INFORMATION GAIN FEATURE SELECTION BASED ON FEATURE INTERACTIONS

A Thesis Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Bangsheng Sui

December 2013

INFORMATION GAIN FEATURE SELECTION BASED ON

FEATURE INTERACTIONS

_____

Bangsheng Sui

APPROVED:

_____

Dr. Ricardo Vilalta, Chairman
Department of Computer Science

_____

Dr. Christoph F Eick
Department of Computer Science

_____

Dr. Klaus Kaiser
Department of Mathematics

_____

_____

_____

_____

Dean, College of Natural Sciences and Mathematics

INFORMATION GAIN FEATURE SELECTION BASED ON FEATURE INTERACTIONS

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Bangsheng Sui

December 2013

Abstract

Analyzing high-dimensional data stands as a great challenge in machine learning. In order to deal with the curse of dimensionality, many effective and efficient feature-selection algorithms have been developed recently. However, most feature-selection algorithms assume independence of features; they identify relevant features mainly on their individual high correlation with the target concept. These algorithms can have good performance when the assumption of feature independence is true. But they may perform poorly in domains where there exist feature interactions. Due to the existence of feature interactions, a single feature with little correlation with the target concept can be in fact highly correlated when looked together with other features. Removal of these features can harm the performance of the classification model severely.

In this thesis, we first present a general view of feature interaction. We formally define feature interaction in terms of information theory. We propose a practical algorithm to identify feature interactions and perform feature selection based on the identified feature interactions. After that, we compare the performance of our algorithm with some well-known feature selection algorithms that assume feature independence. By comparison, we show that by taking feature interactions into account, our feature selection algorithm can achieve better performance in datasets where interactions abound.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Computers have become an indispensable part of our world. Computers capture and store huge amount of data every day. For example, there are millions of transactions stored in a bank system in a single day. The size and complexity of these datasets is so great that humans are not able to process and extract useful information from them. In the past, when we were given a dataset, we could generally plot it in a graph and try to extract useful information.

x1	x2	Class
2	2	A
5.3	7.1	B
2.8	1.2	A
7.5	5.1	B
2.7	3.1	A
5.1	4.7	B
1.1	0.8	A
6.8	7.3	B
1.2	2.8	A
6.3	5.8	B

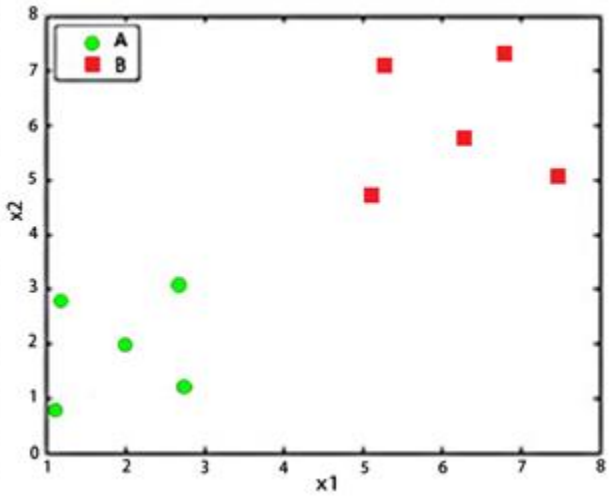

Figure 1.1: A data table and its plotted graph

By comparing the table and the graph in Figure 1.1, we can see that a graph is much easier to deal with than a data table. Unfortunately, if the data have more than three dimensions, it will be difficult for us to view it all at once. It is also impossible for humans to deal with huge amounts of data. Fortunately, we can make computers deal with these massive data for us. This is why machine learning is becoming so popular.

As the dimensionality of the data is getting higher, the efficiency and accuracy of learning algorithms are degraded. The problem of high dimensionality poses a great challenge to learning algorithms. Consider a set of documents, where each document is represented by a vector of occurrence frequencies of each word. There are typically thousands or tens of thousands of attributes. In this case, many types of data analysis become significantly harder as the dimensionality of the data increases. This phenomenon is called the curse of dimensionality [Pang-Ning et al., 2006]. Thus we need to reduce the dimensionality of such high dimensional data in order to apply accurate and efficient learning algorithms; feature selection is a good way to do dimensionality reduction.

1.2 Contribution

Before talking about the contribution of this thesis, we will talk about the benefits of feature selection. There are a variety of benefits from feature selection. A key benefit is that many learning algorithms work better if the number of features in the data is low. This is because we can eliminate irrelevant features and reduce noise by applying feature selection, and we can also improve the efficiency at the same time. Another benefit is that feature selection can lead to a more understandable model because the model may have fewer features. Also, feature selection may allow the data to become easy to visualize.

However, many learning algorithms and feature-selection algorithms assume independence of features, which is not always true. A single feature can be considered irrelevant when treated as an independent feature, but it may become a relevant feature when combined with other features. For example, given a person's weight or height, we are not able to tell the body shape of that person. But if we are given both of these two features, it will be easy for us to tell the body shape. This is because there are interactions between features. In this case, those feature selection algorithms may wrongly remove interacting features. Unintentional removal of these features can result in the loss of useful information and may cause poor learning performance [Zhao and Liu, 2007]. In this thesis, we propose a practical algorithm for feature selection. Instead of treating all single features as independent, our algorithm will deal with feature subsets based on feature interactions.

1.3 Outline

This thesis is organized as follows. In Chapter 2, we introduce some background information about machine learning and feature selection. We will describe the concepts and techniques that we will use in later chapters. In Chapter 3, we introduce the concept of feature interaction. We give an information-theoretic view of feature interaction. Chapter 4 presents the structure and analysis of our algorithm BIFS (Binary Interaction based Feature Selection). In Chapter 6, we evaluate our algorithm with regards to the number of selected features, identified feature interactions and accuracy using selected features. We compare our algorithm with some well-known feature selection algorithms. Finally in the last chapter, we provide our conclusions and discuss limitations and future work.

Chapter 2

Background

2.1 Machine Learning

In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed" [Simon, 2013]. Later a more formal definition was provided: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [Mitchell, 1997]. The key concept of machine learning is learning from data, since data is what we have [Marsland, 2009].

Now, more and more machine learning algorithms have been developed and they are widely used to help humans to analyze complex learning problems; for example, voice recognition, pricing system, computer games and automatic systems. All these learning algorithms can be classified into 5 categories: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, and evolutionary learning.

- **Supervised Learning**

  Supervised learning can be defined as learning from labeled training data. A supervised learning algorithm learns the training data and produces a predictive function. The predictive function will be used to decide the class label for unseen instances [Mohri, Rostamizadeh and Talwalkar, 2012].

- **Unsupervised Learning**

  In contrary to supervised learning, unsupervised learning is learning from unlabeled training data. In the training step, no error or score will be given to evaluate generated solutions. In statistics, unsupervised learning is known as density estimation [Duda et al., 2001].

- **Semi-supervised Learning**

  Semi-supervised learning algorithms learn from both labeled and unlabeled data. Labeled data is often expensive and difficult to obtain. Meanwhile, unlabeled data is relatively easy to collect, but it contains less information than labeled data. By using a large amount of unlabeled data with a small amount of labeled data, semi-supervised learning requires less human effort and tends to higher accuracy [Zhu, 2008].

- **Reinforcement Learning**

  Reinforcement Learning deals with how software agents should take actions in an environment to maximize a reward. In a traditional reinforcement learning model, an agent is connected to the environment via perception and action [Kaelbling et al., 1996]. Whenever the agent takes an action, the action will change the state of the environment. The value of this state transition will be returned to the agent as a reward. The agent should learn to take actions that can maximize the long-run sum of reward.

- **Evolutionary Learning**

  Evolutionary learning is inspired by biological evolution. The learner will start with a population of solutions. Then scores will be assigned to each solution within the population. After that, only a percentage of the solutions will be retained. The empty space of the population will be filled with new solutions generated by genetic operators including crossover, mutation and replication [Holland, 1992].

Supervised learning is the most common type of learning. This thesis will mainly discuss feature selection for a supervised learning problem.

2.2 Classifiers

One instance of supervised learning algorithms is a classifier. Classifiers are machine learning algorithms that solve the problem of classification. A classification problem can be defined as of identifying class labels for new observations on the basis of a training set of data whose class label is known. Examples include fraud detection, face recognition, and galaxy identification. A classifier's task is to build a classification model that can be used for prediction of class labels of unknown data. The process of applying classification to a real-world problem is described in Figure 2.1.

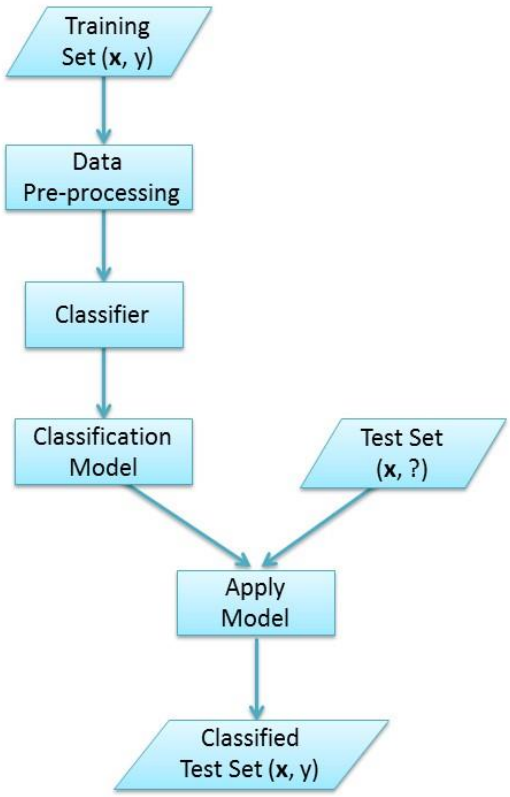

Figure 2.1 The process of classification

The input for a classification task is a collection of labeled records. Each record is represented by a tuple (**x**, y), where **x** is a vector representing the feature set and y is the class label. A first step is data pre-processing. A real-world dataset is usually not suitable for classification directly. In most cases, it contains noise, missing value, irrelevant and redundant features, and therefore requires significant pre-processing [Zhang et al., 2002]. The pre-processed data enables the classifier to operate faster to build a more accurate classification model. A classification model is also known as a target function that maps each feature set **x** to one of the class labels y. After the classification model is built, the mapping function will be used to predict the class label for the test set which consists of records with unknown class labels.

There are many well-known machine learning classifiers, such as decision tree, support vector machine, naïve bayes, linear regression, and multilayer perceptron. Here we will introduce two popular ones: decision trees and support vector machines, which we will use in our experiments.

2.2.1 Decision Trees

A decision tree learning algorithm approximates a target concept using a tree representation, where each internal node corresponds to a feature and every terminal node corresponds to a class [Grosan and Abraham, 2011]. There are two kinds of nodes in a decision tree, internal nodes and terminal nodes. The internal node splits the dataset into different branches according to the different values of the corresponding feature. Root node is a special kind of internal node. The terminal node has a class label assigned to it and all the data records in the terminal node will be predicted as the class label assigned to the node. Figure 2.2 shows an example of a decision tree for the data in Table 2.1.

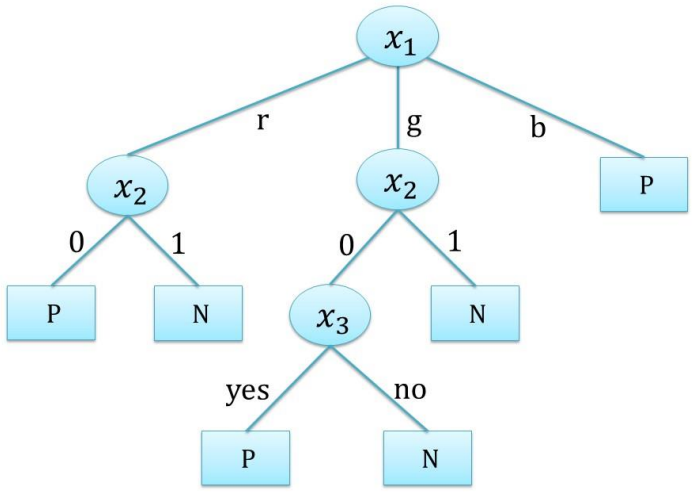

Figure 2.2 A decision tree representation of Table 2.1

Decision trees are a way to represent rules underlying data with hierarchical, sequential structures that recursively partition the data [Murthy, 1998]. The hierarchical structure is very similar to our own logic. This is the reason why decision trees have higher interpretability than other machine learning classifiers. By comparing Figure 2.2 and Table 2.1, we can find that a decision tree is much easier to understand and get information from, than a data table.

x_1 (r, g, b)	x_2 (0, 1)	x_3 (Yes, No)	Class (Positive, Negative)
r	0	Yes	P
r	1	No	N
r	1	Yes	N
r	0	No	P
g	0	Yes	P
g	0	No	N
g	1	Yes	N
g	1	No	N
b	1	Yes	P
b	0	No	P
b	1	No	P

Table 2.1 Dataset example

Given a dataset, there may be more than one possible decision tree representation for it. For example, Figure 2.3 is another decision tree for Table 2.1. Thus we need to choose the best one among all possible decision trees. However, constructing an optimal decision tree is an NP-complete problem and thus efficient heuristics have been found to construct near-optimal decision trees [Kotsiantis, 2007]. The key point to build an optimal decision tree is to find the best feature to divide the training set. There are plenty of methods to judge the goodness of a feature to divide the training set. Among these methods, information gain is one of the most popular one. We will talk about entropy and information gain in section 3.2.

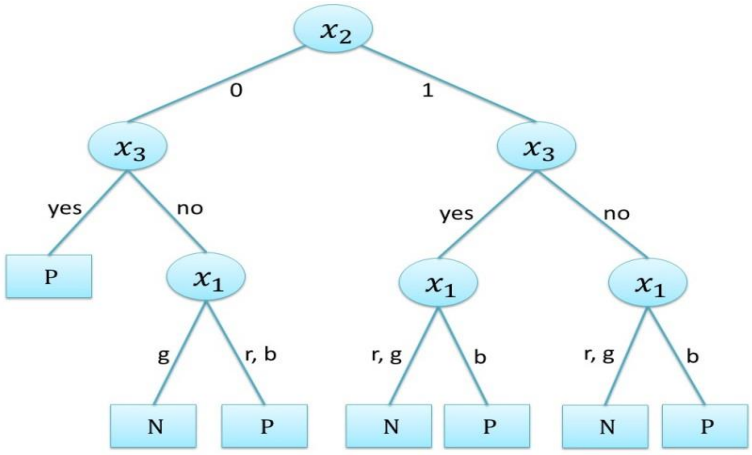

Figure 2.3 Another decision tree representation of Table 2.1

When we talk about decision trees, we must mention the overfitting issue. Before talking about overfitting of decision trees, we will introduce the definition of overfitting of a hypothesis. Assume a hypothesis space *H*. We say a hypothesis *h* in *H* overfits a dataset *D* if there is another hypothesis *h*' in *H* where *h* has better classification accuracy than *h*' on *D* but worse classification accuracy than *h*' on dataset *D*', where *D*' comes from the same data source as *D* [Mitchell, 1997]. A decision tree overfits the data if we let it grow deep enough so that it begins to capture "aberrations" in the data that harm the predictive power on unseen records. Figure 2.4 shows the training and testing accuracy of the decision tree.

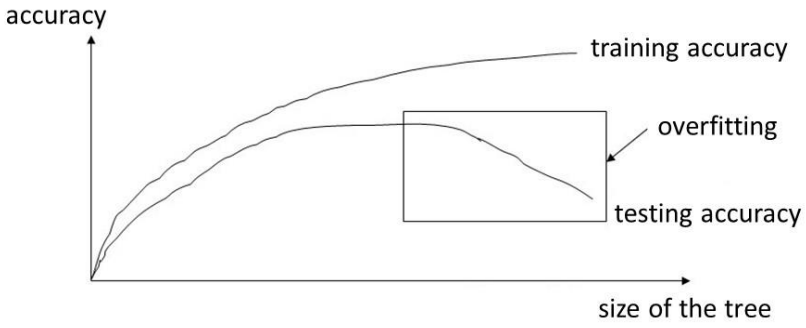

Figure 2.4 Training and testing accuracy of the decision tree

There are mainly two causes that lead to overfitting of decision trees. One cause is the presence of noise. Noise gives incorrect information about the dataset. Another cause is lack of representative samples. In this case, a decision tree may capture coincident patterns due to the small size of the representative sample. And sometimes coincident patterns can also be caused by irrelevant features. Here we will introduce another cause for overfitting. Sometimes classifiers, like decision trees, may use the information provided by redundant features repeatedly. This will cause a tree to be grown too deep. Irrelevant features and redundant features can both be eliminated by feature selection. Thus feature selection is a good way to avoid overfitting in decision trees.

2.2.2 Support Vector Machines

Support vector machines are state-of-the-art supervised learning algorithms that are used for classification and regression analysis. A support vector machine maps the original dataset to a higher dimensional space. Then the support vector machine will construct a hyperplane or set of hyperplanes with sufficiently high dimensions in the new space. The support vector machine was first introduced in [Guyon, Boser and Vapnik, 1992].

To understand how a support vector machine works, we must start from linear discriminant functions, also known as linear classifiers. Consider a binary classification problem containing *n* examples. Each example can be represented by a tuple ($\mathbf{x}_i$, y_i) (i = 1, 2, 3, …, n), where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik})^T$ is the transpose feature vector and y_i is the class lable of the i^{th} example. For convenience we assume the class labels are 1 or -1. Then a linear discriminant function can be defined as:

$$f(x) = \mathbf{w} \cdot \mathbf{x} + b \,,$$

where b is the bias of the model. The key concept for defining a linear discriminant function is the dot product $\mathbf{w} \cdot \mathbf{x}$ which is defined as:

$$\mathbf{w} \cdot \mathbf{x} = \sum_{i}^{n} w_i \cdot x_i \,,$$

where $\mathbf{w}$ is the known weight vector. A linear decision boundary of a linear discriminant function can be expressed as:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \,.$$

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two points on the decision boundary, then

$$\mathbf{w} \cdot \mathbf{x_1} + b = 0 \,,$$

$$\mathbf{w} \cdot \mathbf{x_2} + b = 0 \,.$$

Subtracting the two equations will get:

$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 0 \,.$$

Since $\mathbf{x}_1 - \mathbf{x}_2$ is a vector that is parallel to the decision boundary, $\mathbf{w}$ must be a vector that is perpendicular to the decision boundary.

Linear discriminant functions have their advantages, one of them being that they often have simple training algorithms that scale well with the number of examples [Hastie et al., 2001]. However, sometimes the dataset can't be linearly separated in a fixed dimensional space. We may want to extend linear discriminant functions to handle non-linear decision boundaries. Fortunately, we can generalize a linear discriminant function to a non-linear discriminant function by using a mapping function $\varphi: X \to F$. The φ function maps the data from the original space X to a new feature space F. The generalized discriminant function is defined as:

$$f(x) = \mathbf{w} \cdot \varphi(\mathbf{x}) + b \, .$$

Consider an example of a two dimensional input space with the φ-fucntion

$$\varphi(\mathbf{x}) = (1, x_1, x_1 x_2, x_1^2, x_2^2) \, ,$$

then the dot product $\mathbf{w} \cdot \varphi(\mathbf{x})$ is calculated as:

$$\mathbf{w} \cdot \mathbf{x} = w_1 + w_2 \cdot x_1 + w_2 \cdot x_1 x_2 + w_3 \cdot x_1^2 + w_4 \cdot x_2^2 \, .$$

The new discriminant function will build a non-linear decision boundary in the two dimensional input space.

Now let's talk about the margin of SVMs. Figure 2.5 shows the margin of a linear decision boundary for the data from Figure 1.1.

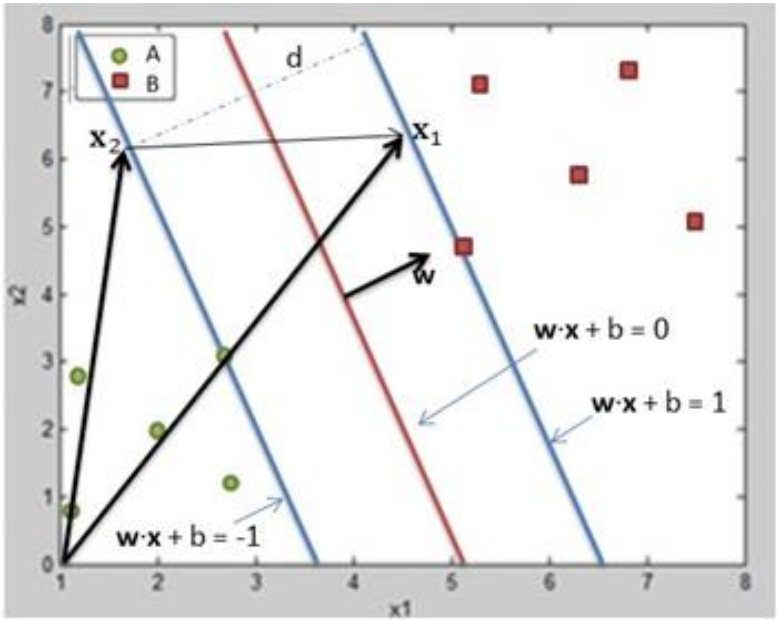

Figure 2.5 Margin of support vector machines

For the data from Figure 1.1, the linear classifier predicts class labels for any test example in the following way:

$$y = \begin{cases} B, & if\ \mathbf{w} \cdot \mathbf{x} + b > 0 \\ A, & if\ \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases}.$$

For all the test examples above the decision boundary, the linear classifier classifies them as class B. Similarly, test examples below the decision boundary are classified as class A.

Consider the red and green points that are closest to the decision boundary in Figure 2.5. It is possible to find the parameters **w** and b so that the two hyperplanes h_1 and h_2 can be expressed as follow:

$$h_1 = \mathbf{w} \cdot \mathbf{x} + b = 1\,,$$

$$h_2 = \mathbf{w} \cdot \mathbf{x} + b = -1\,.$$

The margin of the linear classifier is defined as the distance *d* between these two hyperplanes. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two points from hyperplane h_1 and h_2 respectively. Then we have:

$$\mathbf{w} \cdot \mathbf{x_1} + b = 1 ,$$

$$\mathbf{w} \cdot \mathbf{x_2} + b = -1 .$$

Subtracting the two equation will get:

$$\mathbf{w} \cdot (\mathbf{x_1} - \mathbf{x_2}) = 2 .$$

Since **w** is vector that is perpendicular to the decision boundary, the above equation can be written as:

$$\|\mathbf{w}\| \cdot d = 2 .$$

Now we can see that the margin of a linear classifier is only related to the module of the weight vector **w**:

$$d = \frac{2}{\|\mathbf{w}\|} .$$

The goal of support vector machines is to find the decision boundary with the largest margin. The learning task of a linear support vector machine can be formalized as follows:

$$min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} \; subject \; to \; y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1, i = 1,2,3 \ldots, n ,$$

where *n* is the number of training examples. We can write this in a dual form as:

$$max \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} .$$

Details about deriving the equation can be found in [Pang-Ning, 2006]. Recall that a linear discriminant function can be extended to handle non-linear decision boundaries by using a φ function. For a non-linear support vector machine, the learning task can be summarized as:

$$min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} \; subject\ to\ y_i(\mathbf{w} \cdot \varphi(\mathbf{x}_i) + b) \geq 1, i = 1,2,3 \ldots, n\,,$$

And we can also write it in a dual form as:

$$max \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)\,.$$

Here the dot product $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$ is called a kernel function. The kernel function is a measure of the similarity between two examples in the transformed space. There are three different types of kernel functions that are commonly used:

- Polynomial kernel with degree *s*:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^s$$

- Sigmoid kernel with parameter *k* and δ:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(k\mathbf{x}_i \cdot \mathbf{x}_j - \delta)$$

- Radial kernel with parameter δ:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-(\mathbf{x}_i - \mathbf{x}_j)/2\delta^2)$$

Choosing which kernel and parameters to use is a tricky problem. Most of the time, we will just run support vector machines with different kernels parameters and find the one that works best.

2.3 Evaluation of Classification Models

Since we have so many classifiers to choose from to solve a specific classification problem, we must have a measure to judge the goodness of a classifier. This step is called validation or evaluation. The evaluation of a classifier is most often based on its predictive accuracy. Before talking about accuracy, let's talk about the confusion matrix first. A confusion matrix describes the number of correctly and incorrectly predicted examples by the classification model. Table 2.2 shows the confusion matrix of a binary classification problem.

		Predicted Class	
		Class = 0	Class = 1
True Class	Class = 0	f_{00}	f_{01}
	Class = 1	f_{10}	f_{11}

Table 2.2 Confusion matrix of a binary classification problem

In Table 2.2, each entry f_{ij} denotes the number of examples whose true class label is i and predicted class label is j. For example, f_{11} is the number of examples from class 1 correctly predicted as class1 and f_{10} is the number of examples from class 1 incorrectly predicted as class 0. We can tell the goodness of a classifier for a binary classification problem easily from the confusion matrix. But for multiclass classification problems, we need a generalize measure metric to evaluate the performance of a classifier, such as accuracy, which is defined as follows:

$$accuracy = \frac{Number\ of\ correctly\ predicted\ records}{Total\ number\ of\ predicted\ records}.$$

For a binary classification problem, the definition of accuracy can be expressed as:

$$accuracy = \frac{Number\ of\ correctly\ predicted\ records}{Total\ number\ of\ predicted\ records} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}.$$

Based on the definition of accuracy, we have at least three methods to evaluate the performance of a classifier.

- **Holdout Method**

In the holdout method, the original data are partitioned into two parts, the training set and the test set. Most often, two thirds of the original dataset is used for training and one third is used for testing. Then a classification model will be induced from the training set and the performance of the classifier will be evaluated by the test set. There is one well-known limitation for the holdout method. The performance of the classifier may be highly affected by the composition of the training and testing sets. A small training set size may result in a classification model with large variance. On the other hand, if the training set is too large, the accuracy computed from the small test set may be less reliable.

- **Cross-Validation**

In cross-validation method, the original data are randomly divided into mutually exclusive and equal-sized subsets. For each subset, the classifier is trained with the union of all other subsets and tested on that subset [Kohavi, 1995]. If the original dataset is segmented into *k* subsets, then the method is called *k*-fold cross validation. The cross validation method ensures that each record from the original dataset is used the same number of times for both training and testing. In cross validation method, the total accuracy is obtained by averaging the accuracies for all k runs, which can be calculated as follows:

$$accuracy = \frac{Total\ number\ of\ correctly\ predicted\ records\ for\ all\ k\ runs}{Total\ number\ of\ predicted\ records}.$$

- **Bootstrap**

Bootstrap was introduced by EFron [Efron and Tibshirani, 1993]. In the bootstrap method, the training records are sampled with replacement uniformly from the original dataset. For a dataset of size *n*, the probability of a record not being chosen by *n* bootstrap samples is $(1 - 1/n)^n$. When *n* is sufficiently large, the probability approaches $e^{-1} \approx 0.368$. Thus the probability of a record being chosen by at least one bootstrap sample is 0.632. The accuracy of bootstrap methods can be calculated by the .632 bootstrap approach as below:

$$acc_{boot} = \frac{1}{b}\sum_{i=1}^{b}(0.632 \times \epsilon_i + 0.368 \times acc_s)\ ,$$

where *b* is the number of bootstrap samples, ϵ_i is the accuracy estimate for bootstrap sample *i* and acc_s is the accuracy estimate on the full dataset.

2.4 Feature Selection

Feature selection, also known as variable selection, attribute selection or feature subset selection, is the process of selecting relevant features in terms of a target learning problem. The purpose of feature selection is to remove redundant and irrelevant features. Irrelevant features are features that provide no useful information about the data, and redundant features are features that provide no more information than the currently selected features. In other words, redundant features do provide useful information about the data set, but the information has been provided by the currently selected features. For example, the year of birth and age contain the same information

about a person. Redundant and irrelevant features can reduce the learning accuracy and the quality of the model that is built by the learning algorithm.

In order to apply learning algorithms more efficiently and accurately, many feature selection algorithms have been proposed to help reducing the dimensionality, such as FCBF [Yu and Liu, 2003], CFS [Hall, 2000], ReliefF [Kononenko, 1994], and FOCUS [Almuallim and Dietterich, 1994]. By removing irrelevant features and reducing noise, we can both increase the accuracy and the efficiency of learning algorithms [Guyon and Elisseeff, 2003]. Feature selection has become the key point of much research in areas where high dimensional datasets are involved. These areas include text processing, gene expression, and combinatorial chemistry.

A feature selection algorithm is usually formed by a search technique and an evaluation measure. A search technique is an approach whose task is to propose new feature subsets. These search approaches include best first, exhaustive, genetic algorithm, simulated annealing, greedy forward selection, and greedy backward elimination. An evaluation measure scores the different feature subsets. Some popular evaluation metrics are the followings: correlation, mutual information, error probability, inter-class distance, and entropy. The diagram of feature selection can be summarized in Figure 2.6.

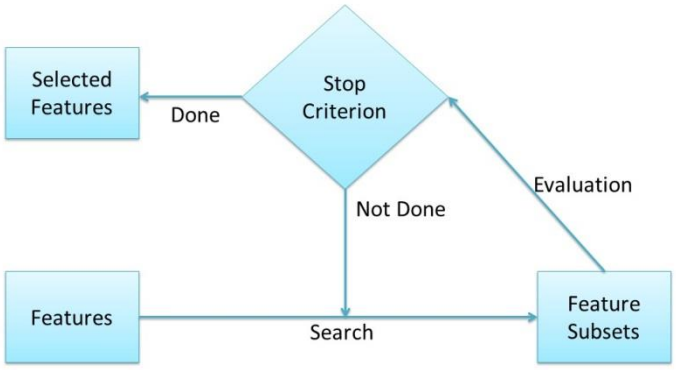

Figure 2.6 Flow of feature selection

In concept, a feature selection algorithm will search over all possible subsets of features to find the optimal subset. This process can be computationally intensive, so we need some sort of stopping criterion. The stopping criterion is usually based on conditions involving the number of iterations and the evaluation threshold. For example we can force the feature selection process to stop after it reaches a certain number or iterations.

Feature selection algorithms can be classified into three categories: embedded approaches, wrapper approaches, and filter approaches [Kohavi and John, 1997; Das, 2001].

- **Embedded Approaches**

Feature selection is performed as part of the learning algorithm. During the run of the learning algorithm, the algorithm itself decides which attribute to use and which to eliminate. By incorporating feature selection as part of the training process, embedded approaches may achieve higher efficiency than other feature selection approaches. One example of an embedded approach is the algorithm for building decision trees, such as CART [Breiman et al., 1984].

- **Wrapper Approaches**

Wrapper approaches use the learning algorithm as a black box to do feature selection. In its most general formulation, wrapper approaches use the performance of the learning algorithm as an evaluation measure to access the usefulness of subsets of features. They are remarkably universal and simple. Wrapper approaches seem to require massive amount of computation, but it is not necessarily so when applying efficient search strategies.

- **Filter Approaches**

Filter approaches perform feature selection before the learning algorithm is run. They can be used as a preprocessing step to reduce the dimensionality and avoid overfitting. They are independent and require no information from the learning task. Filter approaches usually require less computation than wrapper approaches. Filter approaches provide a general feature selection strategy for most learning algorithms.

Chapter 3

Feature Interaction

3.1 General View of Feature Interaction

Feature interaction is phenomenon that has been studied for years, e.g., in the field of telephony [Calder et al., 2003]. A feature interaction occurs when the combination of two or more features modify the behavior of some decompositions of the combined feature set in an unexpected way. The concept of feature interaction has been presented several times in machine learning, but with different terminologies. For example, J. R. Quinlan referred to the problem in this way [Qui, 1994]:

> We can think of a spectrum of classification tasks corresponding to this same distinction. At one extreme are P-type tasks where all the input variables are always relevant to the classification. Consider an n-dimensional description space and a yes-no concept represented by a general hyperplane decision surface in this space. To decide whether a particular point lies above or below the hyperplane, we must know all its coordinates, not just some of them. At the other extreme are the S-type tasks in which the relevance of a particular input variable depends on the values of other input variable. In a

concept such as ‘red and round, or yellow and hot’, the shape of the object is relevant only if it is red and the temperature only if it is yellow.

In this description, S-type tasks are actually tasks with interactive features, while P-type tasks contain all independent features. To illustrate an example of feature interaction, see Figure 3.1.

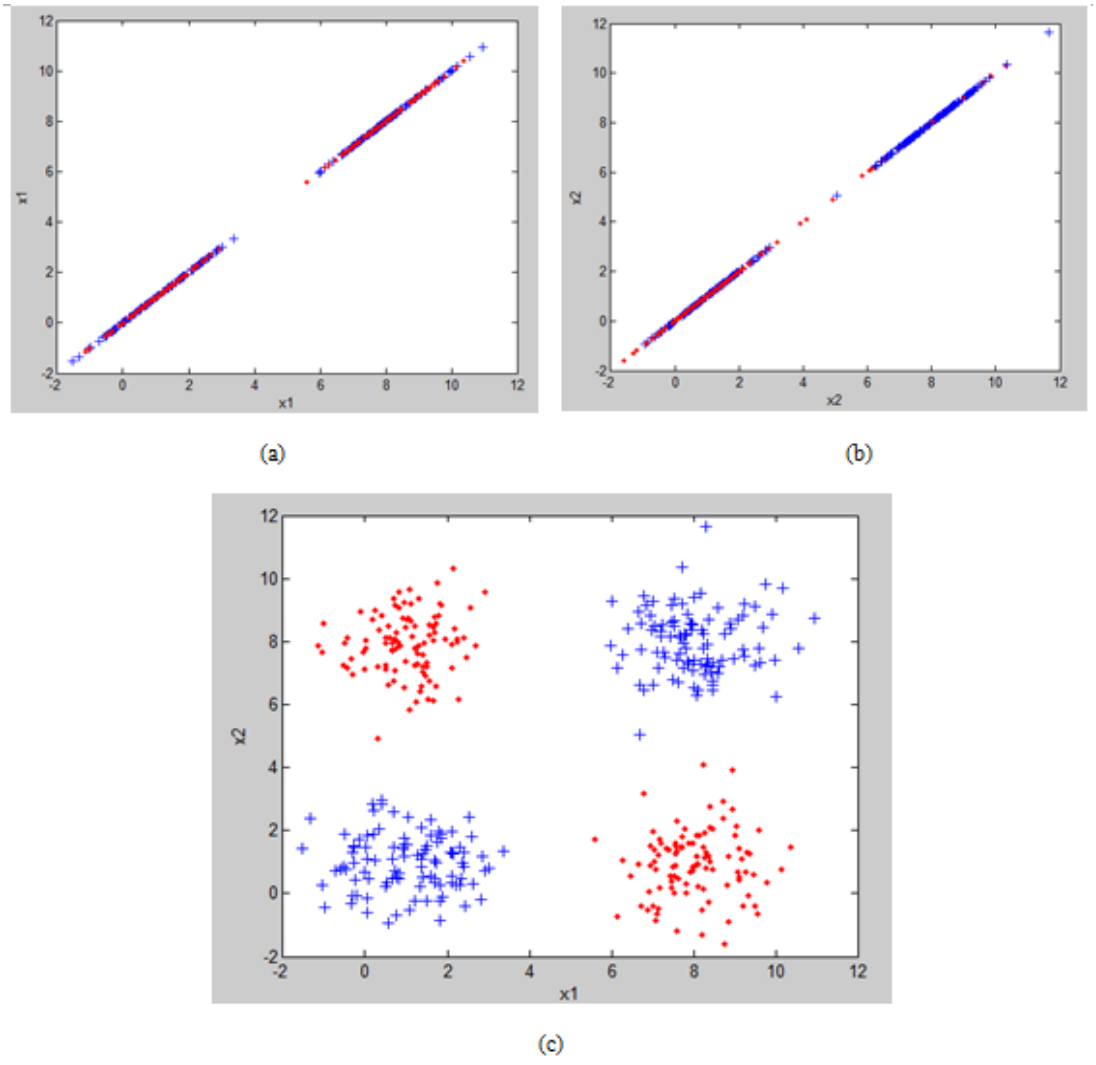

Figure 3.1Example of feature interaction. (a) The data is plotted by projecting on feature x_1. (b) The data is plotted by projecting on feature x_2. (c) The data is plotted using both feature x_1 and feature x_2.

The data of Figure 3.1 is generated by four Gaussians centered on four corners of a square: (1, 1), (1, 8), (8, 8), and (8, 1). The two colors denote two classes. This example is inspired by the XOR problem. As shown in Figure 3.1 (a) and (b), the data is highly overlapped by plotting the data using either x_1 or x_2 only. Neither feature x_1 nor x_2 can provide a separation of the two classes. This means feature x_1 and feature x_2 are useless independently by themselves. However, if we plot the data using feature x_1 and x_2 together as in Figure 3.1 (c), we can see the two classes can easily be separated. This is because there is interaction between these two features. Due to the existence of feature interaction, apparently useless features can be useful when they are taken together.

Feature interaction sometimes can also degrade the prediction power of the combination of features. The significance of feature interactions is well studied in [Jakulin and Bratko, 2003] and [Jakulin and Bratko, 2004]. They defined the strength of feature interactions *IS* as:

$$IS \coloneqq f(C, X_1, X_2, \dots, X_k) - v(\sum_i e_i(C, X_i)\ ,$$

where $f(C, X_1, X_2, \dots, X_k)$ is the evidence function denoting the degree of evidence of choosing class label C given features $X_1, X_2, \dots, X_k$, and $v(\sum_i e_i(C, X_i)$ is the voting sum of the evidence of a single feature X_i. Actually, the voting sum treats every feature as independent. If the assumption of independence of features is true, then

$$f(C, X_1, X_2, \dots, X_k) = v(\sum_i e_i(C, X_i)\ .$$

Otherwise, IS greater than some positive threshold indicates a positive interaction and IS less than some negative threshold indicates a negative interaction. A positive interaction means the

prediction power of a joint set of features $X_1, X_2, \dots, X_k$ is stronger than it of the voting sum which treats all features as independent. A positive interaction indicates the combination of features $X_1, X_2, \dots, X_k$ provides new information about the data in addition to the voting sum. A negative interaction happens when multiple features share duplicate information about the data. In this case, the evidence provided by the evidence function $f(C, X_1, X_2, \dots, X_k)$ is weaker than the voting sum.

3.2 An Information-Theoretic View of Feature Interaction

3.2.1 Entropy

We can now investigate feature interactions in the scope of information theory. In information theory, entropy measures the amount of information that is missing before reception. The entropy we will use to identify feature interactions is Shannon Entropy [Shannon, 1948]. For a data group *S* with *n* class labels, the Shannon Entropy is a measure of its unpredictability or impurity:

$$H(S) = -\sum_{i=0}^{n} p(i) \log p(i),$$

where $p(i)$ is the probability of class i in the data group S. Figure 3.2 shows the entropy function as a function of probability.

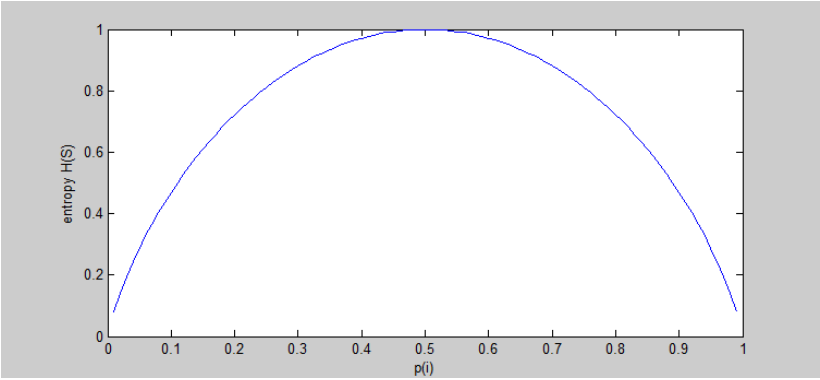

Figure 3.2 Entropy for a problem with two class labels

Entropy is a function concave downward. When all the examples within data group S belong to the same class which means $p(i) = 0$ or $p(i) = 1$, the entropy $H(S) = 1*\log1 + 0*\log0 = 0$. Entropy reaches the largest value 1 when $p(i) = 0.5$ which means half of the examples in data group S belong to one class and half of examples belong to the other class. From Figure 3.2, we can see that the higher the entropy, the less pure the data. Figure 3.2 also tells us that the smallest value of entropy is 0 and the largest value of entropy is 1. This is true for learning problems with two class labels, but not true for learning problems with more than two class labels. Consider a learning problem with n class labels. The most pure situation happens when all the examples belong to the same class. In this case,

$$H(S) = 1 * log1 + 0 * log0 + \cdots + 0 * log0 = 0 .$$

The most impure case is that every $\frac{1}{n}$ of the examples belong to one different class. Then

$$H(S) = -\left(\frac{1}{n} log \frac{1}{n} + \frac{1}{n} log \frac{1}{n} + \cdots + \frac{1}{n} log \frac{1}{n}\right) = -n\left(\frac{1}{n} log \frac{1}{n}\right) = -log \frac{1}{n} = \log n .$$

Thus the smallest value of entropy is 0. But the largest value of entropy is not necessarily limited to 1, it is bounded by $\log n$.

3.2.2 Information Gain

Another key concept of information theory is information gain. In information theory, information gain is a synonym for *Kullback-Leibler divergence*. *Kullback-Leibler divergence* is a non-symmetric measure of the divergence between probability functions *P* and *Q* [Kullback and Leibler, 1951]:

$$D(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} \ .$$

Actually, KL divergence is the expected logarithmic difference between the probability model *P* and *Q*. KL divergence is zero only when the two probability functions *P* and *Q* are equal.

However, information gain can also be defined with mutual information. In particular, information gain *IG*(*A*) is the reduction in the entropy that is archived by learning a variable A:

$$IG(A) = H(S) - \sum_i \frac{S_i}{S} H(S_i) \ ,$$

where $H(S)$ is the entropy of the given dataset and $H(S_i)$ is the entropy of the i_{th} subset generated by partitioning *S* based on feature A. In machine learning, information gain can be used to help ranking the features. Usually a feature with high information gain should be ranked higher than other features because it has stronger power in classifying the data.

Generally, we can also define information gain for a joint set of features as the reduction in the entropy that is archived by learning a joint feature set $\boldsymbol{F}$. The definition is similar to the definition above:

$$IG(\boldsymbol{F}) = H(S) - \sum_i \frac{S_i}{S} H(S_i) \ ,$$

where $H(S_i)$ is the entropy of the i_{th} subset generated by partitioning S based on all features in the joint feature set $\boldsymbol{F}$.

3.2.3 Definitions Based on Information Theory

In order to handle feature selection and feature interactions in practice, we must define the concepts based on a theory which can be applied directly, e.g., information gain. To perform the task of feature selection, we must know what are relevant features. According to [John et al., 1994], a feature f_i is relevant if and only if removing it from a feature set $\boldsymbol{F}$ will harm the prediction power of the feature set. This means a relevant feature contains valuable information about the dataset that cannot be replaced by any other features in the feature set. Thus we can define relevance in terms of information gain as definition 1.

Definition 1 (Feature Relevance) A feature f_i is relevant iff

$$IG(\boldsymbol{F}) > IG(\boldsymbol{F} - f_i) \ ,$$

where $f_i \in \boldsymbol{F}$ and $\boldsymbol{F} - f_i$ is the feature subset resulted by removing feature f_i from feature set $\boldsymbol{F}$.

Here, we use information gain to measure the prediction power of features. According to definition 1, a feature is relevant only if the purity of the state achieved by learning the feature set $\boldsymbol{F}$ is higher than the purity of the state achieved by learning the feature set $\boldsymbol{F}$ - f_i.

Definition 2 (Binary Feature Interaction) f_1 and f_2 are two features. There is binary feature interaction between feature f_1 and f_2 if

$$IG(f_1 \& f_2) \neq IG(f_1) + IG(f_2) \ ,$$

where $IG(f_1 \& f_2)$ is the information gain of the joint set of feature f_1 and f_2. More precisely,

$$IG(f_1 \& f_2) > IG(f_1) + IG(f_2)$$

denotes a positive binary feature interaction between feature f_1 and f_2 and

$$IG(f_1 \& f_2) < IG(f_1) + IG(f_2)$$

denotes a negative binary feature interaction between feature f_1 and f_2.

Actually, feature interactions are not necessarily limited between single features. They can also happen between feature subsets. Thus, we can expand definition 2 as:

Definition 3 (Binary Feature Subset Interaction) $\boldsymbol{F_1}$ is a feature subset with k features and $\boldsymbol{F_2}$ is a feature subset with l features, where $k, l \geq 1$. There is binary feature subset interaction between feature subset $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$ if

$$IG(\boldsymbol{F}_1 \& \boldsymbol{F}_2) \neq IG(\boldsymbol{F}_1) + IG(\boldsymbol{F}_2) \ ,$$

where $IG(\boldsymbol{F}_1 \& \boldsymbol{F}_2)$ is the information gain of the joint set of feature subset $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$. More precisely,

$$IG(\boldsymbol{F}_1 \& \boldsymbol{F}_2) > IG(\boldsymbol{F}_1) + IG(\boldsymbol{F}_2)$$

denotes a positive binary feature subset interaction between feature subset $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$ and

$$IG(\boldsymbol{F}_1 \& \boldsymbol{F}_2) > IG(\boldsymbol{F}_1) + IG(\boldsymbol{F}_2)$$

denotes a negative binary feature subset interaction between feature subset $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$.

Binary feature subset interaction is a generalized definition of binary feature interaction. We can treat definition 2 as a special case of definition 3 when $k, l = 1$. Recall that negative interactions are caused by duplicate information that should be counted only one time. Negative interactions can usually be solved by removing duplicate features. It has been well studied in many well-known feature selections algorithms. However, positive interactions are always ignored by most of the feature selection algorithms which assume independence of features. In this thesis, we focus our research on positive interactions. For the rest of the thesis, we will mainly discuss positive interactions.

After defining the interactions between feature subsets, we can keep expanding the definition of feature interaction to *k*-degree feature interaction.

Definition 4 (*k*-degree Feature Interaction) Let $\boldsymbol{F}$ denote a feature set with *k* features f_1, f_2, …, f_k. $\{F_1, F_2, \dots, F_l\}$ is an arbitrary partition of $\boldsymbol{F}$, where $l \geq 2$ and $F_i \neq \emptyset$. We say f_1, f_2, …, f_k interact with each other iff

$$IG(\boldsymbol{F}) > \sum_{i}^{l} IG(F_i) \quad .$$

Identifying *k*-degree feature interaction requires exponential time. Definition 4 cannot be directly applied to identify interacting features for a high dimensional dataset. In this thesis, we use the idea of closure problem to identify potential *k*-degree feature interactions.

Definition 5 (Closed Interactive Feature Subset) Feature subset ***F*** is a closed interactive feature subset if every possible pair of features in ***F*** has binary feature interaction between them.

Definition 5 gives the definition of closure problem in the domain of binary feature interactions. Closed interactive feature subsets are also considered to be potential feature interactions since every possible pair of features within them must have binary feature interactions. A potential *k*-degree feature interaction is a closed interactive feature subset with *k* features. The purpose of feature interaction identification is to identify all possible feature interactions of maximum size. In order to perform feature selection based on feature interactions, we must ensure that for every identified feature interaction, there is no possible feature interaction containing it. This is the idea of the maximum closure problem.

Definition 6 (Maximum Closed Interactive Feature Subset) Feature subset ***F*** is a maximum closed interactive feature subset if ***F*** is a closed interactive feature subset and adding any new features to ***F*** from the original dataset will cause it to no longer be a closed interactive feature subset.

Definition 7 (*k*-degree Maximum Feature Interaction) Let ***F*** denote a *k*-degree feature interaction. ***F*** is a maximum feature interaction if there is no (k+1)-degree feature interaction that contains F.

Based on the explanation of definition 5, a maximum closed interactive feature subset can be considered as a potential maximum feature interaction. The problem of identifying *k*-degree potential maximum feature interactions can be formulated as:

$$max \sum_{f_i \in \boldsymbol{F}} \sum_{f_j \in \boldsymbol{F}} 1 \quad ,$$

$$subject\ to\ i < j, f_i\ and\ f_j have\ binary\ interaction\ between\ them,$$

where $\boldsymbol{F}$ is the feature set containing all features of the dataset.

With all the definitions above, the purpose of our research is to identify all potential maximum feature interactions and perform feature selection for a supervised learning problem based on feature interactions.

Chapter 4

Algorithm

In this section, we present our algorithm BIFS, which performs the task of identifying all potential maximum feature interactions and feature selection for a supervised learning problem. BIFS is constructed by two main processes: forward identification and backward selection. BIFS performs its task by applying the backward selection after the forward identification. The details of our algorithm BIFS are shown in Figure 4.1.

==========

input:

S: the source dataset

$\boldsymbol{F}$: full feature set with features $f_1, f_2, \ldots, f_n$

α: degree of interaction

β: degree of relevance

output:

$\boldsymbol{F}_{\text{interact}}$: identified potential feature interaction of all k-degrees

$\boldsymbol{F}_{\boldsymbol{select}}$: the selected feature subset

step 1: initialization

$\boldsymbol{F}_{\text{interact}}$ = NULL

$\boldsymbol{F}_{\text{binary}}$ = NULL

for i = 1 **to** n **do**

add $\{f_i\}$ to $\boldsymbol{F}_{\text{interact}}$

end

for i = 1**to** n **do**

for $j = i + 1$ **to** n **do**

if $IG(f_i \,\&\, f_j) > IG(f_i) + IG(f_j) + \alpha$

add $\{f_i, f_j\}$ to $\boldsymbol{F}_{\text{binary}}$

end

end

end

merge($\boldsymbol{F}_{\text{binary}}$, $\boldsymbol{F}_{\text{interact}}$)

step 2: forward identification

$\boldsymbol{F}_{\text{previous}} = \boldsymbol{F}_{\text{interact}}$

for i = 2 **to** n-1 **do** //i+1 is the degree of feature interactions

$\boldsymbol{F}_{\text{temp}}$ = NULL

for j = 1 **to** $\boldsymbol{F}_{\text{previous}}$.size **do**

$A = \boldsymbol{F}_{\text{previous}}.j$

for $k = j + 1$**to** $\boldsymbol{F}_{\text{previous}}$.size **do**

$B = \boldsymbol{F}_{\text{previous}}.k$

if A.size == B.size && $\{A_1, A_2, \ldots, A_{\text{A.size}-1}\}$ == $\{B_1, B_2, \ldots, B_{\text{A.size-1}}\}$ && $\{A_{\text{A.size}}, B_{\text{A.size}}\} \in \boldsymbol{F}_{\text{binary}}$ **then**

remove duplicate items in A

add union(A, B) to $\boldsymbol{F}_{\text{temp}}$

end

end

end

merge($\boldsymbol{F}_{\text{temp}}$, $\boldsymbol{F}_{\text{interact}}$)
$\boldsymbol{F}_{\text{previous}} = \boldsymbol{F}_{\text{temp}}$
end
step 3: backward selection
rank the feature subsets in $\boldsymbol{F}_{\text{interact}}$ in the ascending order of *IGPF*
foreach feature subset *A* in $\boldsymbol{F}_{\text{interact}}$ **do**
if $IG(\boldsymbol{F}_{\text{interact}}) - IG(\boldsymbol{F}_{\text{interact}} - A) \leq \beta$ **then**
delete *A* from $\boldsymbol{F}_{\text{interact}}$
end
end
for $i = 1$ **to** n **do**
if $f_i \in \boldsymbol{F}_{interact}$ **then**
add f_i to $\boldsymbol{F}_{\text{select}}$
end
end
subroutine: merge(*A*, *B*)
for $i = 1$ **to** *B*.size **do**
for $j = 1$ **to** *A*.size **do**
If $B_i\{B_{i1}, B_{i2},\ldots, B_{ik}\} \subseteq A_i\{A_{i1}, A_{i2},\ldots, A_{il}\}$ **then**
delete B_i from *B*
$i = i - 1$
break
end
end
end
union(*A*, *B*)

==========

Figure 4.1 Algorithm BIFS

4.1 Forward Identification

Forward identification is designed to identify all potential maximum feature interactions. Since our identification of *k*-degree potential feature interactions is based on binary feature interactions, we need to identify binary interactions first. However, definition 2 cannot be applied to identify binary feature interactions to real world problems. In real world problems, we may capture fake

feature interactions using definition 2 directly. There are many reasons to the existence of fake feature interactions. For example, they can be the result of noise in the data. Fake feature interactions can also be due to coincidental patterns stemming from non-representative samples of the source data. In this case, the model generated using definition 2 will overfit the training data. And sometimes, we may not be interested in weak feature interactions. We may just want to identify those strong feature interactions which can tell reliable information about the problem. In order to apply definition 2 to real world problems, we introduce a new parameter α to help us filter out weak and fake feature interactions. In our algorithm, we recognize binary feature interaction between two features f1 and f2 if

$$IG(f_1 \& f_2) > IG(f_1) + IG(f_2) + \alpha \ ,$$

where α is called the degree of interaction and $\alpha \geq 0$. α is a user defined parameter.

In the initialization step, $\boldsymbol{F}_{\text{interact}}$ is initialized with sets of every single feature from feature set $\boldsymbol{F}$ and all identified binary feature interactions are stored in set $\boldsymbol{F}_{\text{binary}}$. Then the subroutine merge($\boldsymbol{F}_{\text{binary}}$, $\boldsymbol{F}_{\text{interact}}$) is called to merge $\boldsymbol{F}_{\text{binary}}$ to $\boldsymbol{F}_{\text{interact}}$. The subroutine merge($A$, B) is designed to merge the newly identified (k+1)-degree potential feature interactions A to B which contains potential feature interactions up to k degrees. If a feature interaction B_{i} of size k from B is contained by a feature interaction A_{i} of size k+1 from A which contains all newly identified (k+1)-degree potential feature interactions, then B_{i} will be deleted from the updated B which is the union of B and A.

Step 2 is the main part of forward identification. In this step, our algorithm searches for maximum closed interactive feature subsets of sizes from 3 to n, where n is the number of features of the dataset. Recall that maximum closed interactive feature subsets are considered to

be potential maximum feature interactions, the purpose of step 2 is to identify potential maximum feature interactions from 3-degree to *n*-degree. 2-degree potential feature interactions are identified in the initialization step. In order to identify (*k*+1)-degree potential feature interactions, our algorithm will examine every possible pairs of feature interactions of size *k* from $\boldsymbol{F}_{\text{interact}}$. If the two feature interactions of size *k* contains *k*-1 features in common and the rest two features have binary feature interaction between them, the union of these two feature interactions will be identified as a (*k*+1)-degree potential feature interaction. By doing this identification process for up to *n*-degree potential feature interactions, we can successfully find all potential maximum feature interactions.

4.2 Backward Selection

At the beginning of backward selection, all the identified feature interaction subsets will be ranked based on *IGPF* in the ascending order. *IGPF* stands for information gain per feature. It is defined as:

$$IGPF(\boldsymbol{F}) = \frac{IG(\boldsymbol{F})}{n} ,$$

where $\boldsymbol{F}$ is a feature subset containing *n* features. Then our algorithm will perform a relevance check for all feature interaction subsets in $\boldsymbol{F}_{\text{interact}}$. Irrelevant feature interaction subsets will be deleted from $\boldsymbol{F}_{\text{interact}}$. At last, features remained in $\boldsymbol{F}_{\text{interact}}$ are selected as the best features for the learning problem. Definition 1 gives an idea about how to judge the relevance of a single feature. It is possible that some subsets in $\boldsymbol{F}_{\text{interact}}$ contain only one feature because this feature has no interaction with other features. But most of the subsets in $\boldsymbol{F}_{\text{interact}}$ contain more than one feature. In this case, we need to know how to judge the relevance of a feature subset. The definition of the

relevance of a feature subset is very similar to the definition of the relevance of a single feature in terms of information gain:

Definition 8 (Relevance of Feature Subsets) A feature subset $\boldsymbol{F}_i$ is relevant iff

$$IG(\boldsymbol{F}) > IG(\boldsymbol{F} - \boldsymbol{F}_i)\,, \qquad subject\ to\ \boldsymbol{F}_i \in \boldsymbol{F}\ .$$

Like definition 2, definition 8 cannot be directly applied to assess the relevance of feature subsets for real world problems either. Due to the existence of noises, outliers and coincidental patterns, irrelevant features may have very weak relevance according to definition 8 and can be wrongly identified as relevant features. In this case, the model trained by the selected features tends to overfits the training data. Thus we introduce another parameter β to help us filter out feature interaction subsets with weak relevance. Our algorithm considers a feature subset as relevant if

$$IG(\boldsymbol{F}) > IG(\boldsymbol{F} - \boldsymbol{F}_i) + \beta\ ,$$

where β is called the degree of relevance and $\beta \geq 0$. β is also a user defined parameter. In most cases, we can use zero as the value for β. But for complicated datasets which may contain noise, outliers and coincidental patterns, α and β need to be adjusted.

4.3 Complexity

To analyze the complexity of BIFS, we must start from analyzing the complexity of the subroutine merge. The complexity of the subroutine merge depends on the size of the two sets being merged. The largest size of $\boldsymbol{F}_{\text{interact}}$ is 2^n [Benjamin and Quinn, 2003]. Thus the worst case complexity of the subroutine merge requires exponential time. However, it requires much less time in practice since there are often not that many maximum closed interactive feature sets in a

dataset. Let λ denote the size of $\boldsymbol{F}_{\text{interact}}$. Checking whether a feature interaction B_i is contained by another feature interaction A_i requires a complexity of $O(n)$. The complexity of subroutine merge can be expressed as $O(n\lambda^2)$. In the initialization step, the maximum size of $\boldsymbol{F}_{\text{binary}}$ is n^2. The size of $\boldsymbol{F}_{\text{interact}}$ is n and the sizes of subsets in $\boldsymbol{F}_{\text{interact}}$ are all one. Thus the complexity of subroutine merge is $O(\text{n}^3)$. Calculating of the information gain for one and two features has a complexity of $O(m)$, where m is the number of examples of the datasets. Thus the complexity of the initialization step can be bounded by $O(max(mn^2, n^3))$. In most cases, the number of examples is larger than the number of features. So the complexity of the initialization step can be expressed as $O(mn^2)$.

Recall that identifying *k*-degree feature interactions using definition 4 requires exponential time, the complexity of identifying a *k*-degree potential feature interaction using the idea of closure problem requires only $O(n)$. However, the complexity of identifying all potential maximum feature interactions depends on the size of $\boldsymbol{F}_{\text{previous}}$. Since the maximum size of $\boldsymbol{F}_{\text{previous}}$ is 2^{n-1}, the size of $\boldsymbol{F}_{\text{previous}}$ can be bounded by λ. In our algorithm, we store all feature interactions in a hash structure. By using the hash structure, the complexity of checking whether a union of two *k*-degree potential feature interactions from $\boldsymbol{F}_{\text{previous}}$ can form a (*k*+1)-degree potential interaction is reduced to $O(n)$ successfully. Now we can draw that the forward identification step has a complexity of $O(n^2\lambda^2)$.

In order to rank the feature subsets in $\boldsymbol{F}_{\text{interact}}$, we need to calculate the information gain for all these feature subsets. This requires a complexity of $O(\lambda mn)$. Since the information gain has been calculated in the ranking step, the complexity of checking the relevance of these feature subsets requires only $O(\lambda)$. Constructing the feature subset $\boldsymbol{F}_{\text{select}}$ has a complexity of $O(n)$.

From above all, the complexity of our algorithm BIFS can be summarized as:

$$O\left(max(mn^2, n^2\lambda^2, \lambda mn)\right)$$

where m is the number of the examples, n is the number of features, and λ is the number of all identified potential maximum feature interactions.

Chapter 5

Results and Discussion

BIFS is implemented using JAVA and all the experiments are conducted in the WEKA framework. In this section, we first talk about how the two parameters α and β can affect the performance of BIFS. And then we will compare BIFS with information gain feature selection of single independent features to show the difference. At the end, we will evaluate the performance of BIFS by comparing it with some well-known feature selection algorithms. We will show the results with regards to the number of selected features and the predictive accuracy using selected features. We apply two learning algorithms to test the performance of our feature selection algorithms. They are C4.5 from the decision tree family and SMO from the SVM family. For evaluation of the models, we use additional testing datasets if they are provided in the original data source. Otherwise, we will use 10-fold cross validation. The datasets we use for our experiments are 4 artificial dataset and 2 real datasets which have been identified having feature interactions by [Jakulin, 2005]. We also include another two datasets: the Tic-Tac-Toe Endgame and the SPECT. All these 8 datasets can be found in the UCI ML Repository. Information about these 8 datasets is summarized in Table 5.1.

Dataset	Number of features	Number of examples	Number of classes
Corral	6	32	2
MONK-1	6	124	2
MONK-2	6	169	2
MONK-3	6	122	2
soy-large	35	306	19
zoo	16	101	7
Tic-Tac-Toe Endgame	9	958	2
SPECT	22	267	2

Table 5.1 Summary of the datasets

5.1 α parameter and β parameter

The α and β parameters are introduced to improve the performance of BIFS when dealing with datasets containing noise, outliers and coincidental patterns. Before describing the two parameters, let's examine how BIFS deals with known feature interactions for four synthetic datasets. The first dataset is Corral which has six boolean features A_0, A_1, B_0, B_1, I, R. The class label is defined by $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$ and features A_0, A_1, B_0, B_1 are independent of each other. Feature I is irrelevant and feature R is redundant. The other three datasets are from the MONKs Problem. The MONKs data has six features a_1, a_2, a_3, a_4, a_5, a_6 and three target concepts:

- MONK-1: ($a_1 = a_2$) or ($a_5 = 1$)
- MONK-2: Exactly two of $\{a_1 = 1, a_2 = 1, a_3 = 1, a_4 = 1, a_5 = 1, a_6 = 1\}$
- MONK-3: ($a_5 = 3$ and $a_4 = 1$) or ($a_5 \neq 4$ and $a_2 \neq 3$) (5% class noise added to the training set)

The maximum feature interactions (MFIs) and selected features identified by BIFS are shown in table 5.2.

	Target Concept (Relevant Features)	Identified Relevant MFIs	Selected Features
Corral	$(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$	$\{[A_0, A_1, B_0, B_1]\}$	A_0, A_1, B_0, B_1
MONK-1	$(a_1 = a_2)$ or $(a_5 = 1)$	$\{[a_5], [a_1, a_2]\}$	a_1, a_2, a_5
MONK-2	Exactly two of $\{a_1 = 1, a_2 = 1, a_3 = 1, a_4 = 1, a_5 = 1, a_6 = 1\}$	$\{[a_3, a_5], [a_4], [a_6], [a_1], [a_2]\}$	$a_1, a_2, a_3, a_4, a_5, a_6$
MONK-3	$(a_5 = 3$ and $a_4 = 1)$ or $(a_5 \neq 4$ and $a_2 \neq 3)$	$\{[a_2, a_4, a_5], [a_1, a_4, a_5]\}$	a_1, a_2, a_4, a_5

Table 5.2 Relevant features and MFIs identified by BIFS for the four synthetic datasets

All the results in table 5.2 are generated by BIFS using a value of 0.05 for both α and β. Maximum feature interactions like $[a_5]$ which contains only one feature means the feature has no interaction with other features. The reason why we don't use 0 is that we don't want BIFS to capture very weak feature interactions, and features having little relevance with the target concept. For Corral, BIFS removes the irrelevant feature *I* and redundant feature *R*. It identifies the 4-degree feature interaction $[A_0, A_1, B_0, B_1]$ successfully. For the MONKs problem, BIFS identifies the correct relevant features and feature interactions for MONK-1 and MONK-2. For MONK-3, BIFS does identify the relevant feature interaction (a_2, a_4, a_5), but it also captures the irrelevant feature a_1 due to the noises. In this case, the values of α and β need to be adjusted. We need larger values for α and β to eliminate the affect caused by noises. For example, if we increase the value of β to 0.2, BIFS can get rid of the noises and recognize only the relevant maximum feature interaction $[a_2, a_4, a_5]$.

To illustrate how the values of α and β can affect the performance of BIFS, we show the results generated by BIFS using different values of α and β for the same four datasets.

0.0 / 0.0	All Identified MFIs	Identified Relevant MFIs	Selected Features
Corral	{[A_0, A_1, B_0, B_1], [I, R]}	{[A_0, A_1, B_0, B_1]}	A_0, A_1, B_0, B_1
MONK-1	{[a_1, a_2, a_3, a_6], [a_2, a_3, a_4, a_5, a_6]}	{[a_1, a_2, a_3, a_6], [a_2, a_3, a_4, a_5, a_6]}	a_1, a_2, a_3, a_4, a_5, a_6
MONK-2	{[a_1, a_2, a_3, a_4, a_5], [a_1, a_2, a_3, a_4, a_6]}	{[a_1, a_2, a_3, a_4, a_5], [a_1, a_2, a_3, a_4, a_6]}	a_1, a_2, a_3, a_4, a_5, a_6
MONK-3	{[a_1, a_2, a_3, a_4, a_5, a_6]}	{[a_1, a_2, a_3, a_4, a_5, a_6]}	a_1, a_2, a_3, a_4, a_5, a_6

(a) α = 0.0, β = 0.0

0.0 / 0.2	All Identified Feature Interactions	Identified Relevant MFIs	Selected Features
Corral	{[A_0, A_1, B_0, B_1], [I, R]}	{[A_0, A_1, B_0, B_1]}	A_0, A_1, B_0, B_1
MONK-1	{[a_1, a_2, a_3, a_6], [a_2, a_3, a_4, a_5, a_6]}	{[a_1, a_2, a_3, a_6], [a_2, a_3, a_4, a_5, a_6]}	a_1, a_2, a_3, a_4, a_5, a_6
MONK-2	{[a_1, a_2, a_3, a_4, a_5], [a_1, a_2, a_3, a_4, a_6]}	{[a_1, a_2, a_3, a_4, a_5], [a_1, a_2, a_3, a_4, a_6]}	a_1, a_2, a_3, a_4, a_5, a_6
MONK-3	{[a_1, a_2, a_3, a_4, a_5, a_6]}	{[a_1, a_2, a_3, a_4, a_5, a_6]}	a_1, a_2, a_3, a_4, a_5, a_6

(b) α = 0.0, β = 0.2

0.05 / 0.0	All Identified MFIs	Identified Relevant MFIs	Selected Features
Corral	{[A_0, A_1, B_0, B_1], [I, R]}	{[A_0, A_1, B_0, B_1]}	A_0, A_1, B_0, B_1
MONK-1	{[a_5], [a_1, a_2], [a_4], [a_3], [a_6]}	{[a_5], [a_1, a_2]}	a_1, a_2, a_5
MONK-2	{[a_3, a_5], [a_4], [a_6], [a_1], [a_2]}	{[a_3, a_5], [a_4], [a_6], [a_1], [a_2]}	a_1, a_2, a_3, a_4, a_5, a_6
MONK-3	{[a_2, a_4, a_5], [a_1, a_4, a_5], [a_6], [a_3]}	{[a_2, a_4, a_5], [a_1, a_4, a_5]}	a_1, a_2, a_4, a_5

(c) α = 0.05, β = 0.0

0.05 / 0.2	All Identified MFIs	Identified Relevant MFIs	Selected Features
Corral	{[A_0, A_1, B_0, B_1], [I, R]}	{[A_0, A_1, B_0, B_1]}	A_0, A_1, B_0, B_1
MONK-1	{[a_5], [a_1, a_2], [a_4], [a_3], [a_6]}	{[a_5], [a_1, a_2]}	a_1, a_2, a_5
MONK-2	{[a_3, a_5], [a_4], [a_6], [a_1], [a_2]}	{[a_3, a_5], [a_4], [a_6], [a_1], [a_2]}	a_1, a_2, a_3, a_4, a_5, a_6
MONK-3	{[a_2, a_4, a_5], [a_1, a_4, a_5], [a_6], [a_3]}	{[a_2, a_4, a_5]}	a_2, a_4, a_5

(d) α = 0.05, β = 0.2

Table 5.3 MFIs and relevant features identified by BIFS using different value of and for the four synthetic datasets

From Table 5.3 (a) and (b), we can see that BIFS fails its task of feature selection. Since we use 0.0 as the value for α, weak feature interactions are captured by BIFS. In this case, even we increase the value of β to *0.2*, BIFS is not able to remove the irrelevant features because the irrelevant features are grouped together with relevant features as feature interactions. If we increase the value of α to 0.05, BIFS is capable of identifying meaningful feature interactions. Table 5.3 (c) shows that BIFS identifies all relevant feature for Corral, MONK-1, and MONK-2. For MONK-3, a value of 0.2 is needed for β to remove the irrelevant feature interaction [a_1, a_4, a_5]. Corral is a special case among these four synthetic datasets. For the four combinations of values of α and β in Table 5.3, BIFS can identify all the meaningful feature interactions and relevant features for Corral because Corral is an ideal dataset without any noises or outliers.

Choosing the best values for α and β is a tricky problem. There are several reasons for this. The larger the value of α, the fewer feature interactions BIFS can find. The larger the value of β, the fewer feature interaction subsets can be remained by BIFS. Especially when few feature interactions are detected by the BIFS, the relevance of features will be determined by the information gain of a single feature and the information gain of a single feature is usually limited compared to feature combinations. In this case, we will need a small value of β. In addition to these, there is one important thing that we should keep in mind. A small value of α which is close to zero may cause that irrelevant features be grouped together with relevant features. It would be very difficult for BIFS to remove irrelevant features.

5.2 Improvement of BIFS against Information Gain Feature Selection

The traditional information gain feature selection (IGFS) algorithm is based on the assumption of feature independence. As discussed before, this is not always true, since there may be feature interactions between features. Thus, features that contain little information itself may be more useful than some relevant features when they are used together with some other features. In this case, traditional information gain feature selection may remove such important features and removal of these features can usually result in loss of information.

In this section, we run experiments for the four synthetic datasets in the framework of WEKA. We compare the performance of BIFS and the traditional information gain feature selection by evaluating the accuracies of two supervised learning algorithms on the four datasets using the selected features. The two learning algorithms are C4.5 and SMO which are available in WEKA. For traditional information gain feature selection, WEKA only provides the ranking of features using the information gain attribute evaluator. In order to compare the performance of BIFS and traditional feature selection, we force WEKA to select the same number of features for traditional information gain feature selection as BIFS. For evaluation of the built models, we use additional testing datasets for the MONKs problem and 10-fold cross validation for Corral. Table 5.4 shows the features that are selected by the two feature selection strategies and table 5.5 shows the results of the two learning algorithms. For SMO, we run the algorithm with a polynomial kernel of three different values: 1(SMO-1), 2(SMO-2), and 3(SMO-3).

	Feature Ranking	Selected Features
Corral	R, A_0, B_1, A_1, B_0, I	R, A_0, B_1, A_1
MONK-1	$a_5, a_1, a_4, a_2, a_3, a_6$	a_5, a_1, a_4
MONK-2	$a_5, a_4, a_6, a_1, a_2, a_3$	$a_5, a_4, a_6, a_1, a_2, a_3$
MONK-3	$a_2, a_5, a_1, a_6, a_4, a_3$	a_2, a_5, a_1

(a) Selected features of IGFS

	Identified Relevant MFIs	Selected Features
Corral	$\{[A_0, A_1, B_0, B_1]\}$	A_0, A_1, B_0, B_1
MONK-1	$\{[a_5], [a_1, a_2]\}$	a_1, a_2, a_5
MONK-2	$\{[a_3, a_5], [a_4], [a_6], [a_1], [a_2]\}$	$a_1, a_2, a_3, a_4, a_5, a_6$
MONK-3	$\{[a_2, a_4, a_5]\}$	a_2, a_4, a_5

(b) Selected features of BIFS

Table 5.4 Selected features of IGFS and BIFS

Decision Tree/C4.5			
	Full Set	BIFS	IGFS
Corral	62.50	81.25	65.63
MONK-1	75.70	88.89	72.22
MONK-2	65.05	65.05	65.05
MONK-3	97.22	97.22	97.22

(a) Accuracy (%) of C4.5

SVM/SMO			
	Full Set	BIFS	IGFS
SMO-1			
Corral	84.38	81.25	78.13
MONK-1	72.22	72.22	66.66
MONK-2	67.13	67.13	67.13
MONK-3	97.22	97.22	97.22
SMO-2			
Corral	96.88	100.00	87.50
MONK-1	100.00	100.00	65.74
MONK-2	97.92	97.92	97.92
MONK-3	91.90	100.00	97.22
SMO-3			
Corral	93.75	100.00	81.25
MONK-1	100.00	100.00	65.74
MONK-2	76.85	76.85	76.85
MONK-3	93.98	100.00	97.22

(b) Accuracy (%) of SMO

Table 5.5 Accuracy (%) of C4.5 and SMO on selected features of BIFS and IGFS

For Corral, the selected features of BIFS and IGFS are almost the same. The only difference is that BIFS chooses feature B_0 instead of *R* which is selected by IGFS. IGFS considers feature *R* is the best feature and it destroys the feature interaction [A_0, A_1, B_0, B_1] by removing feature B_0 from the selected features. This is because IGFS evaluates each feature based on its own information gain independently and feature *R* contains much more information itself than feature B_0. On the contrary, BIFS considers that feature B_0 is more important than feature *R* since it detects a very powerful feature interaction which contains B_0. Because the feature interaction [A_0, A_1, B_0, B_1] contains much more information than feature *R*, BIFS removes feature *R* from the selected features. According to Table 5.4, feature B_0 is ranked fifth among the six features while feature *R* is ranked the first. And from Table 5.6, we can see that the information gain of feature B_0 is 0.004708 while the information gain of feature *R* is 0.287031. However, we can see that both C4.5 and SMO can achieve higher accuracy using the selected features of BIFS for Corral according to Table 6.5. Actually the feature interaction [A_0, A_1, B_0, B_1] can achieve an information gain of 0.9887, while the sum of their own information gain is only 0.1121. The same thing happens to MONK-1 and MONK-3. IGFS destroys the feature interaction [a_1, a_2] by removing feature a_2 from MONK-1 and the feature interaction [a_2, a_4, a_5] by removing feature a_4 from MONK-3. The accuracies achieved using the selected features of BIFS are higher than the accuracies achieved using the selected features of IGFS for both MONK-1 and MONK-2. Ignorance of feature interactions causes poor performance for feature selection algorithms like IGFS when dealing with datasets containing feature interactions.

Features	*R*	A_0	B_1	A_1	B_0	*I*
Information Gain	0.287031	0.075273	0.026312	0.005838	0.004708	0.000758

Table 5.6 Information gain of each feature for Corral dataset

5.3 Compare BIFS with Other Feature Selection Algorithms

In this section, we compare our algorithm with some well-known feature selection algorithms. They are CFS, ReliefF and FCBF. All of these three feature selection algorithms are available in WEKA. Besides these, we also include the results of IGFS. Table 5.1 has given a brief description of the 8 datasets we use for the experiments. The first six datasets have been identified with obvious feature interactions. In order to test the performance of BIFS when dealing with datasets without feature interactions, we also include another two datasets, Tic-Tac-Toe Endgame and SPECT. For these two datasets, we ignore whether they have feature interactions or not. We apply C4.5 and SMO with 3 degrees after feature selection to evaluate the performance of the feature selection algorithms. For evaluation of the built models, we use additional test datasets for MONKs and SPECT since they are provided in the original data source and 10-fold cross validation for the rest datasets. The identified MFIs and selected features of BIFS for the Corral dataset and MONKs problem have been presented in Table 5.4(b). Table 5.7 presents the results generated by BIFS for the rest four datasets.

All Identified MFIs	[15], [28], [14], [21], [1, 6, 22], [1, 6, 29], [1, 6, 13], [1, 19], [3, 6, 29], [1, 6, 8], [26], [1, 3, 6, 7], [4, 6, 29], [3, 6, 13], [18], [3, 6, 22], [23], [4, 6, 22], [1, 4, 31], [1, 4, 6, 7, 10], [5, 6, 22], [4, 6, 13], [1, 4, 6, 7, 9], [2, 6, 29], [1, 4, 33], [1, 4, 6, 30], [3, 6, 7, 10], [2, 6, 22], [3, 6, 8], [1, 4, 5, 6, 7], [5, 6, 29], [1, 6, 16], [1, 6, 7, 9, 12], [1, 6, 20], [2, 6, 13], [1, 6, 32], [5, 6, 13], [35], [1, 5, 6, 7, 12], [3, 6, 30], [4, 6, 8], [1, 6, 17], [5, 6, 7, 9], [5, 6, 7, 10], [34], [3, 6, 7, 9, 12], [3, 6, 32], [3, 6, 16], [1, 6, 12, 24], [4, 6, 7, 9, 12], [11, 12, 24], [4, 5, 6, 7, 12], [2, 6, 8], [7, 12, 24], [2, 6, 7, 9, 12], [3, 6, 20], [5, 6, 8], [4, 6, 32], [2, 5, 6, 7, 12], [4, 6, 16], [9, 12, 24], [4, 6, 20], [3, 6, 17], [2, 6, 10], [3, 6, 12, 24], [5, 6, 30], [5, 6, 12, 24], [2, 6, 30], [4, 6, 17], [2, 6, 16], [4, 6, 12, 24], [5, 6, 16], [27], [2, 6, 32], [2, 6, 12, 24], [2, 6, 20], [5, 6, 32], [2, 6, 17], [5, 6, 20], [5, 6, 17], [25]
Identified Relevant MFIs	[15], [28], [14], [21], [1, 6, 22], [1, 6, 29], [1, 6, 13], [1, 19], [3, 6, 29], [1, 6, 8], [26], [1, 3, 6, 7], [4, 6, 29], [3, 6, 13], [18], [3, 6, 22], [23], [4, 6, 22], [1, 4, 31], [1, 4, 6, 7, 10], [5, 6, 22], [4, 6, 13], [1, 4, 6, 7, 9], [2, 6, 29], [1, 4, 33], [1, 4, 6, 30], [3, 6, 7, 10], [2, 6, 22], [3, 6, 8], [1, 4, 5, 6, 7], [5, 6, 29], [1, 6, 16]
Selected Features	25 selected features: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14, 15, 16, 18, 19, 21, 22, 23, 26, 28, 29, 30, 31, 33]

(a) Results of BIFS for soy-large dataset

All Identified MFIs	[4], [2], [9], [3, 7, 13], [1, 7, 13], [6, 8, 10], [1, 7, 8], [7, 8, 10], [3, 7, 8], [8, 10, 11, 14], [6, 8, 11], [3, 7, 12], [3, 7, 11, 14], [3, 11, 14, 15], [1, 7, 11, 14], [1, 7, 12, 16], [7, 8, 11], [8, 11, 14, 15], [1, 11, 14, 15], [1, 12, 15, 16], [10, 11, 14, 15], [8, 11, 15, 16], [3, 7, 11, 16], [1, 7, 11, 16], [5, 11, 14, 15], [3, 11, 15, 16], [14, 15, 16], [6, 11, 16], [1, 11, 15, 16], [6, 11, 14, 15], [10, 11, 15, 16], [5, 11, 15, 16], [7, 12, 15], [7, 11, 14, 15]
Identified Relevant MFIs	[4], [2], [9], [3, 7, 13], [1, 7, 13], [6, 8, 10]
Selected Features	10 selected features: [1, 2, 3, 4, 6, 7, 8, 9, 10, 13]

(b) Results of BIFS for zoo dataset

All Identified MFIs	[5], [3], [1], [7], [9], [6], [4], [8], [2]
Identified Relevant MFIs	[5], [3], [1], [7], [9], [6], [4]
Selected Features	7 selected features: [1, 3, 4, 5, 6, 7, 9]

(c) Results of BIFS for Tic-Tac-Toe Endgame dataset

All Identified MFIs	[11, 16], [9, 11, 13], [5, 6, 7, 13], [5, 6, 9, 13], [2, 9, 13], [8, 11, 15, 22], [6, 9, 13, 19], [1, 16], [8, 11, 15, 20], [2, 8, 15, 22], [6, 16, 19], [11, 20, 21], [10, 16, 19], [8, 10, 15, 19, 22], [6, 8, 15, 19, 22], [3, 11, 20], [4, 15, 17], [11, 14], [6, 8, 12, 15], [2, 20, 21], [1, 4, 7, 15], [7, 10, 14], [2, 8, 15, 20], [4, 7, 10, 15], [3, 5, 7, 20], [1, 9, 15, 20], [5, 6, 16], [6, 7, 14], [1, 7, 15, 20], [5, 6, 8, 15, 22], [9, 11, 15, 20], [4, 11, 15], [5, 8, 15, 20], [6, 19, 21], [10, 18], [10, 19, 21], [5, 9, 15, 20], [5, 7, 15, 20], [4, 5, 6, 7, 15], [3, 5, 6, 7], [1, 15, 22], [9, 10, 15, 19], [2, 9, 15, 20], [5, 20, 21], [6, 9, 15, 19], [6, 9, 12, 15], [2, 4, 15], [2, 14], [4, 10, 15, 19], [5, 6, 21], [3, 6, 19], [10, 14, 19], [5, 6, 9, 15], [4, 6, 15, 19], [6, 14, 19]
Identified Relevant MFIs	[11, 16], [9, 11, 13], [5, 6, 7, 13], [5, 6, 9, 13], [2, 9, 13], [8, 11, 15, 22], [6, 9, 13, 19]
Selected Features	12 selected features: [2, 5, 6, 7, 8, 9, 11, 13, 15, 16, 19, 22]

(d) Results of BIFS for SPECT dataset

Table 5.7 Results of BIFS for the four datasets: soy-large, zoo, Tic-Tac-Toe Endgame and SPECT

	Full Set	BIFS	CFS	ReliefF	FCBF	IGFS
Corral	6	4	2	5	5	4
MONK-1	6	3	4	3	4	3
MONK-2	6	6	3	6	3	6
MONK-3	6	3	3	3	3	3
soy-large	35	25	14	35	9	25
zoo	16	10	10	16	8	10
Tic-Tac-Toe Endgame	9	7	5	9	5	7
SPECT	22	12	7	17	4	12
average	13.25	8.75	6	11.75	5.125	8.75

Table 5.8 Number of selected features for each algorithm

The number of selected features for each algorithm is summarized in Table 5.8. According to Table 5.8, all algorithms significantly reduce the number of features. The average number of selected features for the 8 datasets are 8.75(BIFS), 6(CFS), 11.75(ReliefF), 5.125(FCBF) and 8.75(IGFS) while the average number of features of the full set is 13.25. We can see that BIFS is comparable to the other feature selection algorithms in terms of number of selected features. Now let's examine the effects of the feature selection algorithms on accuracy.

C4.5							
	Full Set	BIFS	CFS	ReliefF	FCBF	IGFS	**win/loss**
Corral	62.50	81.25	65.63	68.75	68.75	65.63	5/0
MONK-1	75.69	88.89	72.22	88.89	72.22	72.22	4/0
MONK-2	65.05	65.05	67.13	65.05	67.13	65.05	0/2
MONK-3	97.22	97.22	97.22	97.22	97.22	97.22	0/0
soy-large	87.30	89.25	85.67	87.30	83.39	87.95	5/0
zoo	92.08	94.06	93.07	92.08	92.08	95.05	4/1
Tic-Tac-Toe Endgame	85.07	82.99	79.44	85.07	79.44	81.73	3/2
SPECT	75.40	74.87	74.87	75.40	66.31	75.40	1/3
average	80.04	84.20	79.41	82.47	78.32	80.03	
win/loss	4/2		5/1	3/2	6/1	4/2	

(a) Accuracy (%) of C4.5 on selected features

SMO-1							
	Full Set	BIFS	CFS	ReliefF	FCBF	IGFS	**win/loss**
Corral	84.38	81.25	71.88	87.5	87.50	78.13	2/3
MONK-1	72.22	72.22	66.67	72.22	66.67	66.67	3/0
MONK-2	67.13	67.13	67.13	67.13	67.13	67.13	0/0
MONK-3	97.22	97.22	97.22	97.22	97.22	97.22	0/0
soy-large	89.90	90.55	91.53	89.90	86.65	89.90	4/1
zoo	96.04	98.02	97.03	96.04	96.04	96.04	5/0
Tic-Tac-Toe Endgame	98.33	75.78	75.26	98.33	74.11	75.68	3/2
SPECT	73.26	79.14	78.61	70.05	76.47	74.33	5/0
average	84.81	82.66	80.66	84.80	81.47	80.64	
win/loss	3/2		5/1	3/2	5/1	6/0	

(b) Accuracy (%) of SMO with polynomial kernel of degree 1 on selected features

SMO-2							
	Full Set	BIFS	CFS	ReliefF	FCBF	IGFS	**win/loss**
Corral	96.88	100.00	68.75	96.88	96.88	87.50	5/0
MONK-1	100.00	100.00	67.13	100.00	67.13	65.74	3/0
MONK-2	97.92	97.92	63.89	97.92	63.89	97.92	2/0
MONK-3	91.90	100.00	97.22	100.00	97.22	97.22	4/0
soy-large	91.21	91.21	90.88	91.21	81.76	87.95	3/0
zoo	96.04	98.02	98.02	96.04	96.04	97.03	4/0
Tic-Tac-Toe Endgame	99.90	83.40	79.33	99.90	79.44	82.57	3/2
SPECT	67.91	65.24	77.01	66.31	71.12	69.52	0/5
average	92.72	91.97	80.28	93.53	81.69	85.68	
win/loss	3/2		6/1	2/2	7/1	6/1	

(c) Accuracy (%) of SMO with polynomial kernel of degree 2 on selected features

SMO-3							
	Full Set	BIFS	CFS	ReliefF	FCBF	IGFS	**win/loss**
Corral	93.75	100.00	68.75	96.88	96.88	81.25	5/0
MONK-1	100.00	100.00	65.28	100.00	65.28	65.74	3/0
MONK-2	76.85	76.85	60.19	76.85	60.19	76.85	2/0
MONK-3	93.98	100.00	97.22	100.00	97.22	97.22	4/0
soy-large	91.86	92.51	90.55	91.86	82.09	89.58	5/0
zoo	97.03	98.02	97.03	97.03	94.06	96.04	5/0
Tic-Tac-Toe Endgame	99.90	90.81	83.51	99.90	83.51	90.50	3/2
SPECT	71.12	64.17	75.94	65.78	71.12	63.10	1/4
average	90.56	90.30	79.81	91.04	81.30	82.54	
win/loss	4/2		7/1	3/2	7/1	7/1	

(d) Accuracy (%) of SMO with polynomial kernel of degree 3 on selected features

Table 5.9 Accuracy (%) of C4.5 and SMO on selected features

	C4.5	SMO-1	SMO-2	SMO-3	Total
Full Set	4/2	3/2	3/2	4/2	14/8
CFS	5/1	5/1	6/1	7/1	23/4
ReliefF	3/2	3/2	2/2	3/2	11/8
FCBF	6/1	5/1	7/1	7/1	25/4
IGFS	4/2	6/0	6/1	7/1	23/4

(a) Win/loss of BIFS against the compared algorithms and full set over all datasets

	C4.5	SMO-1	SMO-2	SMO-3	Total
Corral	5/0	2/3	5/0	5/0	17/3
MONK-1	4/0	3/0	3/0	3/0	13/0
MONK-2	0/2	0/0	2/0	2/0	4/2
MONK-3	0/0	0/0	4/0	4/0	8/0
soy-large	5/0	4/1	3/0	5/0	17/1
zoo	4/1	5/0	4/0	5/0	18/1
Tic-Tac-Toe Endgame	3/2	3/2	3/2	3/2	12/8
SPECT	1/3	5/0	0/5	1/4	7/12

(b) Win/loss of BIFS against the compared algorithms and full set for each dataset

Table 5.10 Summary of win/loss of BIFS against the compared algorithms and full set

According to Table 5.9, the accuracies generated using the selected features of BIFS are comparable with using all features: average accuracy overall datasets 84.20% (BIFS) vs. 80.04% (Full Set) for C4.5, 82.66% (BIFS) vs. 84.81(Full Set) for SMO-1, 91.97% (BIFS) vs. 92.72% (Full Set) for SMO-2, and 90.30% (BIFS) vs. 90.56% (Full Set) for SMO-3. However, the number of features is reduced by 34%: average number of features over all datasets 8.75 (BIFS) vs. 13.25 (Full Set). In Table 5.9, we also show the win/loss of BIFS against the four compared algorithms and the full set for all the 8 datasets. According to Table 5.7, BIFS identifies no feature interaction for Tic-Tac-Toe Endgame, but it detects feature interactions for SPECT while there should not be any feature interactions in SPECT. As discussed before, real world datasets usually contains noises, outliers and coincidental patterns. We need to adjust the value for α and β

to get a good result. In our experiments, we use uniform values of α and β for all the four real world datasets to test the robustness of BIFS. Thus the values may not be good for SPECT and BIFS may detect fake interactions due to the existence of noise, outliers and coincidental patterns.

According to Table 5.9, BIFS achieve more wins than losses over all the compared algorithms and full set for both C4.5 and SMO. For C4.5, the average accuracy of BIFS over all datasets defeats all the compared algorithms and the full set. For SMO, the average accuracy of BIFS is not as high as that of ReliefF and the full set, but it wins over the other three compared algorithms. And BIFS still get more wins than losses compared with ReliefF over all datasets for both C4.5 and SMO: 3/2 (win/loss) for C4.5 and 8/6 (win/loss) for SMO. The total numbers of win/loss are summarized in Table 5.10. From Table 5.10(a), we can see that the total number of wins is much more than the total number of losses when comparing BIFS with CFS, FCBF and IGFS: 23/4 (win/loss) with CFS, 25/4 (win/loss) with FCBF, and 23/4 (win/loss) with IGFS. Though the total number of win/loss over all datasets of BIFS against ReliefF is only 11/8, the performance of BIFS is still much better than ReliefF given that the average number of selected features 8.75 (BIFS) vs. 11.75 (RelefF). According to Table 5.10(b), the total number of win/loss for the first six datasets which have been identified with feature interactions is 77/7 while the total number of win/loss for the last two datasets is 19/20. The performance of BIFS is much better than the compared algorithms when doing feature selection for datasets containing feature interactions. But BIFS is still comparable with the compared algorithms when doing feature selection for datasets containing no feature interaction.

Chapter 6

Conclusion and Discussion

6.1 Conclusion

Feature selection is a very efficient way to help reduce the dimensionality of the data. Feature selection algorithms are widely used to preprocess the data. However, the existence of feature interaction can severely affect feature selection algorithms that are based on the assumption of feature independence. Even well-known feature selection algorithms can remove useful interactive features unintentionally and this will result in loss of information. Through experiments, we present the effects of feature interaction by showing that the information gain of the feature interaction [A_0, A_1, B_0, B_1] of Corral is 0.9887 while the sum of the information gain of the four single features is 0.1121. Ignorance of feature interactions like IGFS will result in poor performance. We prove that a feature should not be evaluated based on its own merit when working independently; feature selection algorithms should evaluate the whole selected feature subset, instead of each single feature within it.

In this thesis, we present an information-theoretic view of feature interaction. We define feature interaction with information gain. Based on that, we develop an algorithm called BIFS to do feature interaction identification and feature selection. We evaluate our algorithm with regards to number of selected features, identified feature interactions, and accuracy achieved using selected features. The experiments are conducted on six datasets which have been identified with feature interactions and two datasets with no feature interaction. For the four synthetic datasets, BIFS identifies the known feature interactions existing in the datasets successfully. We also compare our algorithm with four well-known feature selection algorithms: CFS, ReliefF, FCBF and IGFS (information gain feature selection). In order to test the performance of the feature selection algorithms, we apply two famous supervised learning algorithms on selected features. By comparison, we show that BIFS wins over the compared algorithms over the six datasets identified with feature interactions. For the two datasets without feature interactions, BIFS is still comparable with the other algorithms.

6.2 Limitations and Future Work

BIFS is a robust feature selection algorithm, especially when doing feature selection for datasets with feature interactions. However, it is ongoing research work with many possible enhancements. Some of the current limitations are given below:

- BIFS has variable performance depending on the values of α and β. A good understanding of the data will help choose the right values for α and β. However, choosing the best values for α and β is a tricky problem. For the time being, we do not have an efficient way to solve this issue. What we can do for now is to try several sets of different values and select the one that yields the best result.

- The complexity of BIFS is $O\left(max(mn^2, n^2\lambda^2, \lambda mn)\right)$. It depends on the number of identified feature interactions. Though there are often not that many feature interactions in data, and BIFS has reduced the time complexity from exponential time, the complexity of BIFS can still be high theoretically.

- Feature interactions identified by BIFS are called potential feature interactions which is defined based on binary feature interactions. They are not actual feature interaction according to Definition 4. Though the results show that BIFS does identify the known feature interactions for the four synthetic datasets, we still lack of proof of that potential feature interactions are actual feature interactions.

- The input to BIFS can only contain nominal features. Datasets that contain numeric features must be discretized beforehand. Otherwise BIFS will treat numeric values as nominal values. This can result in a high complexity when calculating information gain.

Bibliography

[1] [Pang-Ning et al. 2006] Pangning Tan, Michael Steinbach, Vipin Kumar. Introduction to Data Mining. ISBN 0-321-32136-7. 2006.

[2] [Zhao and Liu, 2007] Zheng Zhao and Huan Liu. Searching for Interacting Features in Subset Selection. Proceedings of the 20th International Joint Conference on Artificial Intelligence. Pages 1156-1161. 2007.

[3] [Simon, 2003] Phil Simon (March 18, 2013). Too Big to Ignore: The Business Case for Big Data. ISBN 978-1118638170.

[4] [Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*, McGraw Hill. ISBN 0-07-042807-7.

[5] [Marsland, 2009] Stephen Marsland (2009). Machine Learning: An Algorithm Perspective. ISBN 978-1-4200-6718-7.

[6] [Mohri, Rostamizadeh and Talwalkar, 2012] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar. Foundations of Machine Learning. MIT Press ISBN 9780262018258.

[7] [Duda et al., 2001] Richard O. Duda, Peter E. Hart, David G. Stork. (2001). Pattern Classification (2nd Edition), ISBN 0-471-05669-3.

[8] [Zhu, 2008] Zhu, Xiaojin. Semi-supervised learning literature survey. Computer Science, University of Wisconsin-Madison, 2008.

[9] [Kaelbling et al., 1996] Kaelbling, Leslie P., Michael L. Littman, Andrew W. Moore. Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4: 237 – 285, 1996

[10] [Holland, 1992] John H. Holland. Adaption in Natural and Artificial Systems. MIT Press ISBN 0-262-58111-6. 1992

[11] [Zhang et al., 2002] Zhang, S., Zhang, C., Yang, Q. (2002). Data Preparation for Data Mining. Applied Artificial Intelligence, Volume 17, pp. 375-381.

[12] [Grosan and Abraham, 2011] Crina Grosan, Ajith Abraham. Intelligent Systems: A Modern Approach. Intelligent Systems Reference Library Volume 17, 2011, pp 269-280.

[13] [Murthy, 1998] Sreerama K. Murthy. (1998). Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. Data Mining and Knowledge Discovery, 2, 345-389.

[14] [Kotsiantis, 2007] S. B. Kotsiantis. Supervised Machine Learning: A Review of Classification Techniques. Informatica 3, 2007. Pages 249-268.

[15] [Guyon, Boser and Vapnik, 1992] B. E. Boser, I. M. Guyon and V. N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA, 1992. ACM Press.

[16] [Hastie et al., 2001] T. Hastie, R. Tibshirani, and J. H. Friedman. The Elements of Statistical Learning. Springer, 2001.

[17] [Kohavi, 1995] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. International Joint Conference on Artificial Intelligence. 1995.

[18] [Efron and Tibshirani, 1993] An Introduction to the Bootstrap. Chapman & Hall. ISBN 978-0412042317. 1993.

[19] [Yu and Liu, 2003] L. Yu and H. Liu. Feature selection for high dimensional data: a fast correlation-based filter solution. In ICML, 2003.

[20] [Hall, 2000] M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In ICML, 2000.

[21] [Kononenko, 1994] I. Kononenko. Estimating attributes: Analysis and extension of RELIEF. In ECML, 1994.

[22] [Almuallim and Dietterich, 1994] H. Almuallim and T. G. Dietterich. Learning Boolean concepts in the presence of many irrelevant features. Artificial Intelligence, 69(1-2):279-305, 1994.

[23] [Guyon and Elisseeff, 2003] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. J. Mach. Learn. Res., 3:1157-1182, 2003.

[24] [Kohavi and John, 1997] Kohavi and G. John. Wrappers for feature subset selection. Artificial Intelligence, pages 273-324, 1997.

[25] [Das, 2001] Das. Filters, wrappers and a boosting-based hybrid for feature selection. In proceedings of the International Conference on Machine Learning 2001, pages 74-81, 2001.

[26] [Breiman et al., 1984] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees, Wadsworth and Brooks, 1984.

[27] [Calder et al., 2003] M. Calder, M. Kolberg, E. Magill, S. Reiff-Marganiec. Feature Interaction: A Critical Review and Considered Forecast. Computer Networks, 41, pages 115-141. 2003.

[28] [Qui, 1994] J. R. Quinlan. Comparing connectionist and symbolic learning methods, volume I: Constraints and Prospects, pages 445 – 456. MIT Press. 1994.

[29] [Jakulin and Bratko, 2003] Aleks Jakulin and Ivan Bratko. Analyzing attribute dependencies. In PKDD. 2003.

[30] [Jakulin and Bratko, 2004] Aleks Jakulin and Ivan Bratko. Testing the significance of attribute interactions. In ICML. 2004.

[31] [Shannon, 1948] C. E. Shannon. A mathematical Theory of Communication. In The Bell System Technical Journal. Vol. 27, pp. 379-423, 623-656. 1948.

[32] [Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. On Information and Sufficiency. Annals of Mathematical Statistics 22 (1): 79-86. 1951.

[33] [John et al., 1994] G. H. John, R. Kohavi and K. Pfleger. Irrelevant and The Subset Selection Problem. In ICML. 1994.

[34] [Benjamin and Quinn, 2003] Benjamin, Arthur T. and Quinn, Jennifer J.. Proofs that Really Count: The Art of Combinatorial Proof, The Dolciani Mathematical Expositions 27, The Mathematical Association of America, ISBN 978-0-88385-333-7. 2003.

[35] [Jakulin, 2005] Aleks Jakulin. Machine Learning Based on Attribute Interactions. PhD thesis, University of Ljubljana. 2005.