather than just one overall score. One might be interested in considering the variance among these scores to gain an intuition for a method's reliability. However, the rigorous use of basic statistical analysis tools (such as standard deviation) is inappropriate, because the processes producing the scores are highly interdependent due to the repeated reuse of the same instances (see Salzberg 1997, 326). Reunanen's method has an advantage in that the feature selection algorithm, which can be quite computationally intensive, must run only once.

4.3.4 Other approaches to evaluation

Comparing the outer testing set classification accuracy of the best models found by selection algorithms is a straightforward way of evaluating whether one feature selection algorithm is working better than another, or whether feature selection is helpful for a particular problem, but other evaluation approaches do exist. In particular, more complex methods have been proposed for statistically rigorous comparisons of learning algorithms. For example, McNemar's test and a binomial relative of this test are quite powerful tools for comparing two algorithms, and they allow statistical interpretations of the results (Salzberg 1997). Other statistically motivated comparison methods have been proposed by Dietterich (1998), Nadeau and Bengio (2003), and Cohen and Jensen (1997). One might be tempted to use simpler statistical tests (e.g., t-test) to make inferences about the output "observations" of multiple classifier evaluations, for example, partitioning the data several times and observing the resulting outer testing set accuracy, However, it must be underscored that this approach is inappropriate. The main problem inhibiting the use of simpler tests is that multiple iterations of classifier evaluations are not at all independent, due to the fact that the classified instances are drawn from the same dataset (Salzberg 1997).

In this thesis, it was decided to use outer testing set classification accuracy rather than one of the other methods for the following reasons: Test set performance is easy to understand, simple to compute, and allows direct comparison to many of the existing papers about feature selection. Many of the statistically rigorous evaluation methods are only applicable to comparisons of two algorithms to each other (e.g., McNemar's test), not to comparing a group of algorithms. These tests tend to be more complex to implement and do not facilitate direct comparison with published results for feature selection algorithms. While stringent statistical methods would be necessary for making strong claims about a particular feature selection algorithm, the work in this thesis does not aim to do such a thing. The transparency and straightforwardness of testing set classification accuracy aligns satisfactorily with the goal of gaining an increased

general understanding of the benefits and costs one might incur in applying various feature selection algorithms to music classification.

4.4 Defining a specific evaluation methodology

There are many free "parameters" that must be set to completely specify an evaluation methodology like Reunanen's or Loughrey and Cunningham's. For example, should one use outer folds (like Loughrey and Cunningham), repeated sampling to create outer "sets" instead of outer folds, or just a single sampling to create a outer testing and training set (like Reunanen)? If outer CV is used, should stratification be performed, and how many folds should be used? If not, what should be the size of the outer training and testing sets sampled, and how many times should sampling be performed? If CV within the outer training set is used to guide the search of the wrapper-based selection algorithm, as is typical in all the feature selection literature, how many folds should be used? Should stratification be employed? Furthermore, additional considerations involve the choice of classifier to use for the wrapper selection, its parameters (e.g., the value of k for kNN), and the dataset(s) used in the evaluation.

Decisions regarding the setting of all these parameters are not wholly specified in any single paper on feature selection or evaluation in machine learning, and where decisions are specified, they are often not defended (e.g., Loughrey and Cunningham use ten folds for both the outer and inner CV, but they do not explain their decision to do so). However, these decisions must be made somehow if one is to implement and use an evaluation methodology. Ideally, such decisions should be backed up by a combination of theoretical and experimental validation in the existing literature, a precedent of use by machine-learning experts (ideally accompanied by an explanation of their choices), and careful reasoning. If the former are insufficient grounds for making a decision, empirical evaluation may lend insight into the relative value of feasible options. This section addresses these issues for each of the free parameters in turn.

4.4.1 Size of testing and training sets

There is no literature that specifically addresses how data should best be partitioned into outer training and testing sets, or outer folds, for the purposes of evaluating feature selection. Learning curves are a general tool in machine learning, used for examining the testing accuracy of a classifier as the number of training examples increases, and they provide some insight into this problem. An example of a learning curve appears in Figure 11. Typically, the testing performance increases with the number of training examples in an asymptotic way; after a certain training set size (say, *s* instances), the increased performance on the testing data obtained via adding more training instances will likely be negligible. This implies that, before this point, the classifier has not seen enough training data to learn the given problem well, so an outer partition of the data for feature selection evaluation should allot enough data to the outer training set so that at least *s* instances are used for training the classifier. This also implies that increasing the outer training set size to be much greater than *s* will have negligible impact on the accuracy of the classifier (but it will increase the time needed to run feature selection, and it will deplete the outer testing set of its instances, since the training and testing sets are mutually exclusive). In short, the outer partition of the data is an important consideration, and the optimal partitioning scheme may be dataset-dependent.

In practice, many outer partitioning approaches have been used. Kohavi, Langley, and Yun (1997) examine learning curves for each dataset and specify the ratio of outer testing-to-training instances individually. Typically, training sets are small, with the ratio of the number of instances in the testing set to instances in the training set ranging from 30:1 to 3:1. Reunanen's outer partitioning scheme is also somewhat dataset-dependent, and he also maintains larger outer testing sets than training sets. His rationale for the partitioning of a given dataset is not based on learning curves, but on practicality: "to make sure that the training sets do not get prohibitively large in those cases where the dataset has lots of samples." Partition ratios range from 1:1 for small sets to 9:1 for large sets. Wettschereck et al. (1997) use ratios that seem neither consistent nor convenient (e.g., 150:350,

1000:200, 1040:1040), but they do not provide rationale for these choices. Loughrey and Cunningham (2004) consistently use outer 10-fold CV where the training set is larger (that is, a 1:9 testing-to-training size ratio).

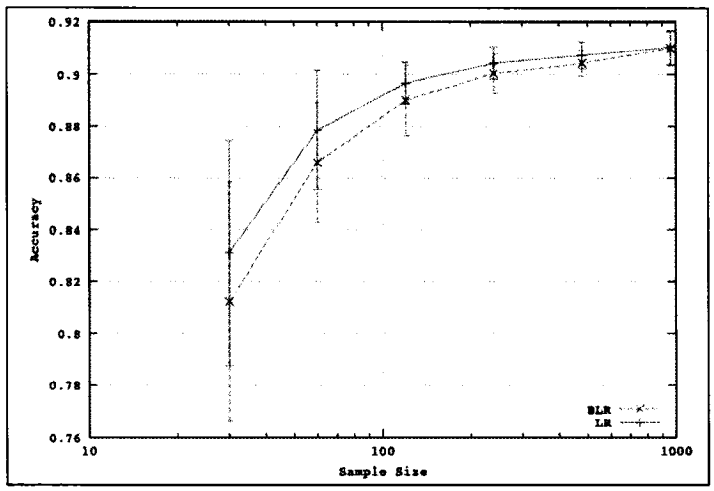

Figure 11: Example of learning curves. This diagram shows curves for two related classification algorithms, on a typical dataset from UCI, from Perlich et al. 2003. The x-axis denotes the training set size, and the y-axis denotes the test-set accuracy.

4.4.2 Outer CV or sampling

Related to the paucity of literature addressing appropriate sizes of outer partitions, there is also a lack of discussion surrounding whether CV or sampling is more appropriate for the outer partitioning scheme. Kohavi (1997) and Reunanen (2003, 2004) both use a single sampling into an outer training and an outer testing set. Reunanen's 2003 study explicitly uses stratification. In this paper, he also notes that CV is a viable choice for the outer partitioning scheme. Loughrey uses 10-fold CV without any remarks regarding this choice. Wettschereck performs sampling, repeated 25 times "to reduce statistical variation" (1997). However, as discussed in Salzberg (1997), such repeated tests cannot easily be used to make claims of significance, because the "trials" are highly dependent.

Because a variety of outer partition methods and outer partition sizes have precedent in the literature, and because of the lack of an authoritative

recommendation in favor of a particular technique, it was decided that empirical investigation of several outer partition methods might lend insight into what are the most sensible choices.

4.4.3 Inner CV

There is more theoretical discussion on the use of CV to compare "models," and this discussion is directly applicable to wrapper feature selection methods. In wrapper selection, each candidate feature subset paired with the base classifier can be considered a model, and wrapper selection typically evaluates competing models using the classification accuracy assessed via CV.

According to Goutte (1997), the number of folds used in CV is generally irrelevant, as long as more than two folds are used. However, Kohavi (1995) explicitly discusses feature selection as a model selection problem, a perspective shared by other researchers (e.g., Moore and Lee 1994), and he discusses the finding that the bias and variance of CV both change according to the number of folds used. He notes that, when one is comparing many competing models, the variance of the evaluation method is of much greater concern than its bias. That is to say, it is better to use an evaluation method that evaluates consistently (say, always underestimates the true performance of every model by 10%, give or take .1%) than a method that evaluates accurately but inconsistently (say, tends to neither overestimate nor underestimate performance on average, but might overestimate or underestimate by 10% for any given model). According to Kohavi, 10-fold CV tends to have a very low variance, so it is a good choice for model comparisons of the sort performed in wrapper feature selection methods. Additionally, he found that stratification resulted in superior bias and variance characteristics, and so recommended its use.

Other literature underscores the idea that, while LOO CV might intuitively seem to be the best choice for evaluating feature subsets, since it is the most "involved" or "rigorous" form of CV, other methods may be more appropriate (see Guyon and Elisseeff 2003; Breiman and Spector 1992; Salzberg 1997). In general, there seems to be consensus that LOO is a suboptimal choice, and that

10-fold CV is reasonable. Salzberg notes, however, that more folds might be more appropriate when the dataset is very small, so that more examples are present in the training set (one can think of this practice as ensuring that asymptote is reached in the learning curve, as discussed above). For a similar reason, when 10-fold CV is used for model selection, it is assumed that the training set will be larger than the testing set, and this practice was employed in all the papers examined in this section.

Given the existing body of literature on this topic, and the general agreement among researchers that 10-fold CV with stratification is appropriate for model selection, no further exploration of this parameter seemed necessary.

4.4.4 Classifiers

There is no literature specifically addressing what classifiers might be best to use for feature selection evaluation. However, it makes sense to avoid using classification algorithms that perform their own explicit or implicit feature selection. Because wrapper-based selection involves many iterations of training and testing a classifier, particularly if inner CV is used, speed of training and testing is a primary concern. Furthermore, it makes sense to employ the same type of classifier in the evaluation of feature selection methods as will be ultimately used for the music genre classification problem. This avoids the need to make the assumption that the relative efficacy of various wrapper-based feature selection methods is unchanged by the classifier employed.

In the literature, kNN classifiers are a popular choice for evaluating feature selection methods (e.g., they are used in Reunanen 2003, 2004; Kohavi et al. 1997; Wettschereck et al. 1997). The effect of feature selection with kNN is conceptually simple: removing a feature removes a dimension of the nearest-neighbor space in which the distance between instances is calculated. Training time is negligible, and testing time is manageable for the sort of datasets used in this work (including the musical genre dataset). k=1 (a 1NN classifier) was used for the evaluations, because it is a choice that has precedent in the literature (e.g.,

Reunanen 2003, 2004), and there is no conclusive evidence that other values of k are, in general, better choices.

4.4.5 Datasets

The question of which datasets one should use to evaluate feature selection is not an easy one. The NFL theorems imply that applying a feature selection method, using a given classifier with a given parameter set, may give quite different results on different datasets. However, if one wishes to learn more about feature selection methods, there are a few properties of datasets that will make such an investigation easier.

4.4.5.1 Number of instances

The number of instances has a large impact on the time it takes to run wrapper feature selection, because it effects the time to train and/or test most classifiers (including kNN). The datasets used should be of manageable size in order to practically run tests.

Number of features: The number of features also impacts the time needed to train and test a classifier (including kNN), and it can also explicitly influence the time to run some feature selection methods (such as forward selection, which performs $O(d^2)$ feature subset evaluations for a dataset with d features). Practicality therefore places an upper limit on the number of features of datasets used in these tests. However, it makes sense to test using datasets with different numbers of features, within this constraint, because it is reasonable that the number of features could influence the performance of a feature selection algorithm.

4.4.5.2 Relationship between number of instances and number of features

The "curse of dimensionality" refers to the need for an exponentially increasing number of training instances as the number of features grows linearly (Duda et al. 2001, 169–70), if classification is to be feasible. Therefore, datasets with varying ratios between the number of instances and number of features should be included.

4.4.5.3 Prior use in the literature

If a dataset has been previously used in the literature to make claims about feature selection's efficacy or about the relative quality of feature selection methods, any results one obtains using this dataset can be compared to published results.

4.4.5.4 Potential for feature selection to improve generalization accuracy

Holding an a priori belief of whether feature selection will be effective, when one is performing tests regarding feature selection's efficacy, seems an odd idea at first. However, for a given evaluation methodology, such as the hierarchical partitioning scheme described above, it is possible to determine whether feature selection has much (or any) chance of finding a better feature subset than the set of all available features. For several UCI sets where it was feasible to exhaustively visit all feature subsets, each subset was evaluated with a classifier using the hierarchical outer/inner partitioning scheme described above. For each outer partition (i.e., each outer fold), the CV accuracy on the outer training set and classification accuracy on the outer testing set were recorded. These values were plotted (Figure 12), and the performance of the classifier for the cases when all features were used was noted. For datasets where the outer training CV accuracy, which is used by any wrapper method to guide its search through the set of feature subsets, is highly predictive of the outer testing classification accuracy, and where there exist feature subsets whose outer test set performance exceeds that using the entire available feature set, feature selection should be able to improve classification accuracy. For other datasets, feature selection will probably not improve classification accuracy.

These plots were created using several variations of the partitioning scheme, including outer sampling and CV, using different numbers of folds for outer and inner CV, and varying the use of stratification; notably, these changes did not greatly impact the results. Additionally, similar plots were created for datasets with too many features to exhaustively evaluate all feature subsets, by randomly generating a predetermined number (e.g., 100) feature subsets. The use

of such plots for reasoning about feature selection's potential to improve classification accuracy is discussed in greater detail in Chapter 5.

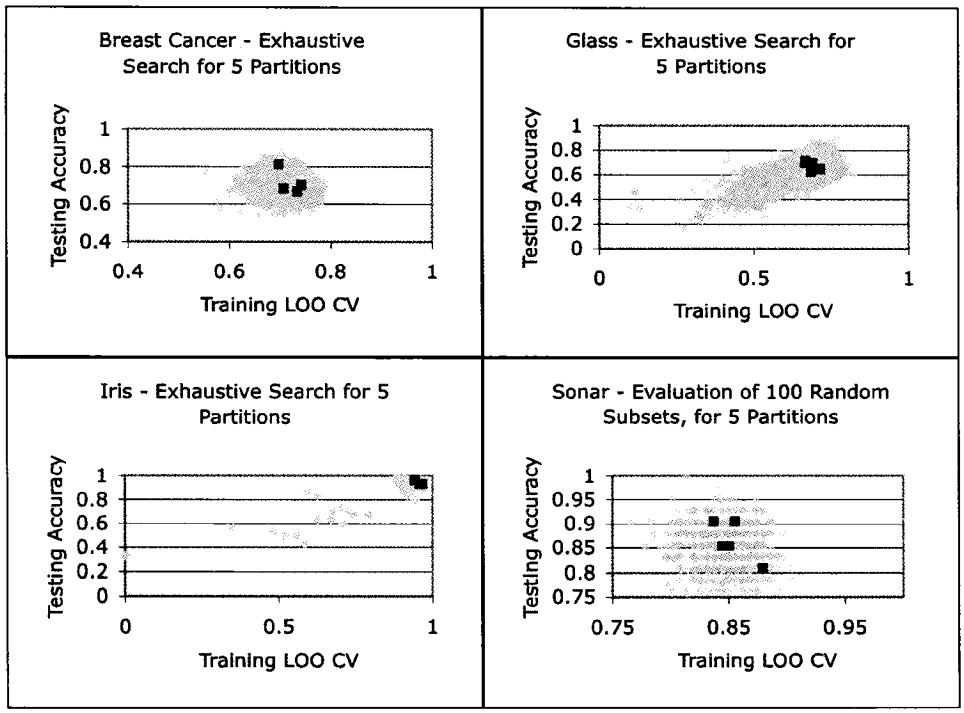

Figure 12: Plots of LOO CV accuracy on the outer training set versus classification accuracy on the outer testing set, for the UCI datasets used in the tests of this chapter. Each plot shows results for 5 separate partitions. The set of all available features is marked with a black square, for each partition. The other 2^d-2 subsets are marked with grey diamonds. See Chapter 5 for more information.

4.5 Empirical evaluation of methodology parameters

4.5.1 Datasets, selection algorithms, and classifiers

Table 1 describes the datasets employed in the testing of this chapter. Each is from the UCI Repository (Newman et al. 1998; see Chapter 3). These datasets are all quite small to facilitate repeated testing, they employ a variety of numbers of features and a variety of ratios of the number of instances to the number of features, and most have been used in a variety of papers about classification and feature selection (e.g., Reunanen 2003, 2004; Kohavi et al. 1997). For one set (Glass), it is likely that feature selection will succeed in improving classification accuracy; for one set (Iris), it is unlikely that feature selection will succeed because of the already high performance using the entire feature set; for two sets

(Breast Cancer and Sonar) it is unlikely that feature selection will succeed because of the lack of correlation between outer training and testing accuracy, but for one of these (Sonar) exhaustive search is infeasible, so one cannot know for certain that a better subset does not exist. The plots of Figure 12 illustrate the basis for these claims.

Table 1: UCI datasets used in the tests in this chapter.

Dataset	# Instances	# Features	# Classes	Improvement expected?
Glass	214	9	7	Yes
Breast Cancer	286	9	2	No
Sonar	208	60	2	No (?)
Iris	150	4	3	No

The feature selection algorithms employed in these tests included forward selection, backward selection, random mutation hill climbing (RMHC), and Monte Carlo selection. RMHC and Monte Carlo selection were both run twice, once evaluating ten feature subsets, and once evaluating fifty. As discussed in Chapter 2, forward selection, backward selection, and RMHC have been discussed in the literature on feature selection. Monte Carlo selection serves here as a "sub-optimal" feature selection method (one would assume it would perform worse than the other, "real" selection methods). None of these selection methods have parameters that must be set (unlike genetic algorithms (GA), a popular selection method that was excluded from these tests for this reason).

A 1NN classifier was used for all feature selection algorithms. The J48 classifier (Witten and Frank 2005) was also evaluated using all features to serve as a comparison to 1NN without selection, with the assumption that relative performance of the various split types would be similar across both classifiers. J48 is an implementation of a standard decision tree classifier, C4.5. Decision trees classify by iteratively partitioning the instances into subsets until each subset is assigned a single class value, where the decision to assign an instance to a particular subset is made one the basis of its feature values; they are therefore quite conceptually different from nearest-neighbor classifiers.

4.5.2 Tests performed

4.5.2.1 First tests: Method of partitioning into outer training and testing sets

The outer partition of the data was initially performed with 9:1, 1:1, and 1:9 ratios of testing set size to training set size. These ratios were chosen based on their previous use in the literature, and because n=10 is a common choice for n-fold CV. The initial tests involved both CV and independent sampling for each partition ratio. For sampling, the same number of trials were performed as for the CV tests with the same ratio; for example, for the 1:9 split with sampling, the data was partitioned ten times into outer testing and training sets. All tests (both CV and sampling) used stratification to maintain identical class proportions between the outer training and testing sets and folds. Feature selection search methods used 10-fold CV of the outer training set with stratification to assess subset quality.

Figures 13 through 15 show the results of this testing. The outer test set classification accuracy of the feature subset selected by each method was compared across all trials for a dataset; the standard deviation of these performance values was averaged over all algorithms and charted in Figure 13. The 1:1 ratio of outer testing set size to outer training set size tended to display the lowest standard deviation among trials for all datasets, for both CV and sampling. The 9:1 ratio tended to have the next lowest standard deviation, with 1:9 performing the worst. This suggests that 1:1 and 9:1 have lower variance, and might be preferred for comparing feature selection algorithms, for much the same reasons as Kohavi recommends using 10-fold CV for model comparison within a wrapper feature selection algorithm. Large differences in standard deviation were not apparent between CV and sampling for any split ratio.

Figure 14 shows the correlation between the outer training set CV accuracy and the outer testing set classification accuracy, using the feature subset selected by an algorithm. This was computed by calculating this correlation value for all trials of each algorithm on each dataset, then averaging this value over all algorithms and then additionally over all datasets. There is some variation in correlation values for different datasets, but in general the 9:1 ratio tends to result

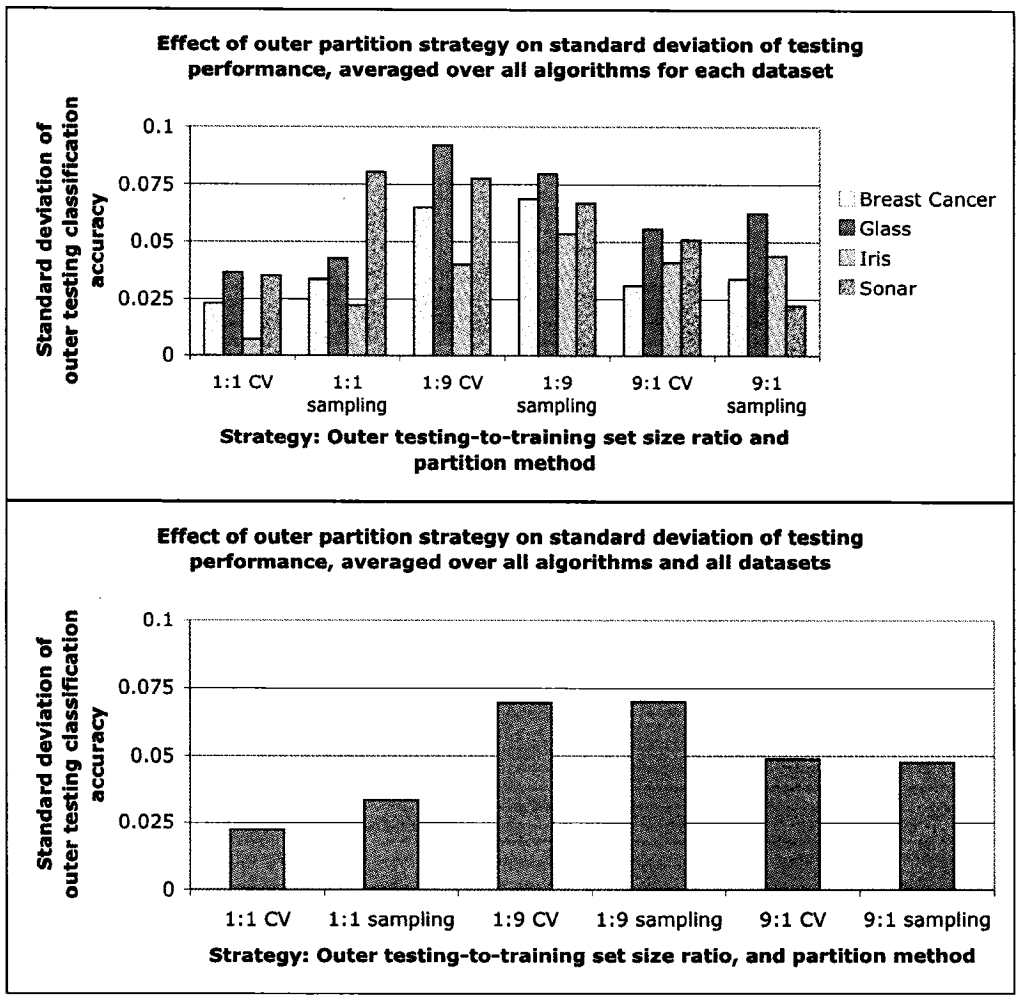

Figure 13: Effect of the outer partition strategy on the standard deviation of outer testing classification accuracy, averaged over all selection algorithms (top) and averaged also over all datasets (bottom).

in a positive correlation, while 1:1 and 1:9 tend to result in negative or negligible correlations. That is, running feature selection on a smaller outer training set tended to result in the chosen best feature subset having outer training set performance that was more predictive of outer testing set performance than running feature selection on a larger outer training set. This is a somewhat surprising result; it seems more intuitive that showing the feature selection algorithm more training data would enable it to find feature subsets that were better fit to the data set.

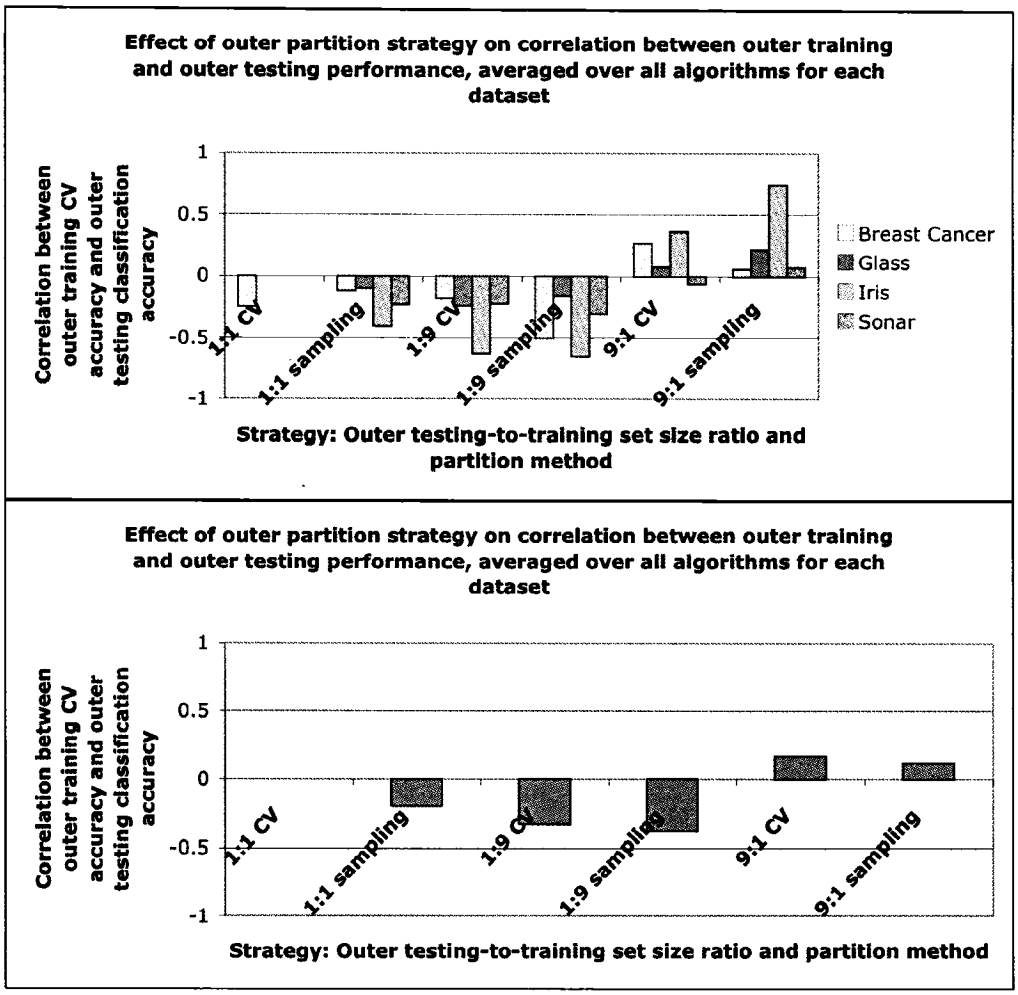

Figure 14: Effect of the outer partition strategy on the correlation between training and testing performance, averaged over all selection algorithms (top) and averaged also over all datasets (bottom).

Figure 15 shows the absolute improvement in classification accuracy over the no-selection 1NN model, averaged over all feature selection algorithms. Glass was the only dataset for which feature selection was predicted to result in an improvement in accuracy, and this improvement is seen using a 1:1 outer partition ratio with outer CV, a 1:1 outer partition ratio with outer random sampling, and 1:9 outer partition ratio with outer CV. The use of a 9:1 ratio and sampling actually gives results contrary to expectation for all datasets.

Given that these results do not conclusively show that one outer partition method consistently leads to results that have low variance and good predictability, the tests were repeated using 1:4 and 4:1 outer testing-to-training

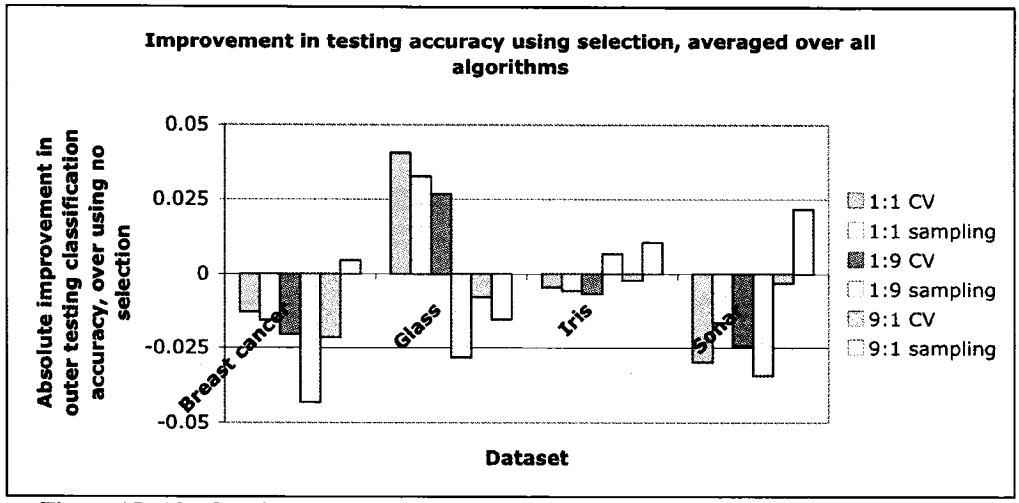

Figure 15: Absolute improvement in accuracy from feature selection, averaged over all algorithms.

size ratios. The standard deviation and average accuracy values for the 1:4 ratio were between the values for 1:1 and 1:9, and the standard deviation and accuracy values for 4:1 were between 1:1 and 9:1. This is unsurprising, but these tests yielded no further insight into what outer partition size ratio might be best.

Based on the above, 1:1 is perhaps the most defensible choice of outer partition size ratio. It has the lowest variability of performance, and it allows accurate assessment of feature selection's efficacy for all datasets. However, 1:1 is not always feasible to use for larger sets, as indicated by Figure 16 showing the polynomial increase in overall evaluation time as the number of instances in a dataset grows. For larger datasets, 4:1 or 9:1 seem to be defensible choices; the strange behavior displayed in mispredicting feature selection's efficacy might not be a problem for larger datasets, where the learning curve is more likely to have reached asymptote for the number of points in an inner training set comprised of 9% of the available data (that is, the inner training folds of the outer training set using a 9:1 ratio and 10-fold inner CV). The decision to use larger outer testing sets for larger datasets is reflected in the work of Reunanen (2003, 2004) and Kohavi et al. (1997) as well. There is certainly room for further work on this topic to better elucidate the trends seen in Figures 13 through 15.

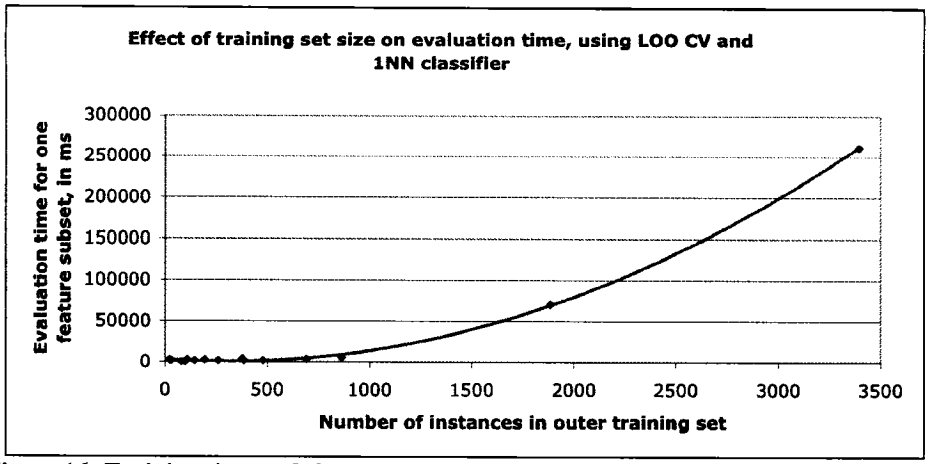

Figure 16: Training time and the number of instances, using a 1NN classifier. Evaluation time increases polynomially when inner *n*-fold or LOO CV is used to evaluate feature subsets.

4.5.2.2 Second tests: Stratification

Further tests were performed to investigate the effects of stratification in the outer partitioning of the data, given that it was unknown whether maintaining class proportions between the outer training and testing sets would consistently affect evaluation results, or the time needed to perform evaluations. Using the 1:1 ratio of outer testing-to-training set size deemed reasonable by the first set of tests, average accuracy and standard deviation were calculated for the same set of feature selection algorithms and datasets, with and without outer stratification. The results appear in Figures 17 through 20. Figure 17 clearly shows that the effect of stratification on absolute accuracy is negligible. Figure 18 shows that the standard deviation may be affected by the use of stratification, but the effects of stratification differ according to the feature selection algorithm used. Figure 19 shows that this is the case even when analysis is restricted to the datasets of Iris and Glass, where the higher degree of correlation between outer training CV and outer testing accuracy demonstrated in Figure 12 suggest that feature selection might behave more predictably. Initial analysis of the impact of stratification on running time for these small datasets showed it to have minimal influence, and Figure 20 shows that this is also the case for much larger datasets. That is, stratification does not appreciably increase computation time in practice. The influence of stratification on running time is particularly small in comparison to

the influence of the choice of feature selection algorithm and size of the dataset, as discussed in Chapters 5 and 6.

Therefore, these tests do not give conclusive evidence in favor of or against the use of stratification in the outer partition, at least for the 1:1 partition size ratio. Given that not using stratification to perform the outer split could potentially give rise to an outer training set whose class proportions are very unrepresentative of the dataset as a whole, one might reason that the variance would generally be higher without stratification than with it. Furthermore, the use of stratification could result in more optimistic estimates of performance of feature selection on the test set, because the training set has identical class proportions; however, this optimism will be present for all algorithms being evaluated (including the no-selection case). Using reasoning similar to Kohavi's, one might conclude that it is better to have lower variance when comparing models (including both comparing multiple selection algorithms, as well as comparing no selection with selection) than to have a more honest assessment of performance. For these reasons, and because no ill effects of stratification were consistently observed in the testing, further evaluation employs stratification in constructing the outer partitions.

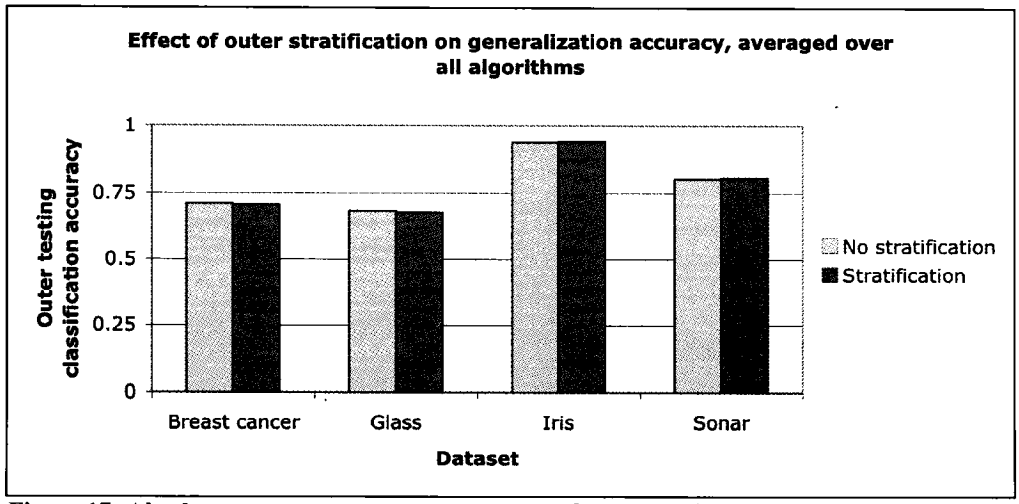

Figure 17: Absolute outer test set accuracy, averaged over all trials and algorithms, with and without stratification for the outer set partition.

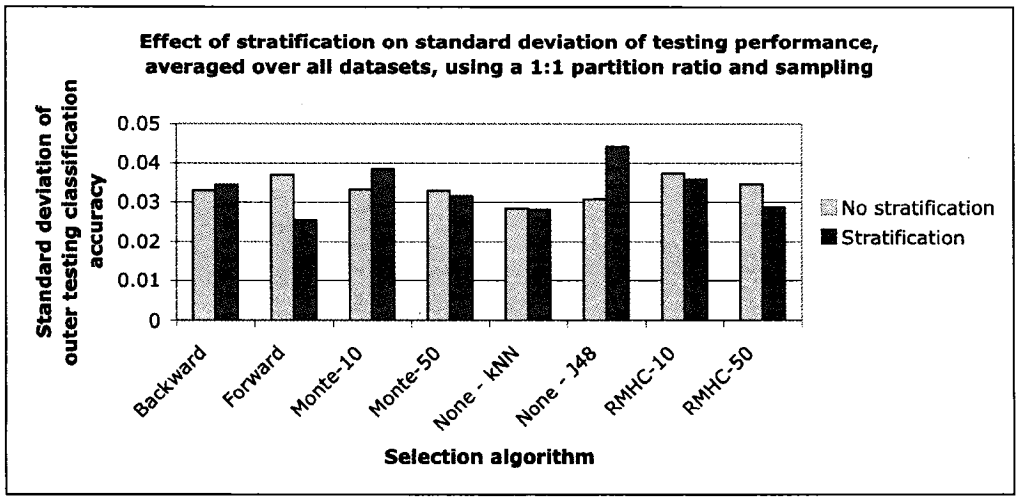

Figure 18: Standard deviation of outer test set accuracy, computed over all trials for an algorithm, averaged over all algorithms and datasets, with and without outer stratification.

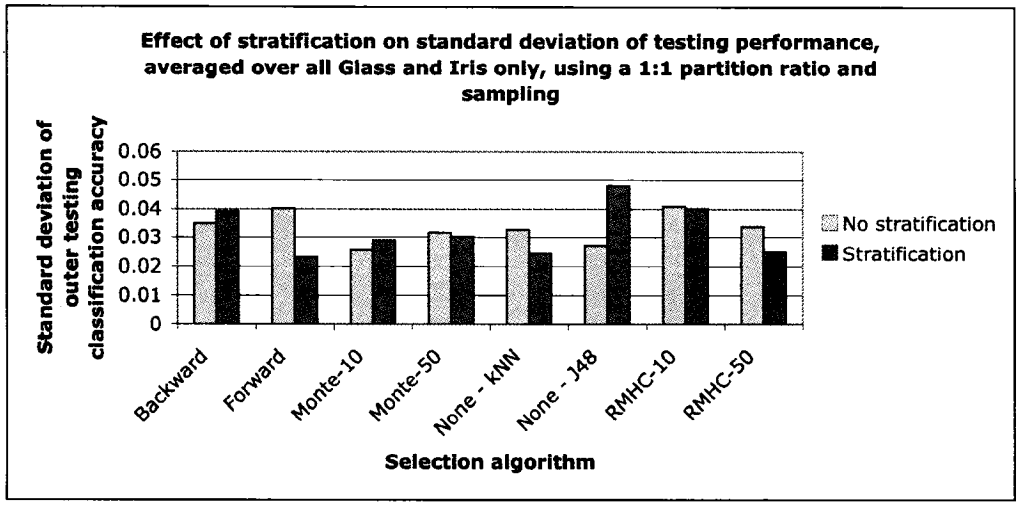

Figure 19: Standard deviation, computed as in Figure 18, for Iris and Glass only.

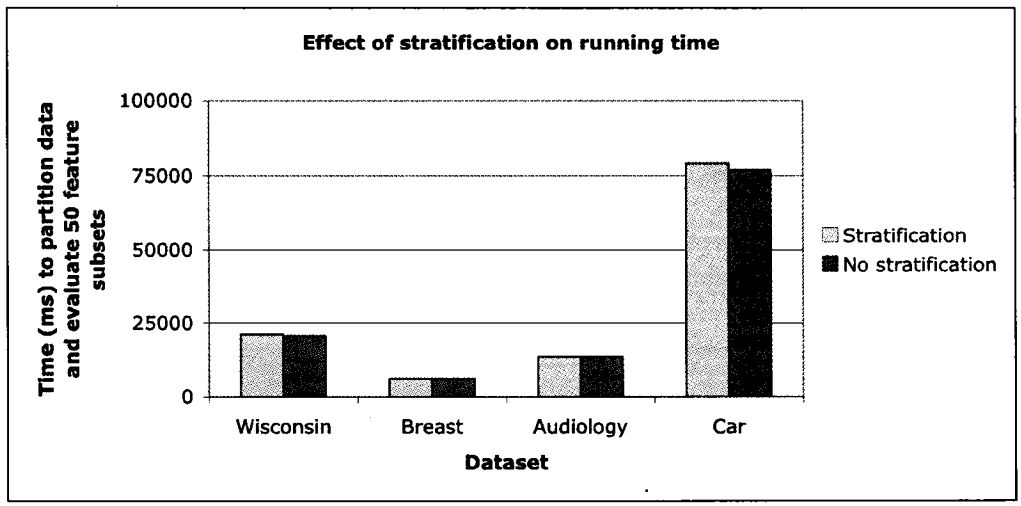

Figure 20: Stratification does not have an appreciable effect on running time.

4.6 Summary of recommendations

Following the discussion above, the following methodology is recommended for evaluation of feature selection methods:

- Evaluate all feature selection methods by examining the classification accuracy of a classifier trained on the data shown to the selection algorithm (i.e., an outer training set), using the selected feature subset, evaluated on data unseen by the selection algorithm (i.e., an outer testing set).
- For small datasets, construct the outer testing and outer training sets using repeated sampling into two sets of equal size (i.e., a 1:1 size ratio), where stratification enforces identical class proportions in the two sets.
- For larger datasets, construct the outer testing and outer training sets using repeated sampling into sets with a testing-to-training size ratio of 4:1 for medium sets and 9:1 for large sets, as determined by practical time constraints. Employ stratification to ensure identical class proportions in both cases.
- Employ 10-fold CV with stratification to guide the feature selection search within the outer training set.
- Compare selection algorithms by comparing the classification accuracy of the outer testing set using the best feature subset found by each algorithm.

This methodology is shown in pseudo code below:

0. N ← number of outer partitions (2 for small sets, 5 for medium sets, and 10 for large sets, as determined by practical time constraints)
1. Repeat N times:
    1.1 Divide complete dataset D into N partitions, D_1 ... D_N, where the size of each partition, $|D_i|$, is
        approximately |D|/N, using stratification to preserve class proportions in all partitions.
    1.2 Define $DOuterTest_i$ ← D - D_i (That is, the outer testing set is the instances *not* in partition D_i)
    1.3 Define $DOuterTrain_i$ ← D_i
    1.4 Divide $DOuterTrain_i$ into ten folds, $DOuterTrain_{i,1}$ to $DOuterTrain_{i,10}$, where the size of each
        fold is approximately $|DOuterTrain_i|/10$, using stratification.
    1.5 Run the feature selection algorithm. Inside the algorithm, score each candidate feature subset

according to its 10-fold CV accuracy using the folds $DOuterTrain_{i,1}$ to $DOuterTrain_{i,10}$.

1.6. $Accuracy_i$ ← classifier accuracy when trained on $DOuterTrain_i$ using the subset returned by the feature selection algorithm, and evaluated on $DOuterTest_i$.

2. OverallAccuracy ← average of all $Accuracy_i$, for i from 1 to N.

Using the above methodology, one can compare average test set performance on subsets found by competing feature selection algorithms to choose which one performs better. Alternatively, one can compare average test set performance of the subset found by a feature selection algorithm to test set performance using the same classifier and all available features to assess whether feature selection is effective in improving accuracy for a particular dataset.

5 EXAMINING FEATURE SELECTION ALGORITHMS

5.1 Will feature selection ever work?

5.1.1 Questions raised by Reunanen

Reunanen (2004) states that the ability of feature selection to improve classification accuracy has been greatly overstated in testing that uses poor evaluation methodology. Furthermore, his own tests suggest that feature selection offers no benefit to classification accuracy of new data for many standard datasets; in many cases, feature selection can select a feature subset that actually leads to worse classification accuracy for new data.

Analysis of forward selection and sequential forward floating selection (SFFS) in an earlier paper (Reunanen 2003) foreshadows this finding: for a given dataset, Reunanen records the feature subsets visited by the feature selection algorithm throughout its search. He plots the estimated "quality" of each subset used by the search algorithm, as measured by the leave-one-out cross-validation (LOO CV) accuracy on the outer training set, versus the actual classification accuracy on the outer testing set. Figure 21 shows these values for each feature subset of size 10 evaluated by forward selection and SFFS. Wrapper selection methods assume a correlation between these two values: they operate by searching for the subset with the best CV accuracy. Their success therefore relies on the fact that a feature subset that allows better CV performance on the data used to guide the selection (i.e., the outer training data) will correspondingly lead to better classification accuracy on unseen data. However, Figure 21 shows that this is not the case: CV accuracy on the outer training set is in fact quite a poor predictor of performance on the outer testing set, even though Reunanen has used stratification to maximize similarity between the training and testing sets. The impact of this phenomenon on the overall performance of feature selection is apparent in Figure 22, which shows the outer training LOO CV accuracy and

outer testing accuracy for the best subset of each size visited by the two algorithms. There is little correlation between these values for either algorithm. It is not clear that picking the subset with the best LOO CV accuracy will result in the best generalization performance.

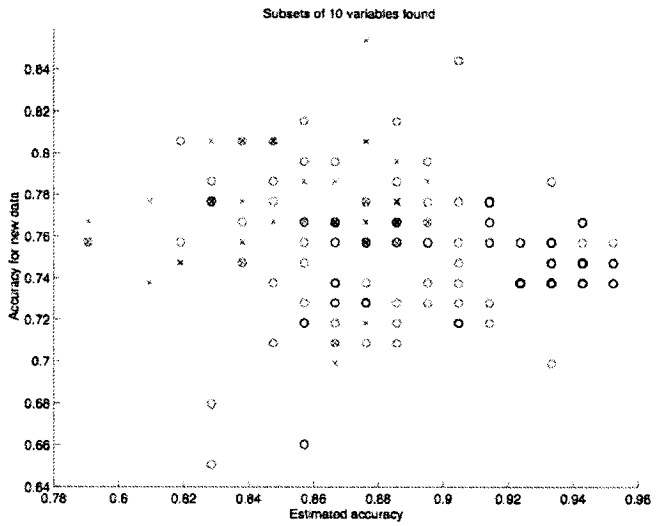

Figure 21: This plot from Reunanen 2003 shows the LOO CV performance on the outer training set ("Estimated accuracy") versus the classification accuracy on the outer testing set ("Accuracy for new data") for all feature subsets of size 10 evaluated by forward selection (marked with 'x') and SFFS (marked with 'o'), on the UCI Sonar dataset.

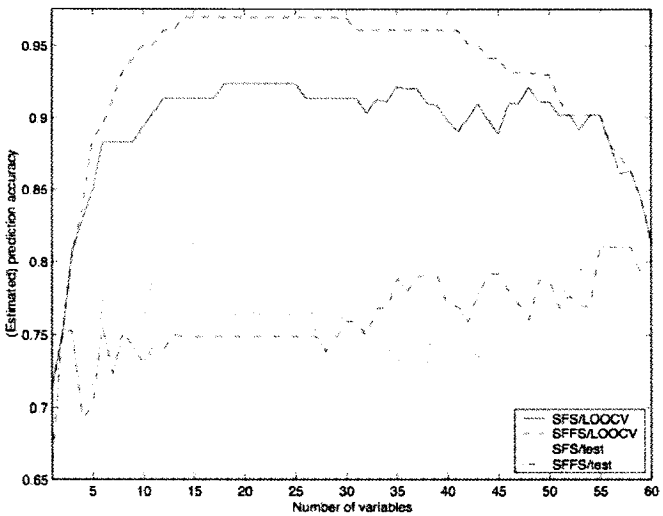

Figure 22: This plot from Reunanen 2003 shows the LOO CV performance on the outer training set and the classification accuracy on the outer testing set for the best feature ("variable") subset of each size evaluated by forward selection (SFS) and SFFS on the UCI Sonar dataset.

In the 2003 paper, Reunanen draws the conclusion that employing algorithms such as SFFS that search intensively for subsets with higher CV accuracy is an inefficient use of time, since finding a subset with marginally higher CV accuracy often does not offer any benefit in increased classification performance on new data. In the 2004 paper, while stopping short of claiming that feature selection will never be effective, he observes that this phenomenon can undermine entirely the ability of wrapper selection methods to find subsets that result in a substantial or consistent increase in classification accuracy over the case in which all features are used.

5.1.2 Creation of new plots

If CV performance on the outer training data is in fact such a poor predictor of generalization performance, this bodes poorly for wrapper selection. However, Reunanen has only plotted the performance of subsets visited by specific search algorithms; it is possible that a correlation between training CV performance and testing performance over all potential feature subsets is obscured when examining only the subsets visited by forward selection and SFFS. Additionally, the high variance of LOO CV (Kohavi 1995) could contribute to decreasing correlation between these values. Using an evaluation method with lower variance, such as 10-fold CV, might improve these results.

To further investigate whether properties of subsets visited by forward selection and SFFS or the use of LOO CV explain the lack of correlation, several new tests were performed. In these tests, all possible feature subsets were evaluated using both 10-fold CV on the outer training set and classification accuracy on the outer testing set. Because the number of possible subsets is 2^d for d features, it was only feasible to perform this testing on datasets where the number of features was small (fewer than 10). Datasets with varying numbers of classes and instances that met this criterion were chosen from the UCI Repository (Newman et al. 1998) for use in these tests.

The datasets were split into outer training and outer testing sets according to the methodology chosen in Chapter 4. That is, stratification was used to split the datasets, and a 1:1 outer testing-to-training size ratio was used for all sets. 10-fold CV with stratification was performed on the outer training set, and this score was compared to the classification accuracy on the outer testing set. A kNN classifier, using one nearest neighbor (k=1) with normalization was used. This is the same classifier used by Reunanen (2003; 2004). Figure 23 shows results for many of the datasets examined.

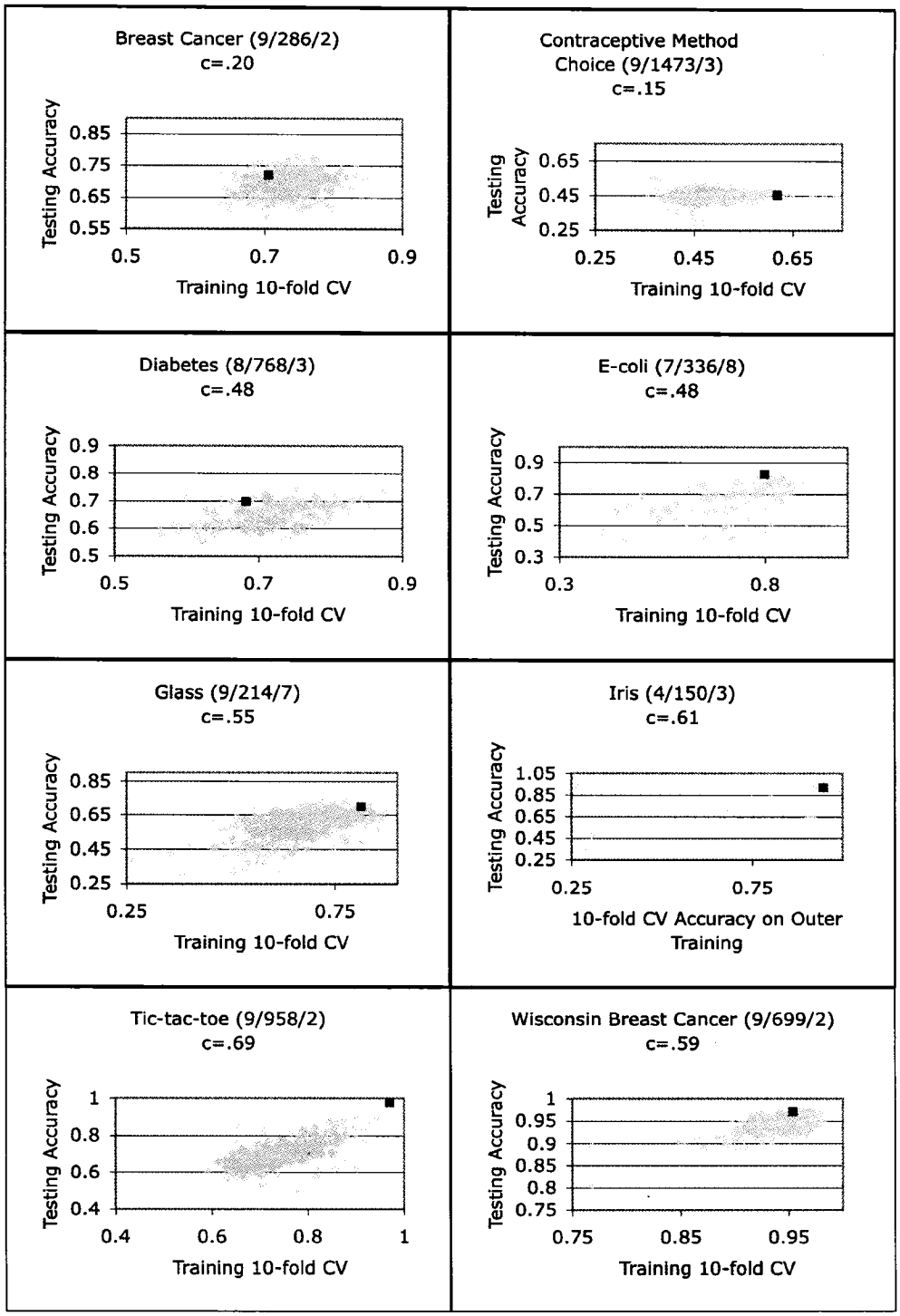

Figure 23: Plots of 10-fold CV accuracy on the outer training set versus classification accuracy on the outer testing set, for several standard UCI datasets, for exhaustive evaluation of all feature subsets. The number of features, instances, and classes are indicated next to the dataset name, and the correlation *c* between training and testing accuracy is indicated beneath. The set of all available features is marked with a black square. The other subsets are marked with grey diamonds.

5.1.3 Analysis of plots

The above plots demonstrate several phenomena that have strong implications for wrapper feature selection. First of all, some plots do show an apparent correlation between the outer training CV performance and the testing performance. The plots for Diabetes, Glass, Tic-Tac-Toe, and Wisconsin Breast Cancer, for example, fall into this category. For these datasets, it is likely that a feature selection method that selects a feature subset with good CV accuracy will allow good classification accuracy on new data. However, for many of these datasets, the classifier performance without selection is already optimal or nearly optimal in terms of classification accuracy on the outer testing set, implying that removing features is unlikely to increase classification accuracy.

Other plots show very little correlation between the subset quality seen by the algorithm and the generalization performance for the subsets. The plots for Breast Cancer and Contraceptive Method Choice fall into this category. Even a perfect feature selection algorithm that always found the subset with optimal outer training CV accuracy would be unlikely to increase classification accuracy for new data.

It is alarming that performance on one portion of the dataset may not be at all predictive of performance on another portion of the dataset, even when stratification has been performed to preserve maximal class similarity between the two portions. Similar experiments were performed for varying values of k, for LOO CV on the outer training set, for a 1:4 testing-to-training size ratio, for different random partitions of the datasets into outer training and testing sets, and for a J48 decision tree classifier (see Witten and Frank 2005). None of these modifications changed whether a dataset displayed correlation between the CV-estimated accuracy and accuracy on new data. Analysis of the number of training instances, the number of features, and the number of classes reveals that neither these characteristics nor simple relationships between them are predictive of this phenomenon. For example, Breast Cancer and Glass both have 9 features and similar numbers of instances (286 and 214, respectively), and Glass has 7 classes (making this a "harder" problem than Breast Cancer, with 2 classes), but Glass

shows a high correlation while Breast Cancer does not. Analysis of why some datasets display high correlation between CV accuracy and classification accuracy of new data and others do not is a very interesting problem, and one that would further elucidate whether and why feature selection is likely to work for a particular dataset. However, such analysis lies outside the scope of this thesis.

5.1.4 Examining correlation for larger datasets

The ability to draw meaningful conclusions from the above plots is limited by the fact that the datasets analyzed above were chosen specifically because they have few features. It is not so interesting to perform automatic feature selection on these datasets: as discussed in Chapter 2, feature selection offers the most appreciable benefits for problems where many features are available.

Therefore, tests similar to those above were performed on datasets with greater numbers of features. Instead of searching exhaustively through the set of all feature subsets, 500 subsets were randomly generated for evaluation. As above, the datasets were partitioned into outer testing and training sets of equal size using stratification. The plots in Figures 24a and 24b show the 10-fold CV accuracy on the outer training set plotted against the classification accuracy on the outer testing set.

These plots show similar behaviors to those in Figure 23. For some datasets, such as Audiology and Vote, there is a clear correlation between outer training CV accuracy and outer testing classification accuracy. Here, the set of all available features is not usually optimal, so feature selection seems to have a good chance of improving performance. For other datasets, such as Ionosphere and Spectf, there is no clear correlation between outer training CV accuracy and outer testing accuracy. Interestingly, the Sonar dataset used by Reunanen (2003) to compare forward selection and SFFS is among this group. This suggests that the problematic behavior exhibited in Figure 21 and Figure 22 may not be an artifact of these selection algorithms, but rather a problem with the Sonar dataset that would impair the success of any wrapper selection algorithm working to select features.

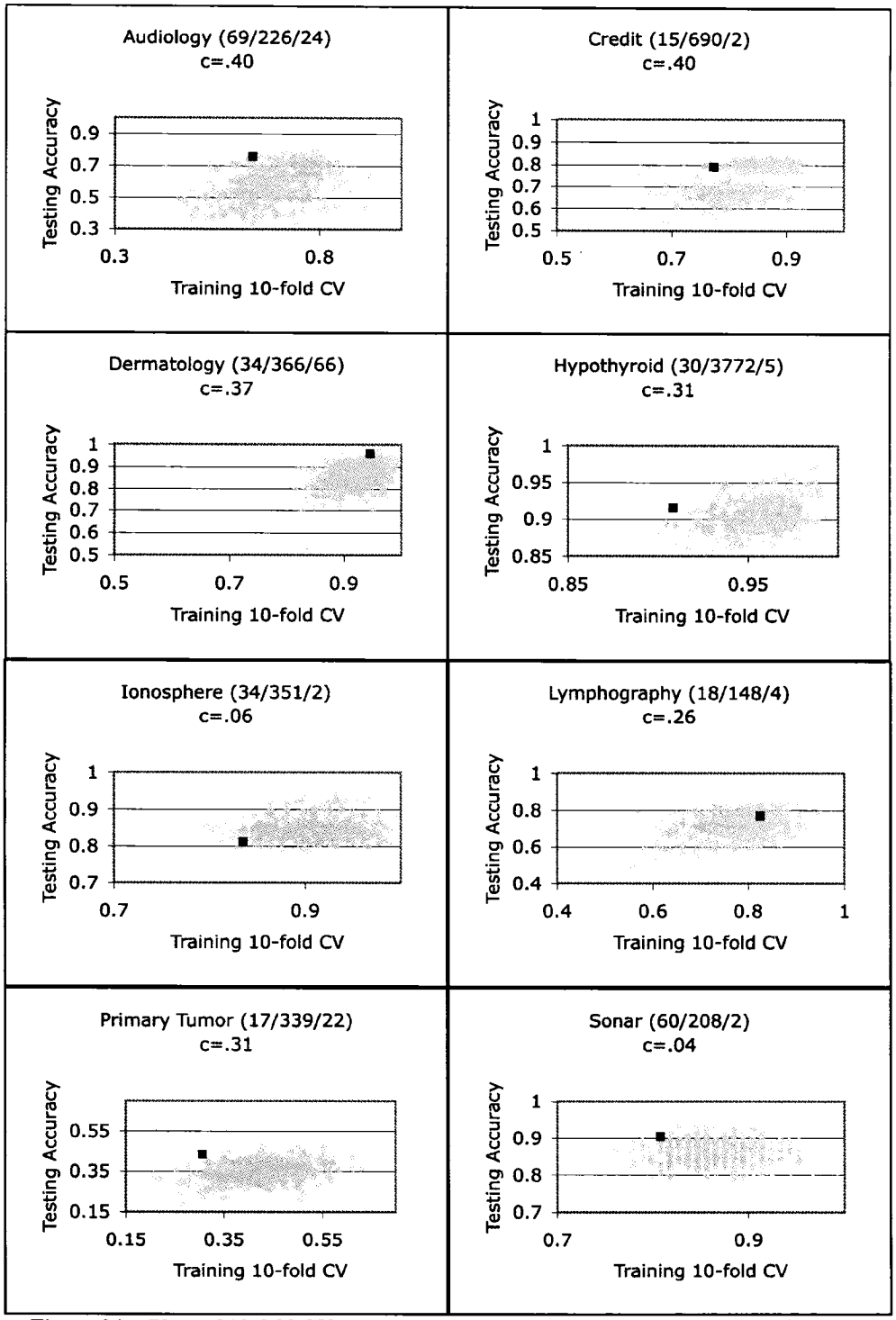

Figure 24a: Plots of 10-fold CV accuracy on the outer training set versus classification accuracy on the outer testing set, for several standard UCI datasets, for 500 randomly selected feature subsets. The number of features, instances, and classes are indicated next to the dataset name, and the correlation *c* between training and testing accuracy is indicated beneath. The set of all available features is marked with a black square, and the other subsets are marked with grey diamonds.

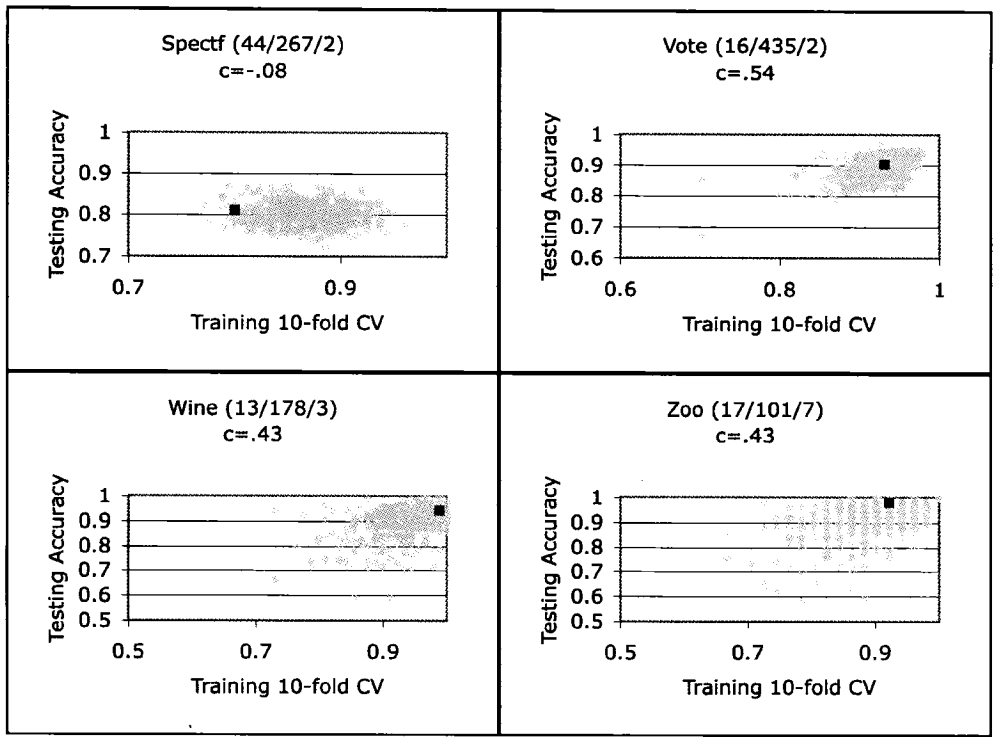

Figure 24b: Plots similar to 24a, for additional datasets.

5.2 Evaluating the performance of algorithms

The above results indicate that, at least for some of the datasets with greater numbers of features, there is a chance that a wrapper feature selection algorithm will select an optimal feature subset that leads to increased classification accuracy on new data. However, these results do not clearly imply that one feature selection method will outperform another, in general or for one of these datasets in particular. On one hand, the correlation between training performance and testing performance is often not so strong that there is a great increase (or any increase) in outer testing set performance when comparing the subsets with the absolute best and the top 10- or 20-best outer training CV scores. More intense search methods, such as SFFS or GAs, may select a feature subset with better training CV performance than a less intense method, but at no real added benefit when classifying new data in practice. The greater time required by these algorithms to perform the feature subset search is therefore wasted. On the other hand, there may be some property of a given selection algorithm that makes

it more likely to find the feature subsets that allow the best generalization performance. For example, backward elimination investigates many subsets with a great number of features, and forward selection visits many subsets with fewer features. If most of the well-performing subsets have nearly all features included, backward elimination might be a better choice.

Tests were done on several datasets (listed in Table 2) for which the above evaluation of 500 random subsets indicated that feature selection could potentially improve accuracy. Several feature selection methods were chosen for investigation. The methods include a "Monte Carlo" random search, forward selection, backward elimination, RMHC, and GAs. These methods vary quite widely in terms of implementation complexity and resource requirements, with Monte Carlo and RMHC being quite simple and fast, forward selection and backward elimination requiring simple implementations but a moderate amount of time to run, and GAs requiring an elaborate implementation and a long time to run.

For Monte Carlo search, 100 feature subsets were randomly generated and evaluated using CV performance on the outer training set. The best of these was chosen as the optimal feature subset. The performance of this algorithm serves as a baseline against which to compare the "real" search methods. Very good performance of subsets selected by this algorithm would indicate that any expense of implementing a "real" feature selection algorithm is wasteful.

RMHC was set to terminate after 100 subsets were evaluated. Initial tests suggested that training CV accuracy of the selected subset did not appreciably improve after 100 iterations. In fact, in the tests below, RMHC converged to the selected subset by the 85^{th} generation or earlier in 17 of the 25 trials.

Some parameters of the GA were fixed: for example, weighted roulette selection (a selection method built into JGAP) was used for selecting chromosomes for crossover based on their fitness values, and the GA was judged to have converged when no improvement in the fitness of the fittest chromosome occurred over five generations. The tuning of mutation rate and population size was explicitly performed. Tuning used datasets for which feature selection was

somewhat likely or very likely to improve accuracy, based on the correlation plots generated above (Audiology, Glass, and Iris), as well a version of Iris with two random features added (for which it was assumed that selection would be beneficial). For each of these datasets, population sizes of 20, 50, and 80 were used in combination with mutation rates of .05, .1, .167, .2, and .25. A kNN classifier was used in all tests, with k=1. All trials were repeated five times, each time with a new split of the data into outer training and testing sets.

For each combination of dataset, population size, and mutation rate, the following values were recorded: total number of chromosome evaluations before convergence, final outer training CV accuracy of the selected subset, and final testing accuracy of the selected subset. There were a wide variety of these values between trials, making it impossible to declare a clearly optimal combination of these parameters. However, in general, a mutation rate of .167 seemed to result in good final testing accuracy for populations of both 50 and 80 chromosomes. There seemed to be a slight increase in testing accuracy for 80 chromosomes, but the number of evaluations was nearly prohibitively high. Therefore, in the interest of the GA converging in a reasonable amount of time, a population of 50 and a mutation rate of .167 were chosen for subsequent tests in this section.

5.3 Results

Each selection method was run five times on each dataset using different random partitions of the data, using the evaluation guidelines established in Chapter 4. The partitions for each one of the five trials were the same across algorithms for a given dataset; that is, the outer training set for the first trial of forward selection for Credit was identical to the outer training set for the first trial of backward elimination for Credit, and so on. The results appear in Table 2. Figure 25 illustrates these results graphically, showing the average change in generalization performance using the feature subset selected by each algorithm for each dataset, averaged over the five trials. A positive change indicates that feature selection improved generalization performance on average. The standard deviation across trials is also indicated for the purposes of illustrating the variation due to different

executions of partitioning the data; this should not be interpreted as a conventional standard deviation measured across independent trials of the classifier, because the dataset remains the same.

Table 2: Average outer 10-fold CV accuracy (CV), outer testing accuracy (T), and number of evaluations (#), for each dataset and feature selection method.

Dataset	Monte Carlo (100 eval.)		Forward			Backward			RMHC (100 eval.)		GA			None	
	CV	T	CV	T	#	CV	T	#	CV	T	CV	T	#	CV	T
Credit	94.0	76.8	94.7	79.2	121	94.4	79.4	121	85.4	82.1	94.1	77.4	1766	79.4	80.5
Dermatology	98.1	92.1	99.6	88.9	596	99.7	94.2	596	96.2	93.0	98.9	87.9	1143	92.8	94.6
Hypothyroid	99.3	91.8	99.4	92.8	436	99.4	93.1	596	98.2	97.6	99.3	93.4	1355	91.1	92.0
Primary Tumor	68.0	40.1	74.9	39.9	154	80.4	40.5	154	43.8	38.7	66.0	39.5	1627	37.4	40.9
Vote	98.5	93.0	98.1	93.8	137	98.9	92.3	137	96.1	94.5	98.0	92.9	1241	92.7	92.0

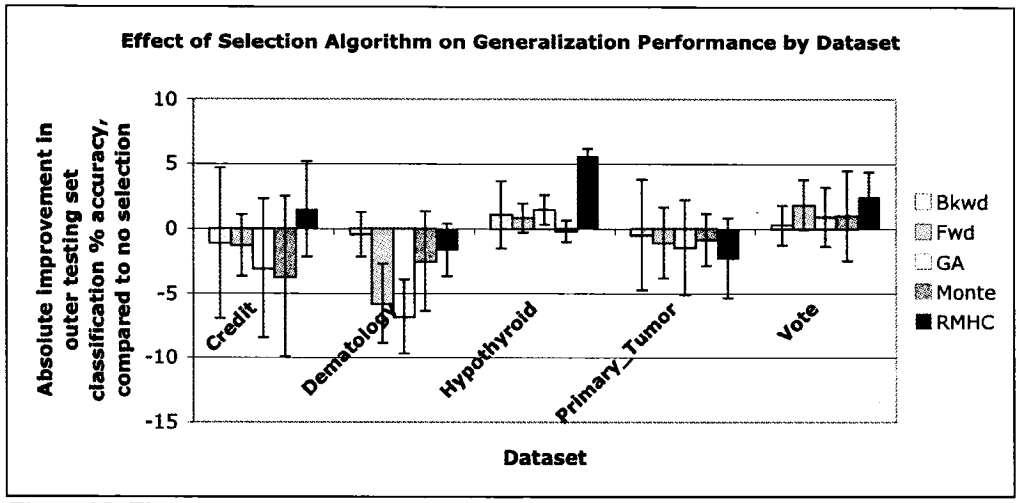

Figure 25: The average absolute change in percentage of outer testing set instances classified correctly, using the feature subset selected by various algorithms, as compared to classification accuracy without feature selection. Results are for five trials for each combination of dataset and selection algorithm. Error bars show standard deviation across the five trials.

These results show clearly that no wrapper feature selection method performs well on all—or even most—of these datasets. On average, feature selection offers no benefits for the Credit, Dermatology, or Primary Tumor datasets. On average, feature selection does tend to increase performance on the Hypothyroid and Vote datasets, but the variation across trials is quite large. Only

GA on Hypothyroid and RMHC on Hypothyroid and Vote outperform the no-selection case by more than one standard deviation.

Figure 26 shows the average change in CV performance on the outer training set compared to the CV performance without selection, for each algorithm and dataset. It is apparent from contrasting Figure 25 and Figure 26 that using only the improvement in cross-validation accuracy on the outer training set overestimates the efficacy of feature selection, in accordance with the findings of Reunanen (2004). For all datasets, all feature selection methods have found feature subsets that outperform the no-selection case by more than one standard deviation, and often the improvement is quite larger. However, Figure 25 makes clear that the ability of wrapper feature selection search algorithms to consistently find well-performing feature sets does not translate into an ability to improve performance on new data, in these cases.

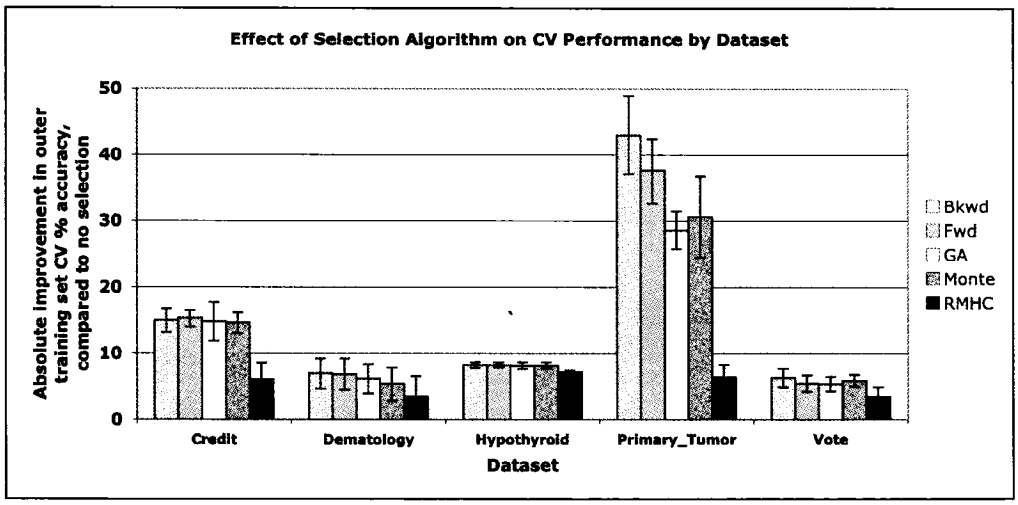

Figure 26: The average absolute change in outer training CV accuracy, using the feature subset selected by various algorithms, as compared to outer training CV accuracy without feature selection. Results are for the same trials as Figure 25.

It is not clear from these results whether any particular feature selection method is to be favored over the others, based on the generalization performance using the chosen feature set. Table 3 shows the change in outer training CV accuracy and the change in outer testing classification accuracy for each algorithm, averaged across results on all datasets. All algorithms but RMHC led

to a decrease in generalization performance on average. Very computationally intense search methods such as GAs may be ruled out for problems such as these; the high cost of running the algorithm is quite unlikely to be worth the small potential for improvement. Interestingly, RMHC has selected the feature subsets with the best generalization performance, when compared to the other algorithms, though it resulted in the least gain in outer training CV performance (even less than randomly evaluating the same number of feature subsets using Monte Carlo selection). RMHC also displayed the lowest variation in performance among trials, though as stated above, this variation was still quite high in comparison to the achieved performance gains.

Table 3: Change in CV accuracy on the outer training set (ΔCV) and change in classification accuracy on the outer testing set (ΔT) for each algorithm, compared to using all available features, averaged across all datasets.

Algorithm	ΔCV	ΔT
Backward Elimination	+15.9 ± 2.4	-0.1 ± 3.2
Forward Selection	+14.6 ± 2.1	-1.1 ± 2.3
GA	+12.6 ± 1.9	-1.8 ± 3.1
Monte Carlo	+12.9 ± 2.3	-1.2 ± 3.3
RMHC	+5.2 ± 1.9	+1.2 ± 2.2

5.4 Implications of this work

This chapter has shown that feature selection is often ineffective at improving classification accuracy on commonly used datasets, either because there is poor correlation between the measure of subset quality used by the selection algorithm and the actual quality of that subset in classifying new data, or because generalization accuracy is already near-optimal when all features are used. However, these findings are insufficient grounds for concluding that feature selection is unlikely to be useful for music classification, particularly because it is possible that many of the sets in the UCI Repository have features that experts have picked as relevant, or are from domains where acquisition of many features is too costly (e.g., medical records where there are practical limits on how much data can be acquired from a patient). This is quite different from audio classification problems, in which arbitrarily many features may be acquired and expert knowledge may be less helpful in determining which are most relevant.

This chapter has also presented empirical evidence supporting Reunanen's (2003) claim that less intense search methods may perform just as well as or better than more involved search methods for feature selection. This observation was shown to hold for those datasets where plots of CV accuracy on the outer training set against classification accuracy on a held-out outer testing set showed wrapper feature selection to have a reasonable chance of improving performance. The additional cost of running GAs, for example, is not worthwhile when analyzed against performance of simpler methods.

The work in this chapter also suggests that scatter plots, such as shown in Figures 23 and 24, may be useful as a tool for understanding the behavior of wrapper feature selection for a given problem. Much further study would be necessary to discern why there is such low correlation between outer training set CV accuracy and outer testing classification accuracy for certain datasets. However, it is clear from such plots that feature selection is unlikely to work for certain problems, where the estimated quality of feature subsets that it uses to guide its search is not predictive of generalization quality.

6 FEATURE SELECTION FOR THE GENRE PROBLEM

6.1 Introduction

The results in Chapter 5 indicate that wrapper feature selection is not guaranteed to find a feature subset that results in better classification accuracy on new data than classification using all available features. However, many of the classification problems used in the experimentation in Chapter 5 and in other papers on feature selection are different from musical genre classification: in musical genre classification, an arbitrarily large number of features may be extracted from the audio data, and domain knowledge is not necessarily helpful in choosing which features are best for classification. It therefore remains an interesting problem to discover whether feature selection may offer benefits to musical genre classification. In this chapter, feature selection is run on a musical dataset that has been commonly used for genre classification, and the results are compared to classification without feature selection and classification with an alternative dimensionality reduction technique.

6.2 Testing setup

For the following tests, a kNN classifier with k=1 was used. As discussed previously, kNN offers the benefits of being a commonly used, well-understood classifier, and it has been applied to music classification successfully in the past (e.g., Fujinaga 1998). The performance of kNN is sufficiently fast to use it in repeated training and testing cycles, such as occurs in wrapper feature selection. Furthermore, many studies on feature selection employ kNN (e.g., Reunanen 2003, 2004; Wettschereck 1997; Kohavi et al. 1997). k=1 was chosen because this is a common choice (e.g., Reunanen 2003, 2004; Kohavi et al. 1997). While there is no standard "best" value of k, it is possible that other values of k would result in more accurate classification of the genre dataset, and a value could be chosen for

the outer training data before applying feature selection. However, the goal of these tests was to evaluate the effects of feature selection rather than specify a complete optimal classification strategy.

It is also possible that another classifier, for example a support vector machine (SVM), would perform equally well as or better than kNN on this dataset. However, most other classifiers (including SVM) have more parameters than kNN that must be set, and they can be much slower to train and test. The search for a good classifier for musical genre is an interesting problem that is outside the scope of this thesis; one benefit of investigating wrapper selection is that many of the results of this work can be applied to musical genre classification using any classifier.

The data used in these tests comes from the Magnatune collection (Magnatune 2006). As discussed in Chapter 2, this collection has been previously used for genre classification, and there are published results for several algorithms on a subset of the dataset. These tests used the entire Magnatune collection available in January 2006, which included 5285 songs from 23 genres, summarized in Table 4 below. Each song in the collection is encoded in MP3 format, in stereo, with a sample rate of 44.1 kHz and a bit rate of 128 kbps. The songs were down-sampled to 16 kHz before extracting features.

Table 4: Distribution of songs in Magnatune database, by genre.

Genre	# Songs	Genre	# Songs	Genre	# Songs
Electronic	658	Metal	68	Ambient	148
Classical	2185	Pop	54	Trip-Hop	7
Rock	594	Other	24	Hard Rock	52
Celtic	41	Funk	15	Retro	14
Ethnic	691	Punk	101	Trance	9
Jazz	75	Techno	10	Folk	71
Blues	120	Acid	9	Punk Rock	37
New Age	273	Industrial	29		

The feature set extracted from the collection is summarized in Table 5 below. Fourteen standard audio features are used, and three of these (MFCC, LPC, and Method of Moments) have multiple dimensions, resulting in 39 distinct features being measured frame-by-frame. The average and standard deviation of

each of these values is calculated over all 1024-sample windows in each song, resulting in a single MP3 being represented by 78 real values. These time- and frequency-domain features range from simple (e.g., Zero Crossings) to complex (e.g., LPC) and include features that are easily interpretable musically (e.g., Strength of Strongest Beat) as well as features whose relationship to the musical signal is less transparent (e.g., Zero Crossings). The feature extraction took two days for all 5285 songs, and the stored features required 43 MB.

Table 5: Features extracted from the Magnatune collection for the genre classification problem, with the number of dimensions and the description supplied by jAudio (McEnnis et al. 2005).

Feature	#	Description
Spectral Centroid	1	The center of mass of the power spectrum.
Spectral Rolloff Point	1	The fraction of bins in the power spectrum at which 85% of the power is at lower frequencies. This is a measure of the right-skewedness of the power spectrum.
Spectral Flux	1	A measure of the amount of spectral change in a signal. Found by calculating the change in the magnitude spectrum from frame to frame.
Compactness	1	A measure of the noisiness of a signal. Found by comparing the components of a window's magnitude spectrum with the magnitude spectrum of its neighboring windows.
Spectral Variability	1	The standard deviation of the magnitude spectrum.
RMS	1	A measure of the power of a signal.
Fraction of low energy windows	1	The fraction of the last 100 windows that have an RMS less than the mean RMS of the last 100 windows. This can indicate how much of a signal is quiet relative to the rest of the signal.
Zero Crossings	1	The number of times the waveform changes sign. An indication of frequency as well as noisiness.
Strongest Beat	1	The strongest beat in a signal, in beats per minute, found by finding the strongest bin in the beat histogram.
Beat Sum	1	The sum of all entries in the beat histogram. This is a good measure of the importance of regular beats in a signal.
Strength of Strongest Beat	1	How strong the strongest beat in the beat histogram is compared to other potential beats.
MFCC	13	Mel-frequency cepstral coefficients.
LPC	10	Linear prediction coefficients calculated using autocorrelation and Levinson-Durbin recursion.
Method of Moments	5	Statistical method of moments of the magnitude spectrum.

In the tests of Chapter 5, RMHC was the best-performing feature selection algorithm for the given classification problems. Additionally, it was found to converge to a reasonable solution quite quickly (usually in fewer than 100 iterations) for a variety of problems. Therefore, RMHC was chosen to apply to the

musical genre classification problem, and 100 iterations were used, as in Chapter 5. Forward selection was also applied in order to compare with RMHC. The number of iterations of forward selection is fixed for a given number of features, and forward selection evaluates far more than 100 candidate feature sets for the given dataset (with 78 features). Therefore, if forward selection were to outperform RMHC, it could suggest that RMHC would need more iterations to converge to an acceptably good subset for this problem. On the other hand, if RMHC were to outperform forward selection, it would underscore the capability of very simple and fast feature selection algorithms to outperform more complicated approaches.

Both feature selection algorithms were evaluated according to the evaluation methodology laid out in Chapter 4. Because of the large size of the dataset, it was split into an outer testing and an outer training set with a 9:1 testing-to-training size ratio. The split used stratification to maintain class proportions. Each algorithm chose a feature subset using 10-fold CV with stratification on the outer training set, and the performance of the finally chosen subset was evaluated by training the kNN on the outer training set and then examining the classification accuracy on the outer testing set. The data was partitioned into outer training and testing sets five times, and each algorithm was run and evaluated once for each pair of outer training and outer testing set. Again, it should be noted that repeating this procedure merely illustrates the amount of variation in results on these datasets that is attributable to the particular partitioning of the dataset. The five repetitions should not be treated as "trials" from which to infer about the "expected" behavior of the algorithms, because the same dataset is used across repetitions.

6.3 Results of evaluating feature selection

6.3.1 Results using methodology of Chapter 4

Table 6 displays the results of feature selection for the Magnatune dataset for the evaluation discussed above. Both RMHC and forward selection were able

to consistently find feature subsets that resulted in better generalization accuracy than using all 78 extracted features.

Table 6: Results of feature selection, for five trials, using kNN for genre classification of the Magnatune collection. For each selection method, the table shows the percentage CV accuracy on the outer training data using the chosen subset, the classification accuracy on the outer testing data using the chosen subset, the number of subsets evaluated by the selection algorithm during its search, the time to run the selection algorithm for each trial, and the number of features in the chosen subset.

Selection Method	CV % on Outer Training	Outer Testing % Accuracy	# Evaluations	Time (min.)	# Features Selected
None	52.3 ± 1.5	54.4 ± 1.3	—	—	78
RMHC	62.8 ± 3.5	61.6 ± 2.0	100	7.4	41.8 ± 2.9
Forward	97.1 ± .3	59.0 ± 1.1	3082	217.6	28.8 ± 9.5

RMHC was able to complete its 100 iterations in fewer than 10 minutes for each of the five trials. In most of the trials, the algorithm stably converged on its finally chosen subset well before the 100th iteration, with the algorithm converging to the finally chosen subset in 60.8 iterations on average. Figure 27 shows the CV accuracy of the best subset found in each iteration, averaged across all trials; from this plot, it appears that 100 iterations is indeed a reasonable number.

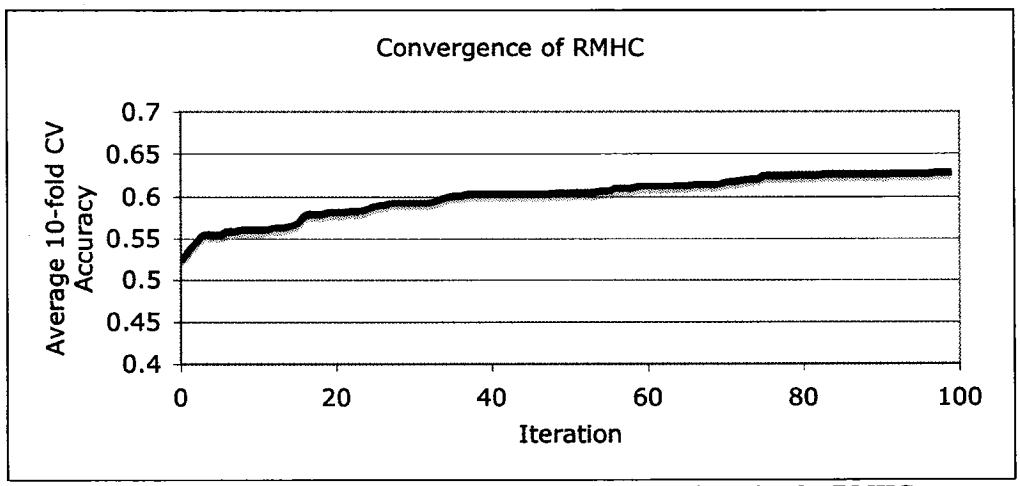

Figure 27: The average outer training CV accuracy by iteration for RMHC.

Forward selection took approximately 30 times longer to run than RMHC and did not produce better generalization performance. This is interesting given forward selection's ability to consistently find a feature subset with much higher

CV performance on the outer training data. These results suggest that it is possible to find feature subsets whose high CV performance does not correspond to high generalization accuracy, a phenomenon that was observed for many datasets in Chapter 5. Evaluation of CV accuracy on the outer training set and classification accuracy on the outer testing set for 1000 randomly generated feature subsets shows that the correlation between these values is rather poor (0.11). Figure 28 provides of a plot of the observed performance, along with marks indicating the performance with no selection, with forward selection, and with RMHC. Like most of the UCI datasets used in Chapter 5, it seems that the poor correlation of CV performance and generalization accuracy for this dataset leaves nothing to be gained in exchange for the additional expense of performing more intensive searches for feature subsets.

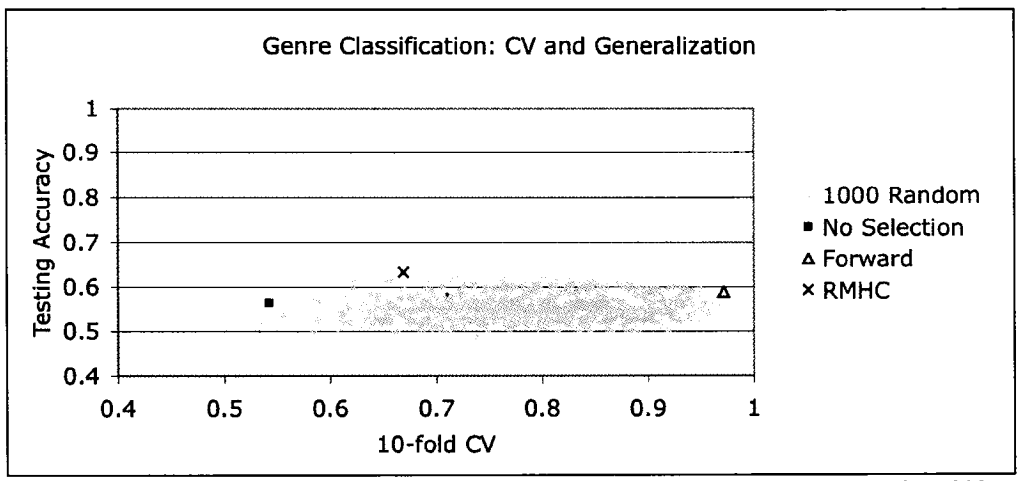

Figure 28: Outer training CV accuracy and outer testing classification accuracy for 1000 randomly chosen feature subsets. The performance using all features, the features chosen by forward selection, and the features chosen by RMHC are also indicated.

6.3.2 McNemar's test

Salzberg (1997) suggests the use of the binomial test or McNemar's test (Everitt 1977) to evaluate one classification method against another in a more statistically meaningful way than a simple comparison of classification accuracy. Because such tests compare only two algorithms to each other, they were not used in previous comparisons of multiple feature selection algorithms. However, here it is possible to evaluate whether RMHC and forward selection significantly

improve classification accuracy for the outer testing set, compared to using no feature selection.

McNemar's test uses the χ^2 distribution to test for significance of two algorithms (e.g., a kNN with all features and a kNN with a reduced feature set selected by RMHC) having the same accuracy. The statistic is computed by the following equation (Salzberg 1997):

$$\chi^2 = \frac{(|s-f|-1)^2}{s+f}.$$

Here, s and f are computed by enumerating all the instances in the outer testing set and tallying the number of instances where one algorithm classifies the instance correctly and one algorithm classifies the instance incorrectly. s is the number of these cases in which the first algorithm is correct, and f is the number in which the second algorithm is correct. (Note that the resulting numeric value is not affected by which algorithm is chosen for s and which for f.)

Because this test compares single trials of each algorithm, the results of the first trial above were used to compute the McNemar's test statistic for RMHC and no selection, RMHC and forward selection, and forward selection and no selection. In all three cases, McNemar's test shows the differences between the algorithms to be significant at $p < .01$.

6.4 Further analysis

These results imply that feature selection can improve classification accuracy for genre classification. However, further analysis compared the gain in accuracy possible with feature selection to the gain in accuracy possible with another dimensionality reduction technique. Principal components analysis (PCA) is a technique for transforming the coordinate system of the feature space of a dataset, such that each new feature is expressed as a linear combination of the original features, and it can be used for dimensionality reduction (Witten and Frank 2005, 306–9). The first axis of the new feature space is placed to correspond to the direction of maximal variance in the data, and the next axis is

placed in the next orthogonal direction with the most variance, and so on. After the variance along remaining dimensions drops below some predetermined threshold, the remaining dimensions can be discarded. This results in a reduction in dimensionality of the new feature space, compared to the original feature space. While PCA is perhaps less intuitive and less straightforward to implement than many wrapper feature selection methods, it is easy to apply in practice. It is built into Weka, and it can run in a fraction of the time of wrapper feature selection.

Testing of PCA used the same five partitions of the dataset as the feature selection tests above. PCA was applied using the outer training data to perform the dimensionality reduction. The resulting remapping into the new feature space was then applied to the outer testing data. The 10-fold CV accuracy on the (dimensionally reduced) outer training data and the classification accuracy of the (dimensionally reduced) outer testing data (after training on the outer training data) were then recorded. Table 7 shows that applying PCA in this manner boosts generalization accuracy to a similar degree as RMHC, but with less variability in accuracy and in a fraction of the runtime. Additionally, PCA results in a greater reduction of dimensionality (to 32.8 features on average, as compared to 41.8) and displays less variability in the number of features than RMHC.

Table 7: Results of dimensionality reduction using PCA, for five trials, using kNN on genre classification of the Magnatune collection, for comparison with feature selection results in Table 6.

Dimensionality Reduction Method	% CV on Outer Training	% Testing Accuracy	# Evaluations	Time (min.)	# Features Selected
PCA	59.7 ± 1.5	62.0 ± 0.5	—	0.3	32.8 ± .4

6.5 Discussion of the benefits and drawbacks of feature selection

6.5.1 Accuracy

The above results suggest that feature selection can potentially improve generalization accuracy for genre classification. Using RMHC, an increase of about seven percent in classification accuracy is possible for this dataset and feature set. However, PCA is an alternative method that requires no special infrastructure to run, that does not require any tuning of parameters, and that takes

a fraction of the time of any wrapper feature selection method studied in this thesis. This suggests that, if an increase in classification accuracy on new data is the sole goal, wrapper feature selection should not be considered as the only or best means of improving performance.

Using feature selection (or PCA), the genre classification method presented here compares modestly with the results of the most recent genre classification contest, MIREX 2005 (Downie et al. 2005). The classification results here cannot be directly compared to MIREX, which used only a subset of the Magnatune collection. Additionally, the training set used in MIREX was substantially larger than the training set used in this work (1005 songs, as compared to 529), which could account in part for higher relative generalization ability of algorithms run on the contest dataset. Nevertheless, the classification method here outperforms four of the thirteen MIREX contestants when considering raw classification accuracy. The performance of the two best MIREX algorithms (69.5% and 68.7% accuracy) still may be beyond the reach of kNN using these features, regardless of the feature selection or dimensionality reduction algorithm employed. However, it seems reasonable that employing more sophisticated features and modeling techniques like those used by the winning algorithms (e.g., aggregating feature measurements at an intermediate time scale of several seconds rather than only computing the average and standard deviation of feature values over the entire song; see Bergstra et al. 2005) could be combined with feature selection to achieve even more competitive performance.

6.5.2 Time and space

6.5.2.1 Classification time

Feature selection does offer clear benefits for classification time. To classify a new instance, a kNN classifier first calculates the distance between the new instance and all training instances in the feature space. The time for this distance calculation increases linearly with the number of features present. The classification is also influenced by the number of instances in the training set (the closest k instances must be chosen after the distance is calculated). However,

Table 8 shows that for a fixed training set size, reducing the number of features does appreciably decrease classification time. This table was created by classifying 5000 instances using a training set of size 5000, then extrapolating to the expected time needed to classify 15,000 songs (the size of a 60GB iPod) and 2,000,000 songs (the size of the iTunes database). Using feature selection or another type of dimensionality reduction can therefore make it more practical to classify a greater number of songs or to increase the size of the training set.

Table 8: Estimated classification time by dataset size and number of features, for a training set of size 5000, using a Dual 2.7GHz PowerPC G5 with 4GB RAM.

Algorithm	# Features	Minutes to classify 15,000 songs	Hours to classify 2,000,000 songs
None	78	16.3	36.1
RMHC	42	9.2	20.5
PCA	33	7.3	16.1
Forward Selection	29	6.4	14.2

6.5.2.2 Feature extraction time and feature storage space

Feature selection can also lead to reduced feature extraction time and storage space. PCA offers a reduction in storage space, but not in feature extraction time, because the same number of original features need to be calculated before the instances are remapped into the lower-dimensional feature space. Table 9 shows the approximate relationship between the number of features and the feature extraction time and feature storage space. Because of the variation in extraction time between different types of features, the feature extraction time in practice will vary according to the types of features selected. Most of the time spent by jAudio to extract features is in fact spent on decoding the MP3 files, a step that takes the same amount of time regardless of the number of features extracted. (Also, it is likely that large collections such as iTunes already have decoded versions of all songs, so this step is not necessary.) Therefore, to highlight the difference feature selection can make, Table 9 shows only the time *saved.*

Table 9: Estimated extraction time savings and absolute storage space by dataset size and number of features. Extraction time is based on the same PowerPC as Table 8 and assumes that the removed features take the same time to compute, on average, as the features that are kept. It is also assumed that the song lengths are equal, on average, to the songs in the Magnatune dataset.

Algorithm	# Features	15,000 songs		2,000,000 songs	
		Hours saved to extract features	Space to store features	Days saved to extract features	Space to store features
None	78	0	62.6 MB	0	8.2 GB
RMHC	42	7.11	48.4 MB	39.5	6.3 GB
PCA	33	0	44.8 MB	0	5.8 GB
Forward Selection	29	9.67	43.2 MB	53.7	5.6 GB

6.5.2.3 Cost of performing selection

The reduction in feature extraction time and classification time should be considered in relation to the cost of the time needed to run feature selection. As discussed in Chapter 5, the time to run feature selection is highly dependent on the selection algorithm. GAs, for example, may require several thousand feature subset evaluations before convergence. However, the work in this thesis has shown very simple methods for feature selection to work more effectively than more intensive methods, and RMHC offers good performance with just 100 evaluations. Using a kNN classifier and a training set of the size used in the experiments in this chapter (529 songs), feature selection can run in under 10 minutes. Even for an outer training set of 5000 songs, feature selection using 100 iterations of RMHC can run in a few hours. The cost of performing feature selection is therefore generally outweighed by the potential savings in feature extraction and classification time for datasets of even 15,000 songs. Of course, feature selection's cost would vary if a different classifier were used. Additionally, PCA takes very little time to perform while offering a similar reduction in classification time as feature selection.

6.5.3 Other benefits

Feature selection can lend insights into the relationship between audio features and musical categories (such as genre labels). If a subset of features leads to very good generalization performance, this suggests that these features are highly informative of the class to be learned. (One cannot draw the inverse

conclusion, that an excluded feature is irrelevant, because such a feature may be redundant with a feature included in the chosen subset.) McKay and Fujinaga (2005) used their finding that features relating to instrumentation are highly beneficial in genre classification of MIDI to motivate discussion of the relationship of timbre and instrumentation to genre. One could similarly explore the features selected for discriminating among particular genres for audio classification; for example, if MFCCs are consistently included in feature subsets that lead to good genre classification of new songs, this suggests that the timbral information captured in this feature is relevant to human genre labels.

7 A NEW, RELATED APPROACH TO OPTIMIZATION OF A CLASSIFICATION-BASED MUSIC MANAGEMENT SYSTEM

7.1 Introduction

This chapter describes work performed at Sun Microsystems during a three-month research internship working on the Search Inside the Music (SITM) project. An initial goal of the internship was to apply feature selection to improve the accuracy of the genre classifier driving the SITM music collection visualization and playlist generation system. Preliminary work revealed that feature selection and other classification optimization techniques were able to improve classification accuracy, but higher classification accuracy did not necessarily translate into subjectively better performance of the overall system. Subsequent work focused on the development of a new optimization technique that works similarly to wrapper feature selection, that aims to maximize an objective evaluation function while searching through dimension subsets, but that operates on a "similarity space" instead of a feature space. Experimentation revealed that this approach can potentially improve subjective performance, and do so in a manner that scales well with collection size.

7.2 The original system

The architecture of the original SITM system appears in Figure 29. This system is discussed in detail in West and Lamere (forthcoming); only an overview of its functionality appears here, as the author did not contribute to design or implementation of the original system. The first component of this system is a feature extractor that extracts audio features from a collection of songs stored as MP3 files. A classifier is then trained to perform either genre or artist classification using some subset of these songs. Then, the trained classifier classifies all songs in the collection.

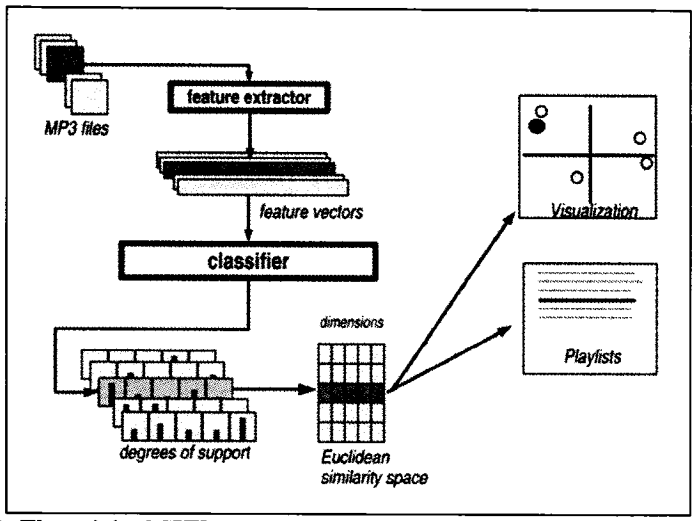

Figure 29: The original SITM system for music collection visualization and playlist generation.

The classifier component differs slightly from classifiers discussed thus far in this thesis, in that the output of classifying a new song n is not a single class label, but a vector of degrees of support $\mathbf{s}^{(n)} = \langle d_1^{(n)}, d_2^{(n)}, \ldots, d_C^{(n)} \rangle$ of length C, the number of available class labels. For example, if the classifier is trained to learn genre from a training set containing the genres "Rock," "Jazz," and "Classical," then the output of the classifier will be a vector $\mathbf{s}^{(n)} = \langle d_{Rock}^{(n)}, d_{Jazz}^{(n)}, d_{Classical}^{(n)} \rangle$, where $d_{Rock}^{(n)}$, for example, indicates the support for the class "Rock." Each vector element is interpretable as the relative support for the claim that the classified song belongs to the corresponding genre. For example, a reasonable classifier output for a jazz/rock fusion piece might look like ⟨1.3, 1.1, .20⟩. (In West and Lamere's original system, the degrees of support are constrained to sum to 1, so they are interpretable as probabilities of membership in each genre.)

One may use the C degrees of support for each song as coordinates mapping the song to a point in a C-dimensional space $\boldsymbol{S}=\mathbb{R}^C$, where each dimension corresponds to a class label of the training set (such as "Rock"). Based on the assumption that similar-sounding songs will be near each other in this space, and different-sounding songs will be further from each other, one can

compute dissimilarity between two songs n and m using a distance metric in this C-dimensional "similarity space":

$$D(n,m) = \left|\mathbf{s}^{(n)} - \mathbf{s}^{(m)}\right| = \sqrt{\sum_{i=1}^{C} \left|d_i^{(n)} - d_i^{(m)}\right|^2}.$$

This objective measure of musical dissimilarity can then be used to generate playlists of the k most similar (least dissimilar) songs to a selected song, by finding the k nearest neighbors in the similarity space. Multidimensional scaling (Kruskal 1964) can be used to project the similarity space onto two or three dimensions, for the purposes of displaying a two- or three-dimensional visualization of the music collection in which similar songs appear close together (one such visualization appears in Figure 30). A user can interact with the collection via this visualization by clicking on a point to play the corresponding song.

Note that this use of MDS is fundamentally different from other common uses in other fields, such as psychology. For example, the algorithm's positioning of dimensional axes is ultimately meaningless beyond the extent that it facilitates visualization of the data. Other techniques could be used to project the data into a visualization space, but the author of this thesis was not involved in the construction of the original system and the choice to use this particular technique; therefore a discussion of other candidate techniques is not relevant here.

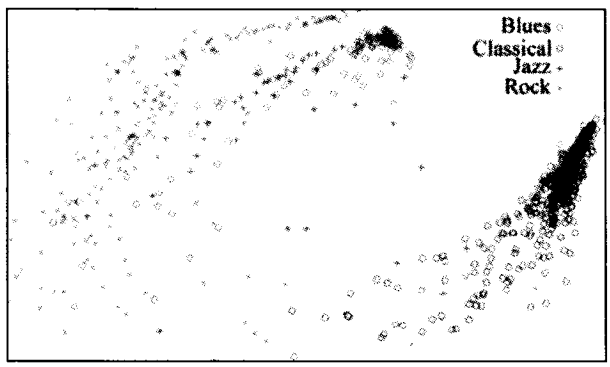

Figure 30: A visualization produced by the original SITM system using a CART genre classifier with 64.7% accuracy, for four Magnatune genres.

West and Lamere's system is not the first system to use objective, distance-based measures of similarity in this way. Logan and Salomon (2001), for example, compute similarity between two songs by applying a distance metric to clustered MFCC features. Aucouturier and Pachet (2002) model MFCC features as a mixture of Gaussians and compute the similarity by applying a distance metric between the Gaussians. Both groups of researchers use these similarity measures to create playlists of similar songs. According to West and Lamere, a drawback of previous approaches is that computing distance metrics between representations of songs' features results in timbrally alike (but otherwise very different) songs being identified erroneously as similar. For example, a classical lute piece is timbrally similar to an acoustic guitar rock song, but listeners would likely not identify such pieces as similar overall because of the stark difference in genre. West and Lamere claim that the classifier component of their system, which can be trained to discriminate between genres, can circumvent this type of problem and therefore produce playlists and visualizations based on a more natural concept of similarity.

7.3 Initial optimization experiments

The focus of the internship was to improve on the subjective quality of playlists and visualizations generated by the existing system outlined above. An initial goal was to investigate ways to improve the classification accuracy of the classifier component of the original SITM system described above, based on the assumption that improving classification accuracy would likely improve the quality of the playlists and visualizations. One variant of the original SITM system employed a Fisher's Linear Discriminant Analysis (LDA) classifier, which classified a subset of the Magnatune collection with 49.6% accuracy, and another variant employed a Classification and Regression Tree (CART) classifier, which obtained 64.7% accuracy. Results of the MIREX 2005 genre contest (Downie) suggested that higher accuracy was possible.

Initial optimization experiments involved varying the type of features, varying the size of the training set, experimenting with different classification

algorithms and different classifier parameters, and applying feature selection and PCA for various combinations of feature sets and classifiers. This experimentation was performed using the Autonomous Classification Engine (McKay et al. 2005). Table 10 provides an overview of the classification accuracy scores resulting from a few of these experiments.

Table 10: 10-fold CV accuracy for genre classification of a subset of the Magnatune collection, obtained in initial optimization experiments.

Classifier	Classifier Configuration	Dimensionality Reduction	10-fold CV
kNN	k=1, Euclidean distance metric	None	70.2
kNN	k=1, Mahalanobis distance metric (see Duda et al. 2001)	None	19.6
SVM	linear kernel	None	67.3
SVM	polynomial kernel, p=3	None	71.9
Naïve Bayes	Gaussian	None	57.3
kNN	k=1, Euclidean distance metric	Feature selection	73.2
SVM	linear kernel	Feature selection	63.6
SVM	polynomial kernel, p=3	Feature Selection	69.8
kNN	k=1, Euclidean distance metric	PCA	66.9
SVM	linear kernel	PCA	68.1
SVM	polynomial kernel, p=3	PCA	66.9
CART		None	64.7
LDA		None	49.6

In general, other results were consistent with those in Table 10, in that feature selection and PCA offered benefits for some classifiers and classifier configurations, and not for others. kNN and SVM classifiers typically performed well with certain parameters; for example, the Euclidean distance metric for kNN and the polynomial kernel with an exponent between 2 and 6 for SVM consistently led to high classification accuracy. Other classifiers (e.g., neural networks and decision trees) either tended to have lower classification accuracy or take so long to train and test that they were not practical for repeated experiments. Varying the types of features and training set size did impact classification accuracy to a degree, but discussion of these experiments lies outside the scope of this thesis.

Results of these initial experiments indicated that several approaches to improving the accuracy of the original classifier component were effective. However, surprisingly, subjective evaluation of the visualizations and playlists

created from the similarity spaces produced by the "optimized" classifier component revealed that they were typically not superior to those produced by the original system using the LDA and CART classifiers.

For example, one classifier that had very high accuracy was an SVM with a polynomial kernel (polynomial degree 3). This classifier obtained 71.9% accuracy, but the visualizations generated from the similarity space created by this classifier were consistently poor (see Figure 31). The SVM was so accurate at identifying the genre of each piece in the collection that it typically assigned a very high degree of support for the correct genre and negligible degrees of support to all other genres. This resulted in pieces from a genre being very tightly clustered together, both in the similarity space and in the two-dimensional MDS projection of the similarity space.

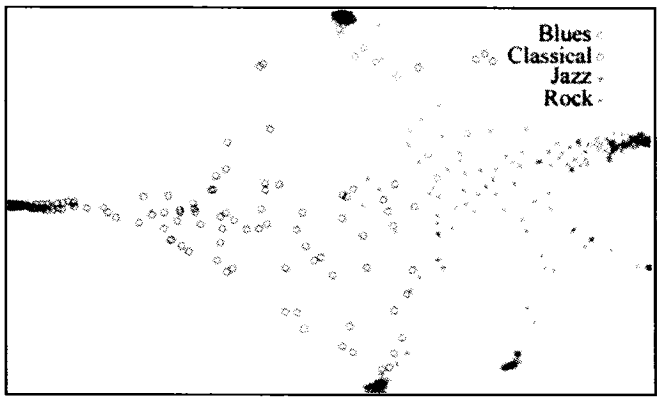

Figure 31: A visualization produced by a modified system using an SVM genre classifier with 71.9% accuracy, for four Magnatune genres.

One negative effect of this clustering is that there is no organization within genres. In the similarity space generated by the original CART classifier, similar pieces within a genre tend to cluster together; for example, many classical orchestral pieces may inhabit one part of the similarity space, and many classical vocal pieces may inhabit another part of the similarity space (though both remain a part of a larger cluster of classical music). It is reasonable that, in classifying a classical vocal piece, for example, the CART classifier may output moderate degrees of support for vocal-heavy genres such as folk, pulling these pieces

toward a different area of the similarity space than non-vocal classical music. In contrast, for most classical pieces, the SVM classifier assigns equally low degrees of support for all non-classical genres. The resulting lack of organization prohibits the visualization from being useful for browsing within a genre.

Another negative effect of this clustering is that there is no organization between genres. In the similarity space generated by the original CART classifier, the jazz pieces tend to be closer to both the blues and rock pieces than to the classical pieces, and this relationship is also apparent in the two-dimensional visualization of Figure 30. However, there is no such order in the SVM-produced similarity space, nor in the visualization of this space; the pieces of a given genre tend to be equally far in the similarity space from the pieces of all other genres. This is explainable by the fact that for a jazz piece, for example, the CART classifier tends to output a high degree of support for jazz, moderate degrees of support for rock and blues, and a low degree of support for classical. The SVM classifier outputs only a high degree of support for jazz, and all other degrees of support are minimal. The SVM visualization is less informative and useful than the CART visualization, because it is incapable of suggesting meaningful relationships between genres.

The tight clustering in the SVM-produced similarity space also negatively impacts playlist generation. Due to the highly accurate nature of the classifier, playlists produced from this space tend to be uniform in genre. However, pieces on these playlists are not necessarily similar otherwise, due to the lack of organization within genre clusters. Table 11 shows an example playlist generated from the CART similarity space, and Table 12 shows an example playlist generated from the SVM similarity space using the same query song. The CART playlist is somewhat cohesive in style and instrumentation, but the SVM playlist is cohesive only in terms of genre.

Applying feature selection and PCA tended not to dramatically change the subjective quality of the visualizations and playlists generated from a similarity space created by a given classifier. Changing the classifier (e.g., kNN versus SVM) or classification problem (e.g., artist versus genre) used to produce the

similarity space tended to have more striking effects on the subjective nature of the playlists and visualizations.

Table 11: Playlist produced using a similarity space created by CART (64.7% accuracy).

#	Track	Artist	Style
1	Vivaldi: Laudate Dominum, RV 600 (Track 12)	La Serenissima	Virtuoso Baroque violin sonatas
2	Uccellini: La Tarantola	Altri Stromenti	17th century Baroque ensemble
3	Mozart: Oboe Quartet in F after K496 / III. Allegretto	American Baroque	Baroque and Classical chamber music
4	Vivaldi: Laudate Dominum, RV 600 (Track 9)	La Serenissima	Virtuoso Baroque violin sonatas
5	Vivaldi: Concerto in F, RV 292/ I. Allegro – adagio – allegro - adagio	La Serenissima	Virtuoso Baroque violin sonatas
6	Vivaldi: Concerto No. 4 in g minor, RV297	American Baroque	Baroque and Classical chamber music
7	Vivaldi: Concerto in F, RV 292 / II. Allegro	La Serenissima	Virtuoso Baroque violin sonatas
8	Mozart: Oboe Quartet in F after K496 / II. Andante	American Baroque	Baroque and Classical chamber music
9	Osward: Gavotta	Da Camera	Celtic Renaissance/Baroque ensemble
10	J.S. Bach: BWV 1041 / II. Andante	Lara St. John	Baroque violin concerto

Table 12: Playlist produced using a similarity space created by SVM (71.9% accuracy).

#	Track	Artist	Style
1	Vivaldi: Laudate Dominum, RV 600 (Track 12)	La Serenissima	Virtuoso Baroque violin sonatas
2	J.S. Bach: Concerto in G Op. 7 No. 6	Sonnerie	Baroque chamber music
3	Diletsky: Came into the church	Kyiv Chamber Choir	Chants of the Ukrainian Orthodox Church
4	Chambonnieres: Suite in g minor	Hanneke van Proosdij	Baroque harpsichord
5	Vivaldi: Laudate Dominum, RV 600 (Track 5)	La Serenissima	Virtuoso Baroque violin sonatas
6	Crecquillon: Bakfark un gay bergier	Jacob Heringman	Renaissance lute
7	Milan: Pavana (no.1)	Jacob Heringman	Renaissance lute
8	Chambonnieres: Suite in C major	Hanneke van Proosdij	Baroque harpsichord
9	Corelli: Trio Sonata Op. 2 No. 1	Brook Street Band	Baroque ensemble
10	Milan: Fantasia 26	Jacob Heringman and Catherine King	Renaissance songs

Ultimately, however, a thorough comparison of over sixty similarity spaces produced using different methods revealed that the classification accuracy

was not at all predictive of the subjective quality of playlists and visualizations. It was therefore decided not to seek further improvement of classifier accuracy via feature selection, classifier tuning, or other means.

7.4 Experiments with a new, related optimization technique

As discussed in previous chapters, wrapper feature selection methods are essentially search methods that use the quality of feature subsets, as determined by a "black-box" evaluator, to guide the search to better subsets. The ability of feature selection algorithms to ultimately select subsets that have very good performance as measured by this evaluator (that is, subsets with very good CV performance on the outer training set) is undisputed by the literature and by the testing performed in this thesis. Rather, the primary problems with feature selection are that the CV performance on the outer training set may not be highly predictive of generalization accuracy obtained using the feature set (as discussed in Chapters 5 and 6), and that higher classification accuracy may not lead to better playlists or visualizations in the context of a real system such as in the SITM project.

A new technique for optimizing the SITM system was therefore created to harness the effective search capability of wrapper feature selection algorithms in a way that seemed likely to have an appreciable impact on the subjective quality of the SITM playlists and visualizations. Instead of applying feature selection to reduce the dimensionality of the feature space used by the classifier, a feature-selection-like algorithm was applied to reduce the dimensionality of the *similarity space* used for playlist generation and visualization. Considering the resemblance of the similarity space to the feature space of a kNN classifier (e.g., playlists of size k are created from the k nearest neighbors in the similarity space), applying dimensionality reduction to optimize the similarity space seemed to be a sensible pursuit. Such a technique necessitated that the black-box dimension subset evaluator consist not of the CV classifier evaluator used for feature selection, but of an evaluator capable of assigning subjectively-relevant scores to subsets of similarity space dimensions.

7.4.1 Objective evaluation metric for similarity spaces

The creation of an objective evaluation metric for similarity spaces (or for similarity-based playlists or visualizations) is not a trivial problem, given that there is no available ground truth denoting the "true" similarity between pairs of songs. Obtaining values to serve as such a ground truth would require extensive testing with human users, and such testing would involve a host of practical problems (e.g., many subjects would be needed to feasibly obtain similarity scores for all pairs of pieces in a collection) as well as theoretical issues (e.g., it may not be reasonable to assume that similarity can be measured as a listener- and context-independent property).

Previous work on similarity-based systems (e.g., Logan and Salomon 2001; Pampalk et al. 2003; Aucouturier and Pachet 2002) has often used genre, artist, or album metadata as a substitute for similarity ground truth, based on the assumption that songs from the same genre (or artist, or album) are more likely to be similar from songs from different genres (or artists, or albums). Based on this work (particularly on the metric used in Pampalk et al. 2003), the following evaluation metrics for similarity spaces were defined:

$$\texttt{genreScore} = \frac{\texttt{avgGenreDistance}}{\texttt{avgTotalDistance}}$$

$$\texttt{artistScore} = \frac{\texttt{avgArtistDistance}}{\texttt{avgTotalDistance}}$$

$$\texttt{albumScore} = \frac{\texttt{avgAlbumDistance}}{\texttt{avgTotalDistance}}.$$

`avgGenreDistance` and `avgTotalDistance` are defined below, where N is the number of songs in the collection, and $D(i, j)$ is the distance-based dissimilarity metric defined above. `avgArtistDistance`, and `avgAlbumDistance` are defined similarly to `avgGenreDistance`.

$$\texttt{avgGenreDistance} = \frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{N} G(i,j)D(i,j)$$

$$G(i,j) = \begin{cases} 1, & \text{Genre}_i = \text{Genre}_j \\ 0, & \text{otherwise} \end{cases}$$

$$M = \sum_{i=1}^{N}\sum_{j=1}^{N} G(i,j)$$

$$\texttt{avgTotalDistance} = \frac{1}{N(N+1)/2}\sum_{i=1}^{N}\sum_{j=1}^{N} D(i,j).$$

While the above measures make intuitive sense, they all have the drawback of highly rating similarity spaces in which all songs from a single genre are very tightly clustered, such as the similarity space produced by the SVM discussed above. To provide some measure of the intra-genre organization apparent in subjectively better spaces, such as the CART-produced space, the following measure was also defined:

$$\texttt{artistGenreRatio} = \frac{\texttt{avgArtistDistance}}{\texttt{avgGenreDistance}}.$$

This measure produces better scores for similarity spaces where artists are more tightly clustered than genres, which is indicative of a degree of organization within each genre cluster.

It was observed that several of the existing similarity spaces performed reasonably well on the above metrics but still produced subjectively bad playlists containing songs from very different genres or artists than the query song. Therefore, two additional metrics were defined. `genre%N` was defined as the percentage of the N nearest neighbors to a query song that share its genre, using each song in the collection as a query and taking the average. `artist%N` was defined analogously. N=20 was used in the evaluation of the similarity spaces, because twenty songs seems like a reasonable length for a playlist, but other values could certainly be used.

All of the above metrics were computed for the sixty-some similarity spaces generated in the initial optimization experiments described above. Comparison of these scores with subjective performance of the playlists and visualizations produced from these spaces revealed that, in general, the best spaces performed well on all of these measures, and the worst spaces performed poorly on one or more of these measures. However, the relationship between

these metrics and the subjective performance was sufficiently complex that no simple combination of metrics was highly predictive of subjective performance. Therefore, a search was undertaken for a combination of metrics was merely able to score the five best existing similarity spaces very well, the five worst similarity spaces very poorly, and the other similarity spaces somewhere in between. The similarity space evaluation function F that uses the combination that best satisfies these criteria appears below:

$$\begin{aligned} F(\text{space}) = .01^2(.7 \times \texttt{genre\%20}^2 + .3 \times \texttt{artist\%20}^2) \\ -.25(\texttt{genreScore}^3 + \texttt{artistScore}^3 \\ + \texttt{albumScore}^3 + \texttt{artistGenreRatio}^3) + 1 \end{aligned}$$

When applied to new similarity spaces, generated by classification algorithms not used to create any of the similarity spaces employed in the tuning of F, this function was generally not predictive of subjective performance.

That such a complex expression is necessary to even crudely approximate subjective quality, and that it still fails to apply well to new models, is not surprising. The individual metrics combined in this expression are quite coarse, and they no doubt fail to describe many meaningful dimensions of visualization and playlist quality. The design of a general measure of playlist or visualization quality would benefit much MIR research (for example, it could be used in MIREX to objectively compare similarity-based systems produced by different researchers), but it is too complex a task for this work. The following work using F proceeded with the assumption that, so long as it was used only as an evaluator for similarity space dimension selection, it needed to be only roughly applicable to the similarity spaces already constructed and to spaces very similar to these spaces (i.e., these spaces with dimensions removed).

7.4.2 Applying dimension selection to similarity spaces

Because of forward selection's ability to find feature subsets with very high evaluation scores (measured by CV performance on the outer training data), and because of its simplicity of implementation, it was used as the algorithm to search for similarity space subsets with the best evaluation scores (measured by

F). The first set of experiments applied forward selection to two of the best similarity spaces to examine whether forward selection was in fact able to find a subset of dimensions with a higher value of *F* than the original spaces with all dimensions, and whether these lower-dimensional spaces were indeed subjectively superior. The two similarity spaces used were the space CART-Genre, created by applying the original CART classifier to a subset of Magnatune including songs from 10 genres, and the space CART-Genre-Artist-Combo, created by combining the dimensions of CART-Genre with the dimensions of the similarity space created by training a CART classifier to classify artists on a subset of Magnatune with 74 artists.

Table 13 shows the results of these experiments. Forward selection on CART-Genre was able to find a dimension subset with a slightly higher score than the original similarity space. Subjectively, this space was quite similar to the original space. Forward selection on CART-Genre-Artist-Combo was able to find a dimension subset with a much higher score (higher than any of the previously-created similarity spaces) and many fewer dimensions than the original space.

Because of the ability of PCA to improve classification accuracy to a degree similar to that possible with feature selection, and because of its speed of application, further experiments applied PCA to the same two original similarity spaces. PCA was indeed able to considerably reduce the number of dimensions of the similarity space, and it was also able to increase the score of the CART-Genre-Artist-Combo space, though not to the same degree as forward selection. There was a marked improvement in the visualization quality when PCA was applied to the previously 84-dimensional CART-Genre-Artist-Combo space, perhaps because of the decreased load on the MDS algorithm used for projecting the space onto two dimensions.

Table 13: Results of applying forward selection and PCA to two of the original similarity spaces.

Model	# Dimensions	*F* evaluation score
CART-Genre	10	1.23
fwdSel-CART-Genre	9	1.24
PCA-CART-Genre	8	1.22
CART-Genre-Artist-Combo	84	1.00
fwdSel-CART-Genre-Artist-Combo	27	1.28
PCA-CART-Genre-Artist-COMBO	15	1.20

The above tests demonstrate that both forward selection and PCA can potentially improve the subjective quality of similarity spaces while reducing the number of dimensions needed to represent the space. If forward selection or PCA is applied to a similarity space containing all songs in a collection, there is no analog to the concern for generalization ability that has presented such problems for wrapper feature selection application and evaluation. That is, if no new songs are to be added after dimensionality reduction is applied, it is irrelevant whether the dimensionality reduction algorithm has produced a dimension subset that harms playlist and visualization quality when new songs are added.

However, it would be convenient if dimensionality reduction could be applied to a similarity space containing only a subset of songs from the collection. For forward selection, this would speed up the evaluation of each subset (calculation of `genreScore`, `artistScore`, and `albumScore` scale polynomially with the collection size) and therefore allow the algorithm to run in less time. Additionally, it would be convenient if forward selection or PCA did not have to be rerun each time new music was added to a collection. For these reasons, further testing examined the effects of adding new data to the similarity spaces after applying forward selection and PCA.

Table 14 shows the results of applying forward selection and PCA to a similarity space containing only a subset of the collection. The similarity space used was produced by a kNN classifier, where k=20. In Test 1, the classifier was trained on 50% of the songs in the collection (selected so that each genre of the collection was proportionately represented). The degrees of support for these songs were output to create a similarity space, which was then evaluated using *F*.

In Test 2, the classifier trained in Test 1 was used to output degrees of support for the other 50% of the songs in the collection; the degrees of support were added to the similarity space including the first 50% of the songs, and this space was evaluated. The evaluation of this space shows that adding new songs (songs not used to train the classifier) to the similarity space decreases the quality of the space, according to *F*.

In Test 3, forward selection was applied to the similarity space containing only 50% of the songs (the space created by Test 1), then the rest of the songs were added to the space. In Test 4, forward selection was applied to the similarity space containing all songs (the space created by Test 2). Tests 3 and 4 demonstrate that forward selection is capable of improving the similarity space produced by this classifier (the scores of the new spaces of 1.07 and 1.08 are somewhat better than the score of 1.03). Also, the scores for Tests 3 and 4 are similar to each other, implying that applying forward selection on a space containing a subset of the collection has effects similar to applying forward selection on a space containing the whole collection. Subjective evaluation of the playlists and visualizations generated from the similarity spaces produced by Tests 3 and 4 revealed them to be indistinguishable in terms of quality. Furthermore, applying forward selection to a space containing a subset of the collection allows selection to run in significantly less time than is required for applying selection to a space containing the entire collection.

Table 14: Effect of applying forward selection and PCA on a subset of the collection rather than the whole collection.

Test	Method	Selection Time	Score
1	Similarity space created by classifying 50% of songs	-	1.09
2	Similarity space of Test 1, with other 50% of songs added	-	1.03
3	Forward selection applied to similarity space produced in Test 1, then other 50% of songs added	1.01 hours	1.07
4	Forward selection applied to similarity space produced in Test 2	3.75 hours	1.08
5	PCA applied to similarity space produced in Test 1, then other 50% of songs added	-	.94
6	PCA applied to similarity space produced in Test 2	-	.93

Tests 5 and 6 demonstrate that applying PCA to the similarity space output by this classifier is not beneficial. However, as is the case with forward selection,

the quality of the space is not much influenced by whether PCA is applied to a space containing a subset of the collection or the entire collection.

In summary, the above testing implies that dimensionality reduction can be performed using a subset of the collection instead of the whole collection. This can allow forward selection to run more quickly, and it can allow one to add new songs to the dimensionally reduced similarity space without fear that the quality of the space will significantly degrade. However, adding new *kinds* of music (for example, new genres) may present a problem; it is likely that dimensionally reduced similarity space will not be appropriate for representation of these new songs.

7.5 Conclusions

Initial work performed during a Sun Microsystems internship found that improving classification accuracy does not necessarily translate into better usability of Sun's SITM music management system. This may not be the case for all systems in MIR that employ classifiers to assist in managing music based on similarity or other properties; it is possible that other systems may benefit more concretely from feature selection and other classification optimization techniques. However, for the SITM project, an approach similar to wrapper feature selection that operated on the similarity space itself showed the most potential for improving system usability.

Applying forward selection to reduce the dimensionality of the similarity space used in the SITM system has been shown to be effective at improving the quality of the playlists and visualizations. The benefits are apparent even when using a heuristically created evaluation metric for similarity spaces. PCA can also be applied to the similarity spaces, particularly to improve visualization quality. Additionally, both forward selection and PCA can significantly reduce the size of the similarity space representations.

It is hoped that this work offers a useful tool to other MIR researchers. Dimensionality reduction could be applied in this manner to any space in which similarity is calculated as a Euclidean distance, regardless of how this space is

generated. Additionally, it is hoped that this work will motivate ongoing discussion on the problem of objectively evaluating systems that use a notion of musical similarity. Further work that produces a better objective evaluation metric than the F function above would be useful for better optimizing similarity spaces using forward selection or other techniques, and it would also allow direct comparisons of similarity systems produced by different means (e.g., by different research groups competing in a MIREX-like similarity contest).

8 CONCLUSIONS

8.1 Objectives accomplished

The work presented in this thesis has involved a thorough investigation of feature selection as an optimization method for musical genre classification. Specific objectives of this work included the presentation of the rationale for investigating feature selection in the context of music classification, the specification of an appropriate methodology for evaluating feature selection, the application of this methodology to evaluate the general efficacy of feature selection and the appropriateness of various existing feature selection algorithms, the application of an appropriate feature selection algorithm to the genre classification problem, and a deepened understanding of the potential for feature selection to be of use in MIR. Each of these objectives has been met.

This work has been motivated by the popularity and usefulness of audio classification in MIR, as well as the existing machine-learning literature on classification and feature selection. This thesis has begun by thoroughly explaining these motivations and putting the current work into context with a discussion of the relevant background in classification, music classification, and feature selection.

After a discussion of the tools used and created for this work, this thesis has presented a critical overview of the literature discussing evaluation practices for feature selection. Aspects of the evaluation process for feature selection for which the literature provides contradictory or insufficient guidance have been identified, and original testing has been conducted to attempt to understand these aspects more thoroughly. An outcome of this work is a specification of a more reasoned and thorough strategy for applying and evaluating feature selection.

This evaluation strategy has then been employed in an exploration of the general questions of whether feature selection is likely to be effective in increasing classification accuracy, and of which algorithms may be most practical and appropriate. This work has identified the problem that, for some datasets

commonly used in the machine learning community, performance on one representative subset of the data is not predictive of performance on another representative subset. This property suggests that, for such datasets, feature selection is not likely to be effective in increasing classification accuracy. This work has also shown that, for datasets without this property, the correlation between the performance measure used by feature selection and the generalization accuracy is still often weak, and/or the set of all available features is likely to result in near-optimal generalization accuracy. In these cases, feature selection may therefore fail to work, or else fail to result in remarkable improvements in accuracy. These findings support the claim of one machine-learning researcher that more intensive search algorithms for feature selection tend not to provide any advantage over less intensive algorithms, despite the increased time needed for them to run. Outcomes of this work include a deepened understanding of whether and why feature selection is likely to benefit a certain problem, as well as the recommendations that simple and fast feature selection algorithms should be used instead of complex and slow algorithms, and that care should be taken to ensure that feature selection actually improves generalization accuracy.

Based on these findings, a simple feature selection algorithm has been applied to a genre classification problem using a commonly used music dataset, popular audio features, and a standard classification algorithm. Using the evaluation methodology previously found to be appropriate, it has been shown that feature selection does modestly improve generalization accuracy on this dataset. However, it has also been shown that a simpler and faster dimensionality reduction technique is capable of increasing classification accuracy to a similar degree. Applying either dimensionality reduction algorithm results in a genre classification system that outperforms a few of the genre classifiers submitted to a recent MIR contest. However, it is not clear that this system is competitive with the highest-performing algorithms in this contest, which use more sophisticated modeling and classification techniques. In addition to elucidating the magnitude of improvement in genre classification accuracy possible with feature selection,

this work has provided an analysis of feature selection's impacts on other aspects of the classification process.

Work in the context of an internship has provided an opportunity to investigate the impact of feature selection on a working music management system that used a genre classifier to construct a measure of musical similarity. While it seems that feature selection is capable of producing improved classification accuracy, the nature of the music management system is such that improved classification accuracy does not translate into improved quality of the playlists and visualizations created by the system. A more successful approach to improving the system has been developed, which involves applying a feature-selection-like algorithm to reduce the dimensionality of a "similarity space," while maximizing an objective measure of the quality of this space that is constructed to correlate roughly to the subjective usability of the system. Outcomes of this work include a deepened understanding of the limits of feature selection to improve the usability of one classifier-based system, as well as an appreciation of the power of wrapper feature selection search techniques and a new means of harnessing this power to optimize any system that relies on a distance-based measure of musical similarity.

8.2 Contributions to MIR and machine learning

The work performed in the context of this thesis has implications for future work in MIR, particularly work involving music classification and the assessment of musical similarity. It is not valid to conclude from this work that feature selection will offer a similar magnitude of improvement in generalization accuracy for other classification problems, or even for audio genre classification using a different dataset or a different set of features. However, this work underscores that the use of an appropriate evaluation strategy for feature selection is absolutely necessary; it has been demonstrated that the use of CV accuracy of the best feature subset found by a selection algorithm may not be at all predictive of the generalization accuracy obtained with that subset. This thesis has outlined a carefully reasoned and specific evaluation methodology that can be used to more

accurately assess feature selection's efficacy. Additionally, this work has shown that the cost of implementing and running complex and intensive feature selection algorithms is not worthwhile for many classification problems. All of these findings may be useful to any classification work in MIR, including not only other audio classification but non-audio classification as well. This work has also shown that feature selection does have the potential to reduce feature extraction time, classification time, and feature storage space, and these benefits are especially relevant to other research in audio classification.

The work performed at Sun Microsystems suggests that improving classification accuracy may not be sufficient to improve the subjective usability of a classification-based system. This finding may be surprising to many researchers in MIR, given the attention paid to increasing classification accuracy in papers published at ISMIR and systems submitted to MIREX. While this work does not imply that increasing classification accuracy is never a useful pursuit, it does suggest that MIR researchers should also place a priority on evaluating the performance of systems in practice. It may be appropriate, for example, to hold a MIREX contest judged according to subjectively evaluated playlist and/or visualization quality in order to encourage MIR researchers to work toward explicitly improving these capabilities of their systems.

The work performed at Sun also suggests that, despite potential limits of feature selection's ability to improve usability of a given system, the optimization approach used in wrapper feature selection can still be a powerful tool. In the case of Sun's system, similarity is computed via a distance in a "similarity space." If one is able to define an objective metric on the similarity space that roughly corresponds to system usability, the "black box" of wrapper feature selection can be replaced with an evaluator of this metric, and feature selection search methods can be used to optimize the space. The use of a distance metric to compute similarity is not unique to Sun's system, and this optimization approach could be employed in any project using such a metric, regardless of how the similarity space is created or how the objective metric is defined.

The work in this thesis is relevant to research in machine learning as well. For one thing, this work supports previous findings that intensive feature selection methods are not useful, that poor evaluation methodology can greatly exaggerate the efficacy of feature selection in improving classification accuracy, and that feature selection may be shown to offer little or no benefits to classification accuracy when evaluated prudently. Furthermore, the exploration of many standard machine-learning datasets using exhaustive feature selection and Monte Carlo methods elucidates some common causes of feature selection failing to be an effective tool. Finally, the evaluation methodology specified in this work can be used to evaluate the efficacy of wrapper feature selection for improving classification in any domain.

8.3 New questions

This work has raised some new questions whose investigation lies outside the scope of this thesis, but whose answers may have interesting implications for other work in MIR and machine learning. First of all, what is the cause of the lack of correlation between CV accuracy and classification accuracy on different representative subsets of particular datasets? What are the implications of this correlation for the use of CV as a predictor of generalization accuracy for these datasets? How reliably might one predict whether a dataset is likely to exhibit this behavior (e.g., by using Monte Carlo evaluations, as have been used in this work)?

Next, what is an appropriate method for objectively evaluating systems that generate playlists or visualizations based on measures of musical similarity? (And does such a method exist?) The work performed at Sun demonstrated that a heuristic metric was sufficient for use in optimizing similarity spaces for playlist and visualization generation, but it would be beneficial to have a more well-reasoned and generally applicable metric for this purpose. Such a metric could replace the heuristic metric currently used for optimization of the similarity space. Additionally, it could be used to judge between different similarity-based playlist generation and/or visualization systems, for example in the context of a MIREX

contest. This is an interesting question, first of all because work at Sun suggested that previously proposed metrics based on genre, artist, or album metadata are insufficiently related to subjective quality of playlists and visualizations to be used alone. Second of all, much work in MIR (including Sun's system) employs the assumption that a listener-independent, context-independent measure of similarity between two pieces of music is relevant to the problem of playlist generation, even though a playlist is in reality used by a particular user in a particular context. If efforts to produce an objective evaluator of generic music similarity prove difficult, it may be because the concept of generic music similarity is itself problematic.

8.4 Conclusions

The work in this thesis has examined the potential for feature selection to be a useful tool for audio genre classification in particular, and for MIR in general. The MIR and machine-learning literature has been used as the foundation for this work when possible, and original empirical work has been performed when the literature has provided insufficient guidance. The outcomes are a deeper understanding of feature selection, both in general and as it applies to music classification, empirical results that have the potential to inform future work in both MIR and machine learning, and the illumination of open questions in these fields.

REFERENCES

Aha, D. 1992. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies* 36: 267–87.

———. 1997. Lazy learning. *Artificial Intelligence Review* 11 (1–5): 7–10.

All Media Guide. Allmusic. http://www.allmusic.com/.

Almuallim, H., and T. Dietterich. 1991. Learning with many irrelevant features. *Proceedings of the Ninth National Conference on Artificial Intelligence*, 547–52.

Alpaydin, E. 1997. Voting over multiple condensed nearest neighbors. *Artificial Intelligence Review* 11 (1–5): 115–32.

Apple Computer Inc. Apple - iTunes - overview. http://www.apple.com/itunes/overview/.

Aucouturier, J.-J., and F. Pachet. 2002. Music similarity measures: What's the use? *Proceedings of the Third International Conference on Music Information Retrieval.*

———. 2003. Representing musical genre: A state of the art. *Journal of New Music Research* 32 (1): 89–93.

Automated Learning Group. ALG: D2K overview. National Center for Supercomputing Applications (NCSA), University of Illinois. http://alg.ncsa.uiuc.edu/do/tools/d2k.

Bergstra, J., N. Casagrande, and D. Eck. 2005. Two algorithms for timbre- and rhythm-based multi-resolution audio classification. *Papers of the MIREX 2005 genre classification contest.* Available: http://www.music-ir.org/evaluation/mirex-results/audio-genre/index.html.

Bergstra, J., N. Casagrande, D. Erhan, D. Eck, and B. Kégel. Forthcoming. Meta-features and AdaBoost for music classification. *Machine Learning.*

Breiman, L., J. Friedman, R. Olshen, and C. Stone. 1984. *Classification and regression trees.* New York: Chapman and Hall.

Breiman, L., and P. Spector. 1992. Submodel selection and evaluation in regression: The x-random case. *International Statistical Review* 60 (3): 291–319.

Cantú-Paz, E. 2000. *Efficient and accurate parallel genetic algorithms.* Boston: Kluwer Academic Publishers.

Cardie, C. 1993. Using decision trees to improve case-based learning. *Proceedings of the 10th International Conference on Machine Learning*, 25–32.

Casagrande, N., D. Eck, and B. Kégel. 2005. Frame-level audio feature extraction using AdaBoost. *Proceedings of the Sixth International Conference on Music Information Retrieval*, 345–50.

Cohen, P., and D. Jensen. 1997. Overfitting explained. *Proceedings of the Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, 115–22.

Cortes, C., and V. Vapnik. 1995. Support vector networks. *Machine Learning* 20: 273–97.

Cover, T., and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13 (1): 21–7.

Cowen, J., and D. Sharp. 1988. Neural nets and artificial intelligence. *Daedalus* 117: 85–121.

Dietterich, T. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* 10 (7): 1895–923.

Downie, J. S. MIREX 2005 contest results. http://www.music-ir.org/evaluation/mirex-results/.

Downie, J. S., K. West, A. Ehmann, and E. Vincent. 2005. The 2005 music information retrieval evaluation exchange (MIREX 2005): Preliminary overview. *Proceedings of the Sixth International Conference on Music Information Retrieval*, 320–3.

Duda, R., and P. Hart. 1973. *Pattern classification and scene analysis*. New York: Wiley.

Duda, R., P. Hart, and D. Stork. 2001. *Pattern classification*. 2nd ed. New York: Wiley.

Eck, D., and N. Casagrande. 2005. Finding meter in music using an autocorrelation phase matrix and Shannon entropy. *Proceedings of the Sixth International Conference on Music Information Retrieval*, 504–9.

Ellis, D. The "Uspop2002" pop music data set. http://www.ee.columbia.edu/~dpwe/research/musicsim/uspop2002.html.

Essid, S., G. Richard, and B. David. 2004. Musical instrument recognition based on class pairwise feature selection. *Proceedings of the Fifth International Conference on Music Information Retrieval*, 560–8.

Fiebrink, R., C. McKay, and I. Fujinaga. 2005. Combining D2K and JGAP for efficient feature weighting for classification tasks in music information retrieval. *Proceedings of the Sixth International Conference on Music Information Retrieval*, 510–3.

Fix, E., and J. Hodges. 1951. Discriminatory analysis, nonparametric discrimination, consistency properties. *Technical Report 21-49-004*. USAF School of Aviation Medicine, Randolph Field, Texas.

Forrest, S., and M. Mitchell. 1993. Relative building-block fitness and the building-block hypothesis. In *Foundations of genetic algorithms 2*, edited by D. Whitley. San Mateo, CA: Morgan Kaufmann.

Fraser, A., and I. Fujinaga. 1999. Toward real-time recognition of acoustic musical instruments. *Proceedings of the International Computer Music Conference*, 175–7.

Freund, Y., and R. Schapire. 1997. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences* 55 (1): 119–39.

Fujinaga, I. 1998. Machine recognition of timbre using steady-state tone of acoustic musical instruments. *Proceedings of the International Computer Music Conference*, 207–10.

Fujinaga, I., S. Moore, and D. Sullivan Jr. 1998. Implementation of exemplar-based learning model for music cognition. *Proceedings of the International Conference on Music Perception and Cognition*, 171–9.

Goutte, C. 1997. Note on free lunches and cross-validation. *Neural Computation* 9:1245–9.

Guyon, I., and A. Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research* 3: 1157–82.

Hastie, T., R. Tibshirani, and J. Friedman. 2001. *The elements of statistical learning*. New York: Springer.

Holland, J. 1975. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Hunt, M., M. Lennig, and P. Mermelstein. 1980. Experiments in syllable-based recognition of continuous speech. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 880–3.

ISMIR2004 audio description contest. http://ismir2004.ismir.net/ISMIR_Contest.html.

Jensen, D., and P. Cohen. 2000. Multiple comparisons in induction algorithms. *Machine Learning* 38: 309–38.

Jiang, D.-N., L. Lu, H.-J. Zhang, J.-H. Tao, and L.-H. Cai. 2002. Music type classification by spectral contrast feature. *Technical Report.* Department of Computer Science and Technology, Tsinghua University, China, and Microsoft Research, Asia.

John, G., R. Kohavi, and K. Pfleger. 1994. Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning*, 121–9.

Kira, K., and L. Rendell. 1992. A practical approach to feature selection. *Proceedings of the 9th International Conference on Machine Learning.*

Kohavi, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the International Joint Conference on Artificial Intelligence*, 1137–45.

Kohavi, R., and G. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence* 997: 273–324.

Kohavi, R., P. Langley, and Y. Yun. 1997. The utility of feature weighting in nearest-neighbor algorithms. *Proceedings of the 9th European Conference on Machine Learning.*

Kruskal, J. 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 29: 1–27.

Kudo, M., and J. Sklansky. 2000. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition* 33: 25–41.

Li, T., M. Ogihara, and Q. Li. 2003. A comparative study on content-based music genre classification. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Li, T., and G. Tzanetakis. 2003. Factors in automatic musical genre classification. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.*

Logan, B. 2002. Content-based playlist generation: Exploratory experiments. *Proceedings of the Third International Conference on Music Information Retrieval.*

Logan, B., and A. Salomon. 2001. A music similarity function based on signal analysis. *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME).*

Loughrey, J., and P. Cunningham. 2004. Overfitting in wrapper-based feature subset selection: The harder you try the worse it gets. *Proceedings of the 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 33–43.

Magnatune. Magnatune: MP3 music and music licensing (royalty free music and license music). http://magnatune.com.

Martin, K., and Y. Kim. 1998. Musical instrument identification: A pattern-recognition approach. *Proceedings of the 136th Meeting of the Acoustical Society of America.*

McEnnis, D., C. McKay, I. Fujinaga, and P. Depalle. 2005. jAudio: A feature extraction library. *Proceedings of the Sixth International Conference on Music Information Retrieval*, 600–3.

McKay, C. 2004. Automatic genre classification of MIDI recordings. Master's thesis. McGill University.

McKay, C., R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga. 2005. ACE: A framework for optimizing music classification. *Proceedings of the Sixth International Conference on Music Information Retrieval*, 42–9.

McKay, C., and I. Fujinaga. 2005. Automatic music classification and the importance of instrument identification. *Proceedings of the Conference on Interdisciplinary Musicology.*

McKay, C., and I. Fujinaga. 2006. Musical genre classification: Is it worth pursuing and how can it be improved? Paper submitted to the *Seventh International Conference on Music Information Retrieval (ISMIR06).* In review.

Moore, A., and M. Lee. 1994. Efficient algorithms for minimizing cross validation error. *Proceedings of the 11th International Conference on Machine Learning*, 190–8.

Mörchen, F., A. Ultsch, M. Nöcker, and C. Stamm. 2005. Databionic visualization of music collections according to perceptual distance. *Proceedings of the Sixth International Conference on Music Information Retrieval*, 396–403.

Nadeau, C., and Y. Bengio. 2003. Inference for the generalization error. *Machine Learning* 52 (3): 239–81.

Newman, D., S. Hettich, C. Blake, and C. Mertz. 1998. UCI repository of machine learning databases. University of California, Irvine, Department of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Pampalk, E., S. Dixon, and G. Widmer. 2003. On the evaluation of perceptual similarity measures for music. *Proceedings of the 6th International Conference on Digital Audio Effects (DAFX-03)*.

Perlich, C., F. Provost, and J. Simonoff. 2003. Tree induction vs. Logistic regression: A learning-curve analysis. *Journal of Machine Learning Research* 4: 211–55.

Perrot, D., and R. Gjerdigen. 1999. Scanning the dial: An exploration of factors in the identification of musical style. Abstract. *Proceedings of the Society for Music Perception and Cognition*, 88.

Poliner, G., and D. Ellis. 2005. A classification approach to melody transcription. *Proceedings of the Sixth International Conference on Music Information Retrieval*, 161–6.

Pudil, P., J. Novovičová, and J. Kittler. 1994. Floating search methods in feature selection. *Pattern Recognition Letters* 15 (11): 1119–25.

Punch, W., E. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody. 1993. Further research on feature selection and classification using genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 557–64.

Reunanen, J. 2003. Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research* 3:1371–82.

———. 2004. A pitfall in determining the optimal feature subset size. *Proceedings of the The 4th International Workshop on Pattern Recognition in Information Systems (PRIS)*, 176–85.

Rotstan, N., and K. Meffert. JGAP: The Java genetic algorithms package. http://jgap.sourceforge .net.

Russell, S., and P. Norvig. 2003. *Artificial intelligence: A modern approach*. 2nd ed. Upper Saddle River, NJ: Pearson Education, Inc.

Salzberg, S. 1997. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery* 1: 317–28.

Siedlecki, W., and J. Sklansky. 1989. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters* 10 (5): 335–47.

Skalak, D. 1994. Prototype and feature selection by sampling and random mutation hill climbing algorithms. *Proceedings of the International Conference on Machine Learning*, 293–301.

Tzanetakis, G., and P. Cook. 2002. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing* 10 (5): 293–302.

Tzanetakis, G., G. Essl, and P. Cook. 2001. Automatic musical genre classification of audio signals. *Proceedings of the Second Annual International Symposium on Music Information Retrieval.*

UCI Machine Learning. UCI machine learning repository content summary. http://www.ics.uci .edu/~mlearn/MLSummary.html.

West, K., and S. Cox. 2004. Features and classifiers for the automatic classification of musical audio signals. *Proceedings of the Fifth International Conference on Music Information Retrieval.*

———. 2005. Finding an optimal segmentation for audio genre classification. *Proceedings of the Sixth International Conference on Music Information Retrieval*, 680–5.

West, K., and P. Lamere. Forthcoming. A model-based approach to perceptual music similarity. *EURASIP Journal on Applied Signal Processing.*

Wettschereck, D., D. Aha, and T. Mohri. 1997. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review* 11: 273–314.

Witten, I., and E. Frank. 2005. *Data mining: Practical machine learning tools and techniques*. New York: Morgan Kaufmann.

Wolpert, D. 1995. The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework. In *The mathematics of generalization*, edited by D. Wolpert. Reading, MA: Addison-Wesley.

Wolpert, D., and W. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1 (1): 67–82.

Xu, C., N. Maddage, X. Shao, and Q. Tian. 2003. Musical genre classification using support vector machines. *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing (ICASSP03).*