A NOVEL METHOD FOR FINDING SMALL HIGHLY DISCRIMINANT GENE SETS

A Thesis

by

JASON H. GARDNER

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2003

Major Subject: Electrical Engineering

A NOVEL METHOD FOR FINDING SMALL HIGHLY DISCRIMINANT GENE SETS

A Thesis

by

JASON H. GARDNER

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Erchin Serpedin (Chair of Committee)	Andrew Chan (Member)
Nancy Amato (Member)	Aydin Karsilayan (Member)
Chanan Singh (Head of Department)	

August 2003

Major Subject: Electrical Engineering

ABSTRACT

A Novel Method for Finding Small Highly Discriminant Gene Sets. (August 2003)
Jason H. Gardner, B.S., University of Washington
Chair of Advisory Committee: Dr. Erchin Serpedin

In a normal microarray classification problem there will be many genes, on the order of thousands, and few samples, on the order of tens. This necessitates a massive feature space reduction before classification can take place. While much time and effort has gone into evaluating and comparing the performance of different classifiers, less thought has been spent on the problem of efficient feature space reduction.

There are in the microarray classification literature several widely used heuristic feature reduction algorithms that will indeed find small feature subsets to classify over. These methods work in a broad sense but we find that they often require too much computation, find overly large gene sets or are not properly generalizable. Therefore, we believe that a systematic study of feature reduction, as it is related to microarray classification, is in order.

In this thesis we review current feature space reduction algorithms and propose a new, mixed model algorithm. This mixed-modified algorithm uses the best aspects of the filter algorithms and the best aspects of the wrapper algorithms to find very small yet highly discriminant gene sets. We also discuss methods to evaluate alternate, ambiguous gene sets. Applying our new mixed model algorithm to several published datasets we find that our new algorithm outperforms current gene finding methods.

DEDICATION

To Carrie, my infinitely patient wife.

ACKNOWLEDGEMENTS

I would like to first and foremost acknowledge the efforts of my thesis advisor, Dr. Serpedin. Throughout this entire process Professor Serpedin has shown a remarkably steady hand which, when combined with his natural initiative, has made this thesis rewarding.

I would also like to thank Professors Amato, Chan and Karysilian for taking time out of their busy schedules and serving on my committee. I realize this favor for what it is and for this favor I wish to express my gratitude.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

INTRODUCTION

Bioinformatics (Luscombe *et al*., 2001) is loosely defined as the cross between biology and information technology. The research potential of bioinformatics is spectacular. As a pure scientific challenge, nothing is more interesting than examination, at a very fundamental level, of life itself; particularly our own. Aside from the purely scientific there is the practical. Many researchers and pundits believe that by merging traditional biology with the power of information technology are entering into a period of, among other things, increased life span, designer drugs and genetic enhancement.

With the completion of the Human Genome Project (HGP) and the invention of microarray technology bioinformatics has entered into what many feel is its golden era. For the first time quantitative genome wide analysis of organisms can be quickly performed. This genome wide evaluation promises spectacular insights into biological mechanisms.

As of now there are four general, although related, areas of research using microarray technology. They are, in no particular order:

1. *Gene Finding*. Gene finding, sometimes called chromosomal mapping, involves both discovery of gene structure and location of the genes on the specific chromosome (e.g., Mathé *et al.*, 2002). Gene finding is much more difficult that initially predicted and far from completely understood. It is also rightfully regarded as the foundation of genomic understanding.

2. *Gene Interaction*: Gene interaction and regulation (e.g., DeRisi *et al.*, 1997). This research seeks to answer questions regarding how genes interact with each other during cellular activity. A typical question would relate to how genes might regulate or interact with each other in a dynamic system sense.

This thesis follows the style of *Bioinformatics*.

3. *Gene Prediction*: This area of research (e.g., Golub *et al.,* 1999) involves using microarray experimental data to predict the outcome of certain biological events. For instance, classifying tumors as either benign or malignant.

4. *Gene Manipulation*: Also called genetic engineering. This is the most complex of the four and involves modifying the genome of an organism to improve the functioning of the organism. The scientific and popular press is filled with examples of this line of research. However, to date the modifications have been limited to single genes but in the future multiple gene modifications will become the norm.

In this research, we are concerned with the general problem of genetic prediction using microarrays. More specifically, we are interested in the particular case cancer prediction.

The current state of the art in cancer diagnosis is to perform a biopsy and have a pathologist[1] determine, using qualitative measures such as color or texture, the type of cancer and the levels of malignancy of that cancer. The general goal of this line of research is to use microarrays to turn a qualitative diagnosis by a pathologist into a quantitative diagnosis. This could also lead to a finer understanding of cancer mechanisms and discovery of new cancer sub-types.

While cancer is the vehicle by which we study microarray classification it by no means is the only application. The same technology and techniques may be used to predict or classify any condition that is genetic in nature.

Research Goals

We view the end results of this line of study to be the design and manufacture of microarray machines capable of detecting thousands of different cancers and cancer sub-

[1] *Pathology* is the medical branch concerned with the diagnosis of bodily diseases, including cancer.

types. Currently, for each and every microarray experiment requires measuring several thousand gene fragments (most irrelevant to the problem at hand) and inducing classifiers over them. This process is repeated for every different experimental question.

For a practical system, this is not optimal. To be efficient, we would like to maximize the number of classifications made per microarray experiment. While the number of measurements in a given experiment (dots) is growing, it is and always will remain finite. To pack ever more experimental questions into a finite microarray experiment therefore requires minimizing the number of measurements required per experimental question. Since, as engineers, we would like to construct these maximally efficient practical systems, we view the key research goal of this thesis to propose and evaluate algorithms that find minimum size feature[2] sets that can maximally discriminate between classes.

Before we proceed any further, it is worthwhile to come up to speed on several issues relating the classification of microarray data.

Background Biology

Before any serious discussion of can begin, a thorough review of the underlying biology is imperative.

Genetics

Genetics can be loosely defined as the study in inherited variation. The Augustinian monk Gregor Mendel laid the groundwork of modern genetics in his underappreciated paper, "Experiments with Plant Hybrids." In this paper he first presented the modern notion of the laws of heredity. Prior to Mendel's work traits were either thought to show up randomly or through a process of averaging. Mendel's work showed that neither of the

[2] Please be aware that throughout this thesis the term 'gene' and 'feature' are used interchangeably.

established folk beliefs was correct, traits were in fact inherited via discreet units of heredity[3].

Mendel further showed that this inheritance could be described by a model whereby each parent has two units of heredity but passed only one of the two units of heredity to his or her children. This idea was formalized in his law of independent assortment[4], which stated that traits are inherited independently of each other in a probabilistic fashion[5].

Deoxyribonucleic Acid (DNA)

While Mendel presented the basics of the laws of heredity, the actual mechanism of heredity remained completely unknown. The next major research step was taken by Oswald Avery who in the mid 1940s conclusively proved that DNA carries genetic information. While the function of DNA was now known, the actual structure of DNA remained a mystery. In stepped James Watson and Francis Crick who in the 1950s proved that the DNA molecule was arranged in the famous double helix[6]. Each side of the 'staircase' was a base pair complement (see below) of the other. During cell division the two strands separate and on each strand a base pair complementary copy is created, reproducing the original molecule. Using this mechanism DNA reproduces itself without changing its structure.

It is now known that DNA is a long molecule composed of four *base pairs*: adenine (A), guanine (G), cytosine (C), and thymine (T). To form the double helix the four base pairs bond in a very specific way. Adenine bonds only with thymine and cytosine only to guanine. This process of bonding with a base pair complementary strand of DNA, or RNA, is called *hybridization*. A simple illustration of hybridization is shown below in Figure 1.

---

[3] Sometimes called Mendel's first law of heredity.
[4] Sometimes called Mendel's second law of inheritance.
[5] We now know that this in not necessarily the case. If two genes reside on the same chromosome they cannot be independent of each other as we inherit genes *en masse* by chromosome.
[6] They later shared the 1962 Nobel Prize (with New Zealander Maurice Wilkins) for this discovery.

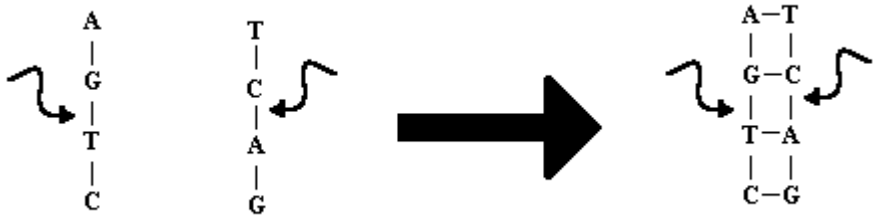

Figure 1. Hybridization of a small DNA fragment.

DNA forms a nucleoprotein and organizes itself in a structure called a *chromosome* that resides in the nucleus of the cell. Chromosomes vary in size and are numbered from largest to smallest where chromosome one would be the largest chromosome. Chromosomes come in pairs; each pair consisting of one chromosome from the mother and the other from the father. Therefore each child inherits one half of its chromosomes from its mother and one half of its chromosomes from its father[7]. During the process of gamete[8] creation, *gametogenesis*, each parent passes randomly to the gamete one chromosome from each chromosome pair.

If DNA is conceptually thought of as a long string of beads tightly wound each bead on the string would be a base pair. A *codon* is an adjacent set of three base pairs. Each of the 64 possible unique codons codes one of 20 different amino acids. Several codons together form one complete gene. Each gene resides at distinct location, or *locus*, on their respective chromosome[9]. Genes are generally considered the lowest unit of heredity[10]. While the length of a codon is fixed, the total length of a gene is highly variable, ranging from several hundred to several hundred thousand base pairs. At a given locus on a particular chromosome there might be different values for different individuals. These alternate locus values are called *alleles* and form the basis of genetic variation in populations.

---

[7] Humans have a total of 46 chromosomes, 23 from mother 23 from father.
[8] Gametes are the sex cells; i.e. the egg and sperm.
[9] Some genes are known to overlap one another, meaning that a particular locus can be part of two separate genes.
[10] Definition of a gene varies widely as is best viewed in the context it is presented.

Ribonucleic Acid (RNA)

Ribonucleic acid (RNA), unlike the double helixed DNA, is a single stranded molecule. Instead of thymine RNA contains the pyrimidine uracil (U) which base pairs with adenine. RNA functions primarily in protein synthesis (see below). As a simplification, RNA can be thought of as the molecule that actually gets the cellular work done. If DNA is thought of as the master instruction book then RNA is the individual instruction.

RNA comes in three different forms, but in all forms RNA is a copy made from a DNA template. The three types of RNA are:

1. *Messenger ribonucleic acid (mRNA)*: mRNA acts as a messenger reading from the master DNA instruction book and conveying the message to the ribosomal sites of protein synthesis.

2. *Transfer ribonucleic acid (tRNA)*: tRNA attaches to amino acids and, using the mRNA instruction, ensures proper construction of protein molecules.

3. *Ribosomal ribonucleic acid (rRNA)*: rRNA is the class of RNA that resides in the ribosomal structures. *Ribosomes* are complexes of rRNA and protein molecules. The main function of ribosomes is to serve as the site of mRNA translation (see below).

While DNA contains the master instruction set for the organism, not every instruction (base pair) is transcribed to RNA. Regions of DNA that are not converted to RNA, and thereby not active, are called *introns*. Regions of DNA that are converted to cellular RNA are called *exons*. Current research indicates that the vast majority of DNA is not coded.

Proteins and Protein Synthesis

Proteins are the basis of cellular life. To oversimplify, they are the complex and numerous molecules that make us who we are. Proteins are involved in every aspect of cellular activity.

Proteins are made of specific amino acids which are connected in very precise ways. Protein synthesis occurs in two steps: *transcription* and *translation*. Transcription is the process, alluded to above, where by DNA is converted to one of the three different types of RNA. Translation is the process of converting the mRNA instructions to a series of amino acids then finally assembling the amino acids into a protein.

The *genome* is defined as the complete collection of all DNA an organism possesses. The *phenotype* is the measurable result of the genome, i.e., the result of the constructed proteins. Differences or traits that are result solely of an organism's genetic make up are defined as *genotypica*l. Measurable properties of an organism that result from the interaction of the organism and the environment are considered *phenotypical.*

Cancer

Cancer is generally considered a condition resulting from one or several genetic mutations. Cancerous cells can be recognized by the following characteristics:

1. *Immortality*: A cancerous cell has the ability to become immortal, meaning it is capable of cellular division without bound.

2. *Independence*: A cancerous cell acts independently of the normal cellular regulating mechanisms.

3. *Invasiveness*: Cancerous cells show a willingness to spread to other tissues in the organism in a process called *metastasis*.

The process of a normally behaved cell becoming cancer is called *oncogenesis* while the genes that cause this transformation are called *oncogenes*. The study of cancerous cells is called *oncology*.

When one cancerous cell divides, eventually the cell will replicate itself into a cluster of similarly cancerous cells called a *tumor*. If the tumor, after surgical removal, does not return the cancer is labeled *benign*. If the tumor does return after surgical removal the cancer is labeled *malignant*.

Oncogenetic mutations can either be inherited or caused by exposure to environmental carcinogens. Oncogenetic mutations come in one of two broadly defined classes: those that are characterized by increased cellular function and those characterized by decreased cellular function. Oncogenetic defects resulting in increased functionality typically cause cancerous cells to replicate without bound (genes turned on). Oncogenetic defects resulting in decreased functionality would typically cause cell suppression mechanisms to fail (genes turned off).

Microarray Technology

Microarrays[11] were developed in the late 1990s to perform simultaneous measurement of thousands of genes. In fact, microarrays can be used to measure all known genes in some simple organisms such as viruses and phages. Before microarrays, analysis of nucleoproteins was performed on a gene by gene basis. This gene by gene evaluation was inadequate, not able to analyze more than a small snapshot of what is believed to be a very complicated system.

[11] "Microarray" is a general term used in this thesis in a very broad sense. Other names for microarrays include DNA chip, DNA microarrays, cDNA microarray, gene array, genome chip and GeneChip®, which is a registered trademark of Affymetrix, Inc.

The key to microarrays is the process of hybridization, discussed above. Remember that two strands of DNA (or RNA) hybridize if and only if they are base pair complementary, meaning that they 'mirror' opposites of each other in a base pair sense. Microarrays exploit this hybridization process to measure the nucleoproteins of a target cell. *Oligonucleotides* are small chains of DNA, typically 20-30 base pairs in length. Oligonucleotides are generally synthesized *in vitro*, meaning in the lab[12].

Microarrays use oligonucleotides and hybridization to measure the type and amount of nucleoproteins in a given cell. To simplify, a microarray is a matrix of small spots, each spot constructed of a number of oligonucleotides. Each of these small spots is termed a *probe*. Cleverly choosing the oligonucleotide determines what nucleoprotein is detected.

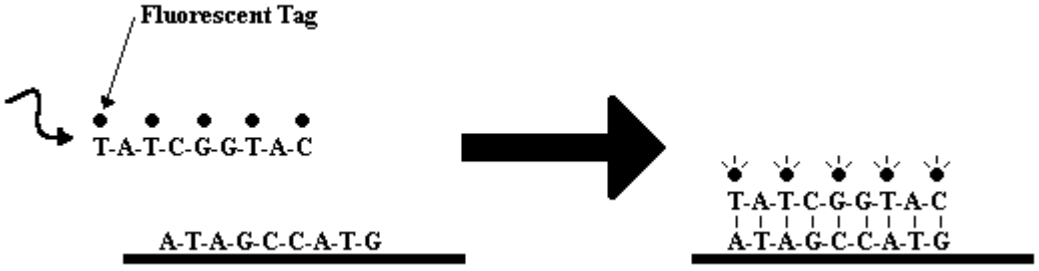

Figure 2. Simplified microarray experiment.

Consider the following simplified example, illustrated in Figure 2, meant to illustrate the basics of microarray functioning. Suppose we wish to detect a specific mRNA (called *expression profiling*). We know beforehand the oligonucleotide TATCGGTAC is unique to this specific mRNA. To detect the presence of the specific mRNA we construct its hybrid complement pair ATAGCCATG and attach it to a thin glass film.

We then amplify and shatter all nucleoproteins in the target cell and label each shattered, target strand with a fluorescing molecule. The fluorescently tagged target nucleoproteins

[12] As opposed to *in vivo*, or in the living body of the organism.

are then washed over the microarray[13]. If any nucleoproteins fragments attach themselves to our probe we have detected the presence of our mRNA.

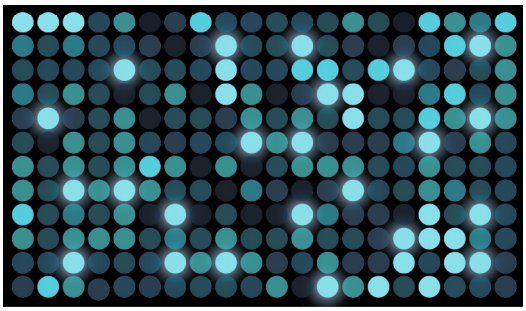

Figure 3. Typical microarray experimental result. (Image credit: U.S. Department of Energy Genomes to Life Program, http://doegenomestolife.org)

Displays can either measure absolute presence (see Figure 3, above) or measure differences between two samples. To measure absolute presence one color is attached to the nucleoproteins and the absolute value of the intensity measured. To differentially detect nucleoproteins two colors are used. One color (red, for instance) is attached to the shattered mRNA fragments of one cell and one color (blue, for example) is attached to the target mRNA fragments of the other cell. As the mixed solution is washed over the microarray if a spot is found to be yellow (red + blue) then there is no difference mRNA expression between the two target cells; if the spot is either red or blue then there is a difference in mRNA expression.

Aside from expression profiling, microarrays can be use to measure all known alleles of a certain gene. This application is called *genotyping*. In this application complementary DNA (cDNA) fragments that are unique to each measured allele are arranged in the spot matrix and labeled target DNA is washed over the slide.

[13] There is nothing standard about microarrays. The technology is very immature and researchers generally make their own microarrays, each with its own slightly different set up.

Basic Classification Theory

It is now worthwhile to review some of the basics of classification theory. We review the simple case of discrimination between two classes knowing that the results are easily generalizable to the multi-class case.

An *observation* is a *d*-dimensional vector **x** (Devroye *et al*., 1996). Each one of the *d*-dimensions is a *feature*, or a measurable property of the sample objects. The *class* is the label associated to the observation and is usually denoted by the variable *y*. While the value of *y* can be any finite label, or *state of nature*, in this discussion we restrict *y* to $y \in \{\omega_{-1}, \omega_1\}$; meaning that all objects classified belong to one of two different labels.

To classify the samples we induce a function:

$$f(\mathbf{x}): \Re^d \rightarrow \{\omega_{-1}, \omega_1\} \quad (1)$$

which maps a vector from the feature space to one of two states of nature. This induced classifier sometimes takes the form of a *classification rule* or *decision rule* (Duda *et al.,* 2001). The Bayes decision rule is perhaps the simplest and most intuitive decision rule. The Bayes decision rule is:

$$\text{Decide } \omega_1 \text{ if } p(\omega_1/\mathbf{x}) > p(\omega_{-1}/\mathbf{x}) \text{ else decide } \omega_{-1} \quad (2)$$

$$\text{Or equivalently, } f(\mathrm{x}) = \operatorname{sgn}[p(\omega_1/\mathbf{x}) - p(\omega_{-1}/\mathbf{x})] \quad (3)$$

where $p(\omega_i/\mathbf{x})$ is the probability the true class is ω_i given **x** is measured, called the *class conditional probability density*. We cannot in general measure $p(\omega_i/\mathbf{x})$. However, if we measure the *prior probability*, $p(\omega_i)$, and the *posterior probability*, $p(\mathbf{x}/\omega_i)$, we can use Bayes rule:

$$p(\omega_i/\mathbf{x}) = p(\mathbf{x}/\omega_i)\, p(\omega_i)/p(\mathbf{x}) \quad (4)$$

$$p(\mathbf{x}) = p(\mathbf{x}/\omega_1)\, p(\omega_1) + p(\mathbf{x}/\omega_{-1})\, p(\omega_{-1}) \quad (5)$$

to find $p(\omega_i/\mathbf{x})$. For most practical problems we never know $p(\omega_i)$ and $p(\mathbf{x}/\omega_i)$ and are therefore forced to estimate. Estimating the prior probability, $p(\omega_i)$, is simple. An obvious estimate is $\hat{p}(\omega_i) = n_i/(n_i+n_j)$, where n_i is the number of samples from class ω_i and n_j is the number of samples from class ω_j. Estimating the class-conditional probability, $p(\mathbf{x}/\omega_i)$, is more difficult; especially if the number of samples are low and the number of features high.

There are two general ways to estimate the class-conditional probability: parametric and non-parametric. To parametrically estimate the class conditional probability, we must first assume a distribution (such as normal, exponential or beta) and then estimate the parameters of the distribution using a parameter estimation technique such as maximum likelihood. Non-parametric estimation does not explicitly rely on the assumption of an underlying model. These methods include Parzen windows and nearest neighbor density estimation.

It is also possible to simply assume a discriminant function and induce the function based on the samples taken. For example, we could assume that the data is linearly separable and use a linear discriminant function, such as a perceptron, inducing the parameters of the discriminant function based on the samples gathered. This method makes no assumptions on the underlying distribution of the feature space which is particularly useful if the feature space is complex or the distribution wholly unknown.

The probability of classification error is simply the probability of incorrect classification. Using the Bayes' classification rule above, if $p(\omega_1/\mathbf{x})$ is greater than $p(\omega_{-1}/\mathbf{x})$ we would label the sample ω_1. Therefore, the probability of correct classification given $p(\omega_1/\mathbf{x}) > p(\omega_{-1}/\mathbf{x})$ and the sample is labeled ω_1 is $p(\omega_1/\mathbf{x})$, while the probability of incorrect classification is $p(\omega_{-1}/\mathbf{x})$. Using this logic and the Bayes classification rule, the probability of correct classification is:

$$p(correct/\mathbf{x}) = \max\left[p(\omega_1/\mathbf{x}), p(\omega_{-1}/\mathbf{x})\right] \quad (6)$$

and the corresponding probability of error is:

$$p(error/\mathbf{x}) = \min\left[p(\omega_1/\mathbf{x}), p(\omega_{-1}/\mathbf{x})\right] \quad (7)$$

Given the above, to minimize error we would like the distribution of $p(\omega_1/\mathbf{x})$ and $p(\omega_{-1}/\mathbf{x})$ to be as different as possible. Different meaning that the difference of the means of the two distributions is very large while the standard deviation of the distributions is very small.

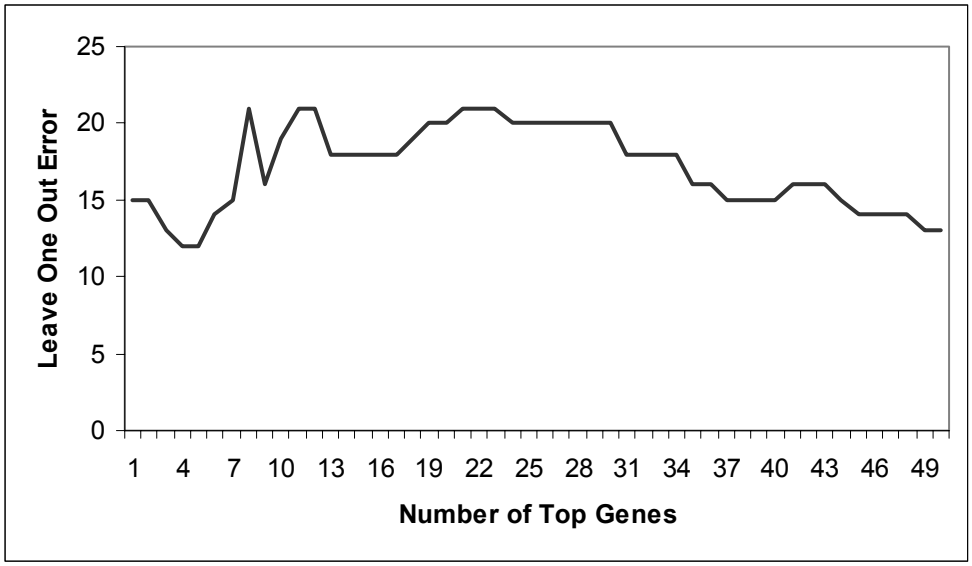

Figure 4. Leave one out error vs. number of features classified over.

While in theory any feature with differing means can be of some use in classification, for practical problems with finite samples and estimated distributions it is often observed that additional features do not necessarily improve classification. Figure 4 above illustrates this point. Using data set C of (Pomeroy *et al*., 2002) we found the 50 most significant genes, using Fisher's Discriminant Ratio. Next, we performed leave one out error

estimation using only the top ranked gene, finding 15/60 errors. This is plotted on the graph as the point (1, 15). We next performed leave one out error estimation on the top two genes, again finding 15/60 errors and plotting at the point (2, 15). The remainder of the graph is filled out in a similar manner.

The results of this exercise clearly show that adding more features, even if the means are very different and the variances are small, does not necessarily improve classification. This graph clearly shows that simply classifying with more features does not necessarily improve classification. In fact, using the full feature set 20 of 60 samples are misclassified. This phenomenon is often attributed to poor density estimation, typically because of poor model selection and too few samples. (Imagine the difficulty of estimating a 10,000 dimensional density with 12 samples.) Feature space reduction is then employed to find those genes that can provide low estimated error. The lesson of this exercise is that care must be exercised in selecting the features of the sub-optimal feature set; adding features in an ad-hoc manner is unlikely to be productive.

FEATURE SPACE REDUCTION

The goal of any feature space reduction algorithm is to retain the necessary and discard the unnecessary. Given a feature space Λ, classification rule ζ, and a classifier $\psi(\Lambda, \zeta)$ that classifies over Λ, we wish to find $\Lambda^+ \subseteq \Lambda$, an optimal feature space that allows the best estimated performance of ψ.

We estimate the accuracy of ψ, given Λ and ζ, by defining an error estimating algorithm, $\xi(\psi,\mathbf{X})$, that estimates the classification error given a particular classifier, implied over a set of features, and a set of samples, $\mathbf{X} = \mathbf{x_1}, \ldots, \mathbf{x_n}$, drawn from the two classes. In this research we use *leave-one-out error estimation*. To estimate the error using leave-one-out, we train (induce) the classifier ψ over (N – 1) samples then test the classifier over the remaining sample. The process is repeated N times, so that every sample is left out exactly once.

While much time and effort has been spent researching different classifiers (e.g., Furey et al., 2000), unfortunately very little has gone into the problem of efficient feature space reduction. We speculate that this is due to the difficulty of the problem at hand; too many features and too few samples.

Evaluating Alternative Sets of Features

Frequently when evaluating alternate sets of genes we are faced with a pair of gene sets where one is not unambiguously better than the other. For example, deciding between a 4 gene set with an estimated 5 errors and a 5 gene set with 4 estimated errors requires introducing the concept of cost. By defining a cost function we can evaluate the total 'cost' of a given gene set and compare it with other dissimilar gene sets. For our research, we define a simple cost function:

$$C = \xi + \kappa * \dim(\Lambda), \tag{8}$$

where ξ is the estimated error, κ is the marginal cost factor, Λ is evaluated gene set and $\subset$ is the total cost of the particular gene set. Larger values of κ value lower dimension gene sets while smaller values of κ value low error solutions. Of course, the value of the penalty is left to the researcher.

As expected, lower costs are always preferred to higher costs. For a fixed cost it is possible that there are many different gene sets with identical costs. If two different gene sets have identical costs we consider the gene sets identical and are indifferent as to which one we prefer. Gene sets with identical costs fall on the same *cost indifference curve*.

We use the cost method because we wish to find gene sets that are both small and highly discriminant. Low estimated error is not good enough by itself but neither is low cardinality. Most researchers up to this point try to find strictly the lowest error gene set. For instance (Ramaswamy *et al*., 2002) found a lowest estimated error gene set, 15 out of 76 wrong for 80% correct, using 128 genes. The researchers asserts that this 128 gene set is the best gene set even though there was an 18 out of 76 wrong for 77% correct gene set using only 8 genes. If the goal is to find small and highly discriminant gene sets there must be a methodology to evaluate trade offs between gene sets, taking into account both the estimated error and the gene set size. By employing the cost method approach we can estimate both properties in conjunction with one another and therefore get a better idea of the overall qualities of the gene sets.

General Feature Reduction Algorithms

The literature distinguishes between two general types of feature reduction algorithms: so-called filter algorithms and wrapper algorithms. The key difference is that wrapper algorithms use an error estimating algorithm and filters do not. To that end we introduce and discuss the general nature of each algorithm below.

Filter Algorithms

Filter algorithms (e.g., John *et al.*, 1994) are both conceptually and computationally simpler than wrapper techniques. Most filter algorithms are based on known statistical measures or heuristically derived. In addition, filter techniques discussed in the literature and below generally have three additional characteristics:

1. Filter techniques assess a features' usefulness with no *a priori* knowledge or consideration to the classifier chosen. (Classifier independence)

2. Filter feature reducers make decisions about the likely suitability of an individual gene, considered without regard to the other genes in the set of genes Λ. (Gene independence)

3. Filter feature space reducers can, in general, detect only linear separations of the samples. (Linear separability)

Given a set of genes Λ**,** a filtering algorithm (function) will evaluate the genes in Λ and, without regard for the classifier, estimate good features. Filters use a so-called filtering function $\Lambda_f = \rho(\Lambda,\varsigma,\alpha)$, where ρ is the filtering function, ς is the ranking criteria and α represents a minimum threshold. For example, if we rank by two sample t-test, the ranking criteria, ς**,** is the p-value of the sample and the function ρ ranks highest t-value to lowest t-value. Those features with a t-value greater than α are included into the set of good features while those features with a t-value less than α are discarded.

Several severe problems, however, arise when employing these methods. First, filtering algorithms completely ignore the effects of the selected feature set on the accuracy of the classifier (Kohavi and John, 1996). The next problem is the arbitrariness of the threshold α. Where is the line that delineates 'good' features that aid in classification from noisy features that only confuse classification? Alternatively, what is the appropriate number of

genes? Is 50 (Golub *et al*., 1999), 100 (West *et al.,* 2001), 500 (Krishnapuram *et al*., 2002) sufficient? These questions highlight a key drawback of filtering techniques; a filtering function can only estimate the relevance of the genes in Λ, it cannot tell how many, or which, genes would actually be useful.

These drawbacks, while substantial, are not meant to imply that filter techniques are useless. Quite the contrary, with (tens of) thousands of genes to evaluate, quick ad hoc feature selection has its place. Another advantage is that these filter techniques quickly and conveniently rank the genes with 'best guess' usefulness. While you would probably feel uncomfortable asserting that the 2^{nd} ranked gene is any more useful than the 3^{rd} ranked gene, you would feel confident that the 2^{nd} is more useful than the 7002^{nd}. They also provide one more key advantage which we will discuss and exploit later.

Wrapper Algorithms

Wrapper algorithms (e.g., John *et al.*, 1994, Kohavi and John 1996) are a general class of algorithms that use more sophisticated and vastly more computationally complex feature set evaluation techniques. However, wrapper algorithms are able to find optimal sets of features.

Wrapper techniques, like filter techniques, conduct searches through subsets of the feature space Λ. They are different than filter methods in that they are not classifier independent or gene independent and, depending on the classifier used, not restricted to linear separations. Wrapper algorithms evaluate the goodness of the set of genes by using ξ to evaluate them. But therein lies the problem; to truly find the best set of genes an estimation of the 'goodness' of each possible subset of features, the power set of the full feature set, must be undertaken. If the feature set is even modestly large this is a computationally daunting, if not practically impossible, task.

For reasons mentioned above, the design of wrapper algorithms is more complex than filter algorithms. Every wrapper algorithm, even a poor one, has four basic parameters; a

starting position, a termination condition, an error estimating function, ξ, and a feature traversal algorithm.

The starting point answers the simple question of where to begin evaluation. For instance, a *backward elimination* algorithm starts with the full set of features while a *forward selection* algorithm starts with no features. Of course, we may also start in the middle or start at a random position.

An ending point answers the obvious question of when to stop evaluating features. A naïve algorithm terminates only after every feature has been evaluated. This is obviously not optimal, as we would like to stop much sooner. More popular choices for termination are estimated error less than a set level, increasing error with more features and more than a certain number of evaluations with no decrease in estimated error.

The choice of an error estimating function determines how the algorithm evaluates each set of features. Equivalently, this parameter determines how we evaluate the goodness of alternative sets of features. The most common method of error estimation is leave-one-out error estimation. There are, of course, other methods to estimate accuracy which we will not cover here.

By far the most difficult question is how to traverse the feature space. If we imagine each of the 2^p possible combinations of the features as a state in a state space, we can view the traversal problem as how to navigate through this state space. At any state the algorithm could go to (evaluate) any other state, conceivably including the state it is already in. To prevent a costly aimless walk through the feature space two common traversal techniques are used. A *greedy algorithm* or *hill climbing algorithm* traverses by finding the best single feature to add (state to travel to next) given its path so far but once adding a feature, the greedy algorithm never re-evaluates its decision to add that feature. Greedy algorithms are popular because they are easy to design, easy to understand and computationally less sophisticated. *Stepwise algorithms* on the other hand consider adding and removing features. Because stepwise algorithms consider discarding genes

they tend to give smaller gene sets. However, these smaller gene sets come at the price of computational complexity.

Current Feature Reduction Methods

We would like to apply statistical classification theory's powerful techniques to the problems of microarrays. However, the well known problem with classification of this microarray data is that the number of genes is much greater than the number of samples provided. This is the 'curse of dimensionality.' For almost every classifier there needs to be many more samples than features. With microarray capable of measuring tens of thousands of genes, the key problem then becomes reducing several thousand genes to small informative sets. To this end there many not entirely disparate measures are used.

Current Filter Algorithms

Currently most researchers use filter algorithms to find small gene sets. These algorithms attempt rate or rank genes, finding those genes that are 'significant' to the given classification problem. To estimate significance we gather a set of samples, $\mathbf{x_1} = \mathbf{x_{11}}, \mathbf{x_{12}}, \ldots, \mathbf{x_{1n}}$ and $\mathbf{x_2} = \mathbf{x_{21}}, \mathbf{x_{22}}, \ldots, \mathbf{x_{2m}}$, drawn from the two classes, ω_1 and ω_{-1}, over a fixed set of features and estimate the features space distributions $p(\mathbf{x_1}/\omega_1)$ and $p(\mathbf{x_2}/\omega_{-1})$, thereby finding the class-conditional probability density. 'Significant' means that the samples show that the class-conditional probability distributions are different and therefore the samples are classifiable with low error.

The literature is filled with a variety of filtering algorithms, most either based on well-known statistical measures or heuristically derived. A good comparative review of several popular criterion methods can be found in (Pan, 2002) and (Troyanskaya *et al*., 2002), but we will review the highlights here.

One of the earliest, most intuitive and most often used filter algorithms used is the *two sample t-test*. The validity of the two sample t-test rests on the assumption of normality of

the underlying distribution. However, this normality assumption, as discussed by (Hunter et al. 2001), is a poor assumption. (Actually we find that the beta distribution fits the data better, both intuitively and empirically.) Because of this, several authors have proposed using non-parametric methods estimating p-values of the two sample t-test by, for example, permuting the data sets. Other groups have tried the *Wilcoxon Rank Sum Test*, another well known parameter free method for testing the differences in distribution of two samples. We have also tried this method but find, in addition to the shortcomings found by Troyanskaya and her colleagues, that the Wilcoxon Rank Sum Test has poor resolution. Hundreds of genes will have the same p-values and discriminating between them is not possible.

Another class of popular filters is based on the idea of correlation with outcome. Various forms of these so-called 'ideal discriminator' methods have been mentioned in the literature (e.g., Golub *et al.,* 1999). These methods seek to find those genes that are maximally correlated to a particular outcome or equivalently they seek to find the genes that are 'most ideal'. Most ideal meaning maximally expressed in one class and maximally repressed in another. *Partial Least Squares*, discussed in (Nguyen *et al*., 2002), is a common technique used in chemometrics which seeks to sequentially maximizes the covariance between the response variables and a linear combination of the predictors. This is similar in spirit to the ideal discriminator method mentioned above in that it finds those genes that are maximally correlated to the outcome.

Fishers Linear Discriminant (FLD) (e.g., Duda *et al.,* 2001) is a popular and well-known matrix based method of feature ranking. FLD attempts to maximize the ratio of between-class scatter and in-class scatter, or similarly it attempts to maximize the so-called Fisher's Discriminant Ratio. FLD involves finding a weight vector, **w,** whose length is the number of genes, which points in the direction of maximum between class differences. By taking the highest magnitude weights (dimensions) and disregarding the lower magnitude weights, feature space reduction is accomplished. However, like all matrix based methods, FLD can be computationally challenging. For example, using FLD

with p genes requires constructing and inverting a p x p matrix. If p is on the order of tens of thousands, this becomes quite challenging.

A method we call *rank and grab* is a common heuristic method in the literature (e.g., Golub *et al.,* 1999). This method first ranks the genes, using any of the filters mentioned above, and then selects some top m genes to form the set of 'good' genes. The choice is either admittedly arbitrary or a function of some p-value cut-off.

Current Wrapper Algorithms

The literature has been very quiet on the subject of wrapper algorithms. Recursive Feature Elimination (RFE), proposed by (Guyon *et al*., 2002) and used by (Ramaswamy *et al*., 2001) is one of the few practical wrapper type feature reduction algorithms the authors have found. RFE trains the weights of an SVM using the full feature set then reduces the data set by iteratively trimming the smallest 10% magnitude weights. Of course RFE works, but it is restricted to SVMs only. For example, RFE is not generalizable to other classifiers such as kNN or neural networks.

Another popular heuristic wrapper algorithm related to rank and grab we call *periodic estimating searches* (e.g., Ramaswamy *et al*., 2002, Pomeroy *et al.,* 2002). Genes are again ranked using any filter. Next, all top gene sets between one and a fixed m are estimated and the best selected. (For example, estimate error on the top ranked gene, the top two ranked genes, the top three ranked and so on up to the top m ranked genes and select the best gene set.)

The methods mentioned above do indeed dramatically reduce the feature space. However, as we will discuss shortly, they suffer from some serious drawbacks.

Wrapper Algorithms and the Microarray Problem

We begin our investigation by thoroughly reviewing the existing feature reduction algorithms and enumerating their respective strengths and weaknesses. Filtering techniques are not evaluated because of their lack of error estimating ability. We do however review several different wrapper algorithms and evaluate their applicability to the microarray problem. We do not review RFE because of its lack of generality.

Exhaustive Searches

The exhaustive search is perhaps the conceptually simplest wrapper algorithm. These searches evaluate every possible gene set, finding the best ones by brute force. This algorithm works for small feature spaces but it is computationally not feasible for microarray problems. For example, suppose there are P genes in the gene space. To evaluate all possible gene sets requires

$$\sum_{i=1}^{P} \frac{P!}{i!(P-i)!} = 2^P - 1 \tag{9}$$

calculations. If P is small, say P = 5, this algorithm needs to evaluate only 31 separate feature sets. This is possible. However, if P = 16,063, as it is in one of our datasets below, then we would need to perform 2.78×10^{4835} evaluations. Obviously, this algorithm has impossible computing requirements and is unsuitable for microarray problems. Further, we assert that the vast majority of the evaluations are on gene sets that are very likely unpromising. Either the gene sets are too large or they contain genes that are irrelevant. Consequently, we will not investigate this algorithm further and not consider it a candidate for our problem.

Limited Exhaustive Searches

Limited exhaustive searches find small gene sets by checking only the small gene sets. Instead of searching the power set of the gene space, limited exhaustive searches check only gene sets of a size less than or equal to p. For example, the limited exhaustive algorithm would evaluate all single features individually then all possible 2-way combinations of features, then all triplets of features, followed by all quartets of features and so on up to the preset size, p. Like exhaustive searches, this algorithm is very computationally expensive. Again, given a microarray experiment with P measured features, checking all gene sets up to size p requires:

$$\sum_{i=1}^{p} \frac{P!}{i!(P-i)!} \tag{10}$$

separate estimations. For a relatively modest P of 10,000 and $p = 4$ this would require performing ~4×10^{14} separate estimations. On the positive, in contrast to exhaustive searches this algorithm will not waste computational resources on gene sets that are too large; however, this algorithm certainly wastes the vast majority of its resources evaluating unpromising sets.

A limited exhaustive search was used by (Kim *et al.*, 2002). In their research they searched every p = 1 and p = 2 gene set and then looked at a subset of 10,000,000 combinations for each $3 < p < 10$. Even this sub-optimal search took two hours to perform on a 72 cluster CPU. To further investigate their algorithm they await a 5,000 CPU supercomputer. Similar to exhaustive searches, we consider limited exhaustive searches unsuitable to the feature reduction problem due to their computational complexity.

Greedy Searches

Greedy algorithms were designed to mitigate the huge computational burden of limited and full feature exhaustive searches.

Greedy algorithms at every step traverse to the state which would give the best marginal benefit. Disregarding any global strategy, greedy algorithms simply focus on the next step. Once this state is found, the greedy algorithm moves to this state, greedily adding the feature but never reconsidering the decision to do so.

Consider a forward selection wrapper algorithm that traverses using a traditional greedy algorithm, arbitrary error estimation algorithm and arbitrary termination condition. Because this is a forward selection algorithm, the initial set of suboptimal genes is empty. The algorithm then begins by estimating, using ξ, every one of the p genes individually and adds the top rated single gene to the set of optimal genes. The algorithm then iterates again, this time searching through the remaining p-1 genes, evaluating each gene in concert with the previously added gene, adding the best gene. This is repeated until the termination condition is met.

Greedy algorithms often work well. Consider Figure 5, below, which is an illustrative example of a forward selection greedy algorithm finding the minimum cost gene set (gene 1 and gene 2). Starting with no features selected the algorithm unambiguously iterates to the lowest cost feature set. The algorithm terminates once there is no path that leads to a state better than the current state. Notice that a backwards elimination would have performed better. (In the figure below the dashed line is greedy path, dashed state is solution state.)

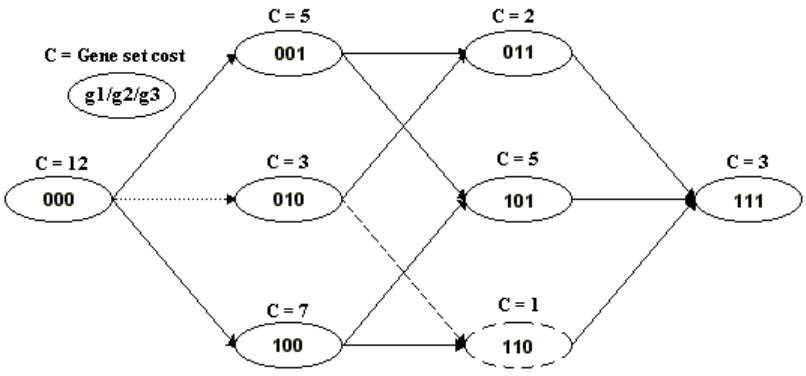

Figure 5. Forward selection greedy algorithm finding optimum gene set.

Greedy algorithms have much lower computational complexity than the exhaustive algorithms studied above. The total computational load depends on the size of the feature space and the number of iterations required to satisfy the terminating condition. If P is the number of features in the feature space and n is the number of iterations required to satisfy the ending condition then the total number evaluations required by the greedy algorithms is:

$$\sum_{i=0}^{n-1}(P-i) \tag{11}$$

If a modestly sized microarray experiment with P = 10,000 genes terminates in n = 5 iterations, the greedy algorithm will perform a total of 49,900 different gene set evaluations. This is clearly less expensive than the exhaustive searches but large nonetheless.

Greedy Algorithm Applied to Microarray Data

However, greedy algorithms can fail; particularly in microarray problems. We applied a greedy algorithm to several different microarray data sets and found that greedy algorithms failed for all data sets[14] but one. We determined that the greedy algorithms failed because, at a given step, there were often hundreds of equivalent paths available, none better than another. Due to this lack of a clear steepest descent[15] we are forced to either exhaustively follow each path (which is computationally too complex) or choose paths based on some arbitrary criteria such as first found. If we exhaustively follow each possible path the computational complexity is too high. If we use an arbitrary criterion, we will find less than optimal gene sets. Remember that we do not need to find thousands

[14] We evaluated ten different data sets from a variety of sources.
[15] Our error estimating algorithm, of course, results in integer errors. With error estimating algorithms that do not result in integer errors, such as bootstrapping or jackknifing, estimated errors that are not statistically different are considered identical.

of low cost solutions. Finding a single low cost solution for our research goals is equivalent to finding every low cost solution.

We apply this algorithm to two different data sets; data set B and data set C, both found in (Pomeroy *et al.,* 2002). For a complete description of the data sets see the results section below.

The first data set, data set C, was chosen because it was the only data set we evaluated where the greedy algorithm worked well. We applied the algorithm and found that it iterated four times, settling on a four gene solution with an estimated 5 errors (total cost of 7 cost units). After every iteration of the algorithm there was a clear best path. Neither PES nor our own algorithm could outperform this result.

The second data set, B, was chosen because it performed in a more typical poor fashion. After the first iteration there were five one gene sets with a lowest estimated three errors. We evaluated each of the five possible paths and found that each of the five paths had several hundred two error gene sets. Iterating further, we easily found hundreds of one error gene sets and thousands of zero error solutions. Because of the total number of possible paths (estimated at perhaps ten thousand) we did not do an exhaustive path search[16]. Remember that we do not need to find thousands of low cost solutions. Finding a single low cost solution for our research goals is equivalent to finding every low cost solution.

Periodic Estimating Searches

Periodic Estimating Searching algorithms are popular (e.g., Ramaswamy *et al.*, 2002 and Pomeroy *et al.*, 2002) gene finding algorithms because they produce reliable results at relatively low computational cost. As described above, PES algorithms work by first filtering the data then evaluating every gene set of top genes from one to a fixed *m*.

[16] We did however find that with real world data there are many hundreds, if not thousands, of identical error gene sets.

The computational simplicity and relatively reliable results of PES makes it very attractive. The computational cost is relatively free of the number of features in the feature set and can be set by the researcher. In fact, the number of evaluations required by the algorithm is *m*.

We can find no better algorithm used by any researcher to date. Because of this we use it as a benchmark to compare our algorithm against. However, PES has one glaring error for our purposes; failing to take into account the size of the 'optimal' gene set. We save a more exhaustive discussion of PES for the results section below.

PROPOSED SOLUTION

After reviewing the existing solutions, we propose our own solution in the hopes that it retains the computational simplicity of PES while finding lower cost gene sets.

Preliminary Definitions

Now, we present and discuss some general definitions that are critical to understanding the microarray feature reduction problem.

Definition 1: Given a set of every possible feature Λ_∞ and *known* distribution *D*, a **truly optimal set of features** $\Lambda^+ \subseteq \Lambda_\infty$ is defined as the smallest set of features which classify with Bayes' Error.

Definition 2: Given a set of features $\Lambda \subset \Lambda_\infty$ and a *known* distribution of those features, an **optimal set of features** $\Lambda' \subseteq \Lambda \subset \Lambda_\infty$, is defined as the lowest cardinality set of features Λ' that classifies with lowest error over a given subset, Λ, of the set of all possible features, Λ_∞.

Definition 3: Given a set of features $\Lambda \subset \Lambda_\infty$, a finite set of labeled samples $s_1, s_2, s_3, .., s_n \in S$ drawn from an *unknown* distribution *D*, and an error estimating algorithm, ξ, a **suboptimal set** of features is defined as the lowest cardinality set of features $\Lambda^* \subseteq \Lambda \subset \Lambda_\infty$ that classifies the samples with lowest estimated error.

The first key idea is that if we know the underlying distribution of every possible feature we can certainly construct a Bayes classifier which has lowest possible error. If *D* is known and every feature is measured perfectly, there can be no ambiguity as to which features aid in classification, and thus we can construct a truly optimal set of features. However, in a real microarray experiment we must decide *a priori* which genes to

measure. Thus we can consider Λ to be the set of features we include in any microarray experiment. We then consider the optimal set of features to be the best set of features given the features we include in our experiment and given a known distribution.

However, the distribution of D is never known and microarray experiments are not constructed with perfect knowledge. Thus, we search for a suboptimal set of features that will provide at the lowest cardinality the best estimated error given a set of labeled samples drawn from D and the measured set of features Λ.

The third key idea is that the estimated error and therefore the suboptimal set of features depend on the labeled samples drawn from D and the genes measured by the particular experiment. Different samples drawn from the same distribution might give a different set of sub-optimal features. Further, different experimental set ups will likely give different sets of suboptimal features. Remember, if we knew the underlying distributions of the features we could certainly find a truly optimal set of features. However, it is assumed that we will never know this underlying distribution so we will never know the truly optimal set of features. Further we can never design an experiment that includes all possible relevant features. Thus we can only *guess* at the optimal set of features, these guesses being suboptimal sets of features, basing the guess on the samples drawn and the genes included in the experiment. *If different samples were drawn and a different experiment conducted, we will likely get a different set of suboptimal features.*

Lastly we must consider the gene finding and error estimating algorithms used. Of course we would not expect every error estimating algorithm to estimate error identically, even if presented with identical samples. Further we would not expect that different gene finding algorithms give identical results. We acknowledge that the choice of error estimating algorithm and gene finding algorithms heavily influences our rankings and estimations. This further adds to the uncertainty as to an optimal set of genes.

Note that for each classification problem there is one truly optimal set of features. Likewise, each unique experimental set up will yield a corresponding optimal set of

features. However, for every experimental set up there are several suboptimal sets of features.

Knowing the above, we must be satisfied with finding any small convenient set of features that classifies with lowest estimated error. We must further acknowledge that we will get different sets of these suboptimal features depending on the parameters mentioned above (error estimating function, gene finding algorithm, experimental set-up, and samples drawn).

Relevance

That brings us to the topic of relevance of an individual feature. The topic of relevance is not inconsequential. Many genes can be 'significant' in the sense that they are not randomly expressed across classes or that they have high p-values. However, we do not seek significant genes. We seek only the necessary *relevant* genes. In other words we seek only those genes that will aid our classification efforts. Consider the following definitions, which were first given by (John et al. 1994) in a classic paper on the subject.

Definition 4: A feature is **strongly relevant** if the feature cannot be removed without the loss of prediction accuracy.

Definition 5: A feature is **weakly relevant** if knowledge of the feature can sometimes add to prediction accuracy.

Definition 6: A feature is **relevant** if the feature is either strongly or weakly relevant.

Definition 7: A feature is **irrelevant** if the feature is not relevant.

For example, suppose we are to classify human adults as either male or female. An example of a strongly relevant feature would be the sex chromosome (XX vs XY). A

weakly relevant feature might be height, lean body mass or shoe size. An irrelevant feature would be number of thumbs, home town, or month of birth. If we disallow the measurement of the sex chromosomes, we expect more inaccurate classifications than if we could measure the sex chromosomes. Further, we would expect that knowing a sample's (person's) height, shoe size or lean body mass might improve prediction accuracy but knowledge of the sample's home town, number of thumbs or month of birth would not.

Assumptions

We begin the discussion of our algorithm with the assumption that truly optimal set of features (genes) can *never* be known or found. We further assert that finding the optimal set of features is also impossible and moreover is not even worth estimating. *Only a suboptimal set of features is worth finding (estimating)*. We justify this in the problem of microarrays thusly:

The human body is estimated to have on the order of tens of thousands of genes. Each gene has many possible alleles. To measure every possible gene and all possible alleles of those genes in any one experiment or set of experiments is considered not possible. We then assert there is no reason to believe that all relevant genes are incorporated in *any* microarray experiment, no matter how well designed. As discussed above, searching for an optimal set of features, using any algorithm, depends exclusively on the genes incorporated in the experiment, the samples drawn and the algorithms used. If the gene is not in the experiment, no predictions can be made about it. Since we cannot be sure that some relevant genes were not left out, and we recognize the randomness of any sample, we cannot be sure of *any* set of features we find is truly the best set in any global sense. Therefore, we should focus our search on finding convenient 'good enough' or 'best estimation' suboptimal sets.

These assumptions free us from worrying about whether we have found the 'right' answer. By realizing that we will never get the 'right' answer, we can focus our search on

finding 'good enough' answers given the experiment and samples drawn, and disabuse ourselves from the thought that the features we find have any actual meaning outside of simple indicators. We should therefore only search for answers that work, given the samples drawn and parameters of the experiment[17].

Our goal is to find a very small set of genes that provides highly accurate estimated classification. We know that a filter algorithm will never take us to this goal by itself. We also know that a wrapper algorithm will take us to our goal, but needs to be designed properly to avoid taking too long to get there. Further we know that for any wrapper algorithm to be practically useful we need to design the algorithm by assigning the four parameters cleverly.

As discussed above, filter gene reduction techniques have the advantage in that they can quickly sort through the data and find 'best guesses' as to which features will be useful. For our algorithm, we assume the following general properties of a 'good' filtering algorithm with respect to the data at hand:

1. Strongly relevant features will generally be highly ranked by any good filtering algorithm.
2. Weakly relevant features will also tend to be highly ranked by any good filtering algorithm, but generally not ranked as high as strongly relevant features.
3. Irrelevant features will tend to be low ranking genes.
4. Most features will be irrelevant.

Our Modified Algorithm

We now propose our algorithm. What we propose is a modification of the traditional greedy (and step wise algorithms) that retains the simplicity of PES algorithms, but finds lower cost gene sets. We present it in the following manner.

[17] Remember that while investigating greedy algorithms we found that there are many gene sets with identical cost. Since theses identical cost gene sets are on the same indifference curve, we have no reason to prefer one over the other. We need not search for *every* low cost gene set; we need only to find one.

If we consider the filtered set of ranked genes, we notice that the ranked set of genes provides some natural answers to our problems. Specifically, we should expect that the highest ranked gene is most likely to be a strong feature. Naturally we would like to start where it is likely that the strong features reside, therefore we assign the starting point of our modified wrapper algorithm to be the highest ranking gene. We further expect that the next highest ranked gene is the next most likely to be a strong feature so we would like to investigate this gene next. Continuing this logic we assert that traversing down the ranked list is the optimum search path. We then assign our feature traversal algorithm accordingly.

However, as we traverse down the ranked set of genes we realize that not every gene is needed. Some genes of the top ranked will be strongly relevant, some weakly relevant and some possibly irrelevant. We propose traversing down the list greedily, adding the next gene down the ranked list only if it lowers cost. By greedily traversing the ranked list we hope to add to our optimum set of genes all strongly relevant genes and only those weakly relevant genes that are shown to improve the gene set. This is opposed to the PES algorithm, which adds every gene on the ranked set whether it is relevant or not. This modification we expect will substantially reduce gene set sizes and therefore reduce costs.

As a result of assumption 4 above, the initial set of genes should again be empty. The termination condition is less well defined and can be tailored to the problem at hand or the fancy of the researcher. We find that stopping when a zero estimated error gene set is found or stopping after the algorithm investigates the top 50 ranked genes to be wholly adequate. (In fact, we find that after the first several genes have been investigated rarely does the gene set change significantly. We further find that if investigation continues into the hundreds or thousands, it is almost certainly a waste of time.)

Our algorithm requires specifying only a filter, a traversal scheme (greedy vs. stepwise), a terminating condition and an error estimating algorithm. Any filter or error estimating

algorithm can be used but the better the filter, the shorter the search and the better the error estimating algorithm the more confident we feel in the answers.

RESULTS

To compare techniques we will use six distinct datasets we found in the literature. (All four can be found at http://www-genome.wi.mit.edu/cgi-bin/cancer/datasets.cgi) We evaluate the performance of our method using two filter methods: Fisher's Discriminant Ratio and our own Ideal Mean Square Error (IMSE) filter. (See Appendix A for details.) The two traversal schemes, greedy and stepwise, are also used for illustrative and comparative purposes. Error estimation was done using a kNN-leave one out error estimating algorithm. Combining the two filters with the two traversal schemes gives a total of four combinations to evaluate. Each of these four combinations results in (possibly) different suboptimal-candidate set of genes which we present for illustration and comparison. Note that unless otherwise specified, only the top 100 ranked genes are evaluated. The results of our algorithms are then compared to what would be achievable using a PES algorithm filtered with the same filter.

We will employ the cost methodology discussed above in evaluating the gene sets. The marginal cost factor, κ, for this thesis is set to ½.

Again, while reading the results of the research, it is important to keep in mind that any combination of filter algorithms, traversal schemes, ending conditions and error estimating algorithms could be used. For this research we chose to use FDR because it generally performed well and chose IMSE because it generally performed poorly. Our error estimation algorithms were selected strictly by familiarity and simplicity.

Metastatic vs. Primary Adenocarcinomas

In (Ramaswamy *et al.*, 2002) the molecular nature metastasis is studied using gene expression data. Specifically, the researchers seek to discriminate between metastatic adenocarcinomas and primary adenocarcinomas. In this study 16,603 genes expressions were measured over a total of 76 samples. Twelve of these samples are from metastatic

adenocarcinoma nodules and the remaining 64 samples are drawn primary adenocarcinomas.

In their research Ramaswamy and his colleagues provide a clear illustration of existing methods for finding small highly discriminant gene sets. Ramaswamy and his colleagues ranked the data using a filter quite similar to FDR and then estimated error on gene set sizes of p = 1, 2, 4, 8, 16, 32, 64, 100, 128, 150, 200, 256, 512, 1024, 2048, 4096, 8192 and 8716 finding that the gene set containing the top128 genes[18] provides the best estimated error of 15/76 (cost = 79). It is not stated specifically why these numbers are chosen. The error estimating algorithm used a weighted voting classifier similar to the one found in (Golub *et al.*, 1999) and leave one out error estimation.

This is clearly a PES algorithm, but in this application not every gene size is evaluated and gene sets that are clearly too big to be useful are evaluated. Using this as a baseline we begin our investigation. We first try to replicate their results and find that by taking 128 of the top genes, as ranked by FDR, we estimate 13/76 errors. This is similar to the original results[19].

We now begin our investigation of the data, taking gene set costs into account. First, all 16,063 features were ranked using our FDR and IMSE filter and the PES algorithm applied. We select k = 3 due to the small number of metastatic samples. The x-axis of Figure 5 represents the number of genes in the gene set. For example, the point 50 represents taking the top 50 rated (by the corresponding filter) genes and estimating the cost of that 50 gene set with a 3NN-leave one out error estimating algorithm. The y-axis represents the estimated cost of the geneset.

We first apply the 3NN-FDR-PES algorithm to the data and find that there is a low cost gene set with an estimated 7 cost units (4 errors with 6 genes). The 3NN-IMSE-PES

[18] In the original paper p =128 was stated as the best gene set. However, in the supplementary notes the group states that the 100 gene geneset and the 128 gene geneset had identical errors.
[19] We find that results are never exactly reproducible. We believe this is because the highly heuristic nature of the classifiers used and the slight differences in how those algorithms are actually implemented.

algorithm finds a 12.5 cost unit gene set (12 errors with 1 gene), finding this lowest error gene set on the first iteration.

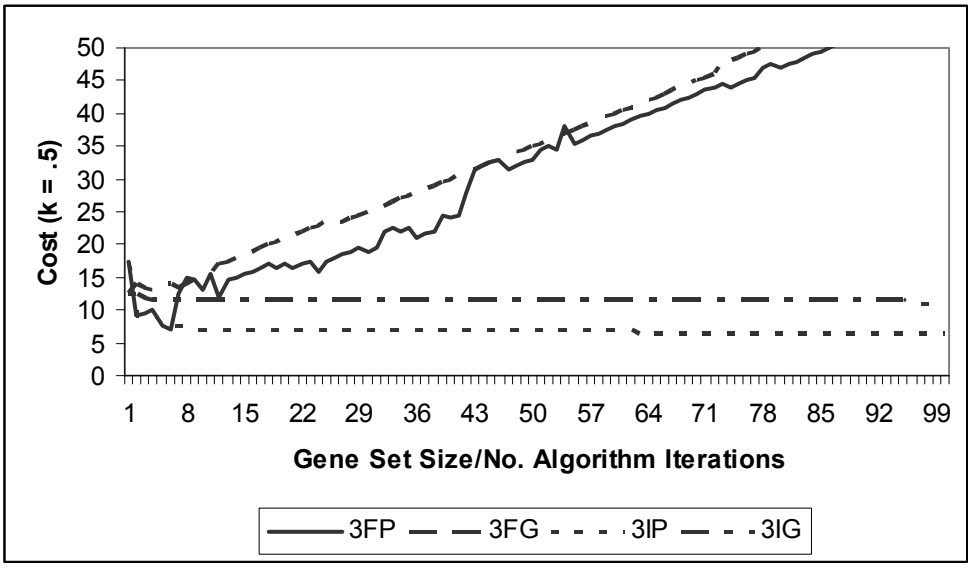

Figure 6. Cost curves for metastatic vs. primary adenocarcinomas dataset. (Key is arranged as kNN/Filter/Traversal. i.e. 3FP = 3NN-FDR-PES and 3IG = 3NN-IMSE-Greedy)

Using these results as our baseline, we now introduce our algorithms and evaluate their performance. Figure 6 also presents the results of our modified algorithms. The x-axis of the figure now represents the number of iterations of the algorithm; or how far down the ranked set of genes the algorithm went in finding its solution. The y-axis shows the estimated cost after each iteration of the algorithm. First we apply the 3NN-FDR-Greedy algorithm and find a 6.5 cost unit solution (4 errors using 5 genes) in 63 iterations. The 3NN-FDR-Stepwise algorithm finds a 5.5 cost unit solution (4 errors using 3 genes) in 31 iterations. (The stepwise results are never shown to avoid clutter on the graph.) The 3NN-IMSE-Greedy algorithm found an 11 cost unit (9 errors using 4 genes) in 96 iterations. The 3NN-IMSE-Stepwise algorithm finds a 9.5 cost unit (8 errors using 3 genes) in 14 iterations.

We will save a complete analysis of all the results in this section for later. But preliminarily we notice that our algorithm outperformed the PES algorithm in all four instances.

Leukemia Dataset

Our second dataset is the popular dataset first presented in (Golub *et al*., 1999). Golub and his colleagues investigated genetic differences between acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). 7129 genes were quantitatively measured over 72 samples, 25 AML samples and 47 ALL samples.

In the original work, the researchers divided the samples into testing and training sets. The training set consisted of 38 bone marrow samples, of which 27 were ALL and the remaining 11 were AML. The researchers then applied their own heuristic algorithm and found that 1100 genes were highly correlated with between class distinctions. Top genes were evaluated using leave one out error estimation and their own 'weighted voting' classifier. Golub and his colleagues' weighted voting scheme allows each gene a 'vote' (by nearness) on an unknown sample with the vote being weighted in the final tally by idealness of the voting gene. They also define a criterion called prediction strength that measures how close the vote was. If the vote was close, the prediction strength is low; if the vote was a landslide the prediction strength is high.

The group settled on a 50 gene model (chosen arbitrarily) that reflects the 50 genes most correlated with discrimination between the classes. This is a good illustration of the Rank and Grab algorithm discussed above. Applying their error estimating algorithm and filter to the top 50 genes, they found that 36 of the 38 samples could be predicted with high prediction strength. The other two were ambiguous, having low prediction strength. All 36 of the high prediction strength samples were classified accurately (cost = 27).

Next their attention was focused on the remaining 34 samples, of which 14 samples are AML and the remaining 20 ALL. These samples were considered the independent or test

set. While the first 38 samples were drawn from bone marrow alone, the second set of 34 had 10 samples that were drawn from peripheral blood samples and 24 samples drawn from bone marrow. The same classifier used on the training set was applied to the test set and 29 of the 34 samples were classified with high prediction strength. The 29 high prediction strength samples were accurately classified.

We now establish a baseline by applying the PES algorithms. We selected k = 3 due to the small number of samples from both classes. Using the 3NN-FDR-PES algorithm, we find that there is a minimum cost of 2, which was found in two gene sets (1 error at 2 genes and 0 errors at 4 genes). The 3NN-IMSE-PES algorithm found a minimum cost gene set with cost of 3.5 (1 gene set with 3 estimated errors) found on the first iteration. From these results, it appears that FDR is a better filter than IMSE for this particular problem. These results are presented in Figure 7, below.

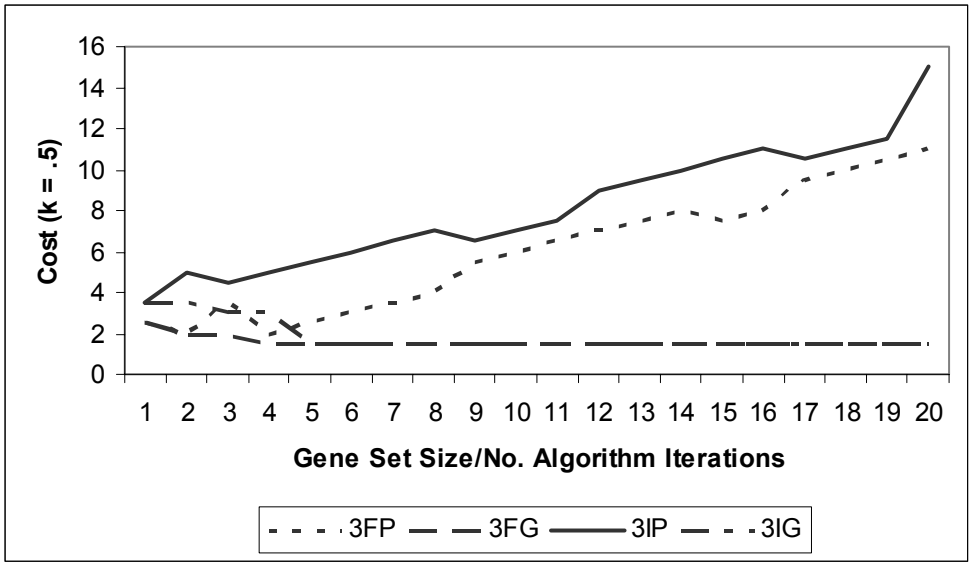

Figure 7. Cost curves for leukemia training dataset.

The figure also displays the results of our greedy algorithms. (In this particular case both the greedy and stepwise versions of the algorithm perform identically for both filters.) The figure shows the 3NN-FDR-Greedy and Stepwise algorithms finding a 1.5 cost gene set (3 genes at 0 errors) in only four iterations. The 3NN-IMSE-Greedy and Stepwise

algorithms find another 1.5 cost gene set (3 genes at 0 errors) result, again in four iterations, but with two of the three genes different from the 3NN-FDR-Greedy and Stepwise gene set. Notice how quickly the four algorithms find zero error solutions.

We now diverge from the original research. We now simply lump the samples into two classes with 25 samples of AML and 47 samples of ALL. Due to the larger number of samples we use a 5NN classifier instead of a 3NN classifier. This was not done in the original research but we feel it is a worthwhile exercise nonetheless.

As a starting point we notice, using the 5NN-FDR-PES algorithm, that there are two genesets with a minimum cost of 5 (3 errors using 4 genes and 2 errors using 6 genes). The 5NN-IMSE-PES algorithm finds a minimum cost of 8.5 (7 errors using 3 genes). For comparison, the minimum error solutions found by both algorithms were 1 error using 28 genes and 4 errors using 14 genes for FDR and IMSE respectively.

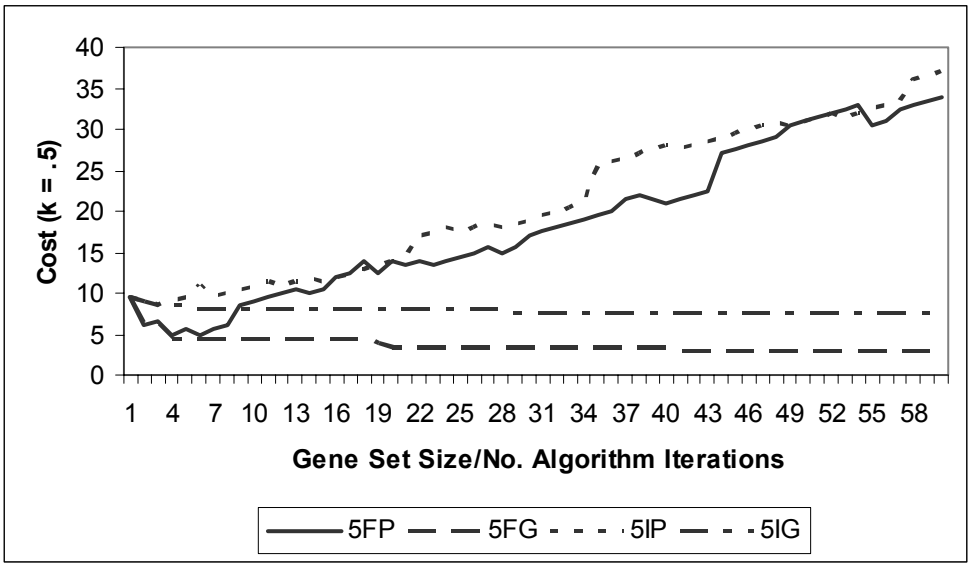

Figure 8. Cost curves for leukemia combined dataset.

Using the results of the PES algorithms as a baseline we apply the four algorithms to the combined data set. Figure 8, above, shows the results of this exercise. The FDR-5NN-Greedy algorithm finds a minimum cost of 3 (6 genes with zero errors) in 41 iterations

while the 5NN-FDR-Stepwise algorithm found a 2.5 cost gene set (3 gene set with one error) in 5 iterations.

The 5NN-IMSE-Greedy algorithm finds a 7.5 cost gene set (5 genes with 5 errors) in 29 iterations while the 5NN-IMSE-Stepwise algorithm found a much better 3.5 cost unit (3 genes with 2 errors) gene set in 68 iterations.

We would like to press a point here. When we applied the FDR-3NN-Greedy algorithm to the 34 sample training set of data we found a 1.5 cost unit geneset (3 gene with an estimated zero errors). When we applied the same FDR-5NN-Greedy algorithm to the combined data set of 72 samples we found a 3 cost unit geneset (zero estimated errors using 6 genes). Although the additional samples were drawn from the same distribution and the error estimation algorithms changed only slightly we notice a very different low cost solution. In fact, no genes are common between the two zero error gene sets. The algorithm found two completely different zero error solutions. This was not a surprise; the different samples and different error estimation algorithms did indeed produce different sub-optimal sets of genes.

Prostate Cancer Survival

Our third dataset was presented in (Singh *et al*., 2002). Singh and his colleagues investigated predicting clinical outcomes of prostate cancer. Twenty-one patients were evaluated, eight having relapsed within four years and thirteen not. In the original work, Singh and his colleagues used a 2NN classifier[20]. The researchers found that they could separate the samples with 'greater than 90% accuracy'[21], using five genes. The five genes were found by ranking the genes, using a filter similar to FDR, and then classifying over all geneset sizes of 1 to 200 and selecting the lowest error geneset. (In other words, they used the PES algorithm.)

[20] Not known why 2NN was used or to which class ties were assigned.
[21] We assume this means that they had one misclassification.

We first investigate what happens when genes are added to the gene set using arbitrary gene set sizes. The two algorithms 3NN-FDR-PES and 3NN-IMSE-PES were applied to the data and the results shown in Figure 9. As we can see from the figure, the 3NN-FDR-PES algorithm finds 4 cost gene set (1 estimated error using 7 genes). We believe this is similar what Singh and his colleagues found. The 3NN-IMSE-PES algorithm does better, finding a 4 cost gene set (3 estimated errors using 2 genes).

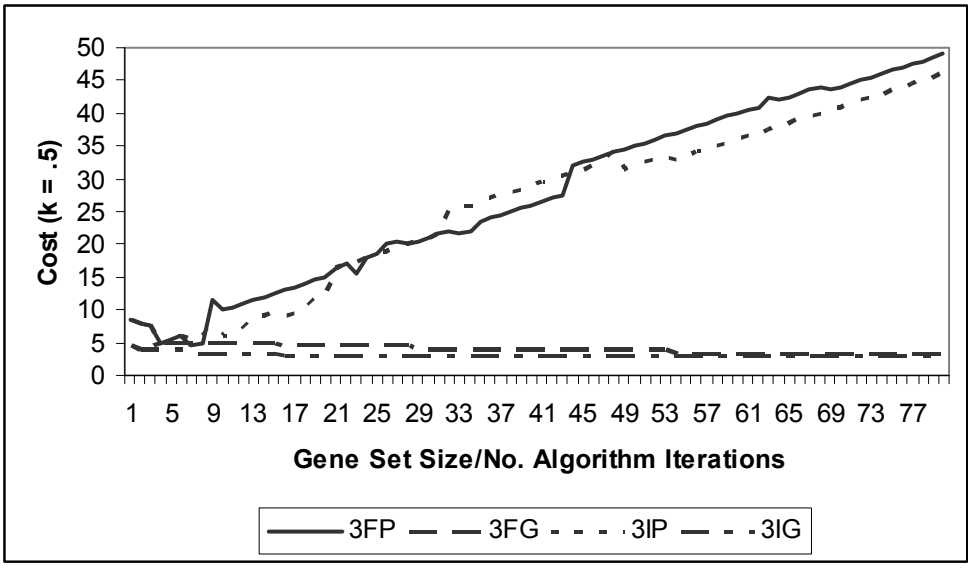

Figure 9. Cost curves for prostate cancer survival dataset.

We try all four different combinations of induction algorithms and filtering algorithms and find notable results. The results of the two greedy combinations are presented in Figure 8, above. Again, all four algorithms found gene sets that had lower total cost than the existing PES algorithms. The 3NN-FDR-Greedy algorithm finds a 3.5 cost gene set (7 genes with no errors) in 54 iterations while the 3NN-FDR-Stepwise algorithm found a 1.5 cost gene set (3 genes with no errors) in 17 iterations of the algorithm. What is interesting about the runs of these algorithms is that two separate zero error solutions were found, one with 7 genes another with 3, but neither having any genes in common. The 3NN-IMSE-Greedy algorithm found a 3 cost gene set (4 genes with 1 error) in 16 iterations and 3NN-IMSE-Stepwise algorithm found a 2.5 cost gene set (3 genes with 1 error) in 16

iterations of the algorithm. Again, we save further analysis until later in the results section.

Medulloblastoma Survivor Prediction

In (Pomeroy *et al*., 2002) Pomeroy and his colleagues investigated whether the outcome ('survivor' vs 'failure') after treatment for medullablastomas could be predicted from gene expression data. Data was gathered for 7129 genes over 60 samples, with 39 successes and 21 failures.

In the original work the data was filtered by a filter very similar to FDR and then error was estimated using a self-developed 'weighted kNN' classifier choosing k to be 5. They evaluated all gene sets of size two to two hundred and found an eight gene set to be optimal. Again, this is clearly a PES algorithm. This eight gene set classified with an estimated error of 13 out of 60 samples for a total cost of 17 cost units. We found roughly the same thing. As is obvious from the accompanying figures the data is difficult to separate.

Observation of Figure 10 gives us our starting points. We again establish a baseline using the two PES algorithms. The 5NN-FDR-PES algorithm finds a 14 cost unit solution (12 estimated errors using 4 genes). Applying the 5NN-IMSE-PES algorithm to the data results in a two different 22 cost unit solutions (21 errors with 2 genes and 20 errors with 4 genes).

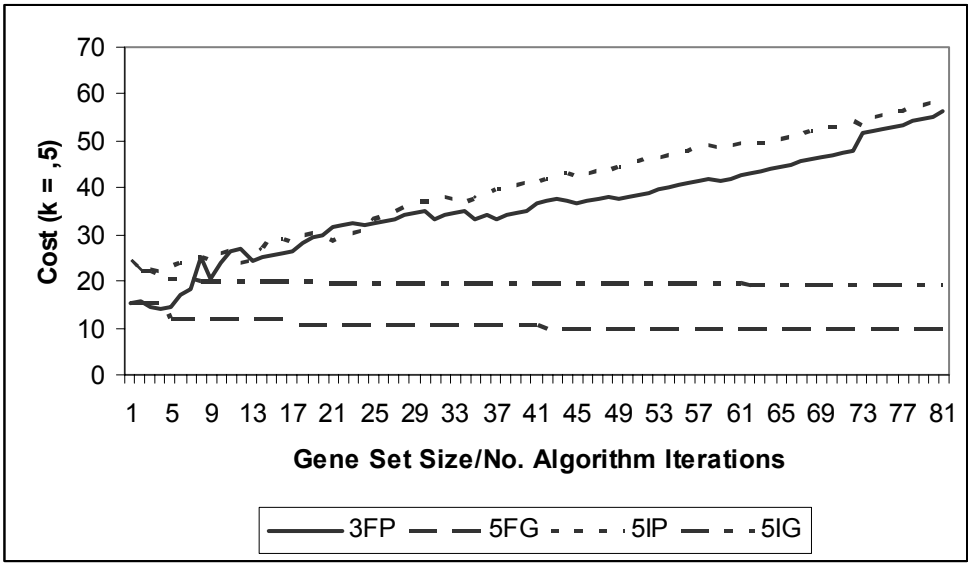

Figure 10. Cost curves for medullablastoma survivor prediction dataset.

After using the PES algorithms to establish a baseline, we again apply our four algorithms. Applying the 5NN-FDR-Greedy algorithm results in a 10 cost unit solution (8 errors using 4 genes) in 42 iterations of the algorithm. The 5NN-FDR-Stepwise algorithm has identical results. The 5NN-IMSE-Greedy algorithm finds a 19 cost unit solution (16 estimated errors using 6 genes) in 62 iterations while the 5NN-IMSE-Stepwise algorithm does much better, finding a 14 cost unit solution (11 estimated errors using 6 genes) in 80 iterations. Interestingly, the 5NN-IMSE-Stepwise algorithm found a geneset with the same number of genes as the 5NN-IMSE-Greedy algorithm but with 5 fewer errors. Further, the genesets had only two genes in common.

Desmoplastic vs. Classic Medulloblastomas

A different question asked in the same paper as above was whether desmoplastic and classic medulloblastomas are distinguishable from gene expression patterns alone. Again 7129 gene expression profiles were gathered, this time over 34 samples. Of the 34 samples 25 are of classic medulloblastomas and 9 are of desmoplastic medulloblastomas.

Pomeroy and his colleagues found that using their weighted kNN classifier (unknown k) they could distinguish between the classes making only 1/34 errors. From reading their research we were unable to the number of genes but the genes were again filtered using a filter roughly equivalent to FDR.

Again we set about preliminarily evaluating the data. For our comparison, we select k = 3 and apply the PES algorithms. The 3NN-FDR-PES algorithm finds a 5 cost unit gene set (4 genes with 3 errors) while the 3NN-IMSE-PES algorithm finds a 3.5 cost gene set (1 gene with 3 errors). This is shown in Figure 11, below.

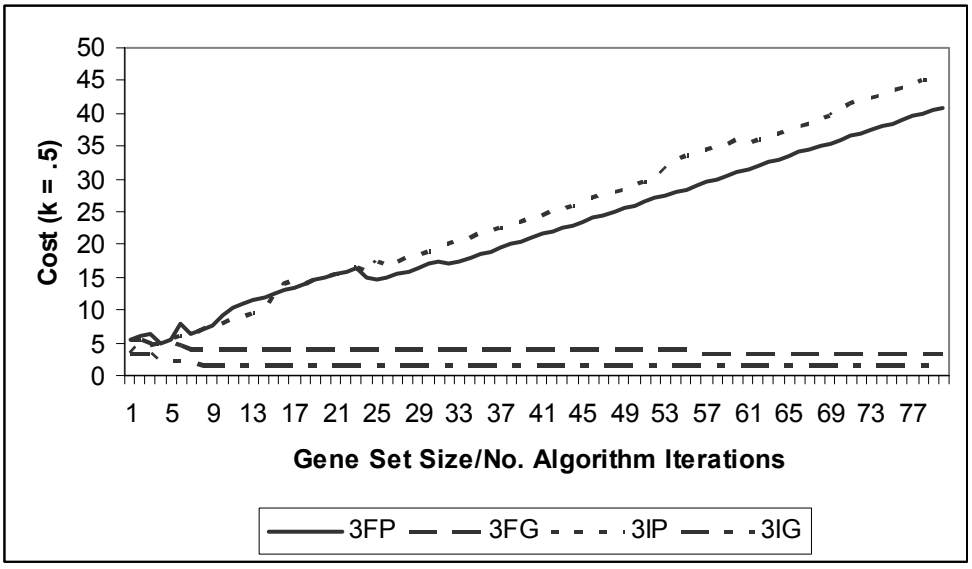

Figure 11. Cost curves for desmoplastic vs. classic medulloblastoma dataset.

Using the results above as a benchmark, we now apply our four algorithms. The 3NN-FDR-Greedy algorithm finds a 3.5 cost unit gene set (5 genes with 1 error) in 56 iterations while the 3NN-FDR-Stepwise algorithm finds a 3 cost unit gene set (4 genes with 1 error) again in 56 iterations. Continuing our evaluation, we apply the 3NN-IMSE-Greedy and 3NN-IMSE-Stepwise algorithms and find identical results. Both algorithms bind 1.5 cost gene sets (3 genes with 1 error) in 8 iterations. On this dataset, it appears that IMSE is the better filter.

Results Summary

We will now do some statistical analysis to evaluate whether the differences in cost are significant. We seek to answer the following three questions: First, are the differences in cost between the PES algorithms and our modified greedy algorithm significant in the statistical sense? Similarly, are the differences between the PES algorithm and our modified stepwise algorithm significant? Last, does the stepwise algorithm outperform the greedy algorithm in the statistical sense?

We seek to answer the above questions at the 95% confidence level using t-tests. We have the following data gleaned from our results above. Let P, G and S be 12 x 1 random column vectors representing the cost results of running the PES, modified greedy and modified stepwise algorithms respectively. By reviewing the results above, we find that:

$$P = [8.5, 4.5, 3.5, 22, 3.5, 5, 4, 5, 14, 2, 7, 12.5]^T$$
$$G = [7.5, 3.5, 1.5, 19, 1.5, 3, 3, 3.5, 10, 1.5, 6.5, 11]^T \text{ and}$$
$$S = [3.5, 1.5, 14, 1.5, 2.5, 2.5, 3, 10, 1.5, 5.5, 9.5]^T$$

Since we are not interested in the values of P, G and S explicitly we define two new random column vectors, PG and PS. The values of PG and PS are the differences between the P and G vectors and the P and S vectors above. Therefore:

$$PG = P - G = [1, 1, 2, 3, 2, 2, 1, 1.5, 4, 0.5, 0.5, 1.5]^T \text{ and}$$
$$PS = P - G = [5, 3, 2, 8, 2, 2.5, 1.5, 2, 4, 0.5, 1.5, 3]^T$$

If our modified algorithms performed no better we would expect that the random vectors PG and PS would not differ from zero significantly. If the algorithms did indeed perform better then we would expect that PG and PS would be greater than zero.

PES vs. Modified Greedy

To evaluate whether PG is different from zero we use the following null and alternative hypotheses:

$$H_o: \mu_{pg} = 0 \text{ vs.}$$
$$H_1: \mu_{pg} > 0$$

Since we would like to evaluate at the 95% confidence level, we apply the one sample t-test and evaluate with $\alpha = .05$ and 11 degrees of freedom. To reject the null hypothesis in favor of the alternative hypothesis we would need T to be greater than $t_{\alpha, n-1} = t_{.05,11} = 1.796$.

First we find $\bar{x}_{PG} = \sum_i x_i = 1.6667$ and $S^2_{PG} = \sum_i (x_i - \bar{x})^2 = 1.0606$ and apply the standard one sample t-test formula found in any statistics book:

$$T = \frac{\bar{x}_{PG}}{\sqrt{S^2_{PG}/n}} = \frac{1.6667}{\sqrt{\frac{1.0606}{12}}} = 5.6061$$

Since $5.6061 > 1.796$ we reject the null hypothesis in favor of the alternative hypothesis and conclude that the differences between the two algorithms are significant. In fact we would reject the null hypothesis in favor of the alternative for confidence levels greater than 0.9995 or $\alpha = 0.0005$.

PES vs. Modified Stepwise

To evaluate whether PS is different from zero we use the following null and alternative hypotheses:

$$H_o: \mu_{ps} = 0 \text{ vs.}$$
$$H_1: \mu_{ps} > 0$$

Again we would like to evaluate at the 95% confidence level and therefore we again apply the one sample t-test and evaluate with **α** = .05 and 11 degrees of freedom. To reject the null hypothesis in favor of the alternative hypothesis we would again need *T* to be greater than $t_{\alpha, n-1} = t_{.05, 11} = 1.796$.

First we find $\bar{x}_{PS} = \sum_i x_i = 2.9167$ and $S^2_{PS} = \sum_i (x_i - \bar{x})^2 = 3.9924$ and again we apply the standard one sample t-test formula:

$$T = \frac{\bar{x}_{PS}}{\sqrt{S^2_{PS}/n}} = \frac{2.9167}{\sqrt{\frac{3.9924}{12}}} = 5.0566$$

Since 5.0566 > 1.796 we reject the null hypothesis in favor of the alternative hypothesis and again conclude that the differences between the two algorithms are significant. Once again, we would reject the null in favor of the alternative for confidence levels exceeding α = 0.0005.

Modified Greedy vs. Modified Stepwise

A more interesting question is whether the modified stepwise algorithm outperformed the modified greedy algorithm. We stated earlier in this thesis that stepwise algorithms generally found smaller feature sets than greedy algorithms. We wish to see if that translates to modified stepwise algorithms finding lower cost gene sets than modified greedy algorithms. We define the random column vector GS as the difference between the random column vector G and the random column vector S. Again, we test to see if the mean difference between the two vectors significantly differs from zero. Using the data above:

$$GS = G - S = [4, 2, 0, 5, 0, .5, .5, .5, 0, 1, 1.5]^T$$

And our hypotheses are as follows:

$$H_o: \mu_{gs} = 0 \text{ vs.}$$
$$H_1: \mu_{ss} > 0$$

To reject the null hypothesis in favor of the alternative, at the 95% confidence level, we apply the one sample t-test again and evaluate again with $\boldsymbol{\alpha} = .05$ and 11 degrees of freedom. To reject the null hypothesis in favor of the alternative hypothesis we would again need T to be greater than $t_{\alpha, n-1} = t_{.05, 11} = 1.796$.

We find $\bar{x}_{GS} = \sum_i x_i = 1.25$ and $S^2_{GS} = \sum_i (x_i - \bar{x})^2 = 2.75$ and once more apply the standard one sample t-test:

$$T = \frac{\bar{x}_{GS}}{\sqrt{S^2_{GS}/n}} = \frac{1.25}{\sqrt{\frac{2.75}{12}}} = 2.6112$$

Since 2.6112 > 1.796 we reject the null hypothesis that there is no difference between the modified greedy and modified stepwise algorithms in favor of the alternative hypothesis that there are differences between the algorithms. We therefore recommend using the modified stepwise algorithm to find the lowest cost gene set.

DISCUSSION AND CONCLUSION

In this paper we have discussed the problem of feature space reduction. In this endeavor, we recognize the inherent problems with filtering algorithms and argue against them in favor of wrapper algorithms. Strict filtering algorithms, those that select genes independently of estimated error, are lacking in that they do not tell us what genes are actually needed.

Realizing that any good feature reduction technique will be a wrapper algorithm, we begin by thoroughly investigating wrapper algorithms as applied to the problems of microarrays. We reject the one existing algorithm used in the literature, Recursive Feature Elimination, due to it lack of generality. We recognize that an exhaustive search of all the power set of the feature set will produce good results but argue against this endeavor based on the prohibitive complexity of the algorithm and the arbitrariness of the results. Limited exhaustive searches, those that evaluate all gene sets of size 1 to m exhaustively, are evaluated but again found wanting. Using limited exhaustive searches we would find one, probably many, best gene sets but we recognize the uselessness of asserting that the results of this search have any meaning in a global sense due to the inherent arbitrariness of the error estimation algorithms and experimental set-up. Thus we are forced to regard this exercise of exhaustively checking all (or some of the smaller) gene sets as futile.

Of course, greedy and stepwise wrapper algorithms were designed to mitigate some of the problems of exhaustive searches. However, after evaluating these algorithms on real-life data we conclude that these algorithms cannot be effectively applied to microarray feature reduction problems. There exists the small, vexing problem of ties which prevents us from making use of these powerful algorithms. With many branches at each step to choose from and no way to pick branches we end up either picking branches arbitrarily or pursuing each of the numerous possible paths, a computationally daunting task. Of

course, any gene set we using these methods will be subject to the arguments above; namely the arbitrariness of asserting a best gene set.

We then evaluated the periodic estimating search (PES) algorithms often used in the literature. Admittedly, these algorithms are heuristic in nature but they perform surprisingly well, especially considering their computational simplicity.

Since the traditional alternatives to the PES algorithm are inapplicable to such a high dimensional problem, we propose a modified wrapper algorithm that retains the computational simplicity of the PES algorithm while improving on its performance. We believe that our algorithm provides several key advantages. First, our algorithm is flexible. The four parameters (filter, error estimator, traversal scheme, and ending condition) can be chosen to suit the problem at hand or the preferences of the researcher. Second, by filtering the data first we restrict our search to promising genes only. The algorithm saves computation by steering the search away from genes that are likely to be irrelevant. Finally, we find that in real problems our algorithm performs well, easily finding low cost gene sets.

REFERENCES

DeRisi, J.L., Iyer, V.R., Brown, P.O. (1997) Exploring the metabolic and genetic control of gene extpression on a genomic scale. *Science* **278**, 680-686.

Devroye, L., Györfi, L., Lugosi, G. (1996) *A Probabilistic Theory of Pattern Recognition.* Springer-Verlag, New York.

Duda, R.O., Hart, P.E., and Stork, D.G. (2001) *Pattern Classification*, 2nd edn, Wiley-Interscience, New York.

Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., *et al.* (2001) Support vector machine classification and validation of cancer tissue samples using microarry expression data. *Bioinformatics* **16**, 906-914.

Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., *et al.* (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531-537.

Guyon, I., Weston, J., Barnhill, S., Vapnik, V. (2000) Gene selection for cancer classification using support vector machines. *Machine Learning*, **46**, 389-422.

Hunter, L., Taylor, R., Leach, S., Simon, R. (2001) GEST: A gene expression search tool based on a novel Bayesian similarity metric. *Bioinformatics*, **17,** Suppl.1, S115-S122.

John, G.H., Kohavi, R., Pfleger, K. (1994) Irrelevant features and the subset selection problem. In: *Machine Learning: Proceedings of the Eleventh International Conference.* Morgan Kaufmann Publishers, San Francisco, CA.

Kim, S., Dougherty, E.R., Barrera, J., Chen, Y., Bittner, M.L., Trent, J.M. (2002) Strong feature sets from small samples. *Journal of Computational Biology*, **9**, 1 127-1 146.

Kohavi, R., John, G (1996) Wrappers for feature subset selection *Artificial Intelligence Special Issue on Relevance.* http://robotics.stanford.edu/~{ronnyk,gjohn}

Krishnapuram, B., Hartemink, A., Carin, L. (2002) Applying logistic regression and RVM to achieve accurate probabilistic cancer diagnosis from gene expression profiles. *Workshop on Genomic Signal Processing and Statistics (GENSIPS*), 2002. http://www.gensips.gatech.edu/

Luscombe, N.M., Greenbaum, D., Gerstein, M. (2001) What is bioinformatics? A proposed definition and overview. *Methods in Medical Informatics*, **40**, 346-358.

Mathé, C., Sagot, M., Schlex, T., Rouze. (2002) Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research*, **30**, 4103-4117.

Nguyen, D.V., Rockne, D.M. (2002) Tumor classification by partial least squares using microarry gene expression data. *Bioinformatics*, **18**, 39-50.

Pan, W. (2002) A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. *Bioinformatics,* **18**, 546-554.

Pomeroy, S., Tamayo, P., Gaasenbeek, M., Sturla, L.M., Angelo, M., *et al*. (2002) Prediction of central nervous system embryonal tumor outcome based on gene expression. *Nature*, **415**, 436-442.

Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C., *et al*. (2001) Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Science*, **98**, 12149-15154.

Ramaswamy, S, Ross, K.N., Lander, E.S., Golub, T.R. (2002) A molecular signature of metastasis in primary solid tumors. *Nature Genetics*, **33**, 49-54.

Singh, D., Febbo, P.G., Ross, K., Jackson, D.G., Manola, J., *et al*. (2002) Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, **1**, 203-209.

Troyanskaya, O., Garber, M., Brown, P., Botstein, Altman. D.R. (2002) Nonparametric methods for identifying differentially expressed genes in microarray data. *Bioinformatics,* **18**, 1454-1461.

West, M., Blanchette, C., Dressman, H., Huang, E, Ishida, S., *et al.* (2001) Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Science,* **98**, 11462-11467.

APPENDIX A

FILTERS USED

Fisher's Discriminant Ratio

One of the simplest and most intuitive metrics is Fisher's Discriminant Ratio. This metric measures differences in distributions. Fisher's Discriminant Ratio is given by

$$F = \frac{(\bar{x}_1 - \bar{x}_2)^2}{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} \tag{A.1}$$

Where $\bar{x}_i = \sum_i x_i$, $s_i^2 = \sum_i (x_i - \bar{x})^2$ and n_i is the number of samples labeled i. The higher the value of *F*, the more likely the samples for each class are from different distributions. We empirically find that this is a good metric for our data. The key is that the further the mean of the two class samples are apart and the smaller their estimated variance the more likely the samples were drawn from separate distributions and are therefore separable. Equivalently, the smaller the difference between the means the more likely the two samples came from the same distribution and thus are not separable. Small sample variances imply lower noise which implies easier separation of the distributions, provided of course that they are different.

Ideal Mean Square Error Filter

Our heuristic ideal mean square filter (IMSE) measures how close the samples drawn approximate an 'ideal' expression pattern in the mean square sense. Suppose the sampled expression levels of both classes of a particular gene are mapped to [0, 1], on a per gene basis, so that the minimum of the combined samples gets mapped to 0 and the maximum of the combined samples gets mapped to 1. If this gene is 'ideal,' all samples of one class will be 1, meaning fully expressed, and all samples of the other class will be 0, meaning fully repressed. To approximate how close two samples are to the ideal we use the following formula:

$$E = \min\left(\sum_j (x_j - 1)^2 + \sum_i x_i^2, \sum_i (x_i - 1)^2 + \sum_j x_j^2\right) \quad \text{(A.2)}$$

Where the equation measures the squared difference between the ideal gene and the p^{th} gene, j is the index of the first sample class and i is the index of the second sample class. If a given gene is expressed ideally, $E_p = 0$; if expressed randomly, $E_p < (i+j)/2$. Genes with lower scores are preferred to genes with higher scores, as they are 'more ideal

APPENDIX B

MODIFIED GREEDY ALGORITHM

ALGORITHM I (Modified Greedy)

Let $\mathbf{\Lambda}$ be the p x N of genes, where p is the number of genes and N is the number of total samples drawn, n_1 from class 1 and n_2 from class 2. Let $\mathbf{\Lambda_r}$ be the ranked set of genes and $\mathbf{\Lambda}^*$ be the optimum set of genes. As before, $\boldsymbol{\psi}$ is the classifier which classifies using rule $\boldsymbol{\zeta}$, $\boldsymbol{I(\cdot)}$ is the error estimating indicator function, and $\boldsymbol{\rho}$ is the filtering function which ranks genes using criteria $\boldsymbol{\varsigma}$. Below is shown the greedy version of the algorithm.

```
INITIALIZE
        Λ ←{Set of all genes}
        Λr←{Ø}
        Λ*←{Ø}
        ΛT ← {Ø}        // Dummy variable
        E = []          // Array of errors
        srchLen = user specified value
BEGIN
        Λr ← ρ(Λ, ς)
        Λ* ← λr1
        E[1] = I(ψ, Λ*, ζ)
        i = 2

        WHILE ((i < searchLength) and (Error>0))
                ΛT ← {Λ*, λri}
                E[i] = I(ψ, ΛT, ζ)
                IF (E[i] < E[i-1] )
                        Λ* ← ΛT
                END
        i = i+ 1
        END
END
```

VITA

Jason H. Gardner was born January 6, 1973 to David Wayne and Francis Elizabeth Gardner in Federal Way, Washington. After graduating from Decatur High School in Federal Way, Washington, in 1991 he enlisted in the U.S. Navy. After being honorably discharged in 1995, the former ET3 Jason Gardner began studies at the University of Washington in Seattle, Washington. Graduating with a B.S. in 1999 he spent the following two years working as an electrical engineer; first for AT&T, then for IBM. In January 2002 entered Texas A&M University as an electrical engineering graduate student at the master's level, completing the degree in August 2003.

He may be contacted at:

1516 Stonebrook Drive
Edwardsville, IL 62025
(618) 659-1047