Prognostic Models for Mesothelioma: Variable Selection and Machine Learning

by

Nathan Hans Vantzelfde

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2005

Author .....
Department of Electrical Engineering and Computer Science
October 5, 2005

Certified by .....
Lucila Ohno-Machado
Health, Science and Technology, Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Prognostic Models for Mesothelioma: Variable Selection and Machine Learning

by

Nathan Hans Vantzelfde

Submitted to the Department of Electrical Engineering and Computer Science
on October 5, 2005, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Malignant pleural mesothelioma is a rare and lethal form of cancer affecting the external lining of the lungs. Extrapleural pneumonectomy (EPP), which involves the removal of the affected lung, is one of the few treatments that has been shown to have some effectiveness in treatment of the disease [39], but this procedure carries with it a high risk of mortality and morbidity [8]. This paper is concerned with building models using gene expression levels to predict patient survival following EPP; these models could potentially be used to guide patient treatment. A study by Gordon *et al* built a predictor based on ratios of gene expression levels that was 88% accurate on the set of 29 independent test samples, in terms of classifying whether or not the patients survived shorter or longer than the median survival [15]. These results were recreated both on the original data set used by Gordon *et al* and on a newer data set which contained the same samples but was generated using newer software. The predictors were evaluated using N-fold cross validation. In addition, other methods of variable selection and machine learning were investigated to build different types of predictive models. These analyses used a random training set from the newer data set. These models were evaluated using N-fold cross validation and the best of each of the four main types of models – decision trees, logistic regression, artificial neural networks, and support vector machines – were tested using a small set of samples excluded from the training set. Of these four models, the neural network with eight hidden neurons and weight decay regularization performed the best, achieving a zero cross validation error rate and, on the test set, 71% accuracy, an ROC area of .67 and a logrank p value of .219. The support vector machine model with linear kernel also had zero cross validation error and, on the test set, a 71% accuracy and an ROC area of .67 but had a higher logrank p value of .515. These both had a lower cross validation error than the ratio-based predictors of Gordon *et al*, which had an N-fold cross validation error rate of 35%; however, these results may not be comparable because the neural network and support vector machine used a different training set than the Gordon *et al* study. Regression analysis was also performed; the best neural network model was incorrect by an average of 4.6 months in the six test samples. The

method of variable selection based on the signal-to-noise ratio of genes originally used by Golub *et al* proved more effective when used on the randomly generated training set than the method involving Student's t tests and fold change used by Gordon *et al.* Ultimately, however, these models will need to be evaluated using a large independent test.

Thesis Supervisor: Lucila Ohno-Machado
Title: Health, Science and Technology, Associate Professor

Acknowledgments

Thank you to Lucila Ohno-Machado for providing the topic of this thesis and the data set used throughout, as well as for offering continual guidance. Thank you also to Jonathon Jackson for providing the neural network software used in this paper, as well as for helping with revisions and editing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Malignant pleural mesothelioma (MPM) is an extremely rare and lethal form of cancer which affects the external lining of the lungs. Generally associated with exposure to asbestos, the disease spreads rapidly; following diagnosis, median survival of patients with MPM is between nine and thirteen months [8]. Treatment of MPM has met with very limited success, in part due to the ineffectiveness of radiotherapy and chemotherapy. Furthermore, the rarity of the disease, coupled with the lack of a universal staging system for MPM, has made organizing standardized clinical studies difficult [26].

One treatment that has shown some promise in extending patient life is extrapleural pneumonectomy (EPP), which involves the removal of the entire affected lung. At least one study has shown a significant increase in patient survival following EPP; furthermore, chemotherapy and radiotherapy have been shown to be more effective when used in combination with EPP [39]. However, the high rate of mortality and morbidity associated with the procedure dictates that eligible patients be selected with care. Ideally, patients with a long expected postoperative survival would undergo EPP, while those with a shorter expected postoperative survival would be treated differently to avoid the risks associated with the procedure. Although patient age and the tumor subtype are loosely correlated with postoperative survival, there are no reliable predictors of this survival.

The goal of this paper is to build a predictive model for postoperative patient

survival using the gene expression levels of extracted tumors. The gene expression levels were collected using microarray slides, which allow for the analysis of thousands of genes in a single experiment. While the gene expression levels potentially hold a wealth of information, analysis of expression levels for thousands of genes can be very difficult, especially when the number of samples is small.

A research study by Gordon *et al* addressed the problem of predicting postoperative patient survival using gene expression levels. Gordon *et al* built a simple predictor based on ratios of gene expression levels; with only four genes, the ratio-based predictor was able to accurately predict all of the training samples as having a survival greater than or less than the median survival, and was able to predict an independent test set with 88% accuracy as well [15]. The analysis described in this paper recreates the results of Gordon *et al* using both the original data set and a newer gene expression level data set, which includes the same samples but was produced from the microarray slides using newer software and so is hopefully more precise.

This paper also describes other machines learning methods for building predictive models, as well as variable selection techniques for eliminating irrelevant genes. Since this paper is focused on computer science and artificial intelligence rather than biology and oncology, the biological roles of the genes used in the predictive models will not be discussed. Models are built using four popular machine learning methods – decision trees, logistic regression, artificial neural networks and support vector machines – and models using different parameters are compared to one another using N-fold cross validation. The best of each type of model is then applied to a small set of test samples excluded from the training set; these models are compared based on sensitivity and specificity, as well as logrank and Cox proportional hazard regression p values. While this paper is mainly concerned with classification – that is, determining whether a patient will survive shorter than or longer than the median survival – some regression analysis – that is, predicting actual survival in months – is preformed as well; regression model performance is given in terms of average sum of squares error.

The ultimate goal of this analysis is to positively affect quality of life and length of survival among patients with MPM, since a reliable model could be used in practice to

guide patient treatment. However, the current unavailability of a large independent test set means some additional data collection and testing of the constructed models must be done before any of the predictive models can be safely used in practice.

Chapter 2

Background

This chapter introduces a number of important concepts used in this paper. Section 2.1 introduces the disease being studied, malignant pleural mesothelioma, and Section 2.2 describes the microarray technology used to collect gene expression data. Section 2.3 describes the machine learning methods used to build the prognostic models. Section 2.4 describes the approaches used to narrow down the number of genes to use as inputs to the predictive models. Lastly, Section 2.5 describes the methods used to compare the various models built using different machine learning techniques and different sets of genes.

2.1 Malignant pleural mesothelioma

MPM is a rare form of cancer affecting the pleural cells that form the external lining of the lungs. As mentioned in Chapter 1, MPM is incurable and extremely aggressive, spreading rapidly over affected surfaces; median survival after diagnosis is between 9 to 13 months and the five-year survival rate is only 3% [8]. MPM is most often caused by exposure to asbestos, and the highest incidence of the disease is found among former shipyard and construction workers, asbestos miners, and workers involved in insulation and automobile brake manufacturing [2, 8]. While these types of workers are at risk for a high level of exposure, even a minimal amount of asbestos exposure in the house can lead to development of the disease [2]. Due to the lengthy latency

period of the disease – over 30 years in most cases [2] – the average age of affected patients is approximately 60 years [8].

Besides asbestos, other minerals such as erionite, the simian 40 virus, and exposure to radiation have been associated with MPM [24]. There may also be a genetic factor involved in the development of the disease [36]. Despite these other potential causes, asbestos exposure can be found in approximately 80% of patients with MPM [2]. The significantly increased use of asbestos following World War II [8], coupled with the relatively recent ban on the use of asbestos and long latency period of the disease, means that the incidence of MPM is likely to increase continually in industrialized countries for the next few decades [36]. Furthermore, asbestos is still being mined in many parts of Asia [36], suggesting that frequency of MPM will increase soon in that part of the world as well.

The most common symptoms of MPM include localized chest pain, generally caused by the cancer invading the chest wall, and shortness of breath. Liquid discharge from the lung lining, called pleural effusion, will likely be discovered upon clinical examination. Other symptoms include fatigue, weight loss, coughing, fever, and profuse sweating [2]. Generally, a biopsy is needed for diagnosis, and can be performed either by open surgery or by video-assisted thoracoscopy [8]. Due to the non-specific nature of the symptoms and the invasive procedure needed for diagnosis, it can take as long as 2 to 3 months from the time symptoms are presented to a physician to the time when the disease is correctly diagnosed [39].

As the cancer progresses, it will likely invade the chest wall and eventually more vital areas such as the heart and esophagus [2]. While metastases in distant parts of the body are found in as many as 80% of patients [26], most patients die of local complications. These complications include difficulty breathing, inability to eat due to narcotics needed for pain relief as well as the cancer spreading to the esophagus, and heart failure due to the cancer invading the heart wall [26].

Treatment for MPM is aptly described as "disappointing" [2]. The rarity of the disease, along with the lack of uniformity in mesothelioma staging standards, has made running studies and evaluating treatments difficult [26]. Radiotherapy is gen-

erally difficult because of breadth of the tumors, not to mention the number of vital organs in the proximity of the cancer. MPM has also proven very resistant to most types of chemotherapy [36].

Surgical procedures, however, can be used to control pleural effusions and to reduce the bulk of the cancer. When used in combination with other types of treatment, these procedures have been shown to significantly increase patient survival [39]. The less radical of two available treatments, pleurectomy, involves removal of the pleural layer. This procedure is effective in controlling effusion and has a mortality rate of less than 2%; major complications are also fairly uncommon. The more radical procedure, EPP, involves the removal of an entire lung, and carries a much higher risk of morbidity and mortality. The mortality rate for EPP is close to 10% [8], and major complications occur in 24% of patients [26]. However, EPP is able to remove more of the bulk of the cancer and makes radiotherapy more feasible due to the removed lung. When used in combination with radiotherapy and chemotherapy, EPP has been shown to significantly increase patient survival, with one study producing two-year and five-year survival rates of 38% and 15%, respectively [39]. Furthermore, in the same study, patients who had the epithelial subtype of MPM – the three subtypes of MPM are epithelial, sarcomatoid and mixed – had an impressive median survival of 51 months [39].

While EPP has considerable promise in extending patient life and eventually providing a cure for MPM, the significant risk of morbidity and mortality associated with the procedure warrants that patient selection be done with care. Although epithelial subtype and young age are associated with a relatively good prognosis, there are really no known factors that are reliable predictors of patient survival; one of the purposes of this work is to find predictors of postoperative patient survival based on gene expression. Certainly, the ability to preoperatively predict patient survival following EPP – using, for example, gene expression levels measured from a tumor biopsy – would be very beneficial to the treatment of MPM; patients who are likely to have long postoperative survival times will undergo the procedure in spite of the risks, while those with shorter predicted survival times will be given less radical treat-

ments. In this way, an accurate prognostic model for MPM survival following an EPP procedure would help to improve patient life expectancy and quality of life.

2.2 Using microarrays to analyze gene expression

While nearly all the cells in an organism contain identical DNA, the genes expressed within the cells can vary substantially. Cells produce different proteins based on their functions or in response to various stimuli. Although not a perfect indicator, the amount of various molecules of mRNA within a cell is correlated with the expression levels of corresponding genes [7].

A microarray is a glass slide with thousand of spots, with each spot containing a strand of DNA replicated millions of times [7]. Each of these DNA strands, called probes, tests for the presence of the corresponding complimentary mRNA molecule.

The most common type of microarrays, called spotted microarrays, compare the relative expression levels of a test sample and a reference sample. The mRNA strands from two the samples are labeled with opposing colors, generally red and green, and the mRNA is allowed to hybridize with the probes. If a spot is red, presumably the target mRNA molecule is in greater abundance – and hence the corresponding gene has a higher expression level – in the red-labeled sample; the opposite is the case if the spot is green. If the spot is yellow, the two samples have the same gene expression level. Greater color intensity indicates higher gene expression level [7].

Another method used in microarrays is that of the Affymetrix company's GeneChip^{TM}. 22 to 40 oligonucleotides are used for each gene, each 25 nucleotides long. 11 to 20 of these oligonucleotides are perfectly complimentary to various sections of the target mRNA. These oligos are the so-called *perfect match* strands. Each of the other 11 to 20 oligonucleotides is identical to a corresponding perfect match strand, except the nucleotide at the central location is replaced by the complimentary nucleotide. These oligos are the *mismatch* strands, which help to identify background hybridization. Thus, the true hybridization can be quantified by subtracting the mismatch value from the perfect match value. The early versions of Affymetrix software calculated

the overall gene expression for a single gene as the average difference for all pairs of oligos, using the formula

$$Average\ Difference \sim \frac{\sum_{i=1}^{n} PM_i - MM_i}{n}, \tag{2.1}$$

where PM_i and MM_i are the perfect match color intensity and mismatch color intensity, respectively, for the i^{th} oligo, and n is the number of oligo pairs used to measure the given gene [7]. This formula, however, can produce negative values for gene expression levels if the mismatch values are large enough. Newer versions of the software give weights to each pair of perfect match and mismatch values; pairs with large mismatch values are given smaller weights to lessen their impact, causing expression levels to always be nonnegative [7].

The Affymetrix company has developed a manufacturing process similar to that used in producing silicon chips [21]; the production of oligonucleotides within the spots is much more uniform than with spotted arrays. However, in spotted arrays, any oligonucleotide strand can be designed and included in the array, which is not possible with a mass-produced GeneChip^{TM} [21].

Whichever type of microarray is used, once hybridization is complete, the microarray slide is scanned, the resulting digital image is then analyzed to calculate color intensities. The resulting data is then normalized to produce comparable measurements. This process of converting the physical microarray slide into a usable data set is not a trivial one and is increasingly becoming the job of specialists [7]. The end result is a gene expression matrix, in which each column represents a single sample and the rows contain expression level values for each of the target genes. Each sample is often annotated with other patient information relevant to the problem being addressed [7]. The gene expression matrices can offer a vast amount of information, and researchers could potentially use differentially expressed genes to aid in determining the diagnosis or prognosis of various conditions [7]. Although microarrays have been in use for less than a decade, they have already been used extensively in oncology research to classify tumor types, discover new subtypes of cancer, and predict patient

prognostic outcome [7].

Effectively analyzing a gene expression matrix can be a difficult and time-consuming task. Researchers generally wish to identify genes that are differentially expressed across various sets of samples, such as healthy and sick patients, or build models which can help decide the diagnosis or prognosis for a patient.

Since microarrays are a new technology, there are few universally accepted analysis methods, and many of the methods used in research reports are seemingly *ad hoc*. Because they generally contain a limited number of samples and a huge number of independent variables (the expression levels for thousands of genes), these data sets suffer what is known as the "curse of dimensionality" [16]. Because of the disproportionately large number of variables compared to the number of samples, a model can always be built to perfectly classify one set of samples from another; the difficulty lies in building models which can accurately and reliably predict previously unseen samples. Methods of machine learning used to build these models are discussed in Section 2.3, and variable selection methods used to reduce the dimensionality of the data are discussed in Section 2.4.

2.3 Machine learning methods

The machine learning methods used to analyze gene expression matrices are rarely *supervised* learning methods; that is, training is guided by an *outcome* variable. In contrast, the use of *unsupervised* learning methods, such as clustering, which do not use an outcome variable, is popular. Since the data set used in this research work contains an outcome variable, patient survival, supervised learning methods are used. In general, the goal of supervised machine learning is to build a predictive model for an outcome variable using a number of *input* variables for the samples in a particular *training set* [16]. The outcome variables can be either categorical or ordered. Categorical variables are those that can take one of a number of values which have no natural ordering, thus separating the samples into a number of classes; whether or not a patient has a particular disease, signified with a 0 or 1, is an example of a

categorical variable. Modeling a categorical variable is known as classification, and can also be viewed as creating one or more decision boundaries in the input space that separate samples of the various classes. Ordered variables, on the other hand, can take a continuous or discrete range of values, which have a natural ordering; the number of days a patient survives following a treatment would be an ordered variable. Modeling an ordered variable is known as regression. Some machine learning techniques are suited for both categorical and ordered outcome variables, while others can only handle one type or the other.

As examples, two simple types of machine learning are linear regression and K-nearest neighbors. Linear regression attempts to fit a line to the training data, so the predictor has the form

$$f(x_i) = \alpha + \sum_k \beta_k x_{ik}, \tag{2.2}$$

where x_i is the input vector for sample i, x_{ik} is the k^{th} input variable for sample i, α is a constant and β_k is the coefficient for the k^{th} input variable. The best-fit line is found by minimizing some error measure, generally residual sum of squares, defined as

$$\textit{Residual Sum of Squares} = \sum_{i=1}^{N} (y_i - f(x_i))^2, \tag{2.3}$$

where y_i is the outcome variable for sample i and N is the number of training samples. Residual sum of squares is generally chosen because it is differentiable as well as simple to calculate [16]. This model is very restrictive, and cannot handle non-linear relationships unless input transformations are explicitly added.

K-nearest neighbors uses the K nearest points in the input space to predict the outcome of a sample. Formally, the K-nearest neighbors model is

$$f(x_i) = \frac{1}{k} \sum_{x_j \in N_k(x_i)} y_j, \tag{2.4}$$

where $N_k(x_i)$ is the set, or neighborhood, of K closest training set points to the input vector x_i [16]. The distance between points is often calculated as Euclidean distance. This type of model, especially when $K = 1$, is overly flexible and sensitive to the

particular choice of training set; that is, the predictions can change significantly if one training sample is added or removed.

In general, for a training set of finite size, there are an infinite number of models which can predict or classify the outcomes of this set perfectly. Because of this, nearly every type of machine learning makes some assumptions about the structure of the underlying data and places some restrictions on the type of models which can be generated [16]. The goal of machine learning is not to model the training set perfectly, but to attempt to model the underlying function which generated the data; a more complex and flexible model can perform better on the training set than a simpler model, but may not generalize well to new samples because it may become too sensitive to the noise in the training set [3]. When a model is too restrictive and cannot reasonably model the data, the model is said to *underfit* the data. When a model is overly complex and captures the noise in the training set rather than the true underlying function which generated the data, the model is said to *overfit* the data. Thus, the complexity of a model must be balanced carefully.

The types of machine learning described in Sections 2.3.1 through 2.3.4 are some of the most commonly used methods. Each is generally more flexible than linear regression but not as flexible as K-nearest neighbors.

2.3.1 Logistic regression

Similar to linear regression, logistic regression attempts to model the probability that a sample belongs to each of the various classes. Because it models class probability, logistic regression is used only for classification problems. Only the equations for the two class problem will be presented here, although the logistic regression for three or more classes is not significantly more complicated. Let $\pi(x_i)$ be the probability that patient with input vector x_i suffers some event of interest. In logistic regression, these probabilities are modeled using the equation

$$\pi(x_i) = \frac{e^{\alpha+\sum_k \beta_k x_{ik}}}{1 + e^{\alpha+\sum_k \beta_k x_{ik}}}, \tag{2.5}$$

where again x_i is the input vector for sample i, x_{ik} is the k^{th} input variable for sample i, α is a constant and β_k is the coefficient for the k^{th} input variable. This equation is known as the *logistic probability function* [10]. This equation produces values that range from 0 to 1; clearly, if $e^{\alpha+\sum_k \beta_k x_{ik}}$ is small, the numerator is close to zero while the denominator is close to one and, if $e^{\alpha+\sum_k \beta_k x_{ik}}$ is large, the numerator and denominator are approximately equal.

Using $\pi(x_i)$, the odds that the patient suffers the event is given by the equation

$$odds = \frac{\pi(x_i)}{1-\pi(x_i)} = e^{\alpha+\sum_k \beta_k x_{ik}}. \tag{2.6}$$

Taking the natural logarithm of this equation gives

$$log\ odds = log[\frac{\pi(x_i)}{1-\pi(x_i)}] = \alpha + \sum_k \beta_k x_{ik}. \tag{2.7}$$

Note that the logit function is defined as

$$logit(p) = log[\frac{p}{1-p}], \tag{2.8}$$

where p can be any value from 0 to 1. Using this definition,

$$logit(\pi(x_i)) = \alpha + \sum_k \beta_k x_{ik}. \tag{2.9}$$

Thus, the logit of the probability of the event is modeled as a linear function of the input variables [10]. However, in this case, the coefficients are often found using maximum likelihood rather than residual sum of squares.

The coefficients β_k have an interesting interpretation in logistic regression. Consider a patient that originally has input vector x_i. This patient's log odds of suffering the event is given by

$$log\ odds = \alpha + \sum_k \beta_k x_{ik}. \tag{2.10}$$

Now, if one of the patient's input variables, x_{ij}, increases by 1, giving a new input

vector x_i', the patient's log odds become

$$log\ odds' = \alpha + \sum_k \beta_k x'_{ik} = \alpha + \sum_k \beta_k x_{ik} + \beta_j. \tag{2.11}$$

Thus, the coefficient β_j is equal to the increase in log odds associated with a 1 unit increase in j^{th} input variable. Furthermore, the value e^{β_j} is the odds ratio – or the factor by which the odds increase or decrease – associated with a 1 unit increase in the j^{th} input variable. In logistic regression, this property is true for all values of all input variables. Also note that logistic regression is a multiplicative model, in the sense that, if both x_{ij} and x_{ik} increase by 1 unit each, the resulting odds ratio is $e^{\beta_j} e^{\beta_k}$, or the product of the two individual odds ratios [10].

It is sometimes necessary to include additional variables that capture the interaction between input variables. That is, it may be the case that two variables, when changed at the same time, have a significantly greater or lesser effect on the odds than the product of the two effects. In this case, it would be necessary to include an additional $\beta_{new} \times x_j \times x_k$ term in the exponent of both the numerator and denominator of the logistic probability function from Equation (2.5). Then, when both variables increase by one, the odds increase by a factor of $e^{\beta_{new}} e^{\beta_j} e^{\beta_k}$, rather than simply a factor of $e^{\beta_j} e^{\beta_k}$ [10].

Logistic regression is a useful machine learning technique because the coefficients have a natural interpretation. In logistic regression, it is common to transform a K-class categorical input into K corresponding binary variables; however, this is only a minor nuisance. Lastly, most statistics packages that can handle logistic regression can also handle stepwise forward and backward variable selection, which makes finding a good model a simple task when there are a large number of potentially informative input variables; stepwise variable selection is discussed in Section 2.4.3.

2.3.2 Classification and regression trees

Decision trees are most commonly used for classification problems but can be applied to regression tasks as well. Building a decision tree amounts to repeatedly splitting

subsets of the training data, starting with the entire training set, into two distinct subsets, until each subset meets some stopping criterion [4]. When splitting is complete, each of the undivided subsets, also called *terminal subsets*, is assigned a class based on the training samples contained in the subset. In general, finding the overall optimal set of splits is intractable, so a greedy algorithm is used instead. At every stage of the greedy algorithm, the best split is chosen based on the samples in the current subset; then, each of the resulting subsets is examined in the same way until the entire tree is created. Splitting is generally determined by some measure of node impurity so that the split with the most pure resulting subsets is preferred. Thus, only three separate features of the building algorithm need to be specified: the method to determine each split, the criterion for stopping, and the process for assigning a class to each terminal subset [4].

When viewed as a tree graph, the nodes of the tree represent the subsets of the training data, with the root of the tree as the entire training set. The split at each node determines which samples are filtered to the left subtree and which samples are filtered to the right subtree. Each leaf node represents a terminal subset, and is labeled with a particular class assignment. Classifying a new sample is easy and can often be done by hand; starting at the root node, classification is done by taking the appropriate splits based on the sample input data and following the tree to a leaf node. The class label of that leaf node is the new sample's predicted classification.

To make calculating splits tractable, some restrictions need to be placed on the types of splits that are allowed. First, each split can depend on only one variable. For an ordered variable, a split must have the form $x_{ij} \leq c$, where c is a constant. For N samples, this restriction limits the number of possible splits to $N-1$. For a categorical variable, a split must have the form $x_{ij} \in S$, where S is a subset of the possible values of x_{ij}. If x_{ij} can take a total of M values, there are a total of 2^{M-1} possible subsets and thus a total of 2^{M-1} possible splits. However, if the outcome variable is binary, the number of possible splits can be significantly reduced. This is done by first sorting the M classes of the categorical variable according to the proportion of samples in each class that have outcomes of 1; that is, if a higher proportion of samples with

$x_{ij} = k_1$ have outcomes of 1 than samples with $x_{ij} = k_2$, class k_1 will come before class k_2. If S_i denotes the subset of classes with the i lowest proportions of samples with outcomes of 1, the optimal split will always have the form $x_{ij} \in S_i$, thus giving a total of $M - 1$ possible splits [16].

These restrictions force all splits to be binary. Some algorithms allow for multi-way splits, but multi-way splits fragment the data too quickly, leaving too few samples in each of the resulting subsets to make good subsequent splits [4]. Furthermore, multi-way splits bias variable selection in favor of those variables with a large number of different values. For instance, each of the N samples could have a different value for a particular variable, and so the tree could be split into N subsets, each containing one sample. While the resulting subsets are completely pure, such a split is unlikely to generalize well to new samples [31].

To determine the best split, some measure of impurity is generally used, and the split generating the purest subsets is chosen. For a function to be a suitable impurity measure, the function must have a maximum when all classes have equal numbers in a subset. The function must have a minimum when a subset contains only samples from one class. Also, the function should be symmetric, so that, in the binary outcome case, $\Phi(p_1, p_2) = \Phi(p_2, p_1)$, where Φ is the impurity measure, p_1 is the proportional of samples in class 1 and p_2 is the proportion of samples in class 2. Functions that are concave – that is, $\Phi'' < 0$ – are preferred, although concavity is not a strict requirement. Functions of this type tend to give preference to splits resulting in one very pure subset and one very impure subset, rather than those resulting in two equally impure subsets. The former type of split has been shown in practice to result in better trees. In general, however, the resulting tree is fairly insensitive to the choice of impurity measure, and so any reasonable measure can be safely chosen [4].

There are three commonly used impurity measures: misclassification rate, cross entropy, and the Gini statistic. The misclassification rate is

$$Misclassification\ Rate = 1 - p_{k,max} = 1 - \max_{m \in M} p_{k,m}, \tag{2.12}$$

cross entropy is given by

$$Cross\ Entropy = -\sum_{m \in M} p_{k,m} \log p_{k,m}. \tag{2.13}$$

and the Gini statistic is calculated by

$$Gini\ Statistic = \sum_{m \neq m'} p_{k,m} p_{k,m'} = \sum_{m \in M} p_{k,m}(1 - p_{k,m}), \tag{2.14}$$

where $p_{k,m}$ is equal to the proportion of samples with class m in node k, and M is the set of possible classes. Cross entropy and the Gini statistic are differentiable and therefore are used more commonly than misclassification rate [16]. Misclassification rate is also not concave and may not produce good trees.

Another feature that must be specified when building trees is the class assignment method within leaf nodes. In most cases, the most common class of sample contained within a leaf node determines that node's class assignment. This implicitly assumes that the cost of misclassifying a sample of class i as class j is the same for any two classes i and j, where $i \neq j$. However, this assumption may not always hold. In medical situations, misdiagnosing a sick patient as healthy is likely to be much more costly than misdiagnosing a healthy patient as sick, since, in the absence of proper treatment, a sick patient could ultimately die. These varying costs can be captured in a cost matrix, C, where C_{ij} is the cost associated with classifying a member of class i as class j. Note that C_{ii} should always be equal to zero. Then, the class assignment for a leaf node can be calculated by minimizing the total misclassification cost, using the equation

$$Node\ Class = \arg\min_{m} \sum_{i} C_{im} p_{k,i}. \tag{2.15}$$

The class assignment within the nodes can be influenced without the use of a cost matrix by increasing – perhaps artificially – the prevalence of the certain classes of samples within the training set [4].

The third and final feature that needs to be specified to build a classification tree is the stopping method. Building can continue until all leaf nodes are completely pure

– that is, until each contains samples of only one class. However, a model using this stopping method may be overly complex and may not generalize well; methods for early stopping may help to reduce this overfitting. Such stopping methods include stopping when a node size falls below a certain amount or when a node purity reaches some percentage level. While attractive in theory, early stopping methods do not work well in practice. It is generally better to build a full tree, stopping when nodes are pure, and then prune the tree to the appropriate size [4].

A classification tree effectively divides the input space into a number of regions, one for each leaf node. The decision boundaries between regions must be perpendicular to the dimensions because of the restrictions placed on the possible splits.

Pruning decision trees

The most common method of pruning is cost-complexity pruning. Before describing the pruning method, it is necessary to give several formal definitions. The total misclassification cost for tree T is given by

$$R(T) = \sum_{k \in leaf(T)} \min_m \sum_i C_{im} p_{k,i}, \tag{2.16}$$

where $leaf(T)$ is the set of leaf nodes of tree T. Adding an additional measure to penalize the size of tree T gives the cost-complexity measure

$$R_\alpha(T) = R(T) + \alpha |leaf(T)|, \tag{2.17}$$

where α is a constant. A larger value of α gives a larger penalty for tree size. For a given value of α, let $T(\alpha)$ be the subtree of T that minimizes the cost-complexity function $R_\alpha(T)$. If two distinct subtrees have the same value of $T(\alpha)$, the smaller of the two subtrees is selected so that $T(\alpha)$ is always unique. Note that, when $\alpha = 0$, the "optimal" subtree is T, and as α increases, the size of the optimal subtree decreases until only the root of the tree remains [4].

For one final definition, let T_k be the subtree of T rooted at node k, and let $\{k\}$

be the subtree of consisting only of node k. Then,

$$R_\alpha(T_k) = R(T_k) + \alpha|leaf(T_k)| \tag{2.18}$$

and

$$R_\alpha(\{k\}) = R(k) + \alpha. \tag{2.19}$$

For some critical value of α, these two equations will be equal, at which point the single node k is preferred to the subtree T_k. This value of α is given by the formula

$$\alpha_{critical} = \frac{R(k) - R(T_k)}{|leaf(T_k)| - 1}. \tag{2.20}$$

This calculations forms the basis of cost-complexity pruning [4].

The cost-complexity pruning algorithm starts with the full tree T and finds the internal node – an internal node is any non-leaf node – with the smallest value of $\alpha_{critical}$. The tree is pruned at this node, and the resulting tree, T_1, is stored. Then, starting with T_1, the remaining internal node with the lowest value of $\alpha_{critical}$ is found, the tree T_1 is pruned at this node, and the resulting tree, T_2, is stored. This process continues until only the root node of the original tree remains. The process is known as *weakest link pruning.* The result is a list of trees in decreasing order of size – T, T_1, T_2, ..., $\{t\}$, where $\{t\}$ is the tree containing only the root node of T. A hold-out set can then be used to evaluate the performance of each of these trees, and the one with the best performance on the hold-out set is selected [4].

Regression trees

Regression trees work in much the same way as classification trees. The goal is to divide the input space into a number of regions, and assign each region an outcome value. With regression trees, however, the outcome assigned to each region is not a class assignment, but rather a constant value to approximate the ordered outcome value. Minimizing an error measure, typically sum of squares, is used to determine the outcome value associated with each region. That is, the constant outcome value

$\hat{k}$ assigned to a region R is

$$\hat{k} = \arg\min_{k} \sum_{x_i \in R} (y_i - k)^2 = average_{x_i \in R} y_i, \tag{2.21}$$

since the arithmetic mean minimizes the sum of squares error, giving a total error of

$$Error = \sum_{x_i \in R} (y_i - \hat{k})^2. \tag{2.22}$$

Thus, when making splits, the goal is to minimize the total error in the two resulting regions. For a split that separates the samples into two regions R_1 and R_2, the total error is given by

$$Error = [\sum_{x_i \in R_1} (y_i - \hat{k}_1)^2 + \sum_{x_i \in R_2} (y_i - \hat{k}_2)^2], \tag{2.23}$$

where $\hat{k}_1$ and $\hat{k}_2$ are the means of the outcomes of training samples in regions R_1 and R_2, respectively. The best split can then be found by searching all variables and all possible split points to find the variable and split point which minimizes Equation (2.23) [16].

As in classification trees, a method is needed to determine when to stop building a regression tree. It is possible to stop early, perhaps when the decrease in sum of squares error from a split is less than some threshold; once again, however, it is generally better to build a full tree and prune back using cost-complexity pruning. Cost-complexity pruning for regression trees is equivalent to method used for classification trees except the cost-complexity measure is now

$$E_\alpha(T) = (\sum_{k \in leaf(T)} \sum_{x_i \in R_k} (y_i - \hat{k}_k)^2) + \alpha |leaf(T)|. \tag{2.24}$$

A hold-out set is again needed to determine which of the resulting list of trees performs the best on predicting new sample values [16].

Advantages and disadvantages of decision trees

Classification and regression trees offer a number of advantages over other types of learning methods. First of all, the algorithm to build trees is very fast computationally, and can handle a large number of input variables, since there are a limited number of splits for each variable. Second, there are very few parameters to specify when building a decision tree [4], none of which require a great deal of fine-tuning. Third, decision trees make no assumptions about the distribution of variables or about the independence of variables [31]. No additional parameters, like the interaction terms needed in logistic regression, are needed to handle variables in decision trees. Lastly, and perhaps most importantly, some decision trees are easy for a human to interpret; in many cases, a surprisingly simple tree may be generated to handle a seemingly complicated classification or regression problem [31].

There are a number of disadvantages associated with decision trees as well. First and foremost, because of the restriction places on possible splits, all decision boundaries must be perpendicular to one of the dimensions, making all decision regions hyperrectangular [31]. This means that decision trees cannot directly handle some simple linear relationships, such as decision boundary of the form $x_{i1} - x_{i2} = 0$, and some more complicated, non-linear relationships. A tree will likely have to make a number of inaccurate splits to handle such a decision boundary, and these splits are likely to generalize poorly. Some tree-building algorithms explore linear combinations of variables when making splits, but this can significantly increase computation time and, more importantly, reduce the interpretability of the resulting tree. Second, trees are instable; a small change in the training set can lead to a completely different tree, since changes in higher splits propagate to lower splits in a tree. Third, while trees are easy to interpret, they may not always give a complete description of underlying structure of the problem due to *variable masking*. That is, while one variable may have been selected for a particular split, another variable may be able to generate a split that is almost as good [4]. The tree gives the false sense that the first variable is very relevant to the problem while the second variable is useless, when in reality the

two may be almost equally important.

Decision trees cannot easily handle problems that, by their nature, have a large number of decision regions. For example, classifying a disease that is present if and only if p out of a potential q factors are present would require a very large number of decision regions, and a classification tree would be unlikely to find the true structure of the problem [31]. Furthermore, while decision trees are computationally suited to handle a large number of input variables, having a large number of input variables increases the chance of finding an irrelevant input that can split a given subset well, and so decision trees suffer, to some extent, the curse of dimensionality that plagues other machine learning methods [31]. Lastly, the greedy algorithm chooses the optimal split based on the current subset of training samples; there is no reason to believe that each split is optimal for the overall tree structure [4].

Despite their limitations, trees have proven to be fairly effective in practice and, because of their interpretability, often give some insight into the underlying nature of the problems being solved.

2.3.3 Artificial neural networks

Artificial neural networks have a strong relationship to biologic neural structures, and indeed their original motivation stems from the mathematical formulation of a neuron by McCulloch and Pitts in the 1940's [32]. A biological neuron is a cell containing a long offshoot called an *axon*, which branches off at the end to form *synapses* with other neurons. Chemicals released at the synapse increase or decrease the potential of the connected neurons. If this potential within a neuron exceeds some threshold, a pulse called an *action potential* travels down the axon and causes the release of chemicals at the synapses, which in turn excite or inhibit other neurons [17].

An artificial neural network consists of a number of interconnected artificial neurons. Each artificial neuron, except those which are connected to the inputs of the network, receive inputs from a number of other neurons. These inputs are weighted and summed, and the output of the neuron is calculated by passing this weighted sum through an activation function. A constant, called a bias, is often added to the

weighted sum. Mathematically, the output of a neuron j is given by

$$y_j = f_j(\alpha_j + \sum_{i \to j} w_{ij} y_i), \tag{2.25}$$

where f_j is the activation function, α_j is the bias, w_{ij} is the weight for the connection between neuron i and neuron j, and y_i is the output of the i^{th} neuron [32]. Note that the neuron output y_j should not be confused with the outcome variable for a sample; the distinction between the two will be clear by context.

In most cases, networks are separated into layers, such that each neuron in a layer receives inputs only from the neurons in the previous layer. The first layer is connected to the input values and the final layer generates the network output values. Layers between the input and output layers are referred to as *hidden layers.* Networks containing connections from nodes in later layers to nodes in the earlier layers, thus forming loops, are called *recurrent networks. Feed-forward networks*, on the other hand, do not contain loops. A feed-forward network with one hidden layer using the appropriate activation functions can approximate any continuous function with arbitrary accuracy if enough hidden neurons are used, and a network with two hidden layers can approximate any function [3]. Only feed-forward networks will be discussed here.

There are many common activation functions used in neural networks. For hidden layer neurons, the sigmoid function, defined as

$$f_\alpha(x) = \frac{1}{1 - e^{-\alpha x}}, \tag{2.26}$$

is the most common function. This function ranges from 0 to 1, with the parameter α determining how fast it goes from one extreme to the other. Another common activation function for hidden neurons is the hyperbolic tangent function, given by

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2.27}$$

The hyperbolic tangent function is similar to the sigmoid function, but, because of

the shape of this function, networks using hyperbolic tangent activation functions tend to train faster than those using sigmoid activation functions. Lastly, the linear function

$$f(x) = x \tag{2.28}$$

can be used in hidden nodes; however, training a network that contains only linear activation functions is equivalent to performing linear regression. For output neurons, the sigmoid activation function is frequently used for classification problems, since its output ranges for 0 to 1. For a K-class classification problem, one output neuron is needed for each of the K classes. For regression problems, output neurons often use linear activation, since the output values are not limited to the range $[0, 1]$.

Note that input neurons are generally just used to pass input values to neurons next layer and so can be seen as using linear activation with no bias.

Training artificial neural networks

The most common algorithm used to train neural networks is *back-propagation.* Back-propagation effectively calculates the error gradient with respect to each weight, and updates each some small amount in the negative direction of the gradient. To see this mathematically, let x_j denote the weighted, summed input to neuron j, given by

$$x_j = \alpha_j + \sum_{i \to j} w_{ij} y_i, \tag{2.29}$$

and let y_j denote the output of the neuron, as given in Equation (2.25). Again, x_j should not be confused with the input vector for sample j; the meaning will be clear from context. Then, the error with respect to weight w_{ij} is

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} y_i = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial x_j} y_i = y_i \cdot f_j'(x_j)\frac{\partial E}{\partial y_j} = y_i \delta_j, \tag{2.30}$$

where δ_j is defined as

$$\delta_j = f_j'(x_j)\frac{\partial E}{\partial y_j}. \tag{2.31}$$

The error measure is often chosen as sum of squares. For output neurons, δ can be calculated directly from the derivatives of the error measure and the activation function. For neurons in earlier layers, δ can be calculated using the δ values for neurons in subsequent layers, using the formula

$$\delta_j = f'_j(x_j)\frac{\partial E}{\partial y_j} = f'_j(x_j)\sum_{j\to k} w_{jk}\frac{\partial E}{\partial x_k} = f'_j(x_j)\sum_{j\to k} w_{jk}\delta_k. \tag{2.32}$$

Finally, once the δ values have been calculated, the weight changes are given by

$$\Delta w_{ij} = -\eta\delta_j y_i, \tag{2.33}$$

where η is known as the learning rate [32].

Using Equation (2.32), the δ values for each neuron depend only on δ values of neurons in subsequent layers. Thus, back-propagation is a two-pass algorithm; the first pass starts with the network inputs and propagates the neuron outputs forward until the network output is calculated, and the second pass starts at the output neurons and propagates the δ values backward until all are calculated. When the weights are updated after each training sample, the training process is said to be *online* or *incremental*. Each training sample can be selected randomly, making the process *stochastic*. The weights may be updated only after a number, potentially all, of the samples, in which case the training is said to be *batch*, and the formula used to update the weights is

$$\Delta w_{ij} = -\eta\sum_{\mu}\delta_j^{\mu} y_i^{\mu}, \tag{2.34}$$

where δ_j^{μ} and y_i^{μ} are the δ_j and output value, respectively, for neuron j and training sample μ [17]. Online training, whether stochastic or not, is generally preferred to batch training because gradient movements tend to cancel in batch training, making online converge faster, and batch training is more likely to get stuck in local minima, instead of finding the global minimum of the training error [32].

A *momentum* term is often added to Equation (2.34), making the formula to

update the weights

$$\Delta w_{ij,new} = -\eta \sum_{\mu} \delta_j^{\mu} y_i^{\mu} + \alpha \cdot \Delta w_{ij,old}, \tag{2.35}$$

where α, the momentum coefficient, must be greater than or equal to 0 and less than 1, and $\Delta w_{ij,old}$ is the weight change from the previous iteration of the algorithm. The momentum term helps to accelerate large drops in the error function, while minimizing oscillations [17].

There are a number of methods to determine when to stop network training. Training can be stopped after a set number of *epochs* – iterations through the complete training set – or after a set amount of computation time. Training can also be stopped when the error, either for the training set or hold-out set, falls below some constant value or decreases by less than some constant amount after a given epoch. Lastly, training can stop once the error on the hold-out set begins to increase [3].

Overfitting is often a problem in neural networks, especially when the training set is small or the network size is overly large. Some form of early stopping may be used to prevent overfitting, but, as in classification trees, determining the appropriate time to stop is difficult, and early stopping generally performs poorly in practice. A more effective way to combat overfitting is known as *regularization*, in which a penalty term is added to the error function to penalize large weights. The most common form of regularization is called *weight decay*, which adds a penalty term to the error measure, effectively making the new error measure

$$\tilde{E} = E + \nu(\frac{1}{2} \sum_{i,j} w_{ij}^2), \tag{2.36}$$

where ν is a constant which controls how much of an effect the regularization penalty has. Adding the penalty term to the error measure effectively adds an additional term $-\nu w_{ij}$ to the weight update equation given in Equation (2.35). Using weight decay regularization, unless they are reinforced by training, the network weights decay exponentially to 0. This helps to avoid overfitting [3].

Pruning can also be used to help prevent overfitting. In neural network pruning, the network is completely trained, then one or more weights are removed and the

network is completely trained again. This process can be repeated one or more times. A simple type of pruning simply removes the weights with the smallest magnitudes. While this is easy to calculate, it performs poorly in practice. A more sensible approach attempts to calculate how much the total error would increase if each weight was removed and prunes the weights with the smallest such values. While this type of pruning performs significantly better in practice, it requires significantly more computation. Two implementations of this type of approach are known as *optimal brain damage* and *optimal brain surgery* [3].

Choosing the appropriate values for the parameters used in building neural networks is generally not an easy task. There are a large number of parameters that need to be set, including the network size, learning rate, momentum, regularization constant ν, the stopping method and the method to generate the initial weights. Small changes in one or more of these factors can significantly change the learning process. For example, too small of a learning rate can cause training to proceed too slowly, whereas too large of a learning rate can cause training to overshoot minima in the error function, thus causing it not to converge [17]. Likewise, choosing a network size that is too small will not allow the network to approximate the appropriate function, and choosing too large a size will make the network more likely to overfit the data; generally, however, it is better to choose too large of a size and avoid overfitting by using regularization or pruning [3]. While there may be guidelines to determine appropriate values for each of the network parameters, a sizable amount of trial and error is needed to find the optimal values.

Advantages and disadvantages of neural networks

Neural networks have received a lot of attention over the past few decades for their ability to approximate any function with arbitrary accuracy; this is by far their biggest strength as a machine learning technique. Training, while often time-consuming, proceeds automatically. Furthermore, neural networks can also easily handle both classification problems and regression problems.

However, there are a number of drawbacks to using neural networks. First of all,

neural networks are not easily interpretable. By looking at a network graph or, even worse, a list of network weights, it is nearly impossible for a human to determine the relationships of the various input variables to the outcome variable. Secondly, there are a large number of parameters that need to be set; choosing appropriate parameter values that can generate a network which will generalize well to new samples can be a difficult process. Third, training, especially back-propagation, can proceed very slowly, especially when there are a large number of input variables or a large number of training samples. Lastly, training has a tendency to overfit the training data or to get stuck in local minima, preventing the network from finding the global minimum of the training error.

Despite their drawbacks, artificial neural networks are generally a very powerful and flexible machine learning technique, and are very common in practice.

2.3.4 Support vector machines

A support vector machine (SVM) attempts to classify a number of samples by nonlinearly mapping the input vectors into a higher-dimensional space, and then finding the *optimal separating hyperplane* between classes in the new space, which is also called the *feature space*. The linear separator in the feature space generally corresponds to a nonlinear separator in the original input space [33]. The mapping from the input space to the feature space can be seen as a function $h(x_i)$, defined as

$$h(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_m(x_i)), \tag{2.37}$$

where $h_m(x_i)$ is known as the m^{th} transformation of x_i [16]. SVMs are generally applied to classification problems; only two-class classification problems will be discussed here, although SVMs can be extended to handle both multiple classes and regression problems.

The general theory behind optimal separating will be discussed before highlighting their use in combination with nonlinear mappings. Note that, for the discussion of separating hyperplanes and SVMs, the outcome variable y_i will be assumed to be

equal to -1 or 1, rather than 0 or 1, depending on the class; this helps to simplify many of the formulas.

Optimal separating hyperplanes

A separating hyperplane, generally defined as $f(x_i) = b + x_i^T w = 0$, divides the input space into two region, with each region containing samples belonging to only one class. With this definition, the vector $w^* = \frac{w}{\|w\|}$ is normal to the hyperplane, and the signed distance from a point x_i to the hyperplane is given by $\frac{1}{\|w\|} f(x_i)$ [16]. For a given set of linearly separable samples, there are an infinite number of separating hyperplanes.

The optimal separating hyperplane is defined as the separating hyperplane with the largest distance to the closest point in each of the two classes; equivalently, the optimal separating hyperplane has the widest separating margin between the two classes. The optimal separating hyperplane is an attractive solution because it is unique and it is the most likely to generalize well to new samples. The optimal separating hyperplane can be found by solving the optimization problem

$$\begin{gathered} \textit{maximize } C \textit{ w.r.t. } b,\ w \\ \textit{subject to } \tfrac{1}{\|w\|} y_i (b + x_i^T w) \geq C,\ \textit{for each training sample } i, \end{gathered} \tag{2.38}$$

where b is a constant, w is a vector of constants, x_i is the input vector for training sample i and y_i is the outcome variable for training sample i. Since $y_i = \pm 1$, the constraints require that all points are on the correct side of the separating hyperplane, and that each is at least a distance of C away from the hyperplane. The hyperplane defined by $f(x_i) = 0$ is unchanged for any constant multiple of b and vector w, and so, setting $\|w\| = \frac{1}{C}$, the optimization problem given in (2.38) simplifies to

$$\begin{gathered} \textit{minimize } \tfrac{1}{2}\|w\|^2 \textit{ w.r.t. } b,\ w \\ \textit{subject to } y_i (b + x_i^T w) \geq 1,\ \textit{for all } i. \end{gathered} \tag{2.39}$$

The equivalent *Lagrange primal* function is

$$L_P = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i[y_i(b + x_i^T w) - 1], \tag{2.40}$$

where α_i is the i^{th} Lagrange multiplier. Equation (2.40) is to be minimized with respect to b and w. Setting the derivatives with respect to b and w equal to zero gives

$$w = \textstyle\sum_i \alpha_i y_i x_i, \quad and \tag{2.41}$$

$$0 = \textstyle\sum_i \alpha_i y_i, \tag{2.42}$$

which can be substituted into Equation (2.40) to the give *Wolfe dual* problem, which is

$$\begin{gathered} maximize\ L_D = \textstyle\sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j \\ subject\ to\ \alpha_i \geq 0\ and\ \textstyle\sum_i \alpha_i y_i = 0. \end{gathered} \tag{2.43}$$

This can be solved by many common software packages [16].

A property of the solution to the optimization problem (2.43) is that it must satisfy the *Karush-Kuhn-Tucker complementarity conditions*, one of which is given by

$$\alpha_i[y_i(b + x_i^T w) - 1] = 0,\ for\ all\ i. \tag{2.44}$$

This implies that, for every point x_i with a nonzero value of α_i, $y_i(b + x_i^T w) = 1$, in which case x_i lies on the margin and is called a *support vector*. All other points have $\alpha_i = 0$ and $y_i(b + x_i^T w) > 0$, so they do not lie on the margin; the constraints for these points do not factor into the optimization problem, nor are these points used in the solution for b or w [33].

Once the optimal values of α_i are found, w can be calculated using equation (2.41) and b can be found by applying equation (2.44) to any support vector. The resulting classifier is given by

$$G(x_i) = sign(b + x_i^T w); \tag{2.45}$$

that is, all points on one side of the hyperplane are classified as one class, and all on

the other side of the hyperplane are classified as the other [33].

The method for finding the optimal separating hyperplane given in previous paragraphs can only be applied to linearly separable problems. It is often useful to be able to find the optimal separating hyperplane when the classes are not linearly separable by allowing some amount overlap between the two classes. This is done by introducing slack variables ξ_i into the optimization problem (2.38), thus giving a new optimization problem

$$\begin{gathered} maximize\ C\ w.r.t.\ b,\ w \\ subject\ to\ \tfrac{1}{\|w\|} y_i(b + x_i^T w) \geq C(1 - \xi_i),\ for\ all\ i, \\ \xi_i \geq 0\ for\ all\ i\ and \sum_i \xi_i \leq K, \end{gathered} \tag{2.46}$$

where K is a constant. Here, ξ_i is proportional to the amount by which the i^{th} sample is on the incorrect side of the margin, and so limiting the sum of all the slack variables to be less than a constant limits the total amount by which all samples are allowed to be on the incorrect side of the margin [16]. Again setting $\|w\| = 1/C$, gives the optimization problem

$$\begin{gathered} minimize\ \tfrac{1}{2}\|w\|^2 + \gamma \sum_i \xi_i\ w.r.t.\ b,\ w \\ subject\ to\ y_i(b + x_i^T w) \geq 1 - \xi_i,\ for\ all\ i \\ and\ \xi_i \geq 0,\ for\ all\ i, \end{gathered} \tag{2.47}$$

where γ has replaced K from the problem given by (2.46); the effects of the two parameters are inversely proportional [16]. The method used to transform and solve the optimization problem given in (2.47) is equivalent to the method used to solve the problem given in (2.39); the description is omitted here for brevity.

For the optimal separating hyperplane problem where some overlap is allowed between the classes, the γ parameter can be adjusted to give various separating hyperplanes. If γ is large, the $\gamma \sum_i \xi_i$ term in problem (2.47) has a large effect on the minimization, and so very little overlap is allowed between the points and the margin; when γ is equal to infinity, all points must be on or outside the margin, as in

the original optimal separating hyperplane. Likewise, if γ is small, the $\gamma \sum_i \xi_i$ term in problem (2.47) has a small effect on the minimization, and so considerable overlap is allowed [16].

SVMs using feature space mappings

SVMs add more flexibility to the linear optimal separating hyperplane method by calculating the separating hyperplane in a higher-dimensional feature space. The separating hyperplane in the feature space often corresponds to a nonlinear separating boundary in the original input space.

Given a mapping $h(x)$, as defined in Equation (2.37), the Wolfe dual function that needs to be maximized to find the optimal separating hyperplane is

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j h(x_i)^T h(x_j), \tag{2.48}$$

and so the optimization problem depends only on the dot product of two transformed vectors, rather than the actual transformed vectors themselves. Because of this, the mapping $h(x)$ itself is never needed, rather only the Kernel function

$$K(x_i, x_j) = h(x_i)^T h(x_j) \tag{2.49}$$

is needed [16].

There are many common kernels used in SVMs. The polynomial kernel of d dimensions is given by

$$K(x_i, x_j) = (x_i^T x_j)^d. \tag{2.50}$$

The polynomial kernel

$$K(x_i, x_j) = (1 + x_i^T x_j)^d \tag{2.51}$$

includes all dimensions up to and including d. Two other common kernel functions are the sigmoid kernel

$$K(x_i, x_j) = (\kappa(x_i^T x_j) + \Theta), \tag{2.52}$$

where κ and Θ are constants, and the radial basis kernel

$$K(x_i, x_j) = e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}, \tag{2.53}$$

where σ is a constant. The appropriate choice of kernel depends on the specific problem being classified and often requires trial and error to find the one which performs the best [33].

Given a Kernel function $K(x_i, x_j)$, the Wolfe dual becomes

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i, x_j), \tag{2.54}$$

which can be solved to give the optimal separating hyperplane in the new feature space. The resulting optimal separating hyperplane in the feature space is given by

$$f(x_i) = h(x_i)^T w + b = \sum_j \alpha_j y_j K(x_i, x_j) + b. \tag{2.55}$$

As always, the classifier based on this separating hyperplane is given by the sign of $f(x_i)$.

In a higher-dimensional space, there will generally always be a hyperplane which can completely separate the training set without any overlap between the classes, so the γ parameter is used as a type of regularization to avoid overfitting. If γ is very large, the separating hyperplane will not contain much overlap between the classes, and so the resulting separator in the original space will likely be highly curved and may overfit the training samples. If γ is small, the separating hyperplane will be less sensitive to the particular choice of training set, and so the boundary will be much smoother in the original input space [16].

Advantages and disadvantages of SVMs

SVMs have become popular recently, in part, because they take advantage of the theoretically attractive method of optimal separating hyperplanes. Also, with the appropriate choice of Kernel function, an SVM can classify any function and can

avoid overfitting with use of the γ parameter.

When using an SVM, the choice of Kernel function requires either prior knowledge of the problem or some amount of trial and error. Also, while it has been claimed that SVMs avoid, at least partially, the curse of dimensionality, this is generally false [16]; that is, SVMs suffer many of the same problems as other methods when the number of inputs is proportionally much larger than the number of samples.

2.4 Variable selection methods

Both a gift and curse of microarray analysis is the vast amount of data that is generated. However, when thousands of genes are analyzed, and thus thousands of values are generated for each sample, there are likely to many irrelevant genes which can complicate the data analysis. In fact, in most cases there are only a few truly relevant genes and thousands of irrelevant ones. With thousands of extraneous genes, it is likely that some of these genes, due to chance and noise in the data, will be highly correlated with the outcome variable but they are, in fact, completely unrelated. Sorting through the vast number of irrelevant genes to find the truly relevant ones is not a trivial task.

The goal of variable selection in the analysis of gene expression matrices is two-fold. First, variable selection allows models to be built using machine learning techniques that are both simple and accurate. Even some of the more simple machine learning methods cannot build a model using thousands of inputs in a reasonable amount of time due to the massive amount of computation required; perhaps the only exceptions is decision trees, which can handle a vast number of inputs because of the heavy restrictions placed on the possible splits. However, even if a model could be built in a reasonable amount of time using all of the available genes, the resulting model would likely be overly complex and generalize poorly to new samples. Such a model would likely be built using very small contributions from a large number of genes, making the model highly sensitive to noise in the training set and therefore causing the model to overfit the training data and generalize poorly. Furthermore, the resulting models

will be more easily interpretable.

The second goal of variable selection in gene expression analysis is to discover the genes that are most important to the problem being solved. This would allow further research to be done on a limited number of genes, which could potentially help to determine the role, if any, of theses genes in the disease or problem being studied.

Since microarray studies began in the late 1990's, a variety of different variable selection methods have been tried. Since the field of microarray gene expression analysis is in its infancy, there are few established methods for variable selection and many of the methods used are seemingly *ad hoc*. Sections 2.4.1 through 2.4.3 describe a few methods that can be used for variable selection; these methods are geared towards gene expression analysis.

2.4.1 Selection based on signal-to-noise ratio

Many variable selection techniques use some form of correlation between each gene and the outcome variable. The genes are ranked according to their correlation with the outcome variable, and a number of the most positively correlated genes are selected along with a number of the most negatively correlated genes. In general, if N of the most informative genes are desired, $\frac{N}{2}$ positively correlated genes and $\frac{N}{2}$ negatively correlated genes are selected. This method for variable selection is attractive not only because it retrieves the genes which are likely to be the most related to the outcome variable, but also because the number of genes can easily be varied by simply selecting fewer or more genes from the ranked list.

In what has become a seminal work on microarray data analysis, Golub *et al* used a measure of correlation known as the *signal-to-noise ratio* (SNR) to select informative genes. The SNR for a gene g is defined as

$$SNR(g) = \frac{\mu_{g,1} - \mu_{g,2}}{\sigma_{g,1} + \sigma_{g,2}}, \tag{2.56}$$

where $\mu_{g,i}$ and $\sigma_{g,i}$ are the mean expression level and the sample standard deviation, respectively, of gene g for samples in class i. A positive value of $SNR(g)$ means that

gene g is more highly expressed in class 1, and, similarly, a negative value means that g is more highly expressed in class 2. Furthermore, a large absolute value of $SNR(g)$ means that gene g is highly correlated with the outcome class. By using the 25 genes with the most negative values of $SNR(g)$ and the 25 genes with the most positive values, Golub *et al* were able to build a model that could accurately distinguish between acute myeloid leukemia and acute lymphoblastic leukemia [11].

A number of other measures exist for determining the correlation of each gene with the outcome variable. For example, the t-statistic is similar to the SNR and is defined as

$$t\ statistic(g) = \frac{\mu_{g,1} - \mu_{g,2}}{\sqrt{\frac{\sigma^2_{g,1}}{N_1} + \frac{\sigma^2_{g,2}}{N_2}}}, \tag{2.57}$$

where $\mu_{g,i}$ and $\sigma_{g,i}$ are again the mean expression level and the sample standard deviation, respectively, of gene g for samples in class i and N_i is the number of samples in class i [25]. Some authors believe that the Pearson correlation coefficient is a reasonable choice for regression problems – or for classification problems, for that matter – and is given by

$$r = \frac{\sigma_{xy}}{\sigma_x \sigma_y}, \tag{2.58}$$

where σ_x and σ_y are the sample standard deviations of x and y, and σ_{xy} is the sample covariance between the two [7], although there are several instances in which this would not work.

The signal-to-noise ratio is used here over other correlation measures primarily because of the success of the famous Golub *et al* study. The same method has proven effective in more recent microarray analysis studies as well [28].

2.4.2 Selection based on Student's t tests and fold analysis

Many studies also use a *Student's t test* in combination with other discrimination methods, such as fold analysis, as a means for variable selection [13, 15, 34]. A Student's t test determines the probability p that two sets of measurements, each sampled from a normal distribution, were actually sampled from the same normal

distribution [10]. Here, the two sets of measurements are the gene expression levels for a particular gene for samples of the first class and samples of the second class, respectively. If the p value is low enough for a particular gene, there is a statistically significant chance that the gene is differentially expressed between the two classes. Generally, a test is considered statistically significant if $p \leq .05$.

There are drawbacks to using a Student's t test as the sole means for variable selection. First of all, if the differential expression of a gene is considered statistically significant if $p \leq .05$, then, in a gene expression matrix containing 10,000 irrelevant genes, approximately 500 genes would be identified simply by chance; however, a number of methods exist to mitigate this effect. Furthermore, using a Student's t test may not be an appropriate approach because gene expression levels are generally not normally distributed. When used, a Student's t test is often used in combination with another selection method such that a variable is selected if and only if it has a statistically significant p value and is also selected by the complimentary selection method.

2.4.3 Stepwise variable selection

One of the benefits of using logistic regression to build a classifier model is that most statistical software packages that support logistic regression also support stepwise forward and backward variable selection. In stepwise forward variable selection, a one-input model is built using each of the input variables. The models are then ranked according to some criterion; in logistic regression, this is often the probability p that the true variable coefficients are zero. The input variable that generates the best one-input model is kept, and a two-input model is built using this input and each of the other input variables. The additional input variable that generates the best two-input model is kept, and this process continues until the number of inputs in the model reaches some specified size [10]. Other criteria, such as cross validation error or the *Akaike Information Criterion* (AIC), are also used in stepwise selection.

Stepwise backward variable selection in logistic regression works in the opposite fashion. That is, a model is built using all inputs and the input variable is with the

largest p value in the model is removed. Then, starting with the new model, the input variable with the largest p value in the new model is removed. This process continues until the number of input variables in the model reaches some set size or until the p value of the next variable to be removed is below some threshold [10].

In the case of gene expression analysis, the number of input variables generally needs to be reduced prior to using stepwise selection, so another form of variable selection is first applied, followed by stepwise selection. Similar methods of stepwise forward and backward selection can be applied to types of models other than logistic regression, but software to perform such analysis is often not readily available.

2.5 Comparing model performance

When a large number of models are built using various machine learning techniques, various parameter values, and various input variables, it is necessary to have standardized methods to compare the numerous models. A number of accepted methods exist for both regression and classification models. Furthermore, there are methods are available for right-censored survival data which allows even right-censored samples to be used in model comparisons; right-censored data and survival analysis are discussed in Section 2.5.3. These methods can be used to evaluate model performance using the training set as well as using a separate test set.

2.5.1 Evaluating regression models

Regression models are typically evaluated based on the model error, which is some measure of the total or average difference between the model predictions and the actual outcomes. The are a number of suitable error measures, but sum of squares is most often used, and is given by

$$Error = \frac{1}{N} \sum_i (f(x_i) - y_i)^2. \tag{2.59}$$

The $\frac{1}{N}$ term in Equation (2.59) makes this an average error over the set of samples, so that the performance of models on sets of varying sizes can be meaningfully compared.

2.5.2 Evaluating classification models

Accuracy – the proportion of samples classified correctly – seems like a natural and intuitive measure of a classification models performance. However, there are a number of drawbacks to using accuracy to evaluate classification models. First of all, accuracy may be deceptive when there are significantly more samples in one class than the other. For example, suppose that, in a set of patients, 1% of the patients have a particular disease. Then, a completely uninformative model could simply classify all patients as healthy and achieve a seemingly good 99% accuracy. Furthermore, even if the number of samples in each class are relatively equal, reporting accuracy as a measure for a model's performance doe not give any sense as to whether a model is more likely to classify a sick person as healthy or a healthy person as sick. Therefore, while accuracy, or perhaps misclassification cost, is a reasonable measure for the performance of a multiple class classification problem, there are more informative measures for two-class models.

Before discussing other performance measures for two-class classification models, it is necessary to first give a few definitions. Suppose a data set consists of patients from two classes; one class contains patients with a particular disease, also called the sick patients, and the other class contains patients without that disease, also known as the healthy patients. Then, for a given classification model, a *true positive* is a sick patient that the model correctly classifies as sick; a *false positive* is a healthy patient that the model incorrectly classifies as sick; a *true negative* is a healthy patient that the model correctly diagnoses as healthy; and finally a *false negative* is a sick patient that the model incorrectly diagnoses as healthy.

For medical diagnostic tests, and two-class classification models in general, performance is often reported in terms of *sensitivity* and *specificity*. Sensitivity is defined as the proportion of patients with the disease who are classified as sick, and can be

calculated using the formula

$$sensitivity = \frac{(true\ positives)}{(true\ positives) + (false\ negatives)}. \tag{2.60}$$

Conversely, specificity is defined as the proportion of patients without the disease who are classified as healthy, and can be calculated by

$$specificity = \frac{(true\ negatives)}{(true\ negatives) + (false\ positives)}. \tag{2.61}$$

Occasionally, the *positive prediction value* (PPV) and *negative predictive value* (NPV) of a test or model are reported; these are defined as

$$PPV = \frac{(true\ positives)}{(true\ positives) + (false\ positives)}, \tag{2.62}$$

$$NPV = \frac{(true\ negatives)}{(true\ negatives) + (false\ negatives)}. \tag{2.63}$$

In many cases, a diagnosis test or classification model has a continuous output, and the cutoff point to determine which patients are sick and which are healthy can often be varied. In general, the sensitivity and specificity of a test or model vary inversely with one another; that is, as the cutoff point is changed to make a test more sensitive, the test also generally becomes less specific and *vice versa*. A *receiver operating characteristic* (ROC) curve is used to capture this tradeoff between sensitivity and specificity. For a given test, an ROC curve plots *(sensitivity) vs. (1 – specificity)*, and can be used to pick the appropriate cutoff point, taking into account the misclassification costs. A 45° line is often included on the ROC plot because it represents a completely uninformative test – that is, a test that performs no better than one which randomly assigns patients to classes. The amount by which a test or model's ROC curve is above the 45° line is a good indication of how informative or powerful the test or model is. A test or model whose ROC curve consistently falls below the 45° line can be made more powerful by simply reversing every prediction the test or model makes. In this way, the area underneath the ROC curve, a value

that is also known as the *C-statistic*, is a good measure of the performance of a test of classification model. An ROC area of .5 means the particular model is completely uninformative, while an ROC are of 1 means the model can perfectly discriminate between the two classes.

In this paper, the performance of classification models will, in general, be reported in terms of their ROC areas. However, because the problem being solved involves right-censored survival, there are additional methods, discussed in Section 2.5.3, which can make use of even the censored samples in the model evaluation.

2.5.3 Survival analysis

Survival analysis is concerned with the length of time it takes before each patient suffers some event of interest; this event will be considered death for the rest of the discussion on survival analysis, but could include a number of other things, such as recurrence of a particular disease. Given a set of patients, a *survival function*, which gives the probability a patient will survive to a given time, can be constructed as

$$S[t] = \frac{m_t}{N}, \tag{2.64}$$

where m_t is the number of patients who have not died by time t, and N is the total number of patients in the data set. Depending on the problem, time t can be measured in, for example, hours, days, or months. In contrast to the survival function, the *cumulative mortality function* gives the probability a patient will have died by a given time, and is defined as

$$D[t] = 1 - S[t]. \tag{2.65}$$

However, since the length of follow-up time with patients is often limited, some patients will still be alive when patient follow-up ends. These patients are said to be *right-censored*. Without modification, the survival function in Equation (2.64) would overestimate the probability of being alive at a given time, since some of the

right-censored patients may have died by that time, without the data reflecting that fact [10].

The *Kaplan-Meier survival function* is constructed in a different way from the survival function given in Equation (2.64) and better approximates the true survival probability when there are some right-censored samples. Let p_i be the probability of dying during the i^{th} time period, which can be calculated by

$$p_i = \frac{n_i - d_i}{n_i}, \tag{2.66}$$

where n_i is the number of patients at risk – that is, the number of patients that have not yet died or been censored – at during time period i, and d_i is the number of patients who die during this time period. Then, the Kaplan-Meier survival function is is given by

$$\hat{S}[t] = \prod_{i=0}^{t} p_t. \tag{2.67}$$

Note that $p_i = 1$ for time periods during which patients do not die, so these values are frequently omitted from Equation (2.67). The *Kaplan-Meier cumulative mortality function* is then given by

$$\hat{D}[t] = 1 - \hat{S}[t]. \tag{2.68}$$

Most statistical software packages can calculate Kaplan-Meier curves, as well as their corresponding confidence intervals. Note that the accuracy of Kaplan-Meier curves decreases as time increases, because each estimate is based on fewer samples as more samples become right-censored; equivalently, the confidence intervals around the curves widen as time increases. A Kaplan-Meier survival curve accurately reflects the underlying true survival curve if the patients are representative for the population as a whole, and the censored patients have the same risk of dying as non-censored patients [10].

Given two Kaplan-Meier survival curves, a test known as the *logrank test* can be used to determine the probability p that the two curves were actually derived from the same underlying survival function [10]. A logrank test can be used to evaluate

the performance of a classification model. That is, two Kaplan-Meier survival curves can be constructed, one for each of the two classes, as using the classification model to place samples into the two classes; a lower p value between the two resulting Kaplan-Meier curves means a model has found a more significant division of classes.

Similarly, a method derived from *Cox proportional hazard regression* can also be used to evaluate classification models. Related to the survival function, the *hazard function* $\lambda[t]$ is the instantaneous rate of death for patients in the data set. For a small time period Δt, $\lambda[t]\Delta t$ can be seen as the probability that a patient who is alive at time t will die by time $t + \Delta t$. The hazard function and the survival function are related by

$$S[t] = e^{-\int_0^t \lambda[x]dx}. \tag{2.69}$$

Furthermore, two groups of patients are said to have *proportional hazards* if

$$\lambda_1 = R\lambda_0, \tag{2.70}$$

where λ_1 and λ_0 are the hazard functions for the two groups, and R is called the *relative risk* between the groups. A Cox proportional regression model assumes that the i^{th} patient has an instantaneous rate of death of the form

$$\lambda_i = \lambda_0 e^{\sum_k \beta_k x_{ik}} \tag{2.71}$$

where x_{ik} is the k^{th} input variable for the i^{th} patient and β_k is the coefficient for this input variable. Similar to logistic regression, the value of e^{β_k} is the relative risk increase associated with a one unit increase in x_{ik}. The probability p that each coefficient β_k is actually zero can be found using statistical software [10].

To use Cox hazard regression to compare models, x_i in Equation (2.71) does not contain the input variables for sample i; instead x_i only contains a single binary variable which denotes the class of sample i as predicted by a classification model. Then, the p value for the coefficient β_0 will equal the probability that samples from the two predicted classes actually have the same relative risk; a lower p value means a

model has found a more significant division of the two classes. Classification models can be compared using this value of p in the same way that the models are compared using the logrank p values; for the most part, the two tests produce similar results.

Note that, for either the logrank test or Cox proportional hazard regression, the classification model being evaluated has found a statistically significant distinction between groups if $p \leq .05$.

2.5.4 N-fold cross validation

When the training set is small, cross validation can be used to evaluate the performance of various models. In cross validation, the training set is broken into K equally sized sets. A model is built using $K - 1$ of these sets and is then tested using the samples in the hold-out set. This process is repeated K times, with each set begin used as the validation set once. The model performance is then reported as the total performance – or sometimes the average performance – of the model on all of the validation sets. In this way, each training sample is used both to build and test the model, so the model can be evaluated without reducing the size of the training set.

In *N-fold cross validation*, also called *leave-one-out cross validation*, each validation set consists of only one sample. Thus, each model is built using all but one of the training samples and is then used to predict the excluded sample. This process is repeated so that each sample is used as the validation set once. N-fold cross validation is particularly useful when the training set is extremely small, so that each model is built using as many as possible of the available training samples.

In classification problems, cross validation performance is often reported as the error rate – or, conversely, the accuracy – of the model when applied to the validation samples. In regression problems, cross validation performance is typically reported as the average sum of squares error of the validation samples.

Chapter 3

Materials and Methods

This chapter describes the specific data set and software used in this paper, as well as exact methods for variable selection, machine learning, and model comparison.

3.1 Mesothelioma data set

Collected at the Brigham and Women's Hospital (BWH) in Boston, the data set used in this paper consisted of microarray gene expression data for 31 patients who underwent an extrapleural pneumonectomy as a result of malignant pleural mesothelioma, without additional preoperative treatment. A total of 60 tumor specimens were collected; the remaining 29 samples were saved to be used as a test set for the model built by the Gordon *et al* study [15]. Unfortunately, at the time this paper was written, the data for this test set was unavailable. However, the data for these additional samples would not have been very helpful to the current analysis since, for the test set, only a select few genes were analyzed using the more precise method of polymerase chain reaction; any model using any other genes would have been unable to take advantage of the test set. The expression data was collected using Affymetrix U95A chips and the resulting gene expression matrix was generated using Affymetrix Microarray Suite, version 5. All of this preprocessing was done at BWH [14].

Note that the gene expression data set used by the Gordon *et al* study was generated using an earlier version of the Affymetrix Microarray Suite and therefore con-

tained negative values for some of the expression levels. The matrix generated using the newer version contains no negative values and so was preferable to the older data set.

For two of the patients in the study, three separate microarrays were used to collect expression information and so the gene expression matrix contained three separate columns for each of the two samples. The median of the three values for each gene expression level was used; using the median for each gene used more of the available data than using only one of the three columns and was less sensitive to outliers and incorrect values than using the mean would have been.

Each of the patients contained in the gene expression data was annotated with several patient and tumor attributes in addition to the gene expression levels and patient survival information. This additional information included patient age and sex, as well as tumor stage and histology. The tumor histology described the tumor subtype as epithelial, sarcomatoid, or mixed.

3.2 Recreating the results of Gordon *et al*

The first goal of this paper was to recreate and verify the results of the Gordon *et al* study, using both the original data set and the newer data set – that is, the data set generated with a newer version of the Affymetrix software. The Gordon *et al* study built a predictive model based on the geometric mean of three ratios of gene expression values. To recreate these results, the samples were first ranked according to their survival. Patients with survivals in the 25^{th} percentile or less and patients with survivals in the 75^{th} percentile or more formed the poor and good outcome classes, respectively.

For each gene, a two-tailed, unpaired Student't t test was performed between the expression levels in the poor and good groups. A gene was considered informative if it had a t test p value of less than .05, if the mean expression level in at least one of the two groups was greater that 500, and if the mean expression level in one of the two groups was at least twice that of the other group. The last two criteria seemed

somewhat *ad hoc*, although the justification was that these criterion helped to select genes which were less sensitive to noise [15]. This method of variable selection will be referred to as the *Gordon method.*

From the list of informative genes, the four genes which were most statistically significantly overexpressed in the poor outcome group – that is, the four genes with the lowest t test p values among those whose mean expression level in the poor group was at least twice that of the good group – and the four genes which were most statistically significantly overexpressed in the good outcome group were used to form sixteen ratios of genes overexpressed in the good group to those overexpressed in the poor group. The resulting ratios were used to predict the classes of training set samples. Using the most accurate of these ratios, new predictors were created by calculating the geometric mean of all possible three ratio combinations. The resulting predictors were again applied to the training set, and the most accurate were kept [15].

The analysis described in this paper, this process of choosing the good and poor groups, selecting the informative genes based on these groups, and finally creating the predictive ratios was performed on both the original data set and the newer data set. Cross validation was performed in each case by leaving each of the training samples out in turn, selecting genes and constructing the most accurate three ratio geometric mean predictor with the remaining training samples, and using this predictor to predict the outcome of the excluded sample [15]. Since the independent test set was unavailable, the predictors were applied to the remaining samples in the data set that were not used in the training set. However, since all of these samples have survivals close to the median, they were the most difficult to predict and certainly were not randomly chosen, so the performance on these samples was generally not indicative of the ability of the models to generalize to new samples.

While the Gordon *et al* study achieved good results, both on the training set and independent test set, their method used only 17 of the available 31 samples in training. The analysis methods described in Section 3.3 divided the data set randomly into a training and test set so that all samples could be used. For the classification problem – that is, trying to predict good or poor outcome based on dying before or after the

median survival, respectively – 21 samples were used in the training set and ten were left to use as a test set. For the regression problem – that is, predicting the actual survival time – only 17 samples could be used in training set and six in the test set because a number of the samples were right-censored.

All calculations used to recreate the results of Gordon *et al* were done in Microsoft Excel.

3.3 Additional gene expression analysis

The second goal of this paper was to apply other variable selection and machine learning techniques to the same prediction problem that was studied by Gordon *et al.* Since the independent test set was unavailable, as many as possible of the available samples needed to be included in either the training or test sets. The analysis described in this paper was concerned with building both classification models to classify samples as being above or below the median survival and regression models to predict actual survival in months; because many of the samples were right-censored, some are not suitable for use in one or both types of models. In order to identify suitable samples, first the median survival needed to be calculated. Of the 31 samples contained in the gene expression matrix, a total of 8 were right-censored. The median survival was calculated by first calculating the median for all of the uncensored samples and then including each of the right-censored samples one by one, from the longest known survival to the shortest, until the next censored sample to be added had a known survival time less than the current median. This method used as much as possible of the available data without biasing the calculation. The resulting median survival was 11 months; this, for the classification problem, patients were considered to have a poor outcome if they had survived less than 11 months and were considered to have a good outcome if the survived for 11 months or longer. Only those samples that were right-censored before reaching 11 months could not included in either the poor or the good groups. A total of 28 samples were qualified for use in the classification problem; the remaining three samples, which were right-censored prior to reaching the

median survival, could only be used to construct Kaplan-Meier curves and compare the various classification models using the logrank test.

For the regression problem, no right-censored samples could be included in either the training or test set, because doing so would have biased the results. Thus, a total of 23 samples were suitable for use in the regression problem.

The unavailability of an independent test set necessitated that the already small data set be separated into a training set and an independent test set. Seven samples were excluded from the samples suitable for the classification problem to be used as a test set, resulting in a training set of 21 samples. While this training set is small, it still four samples larger than the training set used by Gordon *et al.* Six of the excluded samples were also samples suitable for regression, leaving a total of 17 samples for use in regression training. Thus, the training sets for classification and for regression were as similar to one another as possible. To choose the specific training and test sets, the training and test set labels were randomly permuted until the absolute value of the correlation between the training set label and the good or poor outcome label for classification samples was less than .05 and the absolute value of the correlation between the training set label and the survival outcome for regression samples was also less than .05.

All of the analysis methods described in this section and in Sections 3.3.1 through 3.3.3 used the newer data set since it contained no negative expression level values.

3.3.1 Variable selection

In addition to being used to recreate the results of Gordon *et al*, the Gordon method of variable selection described in Section 3.2 was applied to the new training set; the set of selected genes was used as inputs to a variety machine learning methods to generate predictive models.

In contrast to the Gordon *et al* study, Golub *et al* ranked genes according to the signal-to-noise ratio (SNR) of the log expression values. In keeping with this approach, the logarithm of all of the expression levels were calculated; the SNR was calculated for each gene using the training samples. Genes were ranked according to

the resulting SNR values and this ranking was used to select genes as inputs for the machine learning methods used. When N genes were needed, the $\frac{N}{2}$ genes with the most positive values and the $\frac{N}{2}$ genes with the most negative values of the SNR were selected. This method of gene selection will be referred to as the *Golub method.* A total of four gene sets were generated using the Golub method, containing the best 4, 10, 30, and 50 genes, respectively.

Although there was some justification, the choice of 4, 10, 30, and 50 genes was somewhat arbitrary. A gene set with 50 genes was chosen as the largest set because using significantly more than this number of genes maked the computation time, especially in neural networks, prohibitively long; Golub *et al* also used a total of 50 genes in their predictive model [11]. Similarly, the gene set with 4 genes was chosen because Gordon em et al used only 4 genes in their prognostic predictor of mesothelioma [15]. Sets with 10 and 30 genes were reasonable compromises between the two extremes.

A total of five different gene sets – the one using the Gordon method and the four using the Golub method – were generated to use as inputs to the machine learning methods. In order to examine the effects of the additional patient information – such as age, sex, tumor subtype and tumor histology – on the performance of the of constructed models, two input variable sets were generated for each gene set, one containing the additional patient information and one without these additional variables. Thus, a total of ten different input variable sets were generated to use in all the types of machine learning.

Lastly, although decision trees can handle a large number of input variables, with over 12,000 total genes in the gene expression matrix, it is necessary to reduce the number of inputs to speed up computations and reduce the chance of finding irrelevant genes that produce good splits. Thus, an input variable set containing the 500 genes with the lowest Student's t test p values, which included all genes with a p value of less than .05, was also generated to use to build a decision tree model.

All calculations needed for variable selection were done using Microsoft Excel.

3.3.2 Machine learning

Classification trees were built using each of the ten different input variable sets and N-fold cross validation was performed to assess the ability of the trees to generalize to new inputs. A tree was also built using the 500 genes with the lowest Student's t test p values. Each tree was built until the leaf nodes were completely pure and no pruning was performed. The best classification tree model, based on the cross validation error rate, was then applied to the independent test set; if two or more trees had the same error rate, the one based on the fewest input variables was chosen.

Classification trees were built using the demonstration version of See5, release 1.20a. This software was unable to handle regression trees.

Four neural networks for classification were trained for each of the ten input variable sets. All neural networks used batch learning with a learning rate of .25 and a momentum coefficient of .9 and all networks were training for a total of 500 epochs. All the networks used hyperbolic tangent activation functions in both the hidden and output layers; the hyperbolic tangent function was chosen over the sigmoid function because networks with hyperbolic tangent activation functions trained faster than those using sigmoid activations. All inputs, including both gene expression levels and additional patient information variables, were standardized to have zero mean and one variance, and the initial network weights were generated from the uniform distribution ranging from -.7 to .7. This initial weight distribution generally produces good results when the inputs are standardized [16].

One of the four neural networks trained for each input variable set contained eight hidden neurons and used no regularization; the relatively large number of hidden neurons presumably allowed every training set to be perfectly modeled. The second network used only two hidden neurons and had no regularization; the small size of this network presumably forced the model to assume a simpler structure and thus avoided overfitting the training set. The final two networks used eight hidden neurons,but used weight decay regularization with ν coefficients of .5 and 2.5, respectively, to avoid overfitting.

N-fold cross validation was used to assess the performance of each network. Since the neural networks produced continuous results, the cutoff value to determine which samples were predicted as poor and which were predicted as good could be varied to change the sensitivity and specificity of the network. For each of the N networks used in N-fold cross validation, this cutoff value was chosen to maximize the sum of the training set sensitivity and specificity and the excluded training sample was predicted using this cutoff value. Since the neural network cross validation results were due, in part, to the random starting weights of the networks, cross validation was performed five separate times for each network configuration and the resulting error rates were averaged. The best neural network configuration, based on the average cross validation error rate, was then applied to the independent test set; if two or more neural networks had the same average error rate, the one based on the fewest number of inputs, with the smallest structure, or with the highest regularization coefficient was chosen.

Four neural networks for the regression problem were also created using each of the ten input variable sets. These four networks used the same parameters as those used for the classification problem, except linear activation functions were used in the output neurons. N-fold cross validation was then used to find the average sum of squares error for each network. The N-fold cross validation was applied a total of five times and the resulting error values were averaged, thus helping to reduce variance in the cross validation error due to the random starting weights of the networks. The best network in terms of average cross validation error was then applied to the independent test set.

All of the neural networks used in this paper were created using GAINN software written by Jonathan Jackson [19]; minor modifications were necessary in order to perform all of the required calculations.

Four support vector machines were generated for each of the ten input variable sets, using a linear kernel, a degree-two polynomial kernel, a degree-four polynomial kernel, and a radial kernel, respectively. All of the support vector machines were created using SVM^{light} software, version 6.01, with learning module 01.09.04. As in

the case of neural networks, each input was standardized to have zero mean and one variance. The default value of the regularization coefficient, which is set dynamically by the software based on the choice of kernel function, was chosen in all cases and estimates of the cross validation error for each SVM were produced automatically by the software. The best SVM, based on the estimated cross validation error rate, was then applied to the independent test set; if two or more support vector machines had the same estimated error rate, the one based on the fewest input variables or with the simplest kernel function was chosen.

A logistic regression model was also generated using each of the ten data sets. The gene expression level inputs were standardized to have zero mean and one variance. N-fold cross validation was done by constructing a logistic regression model with the included 20 samples and predicting the class of the excluded sample using .5 as a cutoff between the poor and good classes. Additionally, logistic regression models were built using stepwise forward and backward variable selection with a p value threshold of .25 for inclusion or exclusion. N-fold cross validation was also done for the stepwise logistic regression models. Due to the limited number of samples in the training set, the software was unable to perform stepwise variable selection for the larger input variable sets. The best logistic regression model based on cross validation error rate was then applied to the independent test set.

All logistic regression calculations were done using Intercooled Stata for UNIX, version 8.2.

3.3.3 Model comparisons using the test set

A total of four models – the best classification tree, neural network, support vector machine, and logistic regression model, based on cross validation performance – were applied to the independent classification test set. The sensitivity and specificity was calculated for each of these models based on the seven test samples that were not right-censored prior to the median survival. Furthermore, using these seven samples and the three samples that were right-censored prior to the median survival, two Kaplan-Meier curves were constructed for each model based on the predicted class;

the logrank test for these two curves was then calculated.

One model, the best neural network regression model, was applied to the independent regression test set. Results were given in terms of the average sum of squares error for the six test cases.

Chapter 4

Results and Discussion

This chapter presents and discusses the results from the various analyses which were performed. Section 4.1 presents the results from recreating the Gordon *et al* analysis, using both the original data set and the newer data set generated with a more recent version of Affymetrix software. Section 4.2 presents the results from the additional variable selection and modeling analyses performed. Lastly, Section 4.3 provides a discussion of these results.

4.1 Recreated results of Gordon *et al*

A total of 17 samples were included in the training set used by the Gordon *et al* study; nine of these samples had poor outcomes and eight had good outcomes. The complete list of all 17 samples is given in Table 4.1. It is interesting that sample 34 was not included in this training set. This patient had a survival of 34 months, well above the 75^{th} percentile, and, although it was right-censored, there were several other right-censored samples in the training set. In order to recreate their results exactly, this sample was excluded from the training set in this analysis.

Using this training set, the Gordon method of variable selection, described in Section 3.2, identified a total of 46 differentially expressed genes, 24 of which were overexpressed in the good outcome group and 22 of which were overexpressed in the poor outcome group. The full list of identified genes is given in Table 4.2.

Table 4.1: Set of training samples used by the Gordon *et al* study

Sample No.	Patient age at diagnosis, yr	Sex	Tumor histology[a]	Tumor stage	Patient survival, mon	Patient status[b]
114	51	M	mixed	2	2	3
133	69	M	mixed	2	2	3
159	62	M	sarc	2	2	3
89	55	M	mixed	2	3	3
22	33	F	ept	2	5	3
6	39	M	ept	2	5	3
130	55	M	mixed	2	6	3
166	66	M	sarc	2	6	3
67	49	F	ept	2	6	3
76	67	M	ept	1	17	3
109	62	M	ept	2	19	3
33	60	F	ept	2	20	3
68	61	M	ept	2	21	3
2	44	F	ept	2	26	2
90	48	M	ept	2	28	2
74	40	F	ept	1	51	2
72	46	M	mixed	2	53	3

[a]For tumor histology, ept = epithelial and sarc = sarcomatoid.

[b]For patient status, 1 = alive without mesothelioma, 2 = alive with mesothelioma, 3 = dead from mesothelioma, 4 = dead from other causes and U = unknown [15].

The four most statistically significantly overexpressed genes in the good group and the four most statistically significantly overexpressed genes in the poor group were used to construct 16 ratios, using all possible combinations with one gene from each group of four. The resulting ratios were used to predict the classes of the training samples; a sample was considered good if the ratio was greater than one and was considered poor if the ratio was less than one. The full list of calculated ratios and their accuracies on the training set are given in Table 4.3. A total of five such ratios had an accuracy of 88% in classifying the training set, and the 16 ratios had an average accuracy of 77%. Since no expression level standardization – for example, converting every gene expression level to have zero mean and one variance – was done in this analysis, it is interesting that this method works so well to predict the training set. For example, without expression level standardization, a sample could have a

Table 4.2: Genes identified by the Gordon *et al* study

Probe ID	Gene sequence No.	Description	Ratio of good to poor mean	Student's t test p value
37405_at	U29091	Selenium binding protein 1	2.8	0.0033
41755_at	AB023194	KIAA0977 protein	2.1	0.0065
41531_at	Y09723	Zinc finger protein 151 (pHZ-67)	3.0	0.0073
36204_at	Y00815	Protein tyrosine phosphatase	2.0	0.0077
32424_at	D84424	Hyaluronan synthase 1	6.0	0.0094
35698_at	Y00318	I factor (complement)	3.6	0.0103
40456_at	AL049963	BCG-induced gene in monocytes, clone 103	3.7	0.0103
32819_at	AJ223352	Histone 1, H2bk	3.5	0.0142
376_at	AB000220	Sema domain, immunoglobulin domain (Ig)	2.3	0.0181
38459_g_at	L39945	Cytochrome b-5	2.5	0.0182
892_at	M90657	Transmembrane 4 superfamily member 1	2.8	0.0256
40016_g_at	AL050214	DKFZP586H2123 protein	2.1	0.0257
34637_f_at	M12963	Alcohol dehydrogenase 1A	14.8	0.0288
33505_at	AI887421	Retinoic acid receptor responder	3.4	0.0291
37017_at	M22430	Phospholipase A2, group IIA	2.7	0.0302
32317_s_at	U34804	Sulfotransferase family	2.3	0.0305
1042_at	U27185	Retinoic acid receptor responder	3.7	0.0327
41210_at	M81057	Carboxypeptidase B1 (tissue)	4.5	0.0329
35754_at	X78136	Poly(rC) binding protein 2	2.2	0.0368
37157_at	X56667	Calbindin 2, 29kDa (calretinin)	2.3	0.0390
286_at	L19779	Histone 2, H2aa	2.2	0.0439
32275_at	X04470	Secretory leukocyte protease inhibitor	2.9	0.0461
40035_at	AB012917	Kallikrein 11	2.3	0.0483
39400_at	AB028978	KIAA1055 protein	2.1	0.0496
32378_at	M26252	Pyruvate kinase, muscle	0.38	0.0013
38138_at	D38583	S100 calcium binding protein A11 (calgizzarin)	0.43	0.0041
1586_at	M35878	–	0.35	0.0046
40164_at	X69550	Rho GDP dissociation inhibitor (GDI) alpha	0.47	0.0063
36931_at	M95787	Transgelin	0.33	0.0068
41220_at	AB023208	MLL septin-like fusion	0.49	0.0069
36958_at	X95735	Zyxin	0.43	0.0105
41764_at	AA976838	Apolipoprotein C-I	0.40	0.0131
36937_s_at	U90878	PDZ and LIM domain 1 (elfin)	0.49	0.0132
37319_at	M35878	Insulin-like growth factor binding protein 3	0.30	0.0135
38021_at	U53204	Plectin	0.33	0.0170
39330_s_at	M95178	Actinin, alpha 1	0.40	0.0216
32755_at	X13839	Actin, alpha 2, smooth muscle, aorta	0.42	0.0234
35766_at	M26326	Keratin 18	0.46	0.0234
38796_at	X03084	Complement component 1, q subcomponent	0.48	0.0238
429_f_at	X00734	Tubulin, beta, 5	0.44	0.0286
33824_at	X74929	Keratin 8	0.38	0.0288
1451_s_at	U24576	LIM domain only 4	0.43	0.0324
36659_at	X05610	Collagen, type IV, alpha 2	0.25	0.0371
38124_at	X55110	Midkine (neurite growth-promoting factor 2)	0.28	0.0375
39145_at	J02854	Myosin, light polypeptide 9, regulatory	0.29	0.0399
1737_s_at	M62403	Insulin-like growth factor binding protein 4	0.34	0.0456

Table 4.3: Ratio predictors and corresponding accuracies for the original Gordon *et al* data set

Gene overexpressed in good	Gene overexpressed in poor	Training set accuracy, %
41531_at	32378_at	88.2
41755_at	1586_at	88.2
41755_at	40164_at	88.2
41531_at	40164_at	88.2
36204_at	40164_at	88.2
41755_at	38138_at	82.4
36204_at	38138_at	82.4
37405_at	1586_at	82.4
36204_at	1586_at	82.4
37405_at	40164_at	82.4
41531_at	38138_at	76.5
37405_at	32378_at	70.6
41531_at	1586_at	70.6
36204_at	32378_at	58.8
41755_at	32378_at	52.9
32378_at	38138_at	47.1

gene that was related to good prognosis expressed at levels above its mean and a gene related to poor prognosis expressed below its mean, but if the overall means of these two genes were very different, the sample could still have a ratio of less than one. The fact that this method works so well without data standardization may be a result of the seemingly *ad hoc* restrictions placed on the genes in the variable selection process.

Using only the five most accurate individual ratios, new predictors were formed by taking the geometric mean of every possible combination of three ratios. Again, the training samples were classified using these predictors such that a sample was predicted to have a good prognosis if the predictor value was greater than one. The ten predictors had an average accuracy of 94% and none had an accuracy of less than that of the best single ratio accuracy. Two of the resulting geometric mean predictors achieved 100% accuracy on the training samples. The list of ten predictors and their accuracies in predicting the training samples are given in Table 4.4; the first predictor

Table 4.4: Geometric mean predictors and corresponding accuracies for the original Gordon *et al* data set

Ratio 1		Ratio 2		Ratio 3		Training
Good gene	Poor gene	Good gene	Poor gene	Good gene	Poor gene	set accuracy, %
41531_at	32378_at	41755_at	40164_at	41531_at	40164_at	100.00
41531_at	32378_at	41755_at	1586_at	41755_at	40164_at	100.00
41531_at	32378_at	41755_at	40164_at	36204_at	40164_at	94.12
41531_at	32378_at	41531_at	40164_at	36204_at	40164_at	94.12
41755_at	1586_at	41755_at	40164_at	36204_at	40164_at	94.12
41755_at	1586_at	41531_at	40164_at	36204_at	40164_at	94.12
41755_at	40164_at	41531_at	40164_at	36204_at	40164_at	94.12
41531_at	32378_at	41755_at	1586_at	41531_at	40164_at	88.24
41531_at	32378_at	41755_at	1586_at	36204_at	40164_at	88.24
41755_at	1586_at	41755_at	40164_at	41531_at	40164_at	88.24

listed is the one selected by Gordon *et al* to test on the independent test set. Note that neither of two most accurate ratios predicted the remaining ten samples excluded from the training set – which did not include the four samples right-censored prior to the median – with an accuracy of greater than 27%, using the median survival of 11 months as the cutoff between poor and good groups. However, this accuracy was not a good measure of the ability of these ratios to generalize to new samples, since these remaining samples could not be considered randomly selected; they tended to have survivals clustered close to the mean and thus were likely to be the most difficult samples to predict.

All of these reported results were generated specifically for this paper and matched the results of Gordon *et al* exactly.

The N-fold cross validation error rate was then calculated for this type of model. That is, one training sample was excluded and the variables were selected using the Gordon method with the remaining 16 training samples. The four most statistically significantly overexpressed genes in each of the two classes were used to create 16 ratio predictors, the three most accurate of which were then used to form one geometric mean predictor; the resulting predictor was then used to predict the remaining sam-

ple. This process was repeated a total of 17 times, with each training sample being excluded once. In many cases, there were more than three equally accurate individual ratios; in this case, geometric mean predictors were created using all combinations of three ratios and the class of the excluded sample was determined by a majority vote of these predictors.

Gordon *et al* reported an 88% accuracy using this process; however, only a 65% accuracy was calculated in the recreation of their results. Gordon *et al* were somewhat vague in their description of the cross validation process, so it is possible that the discrepancy in the reported results was due to slight differences in exact methods used. This could have included, for example, how the excluded sample was predicted when more than three ratios are equally accurate. It is also possible that there were differences in the data set used in the current analysis from the data set used by Gordon *et al.* These errors could have come from a number of sources, such as transcription errors or the matrix being improperly sorted. However, since all the other values, including Student's t test p values and mean expression level ratios, were identical to those reported by Gordon *et al* and since the data set used in this paper was obtained directly from researchers at BWH, it is unlike that the discrepancy in the reported cross validation error rates was caused by data set errors.

However, another discrepancy arose when recreating other results of Gordon *et al.* Since the epithelial subtype is associated with relatively good patient outcome, Gordon *et al* performed a logrank test on two Kaplan-Meier curves containing epithelial samples and non-epithelial samples, respectively, to determine the statistical significance of the difference in survival between these two groups of patients. In recreating these results, the constructed Kaplan-Meier curves were different from those published in the Gordon *et al* paper. The two Kaplan-Meier curves constructed in the current analysis are shown in Figure 4-1. Of course, since the Kaplan-Meier curves were different, the logrank p values were different as well; Gordon *et al* reported a p value of .129, while the current analysis found a p value of .399.

While there may have been errors in the data set, this represented a mistake by Gordon *et al.* In their paper, they listed the set of training samples with patient and

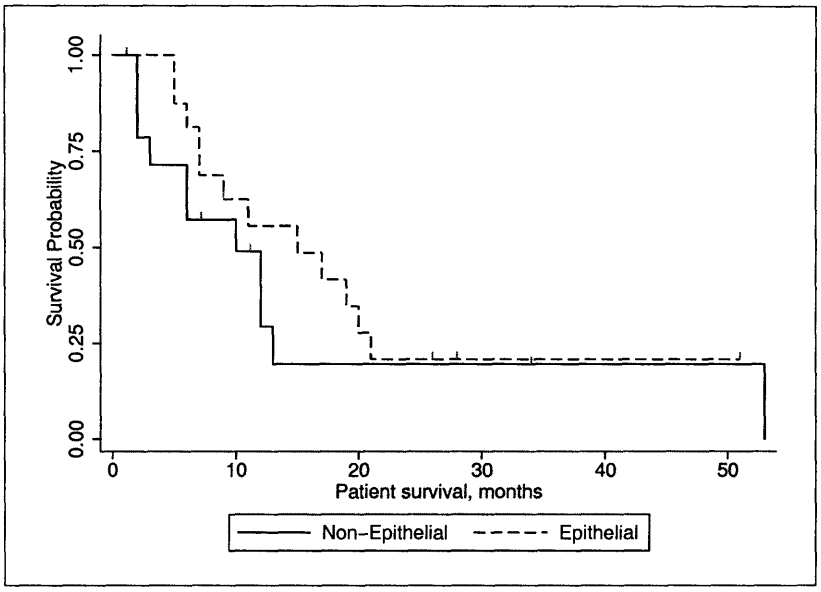

Figure 4-1: Kaplan-Meier survival curves based on tumor histology.

tumor information; this set included the patient with the longest known survival of 54 months – sample number 72 – who also happened to have a mixed subtype tumor. However, this patient was clearly included in the epithelial subtype Kaplan-Meier curve [15] when the patient should have been included in the non-epithelial subtype. However, this was not a serious mistake by any means and did not affect their end results.

4.1.1 Recreated results using the newer data set

After recreating and verifying the results of Gordon *et al* on the original data set, the same process was repeated on the newer data set, generated with a more recent version of Affymetrix Microarray Suite software. This newer data set contained the same samples and patient information; only the expression levels were different. The same 17 samples were used as a training set and the same method for selecting variables and building ratio-based predictors was used. In this case, only 29 genes were identified as differentially expressed; these genes are listed in Table 4.5. Of these 29 genes, 24 were also identified by the original Gordon *et al.* The four most statistically significantly overexpressed genes in each group were then used to form

16 expression level ratios; these ratios were then used to predict the classes of the training set samples. The ratios formed using the newer data set and their accuracies in predicting the training set are listed in Table 4.6. Of the eight genes used to build these ratios, only two genes – genes with probe identification numbers of 32378_at and 40164_at, respectively – overlapped with the eight genes from the original data set; since both of these genes were overexpressed in the poor group, none of the same ratios from Table 4.6 were included in the analysis on the original data set. Therefore, none of the predictors built using the newer data set were the same as the predictors built using the original data set.

With the newer data set, one ratio had an accuracy of 94%, better than any of the individual ratios using the original data set, and seven others predicted the training set with 88% accuracy. New predictors were built using the geometric means of the most accurate ratio and every possible combination of two of the seven next most accurate ratios. The resulting 21 predictors are listed in Table 4.7. Unlike the geometric means built using the original data set, none of these had a training set accuracy of 100%, although they did perform very well with an average accuracy of 89%. Even using all possible three ratio combinations did not produce a single predictor with 100% accuracy on the training set; these results are not presented. None of the five best predictors, which all had a training set accuracy of 94%, predicted the remaining samples – not including samples which were right-censored prior to the median – with greater than a 36% accuracy; again, however, this is not a good indication of the ability of any of these ratio-based predictors to generalize to new samples.

The same method for calculating the N-fold cross validation error rate was then applied to the newer data set. The N-fold cross validation accuracy was calculated as 65%, the same as in the original data set. Thus, the ratio-based method was equally accurate, in terms of cross validation, in both the original data set and the newer data set.

It would be very informative to be able test one or more of the most accurate predictors listed in Table 4.7 against the predictor used by Gordon *et al.* Unfortunately, an independent test set is unavailable. The results on the newer data set do seem

Table 4.5: Genes identified by Gordon method using original training set and new data set

Probe ID	Gene sequence No.	Description	Ratio of good to poor mean	Student's t test p value
37405_at	U29091	Selenium binding protein 1	2.8	0.0036
32424_at	D84424	Hyaluronan synthase 1	3.4	0.0080
41210_at	M81057	Carboxypeptidase B1 (tissue)	4.3	0.0209
892_at	M90657	Transmembrane 4 superfamily member 1	2.6	0.0314
376_at	AB000220	Sema domain, immunoglobulin domain	2.3	0.0339
40456_at	AL049963	BCG-induced gene in monocytes, clone 103	3.1	0.0375
41531_at	AI445461	Transmembrane 4 superfamily member 1	2.7	0.0405
37157_at	X56667	Calbindin 2, 29kDa (calretinin)	2.3	0.0406
33505_at	AI887421	Retinoic acid receptor responder	3.6	0.0434
34637_f_at	M12963	Alcohol dehydrogenase 1A	13.2	0.0485
32378_at	M26252	Pyruvate kinase, muscle	0.41	0.0001
40164_at	X69550	Rho GDP dissociation inhibitor	0.44	0.0015
36958_at	X95735	Zyxin	0.49	0.0030
36931_at	M95787	Transgelin	0.37	0.0044
39330_s_at	M95178	Actinin, alpha 1	0.40	0.0083
32755_at	X13839	Actin, alpha 2, smooth muscle, aorta	0.36	0.0127
39145_at	J02854	Myosin, light polypeptide 9, regulatory	0.38	0.0142
35766_at	M26326	Keratin 18	0.42	0.0149
38796_at	X03084	Complement component 1, q subcomponent	0.46	0.0162
37319_at	M35878	Insulin-like growth factor binding protein 3	0.31	0.0244
35905_s_at	U34995	Glyceraldehyde-3-phosphate dehydrogenase	0.46	0.0253
33824_at	X74929	Keratin 8	0.50	0.0278
1451_s_at	D13666	Osteoblast specific factor 2 (fasciclin I-like)	0.38	0.0308
36659_at	X05610	Collagen, type IV, alpha 2	0.25	0.0310
1664_at	HG3543-HT3739	–	0.30	0.0390
1737_s_at	M62403	Insulin-like growth factor binding protein 4	0.30	0.0409
769_s_at	D00017	Annexin A2	0.50	0.0439
38418_at	X59798	Cyclin D1	0.39	0.0448
36675_r_at	J03191	Profilin 1	0.49	0.0483

very comparable to the results on the original data set; for example, the ratio-based predictor list at the top of Table 4.7 uses two of the same genes as the ratio-based predictor used and tested by the Gordon *et al* study and only two new genes. N-fold cross validation gives an identical error rate between the two data sets, and, although none of the geometric mean predictors predicted the training set in the newer data set with 100% accuracy, the accuracies of the individual ratios and geometric mean predictors are very similar between the two data sets. In fact, the most accurate individual ratio in the newer data set was more accurate than the most accurate in-

Table 4.6: Ratio predictors and corresponding accuracies for the newer data set

Gene overexpressed in good	Gene overexpressed in poor	Training set accuracy, %
37405_at	40164_at	94.1
892_at	32378_at	88.2
32424_at	40164_at	88.2
892_at	40164_at	88.2
37405_at	36958_at	88.2
32424_at	36958_at	88.2
892_at	36958_at	88.2
37405_at	36931_at	88.2
32424_at	36931_at	82.4
37405_at	32378_at	76.5
41210_at	40164_at	76.5
41210_at	36958_at	76.5
892_at	36931_at	76.5
41210_at	32378_at	64.7
41210_at	36931_at	64.7
32424_at	32378_at	52.9

dividual ratio in the original data set. While Gordon *et al* may have found a good predictive model, ratio-based models from the newer data set may perform as well or better as the original model, since the data used to generate these newer models was created using a newer and presumably more precise method of calculating gene expression levels from microarray slides. Only an additional two genes – those with probe identification numbers of 37405_at and 892_at, respectively – would need to be analyzed using PCR in order to test one of the newer ratio-based predictors. Note that, although there is a chance that the probe identification numbers for various genes changed between the two data sets, the list of probe identification numbers and corresponding genes cited from the author's website [12] and the equivalent list obtained directly from researchers at BWH were equivalent for a number of randomly selected genes; thus, it is likely that the probe identification numbers did not change between the two data sets.

Table 4.7: Geometric mean predictors and corresponding accuracies for the newer data set

Ratio 1		Ratio 2		Ratio 3		Training
Good gene	Poor gene	Good gene	Poor gene	Good gene	Poor gene	set accuracy, %
37405_at	40164_at	892_at	32378_at	892_at	40164_at	94.1
37405_at	40164_at	892_at	32378_at	32424_at	36958_at	94.1
37405_at	40164_at	892_at	32378_at	892_at	36958_at	94.1
37405_at	40164_at	32424_at	40164_at	37405_at	36931_at	94.1
37405_at	40164_at	892_at	40164_at	37405_at	36958_at	94.1
37405_at	40164_at	892_at	32378_at	32424_at	40164_at	88.2
37405_at	40164_at	892_at	32378_at	37405_at	36958_at	88.2
37405_at	40164_at	892_at	32378_at	37405_at	36931_at	88.2
37405_at	40164_at	32424_at	40164_at	32424_at	36958_at	88.2
37405_at	40164_at	32424_at	40164_at	892_at	36958_at	88.2
37405_at	40164_at	892_at	40164_at	32424_at	36958_at	88.2
37405_at	40164_at	892_at	40164_at	892_at	36958_at	88.2
37405_at	40164_at	892_at	40164_at	37405_at	36931_at	88.2
37405_at	40164_at	37405_at	36958_at	32424_at	36958_at	88.2
37405_at	40164_at	37405_at	36958_at	892_at	36958_at	88.2
37405_at	40164_at	37405_at	36958_at	37405_at	36931_at	88.2
37405_at	40164_at	32424_at	36958_at	892_at	36958_at	88.2
37405_at	40164_at	32424_at	36958_at	37405_at	36931_at	88.2
37405_at	40164_at	892_at	36958_at	37405_at	36931_at	88.2
37405_at	40164_at	32424_at	40164_at	892_at	40164_at	82.4
37405_at	40164_at	32424_at	40164_at	37405_at	36958_at	82.4

4.2 Results of additional analysis

Additional analyses were performed to identify other differentially expressed genes and build predictive models for survival in patients with malignant pleural mesothelioma. The absence of an independent test set dictated that the data set be split into training and test sets in such a way as to use as much as possible of the available data. Thus, a training set with 21 patients was randomly generated from the suitable samples; eleven of these samples had a good outcome – that is, a survival of 11 of more months – and ten had a poor outcome. Seven of the remaining samples were used as a test set and the final three samples, which were right-censored prior to the median survival, were left to be used in logrank tests. The training set, test set, and

set to be used in logrank tests are shown in Table 4.8; these sets were used in all of the additional analyses. All of the additional analyses used expression levels from the newer data set.

4.2.1 Results of variable selection

The top 50 genes, 25 of which were overexpressed in each of the two outcome classes, as identified by the Golub method of variable selection are listed in Table 4.9. Interestingly, none of the these 50 genes were the same as any of the 46 genes identified by the original Gordon *et al* study or any of the 29 genes identified by recreating their results using the newer data set.

Applying the Gordon method of variable selection to the new training set identified a total of nine genes; these are listed in Table 4.10. Two of these genes were also identified by the original Gordon *et al* study, although neither of these genes was among the eight used to build the ratio-based predictors. Likewise, one of the genes listed in Table 4.10 was also identified by the recreated Gordon *et al* results on the newer data set but again was not among the eight used to build the predictive models. Lastly, none of these nine genes were among the top 50 genes as identified by the Golub method.

It appears that the Gordon method of variable selection is not well suited for use on randomly selected training sets. When applied to the new training set, the method identified only nine genes and only two of these genes were overexpressed in the poor outcome group. This method of variable selection seems to work best when only the best and worst samples are used for training; for many of the genes, the samples with survivals close to the median must pull the mean of the expression levels in each group closer to the overall mean, thus causing the method to identify fewer genes.

4.2.2 Cross validation results from machine learning models

The N-fold cross validation error rates for classification trees are shown in Table 4.11. Notice that, for the Golub set containing 4, 10, and 30 genes, both with and without

Table 4.8: Samples used in training and test sets for additional analysis

Sample No.	Patient age at diagnosis, yr	Sex	Tumor histology	Tumor stage	Patient survival, mo	Patient status[a]
			Training set			
114	51	M	mixed	2	2	3
159	62	M	sarc	2	2	3
89	55	M	mixed	2	3	3
229	33	F	ept	2	5	3
6	39	M	ept	2	5	3
130	55	M	mixed	2	6	3
166	66	M	sarc	2	6	3
67	49	F	ept	2	6	3
167	53	M	ept	2	7	3
86	42	F	ept	2	9	3
213	55	M	mixed	2	11	1[1]
101	71	F	ept	2	11	3
51	49	M	mixed	2	13	3
93	52	M	ept	2	15	3
76	67	M	ept	1	17	3
109	62	M	ept	2	19	3
33	60	F	ept	2	20	3
2	44	F	ept	2	26	2[1]
90	48	M	ept	2	28	2[1]
74	40	F	ept	1	51	2[1]
72	46	M	mixed	2	53	3
			Test set			
133	69	M	mixed	2	2	3
57	61	M	ept	2	7	3
42	64	M	mixed	1	10	3
105	66	M	mixed	2	12	3
212	62	F	mixed	2	12	3
68	61	M	ept	2	21	3
34	52	F	mixed	2	34	4[1]
			Right-censored samples used for logrank test			
82	68	F	mixed	2	1	4[1]
118	74	M	mixed	2	7	4[1]
116	70	M	ept	2	9	U[1]

[a]For patient status, 1 = alive without mesothelioma, 2 = alive with mesothelioma, 3 = dead from mesothelioma, 4 = dead from other causes and U = unknown [15].

[a]Right-censored samples are excluded from regression analysis.

Table 4.9: Top 50 genes identified by Golub method using new training set

Probe ID	Gene Seq. No.	Description	SNR
32629_f_at	U90552	Butyrophilin, subfamily 3, member A1	1.042596
33428_s_at	AF034957	Attractin	1.035815
37973_at	AB018256	Sorting nexin 13	0.875127
41174_at	AF012086	Similar to RAN-binding protein 2	0.841515
39414_at	L23849	—	0.839765
31315_at	D84143	Homo sapiens immunoglobulin lambda light chain	0.83609
304_at	HG961-HT961	–	0.823303
40060_r_at	AF061258	LIM protein	0.792319
35475_at	U37251	Zinc finger protein 177	0.768456
36072_at	AF025770	Zinc finger protein 189	0.76337
31413_at	AF000990	Testis-specific transcript, Y-linked 1	0.758292
37288_g_at	U55258	Neuronal cell adhesion molecule	0.756127
40743_at	M60092	Adenosine monophosphate deaminase 1	0.753599
39815_at	AA883101	Secreted protein of unknown function	0.739854
37142_at	AF038421	GDNF family receptor alpha 1	0.731596
636_at	L43338	—	0.720407
39440_f_at	AA962207	—	0.717747
40606_at	U88629	ELL-related RNA polymerase II, elongation factor	0.716377
39980_at	AB000449	Vaccinia related kinase 1	0.710041
1895_at	J04111	—	0.698625
40292_at	AF027734	Deleted in bladder cancer chromosome region candidate 1	0.694958
35783_at	H93123	Vesicle-associated membrane protein 3	0.693607
35549_at	L05096	Ribosomal protein L39-like	0.692348
35071_s_at	AF042377	GDP-mannose 4,6-dehydratase	0.691798
36501_at	S87759	Protein phosphatase 1A	0.690801
40489_at	D31840	Dentatorubral-pallidoluysian atrophy (atrophin-1)	-1.08847
35151_at	AF089814	Tumor suppressor deleted in oral cancer-related 1	-1.0085
38729_at	M88279	FK506 binding protein 4, 59kDa	-0.96146
33285_i_at	W26762	Hypothetical protein FLJ21168	-0.86635
34541_at	L02867	Paraneoplastic antigen	-0.86036
37386_i_at	X55885	KDEL (Lys-Asp-Glu-Leu) endoplasmic reticulum protein	-0.81889
1797_at	U40343	Cyclin-dependent kinase inhibitor 2D	-0.79297
38429_at	U29344	Fatty acid synthase	-0.79098
2047_s_at	M23410	Junction plakoglobin	-0.78672
32181_at	M60922	Flotillin 2	-0.78316
33909_at	L35013	Splicing factor 3b, subunit 4, 49kDa	-0.78009
1796_s_at	U05681	—	-0.7781
38647_at	AJ131182	Coatomer protein complex, subunit epsilon	-0.76184
207_at	M86752	Stress-induced-phosphoprotein	-0.75549
419_at	X65550	Antigen identified by monoclonal antibody Ki-67	-0.74995
1801_at	U76638	BRCA1 associated RING domain 1	-0.74814
38269_at	AL050147	Protein kinase D2	-0.74517
38828_s_at	AA628946	KH-type splicing regulatory protein	-0.7272
37365_at	X63368	DnaJ (Hsp40) homolog, subfamily B, member 2	-0.7266
40195_at	X14850	H2A histone family, member X	-0.72482
40619_at	M91670	Ubiquitin carrier protein	-0.7223
33014_at	AF059194	V-maf musculoaponeurotic fibrosarcoma oncogene	-0.72116
39704_s_at	L17131	High mobility group AT-hook 1	-0.71493
1486_at	L37127	Polymerase (RNA) II (DNA directed) polypeptide	-0.71294
40580_r_at	M24398	Parathymosin	-0.71203

Table 4.10: Genes identified by Gordon method using new training set

Probe ID	Gene sequence No.	Description	Ratio of good to poor mean	Student's t test p value
33273_f_at	X57809	Immunoglobulin lambda locus	3.3	0.0163
33274_f_at	M18645	Immunoglobulin lambda joining 3	3.1	0.0210
41827_f_at	AI932613	Hypothetical protein LOC51233	2.7	0.0212
33499_s_at	AF067420	Hypothetical protein MGC27165	2.9	0.0227
33501_r_at	S71043	Partial mRNA for immunoglobulin	2.8	0.0232
286_at	L19779	Histone 2, H2aa	2.2	0.0306
32609_at	AI885852	Histone 2, H2aa	2.4	0.0478
35766_at	M26326	Keratin 18	0.47	0.0068
41294_at	AJ238246	Keratin 7	0.21	0.0332

additional patient variables, the resulting classification trees performed the same. The N-fold cross validation error rate increases, however, when the 50 best Golub method genes were used; presumably, the tree building process began to overfit the training data. Notice also that the Gordon method performed very poorly, even worse than simply using the 500 genes with the lowest Student's t test p value.

Furthermore, the additional patient variables – such as age, sex, tumor stage and tumor histology – had little effect on the cross validation error rate, since the error rate was the same for each gene set regardless of whether additional patient variables were included. This is not surprising because, if none of these variables generated good splits, they were simply not included in the classification tree model.

Among all of the input variable sets that gave a cross validation error rate of 9.5%, the input variable set using the best four genes as identified by the Golub method and no additional patient variables was selected to be applied to the test set because it contained the fewest number of input variables.

The cross validation error rates for the neural networks applied to the classification problem are given in Table 4.12. Notice that all of the eleven networks which achieved a mean error rate of zero used regularization, with a coefficient of either 2.5 or .5. All of the top 16 networks either used regularization or had a smaller size to avoid overfitting. In fact, for all input variables sets with the exception of the Gordon gene set, networks with regularization performed better than those with a small size,

Table 4.11: N-fold cross validation results for classification trees

Gene set	Additional patient variables	CV error rate, %	CV error St. dev.
Golub, top 4	No	9.5	6.6
Golub, top 4	Yes	9.5	6.6
Golub, top 10	No	9.5	6.6
Golub, top 10	Yes	9.5	6.6
Golub, top 30	No	9.5	6.6
Golub, top 30	Yes	9.5	6.6
Golub, top 50	No	28.6	10.1
Golub, top 50	Yes	28.6	10.1
Lowest 500 p value	No	28.6	10.1
Lowest 500 p value	Yes	28.6	10.1
Gordon	No	47.6	11.2
Gordon	Yes	47.6	11.2

and both of these types of network structures performed better than the networks with large size and no regularization. It seems that the large network size with no regularization caused the training to overfit the training data.

Both the heavier regularization with a coefficient of 2.5 and the lighter regularization with a coefficient of .5 performed reasonably well, achieving identical results with seven of the ten input variable sets; only in the Golub set with ten genes with additional variables and in both Gordon input variable sets did the heavier regularization perform better than the lighter regularization.

In the case of neural networks, the additional patient variables seemed to increase the cross validation error rates. Except for those built using the Gordon gene set, the networks using the gene set without the additional patient variables performed as well or better than the networks using the same gene set with the additional variables. However, networks using regularization performed equally with or without the additional patient variables, except for those using the Gordon set and the network using the best ten Golub genes with lighter regularization. Apparently, the regularization was enough to counteract the confounding effects of the additional variables.

As was the case with classification trees, the networks using the Gordon gene set

Table 4.12: N-fold cross validation results for neural networks used for classification

Gene set	Additional patient variables	Parameters Hidden size	Parameters Reg. coefficient	CV error rate, % Mean	CV error rate, % St. dev.	CV error rate, % 95% CI
Golub, top 10	No	8	2.5	0.0	0.0	[0.0, 0.0]
Golub, top 10	No	8	0.5	0.0	0.0	[0.0, 0.0]
Golub, top 10	Yes	8	2.5	0.0	0.0	[0.0, 0.0]
Golub, top 30	No	8	2.5	0.0	0.0	[0.0, 0.0]
Golub, top 30	No	8	0.5	0.0	0.0	[0.0, 0.0]
Golub, top 30	Yes	8	2.5	0.0	0.0	[0.0, 0.0]
Golub, top 30	Yes	8	0.5	0.0	0.0	[0.0, 0.0]
Golub, top 50	No	8	2.5	0.0	0.0	[0.0, 0.0]
Golub, top 50	No	8	0.5	0.0	0.0	[0.0, 0.0]
Golub, top 50	Yes	8	2.5	0.0	0.0	[0.0, 0.0]
Golub, top 50	Yes	8	0.5	0.0	0.0	[0.0, 0.0]
Golub, top 10	No	2	0	1.9	4.3	[0.0, 5.6]
Golub, top 10	Yes	8	0.5	1.9	4.3	[0.0, 5.6]
Golub, top 10	Yes	2	0	2.9	2.6	[0.6, 5.1]
Golub, top 30	No	2	0	3.8	2.1	[1.9, 5.7]
Golub, top 50	No	2	0	3.8	4.0	[0.3, 7.3]
Golub, top 10	No	8	0	4.8	4.8	[0.6, 8.9]
Golub, top 30	No	8	0	4.8	5.8	[0.0, 9.9]
Golub, top 10	Yes	8	0	7.6	6.4	[2.0, 13.2]
Golub, top 50	No	8	0	7.6	6.4	[2.0, 13.2]
Golub, top 30	Yes	2	0	8.6	4.0	[5.1, 12.1]
Golub, top 50	Yes	2	0	8.6	2.1	[6.7, 10.4]
Golub, top 4	No	2	0	9.5	0.0	[9.5, 9.5]
Golub, top 4	No	8	2.5	9.5	0.0	[9.5, 9.5]
Golub, top 4	No	8	0.5	9.5	0.0	[9.5, 9.5]
Golub, top 4	No	8	0	9.5	0.0	[9.5, 9.5]
Golub, top 30	Yes	8	0	9.5	8.2	[2.3, 16.8]
Golub, top 50	Yes	8	0	9.5	6.7	[3.6, 15.4]
Gordon	Yes	8	2.5	11.4	2.6	[9.1, 13.7]
Golub, top 4	Yes	8	2.5	14.3	0.0	[14.3, 14.3]
Golub, top 4	Yes	8	0.5	14.3	0.0	[14.3, 14.3]
Gordon	Yes	8	0.5	16.2	2.6	[13.9, 18.5]
Golub, top 4	Yes	2	0	19.0	4.8	[14.9, 23.2]
Golub, top 4	Yes	8	0	21.9	2.6	[19.6, 24.2]
Gordon	Yes	2	0	21.9	9.9	[13.2, 30.6]
Gordon	No	8	0	22.9	4.0	[19.4, 26.3]
Gordon	No	2	0	23.8	0.0	[23.8, 23.8]
Gordon	No	8	2.5	23.8	0.0	[23.8, 23.8]
Gordon	Yes	8	0	25.7	4.3	[22.0, 29.4]
Gordon	No	8	0.5	26.7	4.3	[22.9, 30.4]

performed poorly compared to those using any of the Golub gene sets. Networks using the Gordon gene set made up the six worst networks in terms of mean cross validation error rate, and all eight networks using the Gordon set were among the worst twelve networks. Networks using the gene set with the best four Golub genes performed uniformly poorly as well. With the Gordon gene set, using the additional patient variables improved network performance, except in the large network with no regularization. It appears that the genes included in the Gordon set generalized so poorly that including the additional patient variables – which proved to degrade performance in many of the other networks – actually improved the performance of networks.

It also appears that regularization was able to counteract the effects of overfitting and high dimensionality when the number of input variables was increased. Looking at the Golub gene sets without additional patient variables, the cross validation error for both large networks with no regularization and small networks reached a minimum with ten genes and gradually increased as more genes were added. With regularization, however, the error reached zero with ten genes and stayed constant as the number of genes increased. The same was generally true for the Golub gene sets with additional variables; for large networks with no regularization and for small networks, the mean cross validation error reached a minimum at ten genes and increased as more genes were added. With regularization, however, once the minimum error was reached, the error stayed constant as more genes were added; with the smaller regularization this minimum was reached with 30 genes and with heavy regularization this minimum was reached with ten genes.

If two or more networks had equal mean cross validation error rates, the one with smallest number of inputs, the heaviest regularization, and finally the smallest size was preferred. Therefore, although a total of eleven of combinations of input variable sets and network structures achieved a zero mean cross validation error rate, the network using the best ten Golub genes with no additional variables and having a large size with a large regularization coefficient was chosen to be applied to the test set. In order to reduce the variance of the output due to the random starting weights,

the final model contained a total of five neural networks with identical structure; the final output for each sample was given by the average of the outputs of the individual networks.

The cross validation error rates for the various logistic regression models are given in Table 4.13. Notice in logistic regression that including the additional patient variables did not necessarily lead to degraded performance. For models using forward and backward stepwise variable selection, this is not surprising; if a particular variable was unimportant to the classification problem, it was not included in the final model. Furthermore, Stata software automatically removed many variables due to colinearity, so, even when stepwise variable selection was not used, many of the least important variables were removed before the logistic regression models were built. Notice also that models using the Gordon gene set performed very poorly; the best model using the Gordon gene set had a higher cross validation error than the worst model using any of the Golub gene sets.

It appears that models built with stepwise variable selection, either forward or backward, did not perform consistently better or worse than models built without stepwise selection. For the Golub set with four genes, backward stepwise selection performed the better than any other model but forward stepwise performed no better than simple logistic regression. For the Golub set with ten genes, forward and backward stepwise selection performed worse than the simple models and, for the Gordon gene set, forward and backward selection performed better than simple logistic regression with additional variables but worse than simple logistic regression without these variables. It would be interesting to see the results of stepwise selection on the larger gene sets, perhaps even one containing hundreds of genes, but unfortunately there were too few training samples for Stata software to perform stepwise analysis on these larger input variable sets. This is limitation is specific to Stata software, since many other programs such as SAS are able to perform stepwise selection with large number of input variables.

Although two models had a cross validation error rate of zero, the Golub set with four genes and no additional variables was applied to the test set because it contained

Table 4.13: N-fold cross validation results for logistic regression models

Gene set	Additional patient variables	Stepwise variable selection	CV error rate, %
Golub, top 4	No	Backward	0.0
Golub, top 4	Yes	Backward	0.0
Golub, top 50	Yes	None	9.5
Golub, top 4	No	None	14.3
Golub, top 4	No	Forward	14.3
Golub, top 4	Yes	Forward	14.3
Golub, top 10	No	None	14.3
Golub, top 10	Yes	None	14.3
Golub, top 30	Yes	None	14.3
Golub, top 50	No	None	14.3
Golub, top 4	Yes	None	19.0
Golub, top 10	No	Forward	19.0
Golub, top 10	Yes	Backward	19.0
Golub, top 10	Yes	Forward	19.0
Golub, top 10	No	Backward	23.8
Golub, top 30	No	None	23.8
Gordon	No	None	28.6
Gordon	Yes	Backward	33.3
Gordon	Yes	Forward	33.3
Gordon	No	Backward	38.1
Gordon	No	Forward	38.1
Gordon	Yes	None	38.1

fewer input variables than the other model with zero cross validation error.

The estimated cross validation error rates for support vector machine models are given in Table 4.14. A curious result of the support vector machines was that the estimates of cross validation error rates were the same for each input variable set regardless of the kernel function. This was most likely caused by the fact that the SVM^{light} software only used a select few samples to produce its estimates of the error rate; while this may be a reasonable approach for large training sets, it is not accurate with a small training set like the one used here. Unfortunately, this property could not be changed within the software. The software also produced another estimate, called the *XiAlpha* estimate, of the model error; this estimate was also constant for

each input variable set regardless of kernel function. The fact that the error was constant for each input variable set may have been caused by the regularization coefficient, which was set automatically by the software based on the kernel function. This regularization may have effectively regularized each kernel to the point that each gave comparable results. The actual output was different for each kernel, however. Varying this coefficient did not change the results in any noticeable way.

Because the cross validation and XiAlpha estimates for the error were constant for each input variable set, the results from support vector machines were, for the most part, uninteresting. Using a Golub set with ten or more genes and any kernel function gave a cross validation error estimate of zero and, except for the Golub set with ten genes and additional patient variables, these all gave the same XiAlpha error estimates. Notice again that the Gordon gene set performed worse than all other gene sets with or without additional patient variables.

All other things being equal, the model using the smallest input variable set or simplest kernel function was preferred. Thus, the model using a linear kernel function and the Golub set with ten genes and no additional patient variables was chosen over the other models with cross validation error estimates of zero to be applied to the test set.

Regression results

Results for the regression analysis using neural networks are given in Table 4.15. The error is reported as the average sum of squares error for each of the N-fold cross validation samples. Notice that five of the top six models used a small network size to avoid overfitting. In fact, for five of the ten input variable sets, the network with a small size performed significantly better than any of the other network structures. Networks with the larger regularization coefficient performed better than networks with the smaller regularization coefficient in all but one case and this difference was significant in all but one of the case; in the one case where the smaller coefficient performed better – the Golub set with 50 genes and no additional variables – the difference was negligible.

Table 4.14: Cross validation error estimates for support vector machine models

Gene set	Additional patient variables	Kernel function	CV error estimate, %	XiAlpha error estimate, %
Golub, top 10	No	Linear	0.0	28.6
Golub, top 10	No	Polynomial, degree 2	0.0	28.6
Golub, top 10	No	Polynomial, degree 4	0.0	28.6
Golub, top 10	No	Radial basis	0.0	28.6
Golub, top 10	Yes	Linear	0.0	38.1
Golub, top 10	Yes	Polynomial, degree 2	0.0	38.1
Golub, top 10	Yes	Polynomial, degree 4	0.0	38.1
Golub, top 10	Yes	Radial basis	0.0	38.1
Golub, top 30	No	Linear	0.0	28.6
Golub, top 30	No	Polynomial, degree 2	0.0	28.6
Golub, top 30	No	Polynomial, degree 4	0.0	28.6
Golub, top 30	No	Radial basis	0.0	28.6
Bolub, top 50	No	Linear	0.0	28.6
Bolub, top 50	No	Polynomial, degree 2	0.0	28.6
Bolub, top 50	No	Polynomial, degree 4	0.0	28.6
Bolub, top 50	No	Radial basis	0.0	28.6
Bolub, top 50	Yes	Linear	0.0	28.6
Bolub, top 50	Yes	Polynomial, degree 2	0.0	28.6
Bolub, top 50	Yes	Polynomial, degree 4	0.0	28.6
Bolub, top 50	Yes	Radial basis	0.0	28.6
Golub, top 30	Yes	Linear	0.0	28.6
Golub, top 30	Yes	Polynomial, degree 2	0.0	28.6
Golub, top 30	Yes	Polynomial, degree 4	0.0	28.6
Golub, top 30	Yes	Radial basis	0.0	28.6
Golub, top 4	Yes	Linear	9.5	42.9
Golub, top 4	Yes	Polynomial, degree 2	9.5	42.9
Golub, top 4	Yes	Polynomial, degree 4	9.5	42.9
Golub, top 4	Yes	Radial basis	9.5	42.9
Golub, top 4	No	Linear	14.3	28.6
Golub, top 4	No	Polynomial, degree 2	14.3	28.6
Golub, top 4	No	Polynomial, degree 4	14.3	28.6
Golub, top 4	No	Radial basis	14.3	28.6
Gordon	Yes	Linear	19.1	42.9
Gordon	Yes	Polynomial, degree 2	19.1	42.9
Gordon	Yes	Polynomial, degree 4	19.1	42.9
Gordon	Yes	Radial basis	19.1	42.9
Gordon	No	Linear	23.8	52.4
Gordon	No	Polynomial, degree 2	23.8	52.4
Gordon	No	Polynomial, degree 4	23.8	52.4
Gordon	No	Radial basis	23.8	52.4

Table 4.15: N-fold cross validation results for neural networks used for regression

Gene set	Additional patient variables	Parameters Hidden size	Reg. coefficient	CV average sum of squares error Mean	St. dev.	95% CI
Golub, top 50	Yes	2	0	56.5	5.2	[52.0, 61.1]
Golub, top 50	No	2	0	58.3	4.0	[54.8, 61.8]
Golub, top 10	No	2	0	62.6	9.2	[54.6, 70.7]
Golub, top 30	No	8	2.5	65.9	6.3	[60.4, 71.4]
Golub, top 30	Yes	2	0	66.2	4.2	[62.5, 69.9]
Golub, top 30	No	2	0	67.5	11.1	[57.8, 77.2]
Golub, top 50	No	8	0.5	70.0	7.1	[63.8, 76.1]
Golub, top 50	No	8	2.5	70.9	9.7	[62.3, 79.4]
Golub, top 50	Yes	8	2.5	71.8	3.9	[68.4, 75.2]
Golub, top 30	Yes	8	2.5	72.8	3.7	[69.6, 76.1]
Golub, top 30	No	8	0	74.0	17.3	[58.9, 89.2]
Golub, top 50	Yes	8	0.5	76.2	14.0	[63.9, 88.5]
Golub, top 10	No	8	2.5	80.5	10.9	[71.0, 90.0]
Golub, top 10	Yes	2	0	82.0	22.1	[62.6, 101.4]
Golub, top 30	No	8	0.5	84.0	15.5	[70.4, 97.6]
Gordon	Yes	8	2.5	86.1	10.0	[77.3, 94.9]
Golub, top 10	No	8	0	87.8	19.7	[70.5, 105.1]
Golub, top 10	No	8	0.5	88.0	7.8	[81.2, 94.8]
Golub, top 30	Yes	8	0.5	100.6	29.3	[75.0, 126.3]
Golub, top 50	No	8	0	101.7	30.6	[74.9, 128.6]
Golub, top 10	Yes	8	2.5	106.0	6.7	[100.2, 111.9]
Gordon	Yes	2	0	106.4	18.0	[90.6, 122.2]
Golub, top 50	Yes	8	0	107.9	9.8	[99.3, 116.5]
Golub, top 4	Yes	8	2.5	108.9	5.5	[104.1, 113.6]
Gordon	No	8	2.5	114.5	14.5	[101.8, 127.2]
Golub, top 30	Yes	8	0	115.0	19.4	[98.0, 132.0]
Golub, top 10	Yes	8	0.5	119.4	6.7	[113.5, 125.3]
Gordon	No	8	0.5	127.6	15.3	[114.2, 141.0]
Golub, top 10	Yes	8	0	130.3	26.5	[107.0, 153.5]
Golub, top 4	Yes	2	0	130.4	30.7	[103.5, 157.4]
Golub, top 4	No	8	2.5	132.0	4.6	[128.0, 136.0]
Gordon	No	2	0	139.2	39.7	[104.4, 174.0]
Gordon	Yes	8	0.5	147.3	31.0	[120.1, 174.5]
Golub, top 4	Yes	8	0.5	155.0	29.8	[128.9, 181.1]
Golub, top 4	No	8	0.5	195.7	15.0	[182.5, 208.9]
Gordon	No	8	0	201.6	54.4	[153.9, 249.2]
Golub, top 4	No	2	0	202.9	48.1	[160.7, 245.1]
Golub, top 4	No	8	0	224.7	39.1	[190.4, 259.0]
Golub, top 4	Yes	8	0	262.9	79.1	[193.5, 332.3]
Gordon	Yes	8	0	307.0	122.4	[199.7, 414.3]

A networks with a large size and no regularization only performed better than another type of network structure in one case, the Golub set with ten genes and no additional variables. Thus, it appears that this network structure was very sensitive to overfitting, as was the case in the networks built for the classification problem. Again, the Gordon gene set generated very poorly performing models; the worst model used the Gordon gene set and none of the Gordon models were among the best 15 models. Additional variables did not seem to be very helpful in the regression problem either. In the 20 comparable networks between networks using input variable set with additional variables and those not using additional input variables, the network using the additional variables performed better in only six of these cases, and this difference was significant in only two cases. On the other hand, in twelve of these cases, the network without the additional variables performed significantly better.

As can be seen from Table 4.15, the best network in terms of mean cross validation error performs significantly better than all but the second best network. Thus, the network using the Golub set with 50 genes and additional variables and having a small size was applied to the test set. The final model contained a total of five networks with identical structure; the final output was given by the average of the individual networks.

Classification results using the test set

A total of four models were applied to the test set; the results of these model are given in Table 4.16. The sensitivity, specificity, and accuracy were calculated using a total of seven samples; the logrank and Cox proportional hazard regression tests were calculated using these seven samples plus an additional three right-censored samples.

The final classification tree model uses only two genes, 40489_at and 33428_at, and, in the final logistic regression model, backward stepwise variable selection eliminates all but a single variable, the gene with probe identification 35151_at.

The neural network and support vector machine models performed the best in terms of both combined sensitivity and specificity and ROC area. The logistic regression model results suggest that it was a completely uninformative test since all

Table 4.16: Results of the best of each type of model when applied to the test set

Model type	Gene set	Add. var.	Training set performance Sens.	Spec.	Acc.	ROC area[a]	Logrank p value	Cox p value
Neural network, 8 hidden neurons, reg coeff of 2.5	Golub, top 10	No	1.00	0.33	0.71	0.67	0.219	0.265
Logistic regression, Backward stepwise selection	Golub, top 4	No	1.00	0.00	0.57	0.50	0.432	–[b]
SVM, Linear kernel function	Golub, top 10	No	1.00	0.33	0.71	0.67	0.515	0.529
Classification tree	Golub, top 4	No	0.50	0.33	0.43	0.42	0.235	0.296

[a]ROC area is calculated using the single point (1 - specificity, sensitivity) so that calculations will not favor models that support a range of cutoff values over models with only one possible cutoff.
[b]Test did not converge.

test samples were classified as good outcome, thus giving the test an ROC area of .5. The classification tree model performed even worse than the logistic regression model, achieving an ROC area of .42; thus, the ROC curve actually falls below the 45° line.

The logrank and Cox proportional hazard regression p values used more samples than the sensitivity and specificity calculations, and so may be more reliable measures of model performance. Although the support vector machine model performed well in terms of ROC area, it performed the worst in terms of logrank and Cox hazard regression p values. The logistic regression model also performed poorly in terms of logrank p value – note that the Cox proportional hazard regression p value is not given for the logistic regression model because the calculations failed to converge in Stata software. The classification tree model, despite performing poorly in terms of ROC area, had lower p values than both the logistic regression and support vector machine models. The neural network model, however, performed the best in terms of both ROC area and logrank and Cox hazard regression p values. The Kaplan-Meier survival curves used to perform the logrank tests are shown in Figures 4-2 through 4-5.

Note that none of these four models produced a split with a p value of less than .05; thus, none produced a statistically significant split in terms of patient survival between the two predicted classes. However, since so few samples were used to generate these

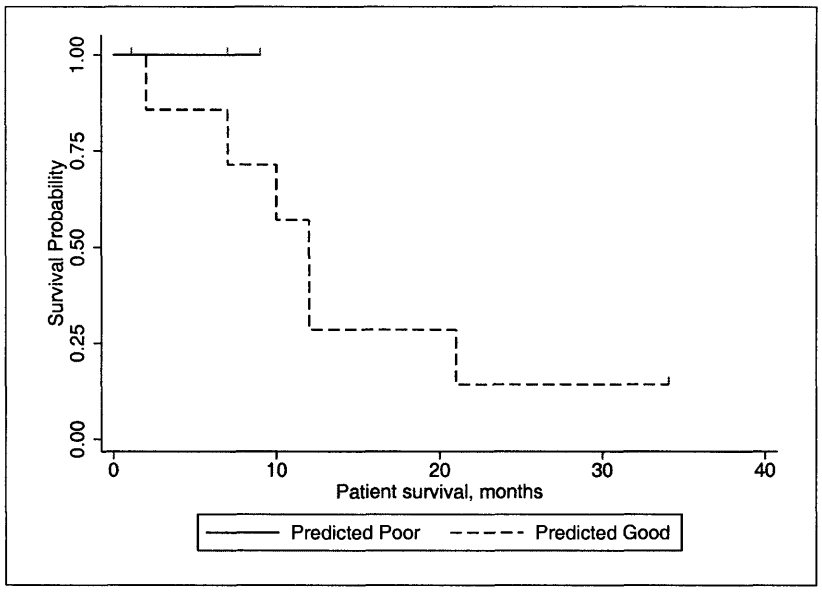

Figure 4-2: Kaplan-Meier survival curves based on the logistic regression model predictions

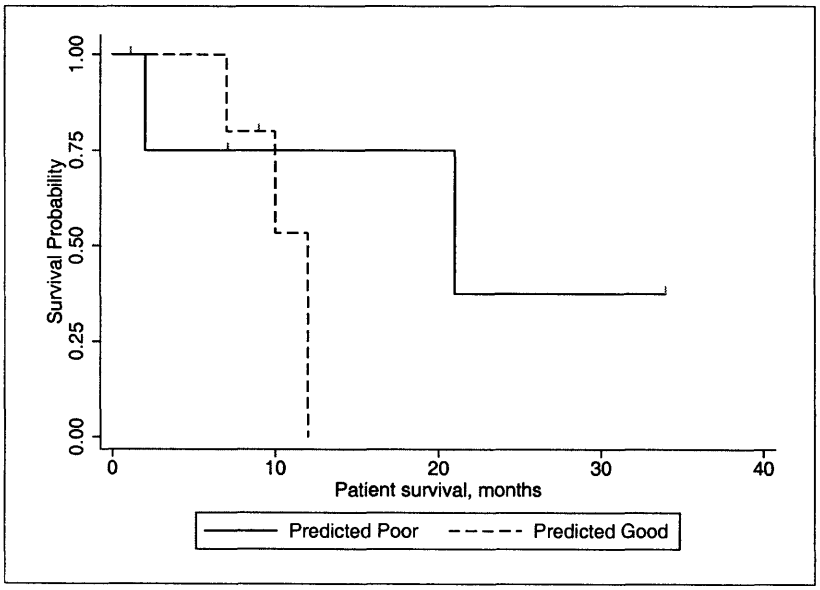

Figure 4-3: Kaplan-Meier survival curves based on the classification tree model predictions

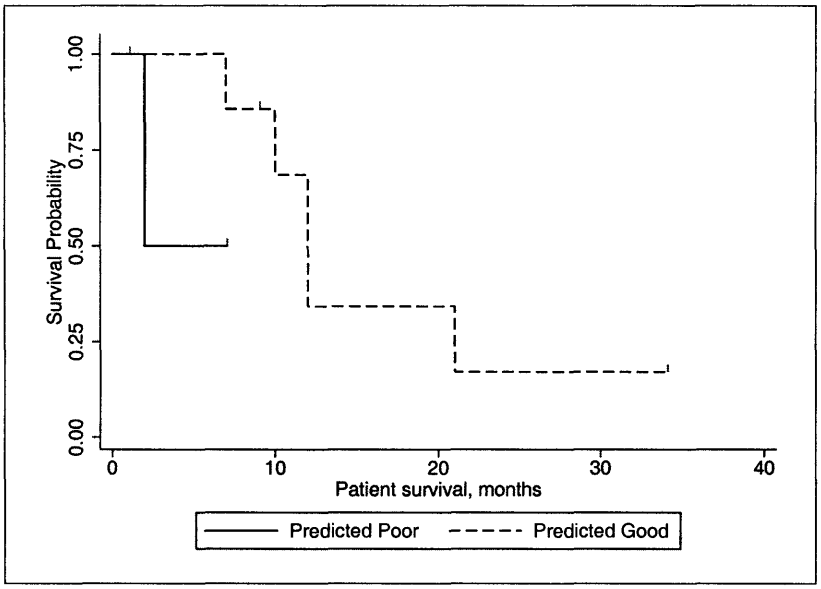

Figure 4-4: Kaplan-Meier survival curves based on the neural network model predictions

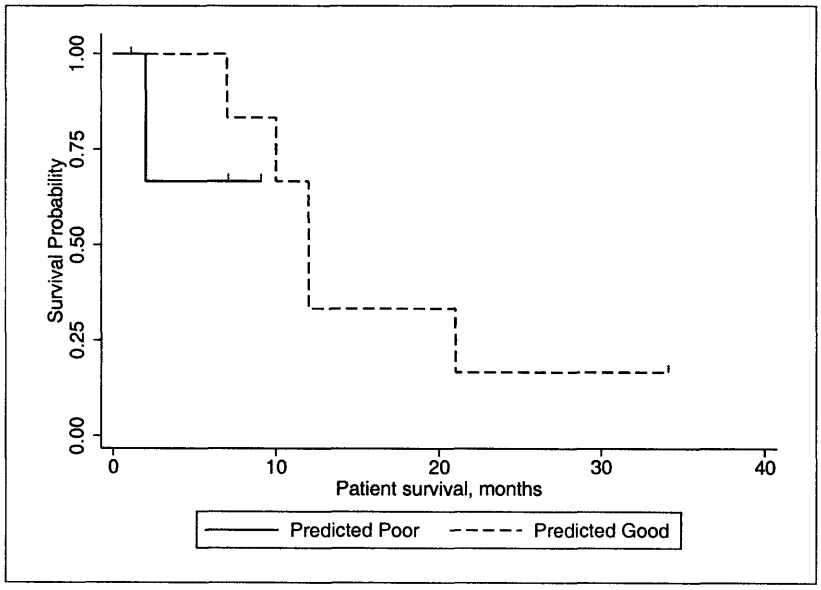

Figure 4-5: Kaplan-Meier survival curves based on the support vector machine model predictions

Table 4.17: Regression results of the best neural network applied to the test set

Model type	Gene set	Additional patient variables	Average training set sum of squares error	Average CV sum of squares error	Average test set sum of squares error	Average test set absolute error
Neural network, 2 hidden neurons, no reg	Golub, top 50	Yes	35.3	56.5	16.0	4.6

values, it is not surprising that these p values were not small, even if the models were very good. In general, since the test set was extremely small, these results have a very high variance. These results are not meant to provide conclusive assessments of the four final models; rather they are given simply to provide some basic evaluation and comparison between the four models.

Neural network regression results using the test set

The results of the final neural network regression model when applied to the test set are given in Table 4.17. The average test error is given both in terms of average sum of squared error, which is the typical error measure, and in terms of average absolute error. Notice that the average sum of squared error on the test set was much lower than both the mean cross validation sum of squares error and the training set sum or squares error; the training set error refers to the error calculated on the training samples using the predictions from the final model. Most of the training set and cross validation error comes from a single sample – sample number 72 – which has a survival much higher than the next training sample. In N-fold cross validation, the network trained using the other samples did not generalize well to this outlying sample; in the final model, the simple structure of each individual network, which helped to avoid overfitting, did not allow this sample to be modeled well.

The neural network model appears to have predicted the survival of the test samples reasonably well; on average, the predictions were off from the actual survival by only 4.6 months. However, as was the case in the classification models, this test set was so small – containing only six samples – that the variance of the test set error

was likely to be extremely high. Thus, these results cannot be trusted as a reliable measure of the ability of the model to generalize to new samples; rather, they are meant only as a simple evaluation of the model's performance on a small number of previously unseen samples.

4.3 Discussion of results

Many of the individual results were discussed as they were presented in Sections 4.1 and 4.2. In general, it appears that the Gordon method did not do well in general to identify differentially expressed genes that build robust predictive models. While the method appears to have performed reasonably well when only the best and worst samples in terms of survival were included in the training set, it only identified a very limited number of genes in the randomly selected training set used throughout much of the analysis. Furthermore, in almost all cases, models built using the genes identified by the Gordon method were clearly inferior in terms of cross validation error to those built using the any number of genes identified by the Golub method. The Golub method had the added benefit of being able to identify any number of genes without needing to change constraints, since more genes can be easily selection from the list ranked by signal-to-noise ratio.

As for the various types of machine learning used to build predictive models, neural networks performed the best in general on a number of different types of input variable sets. Both a small number of hidden neurons and some form of regularization were effectively used to avoid overfitting the training samples, leading to models that generalized fairly well regardless of the dimensionality of the input data. The best neural network model achieved a cross validation error rate of zero and correctly classified five of the seven test samples into good and poor outcome groups. While they performed well, one of the biggest downsides to neural network models, however, is the fact that they are very difficult to interpret.

Several support vector machines also had a cross validation estimate of zero, although this estimate did not actually use all of the training samples. The best SVM

also correctly predicted five of the seven test samples correctly. A curious result of the support vectors machines was that all of the kernel functions produced identical results on every one of the input variable sets.

Classification trees did not perform overly well, either in cross validation or on the test set. The best classification tree model had a reasonable cross validation error rate of 9.5% on the training set but had an ROC area of less than .5 on the test set. The logistic regression also did not perform well, although the stepwise variable selection which can be performed by many statistical software packages was an attractive option. A logistic regression model with backward stepwise selection achieved a zero cross validation error rate but did not generalize well to the – albeit small – test set.

The additional patient variables – such as age, sex, tumor type and tumor histology – did not appear to be helpful in predicting survival for mesothelioma patients. In almost all cases, including these variables led to equivalent or worse performance compared to the corresponding model without these variables. This was expected, however, since the one among these variables that is most often used as a predictor of patient outcome, epithelial tumor subtype, gave a logrank p value of .399 for all of the data set samples, well above the statistically significant level.

The method of building the ratio-based predictors seemed somewhat sensitive to the software used to generate the gene expression matrix, since the most accurate predictors with both the older and newer data sets did share some, but not all, of the same genes. Furthermore, none of the predictors generated using the newer data set classified the training set with 100% accuracy and the variable selection method identified far fewer genes in the newer data set than in the original data set.

Without an independent test set, it is difficult to compare the ratio-based predictor created by Gordon *et al* to any of the other models built with the various machine learning techniques. The method of building these ratio-based predictors did have a high N-fold cross validation error rate of 35%, much higher than the best model built with any of the four machine learning techniques. While this result may be biased because of the different training sets, the method of building ratio-based

predictors may not be the best choice because it does not take advantage of all of the training data – only the samples in the 75^{th} percentile and above and samples in the 25^{th} percentile and below are used – and because it imposes some seemingly *ad hoc* restrictions during gene selection. However, the ratio-based predictors of Gordon *et al* are interpretable and easy to calculate and have been shown to have good performance on a relatively large independent test set. Furthermore, among other benefits, the use of ratios allows gene expression levels collected using various methods to be used in the same analysis [15].

Chapter 5

Future Work

New ratio-based models were built by applying the method of the Gordon *et al* study to the newer and presumably more accurate gene expression matrix; furthermore, using various machine learning methods, four models were built using a randomly generated training set and compared using N-fold cross validation and a small test set. Ultimately, these models will need to be compared to one another – and to the ratio-based predictor originally created by Gordon *et al* – by using a large independent test. A large independent test set would reduce the variance in the results and allow for meaningful and reliable comparisons between the various models. The best model could then be selected to be used in practice to predict survival for patients with mesothelioma, thus helping to guide treatment of the disease.

Therefore, the next step in this analysis is to collect and analyze tumor samples to be used as a test set. Since 29 samples have already been collected at BWH which were not included in the data set [15], these samples only need to be analyzed using microarrays in order to generate a test set; PCR could even be used since a very limited number of genes – ten or less genes in most cases and only four genes in some – were used in each of models. Collecting and analyzing these samples using microarrays or PCR is a job for biologists rather computer scientists, however.

Without an independent set of samples to use to test these models, none of the constructed models can safely be used in practice to guide patient treatment. These models were built based on at most 21 samples and, apart from cross validation,

tested using at most seven uncensored samples; in the absence of a larger test set, the results of these models are unreliable at best. Thus, the collection and analysis of a large test set of new mesothelioma tumor samples are crucial to the success of using variable selection and machine learning methods to positively affect the quality of life of patients with malignant pleural mesothelioma.

Application of the methods in another domain may give some indication of whether the ratio-based classifier has a more general use than the one described by Gordon *et al.* As large data sets become available, extension of the experiments described here to other biomedical domains would help to answer this type of question.

Chapter 6

Contributions

This thesis investigated a number of different methods for building models to predict postoperative survival for patients who underwent extrapleural pneumonectomy as a result of malignant pleural mesothelioma. A patient was classified as having a poor outcome if his or her survival was less than the median survival length of 11 months and as having a good outcome if his or her survival was greater than or equal to 11 months. The method of building ratio-based predictors originally used by Gordon *et al* was recreated using both the original data set and a newer data set which contained the same samples but was generated with a newer version of Affymetrix software. In addition to recreating these results, several variable selection methods and machine learning techniques were explored using the newer data set and a randomly chosen training set. These machine learning techniques included decision trees, logistic regression, artificial neural networks, and support vector machines.

The results showed that neural networks generalized the best of the four main types of models, achieving an N-fold cross validation error rate of zero and a logrank p value on the test set of .219. The best support vector machine model also performed very well. Logistic regression and decision trees generalized poorly, for the most part. Furthermore, the Gordon method of variable selection was shown to perform very poorly compared to the Golub method on the random training set, since models built using the Gordon method genes performed among the worst in terms of cross validation error for all model types.

Lastly, many of the neural network, support vector machine, logistic regression, and decision tree models achieved an N-fold cross validation error of, or close to, zero. However, the ratio-based predictors had a much higher cross validation error of 35%. While the ratio-based method used a different training set than the other models, it would seem that the other models generalize better than the ratio-based ones; an independent test set would be needed to confirm this speculation.

These results may help guide future researchers to select methods appropriate for variable selection and machine learning in microarray data analysis. Furthermore, since a number of tumors have already been collected to be used as a test set, the results presented in this paper may used to guide researchers as to which new genes to analyze in these samples using PCR. This data could then be used to compare the performance of the new models with that of the ratio-based model originally built by Gordon *et al.* The best performing model could then be used reliably in practice to predict patient survival following EPP, thus helping to guide treatment and improve overall survival for patients with MPM. Such a predictor could eventually lead to new types of treatment for a disease which is currently incurable.

Bibliography

[1] American Joint Committe on Cancer. *AJCC Cancer Staging Manual*, fifth edition, 1997.

[2] P. Baas, H. Schouwink, and F. A. N. Zoetmulder. Malignant pleural mesothelioma. *Annals of Oncology*, 9(2):139–149, 1998.

[3] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[4] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA, 1984.

[5] Michael P.S. Brown, William Noble Gundy, David Lin, Nello Cristianini, Charles Walsh Sugnet, Terrence S. Furey, Manuel Ares Jr., and David Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. *Proceedings of the National Academy of Sciences of the United States of America*, 97(1):262–267, 2000.

[6] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[7] Helen C. Causton, John Quackenbush, and Alvis Brazma. *Microarray Gene Expression Analysis: A Beginner's Guide.* Blackwell Publishing, Malden, MA, 2003.

[8] A. Philippe Chahinian and Harvey I. Pass. Malignant mesothelioma. In James F. Holland and Emil Frei III, editors, *Cancer Medicine*, pages 1293–1312. B. C. Decker, Hamilton, Ontario, fifth edition, 2000.

[9] Sandrine Dudoit, Jane Fridyland, and Terence P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457):77–87, 2002.

[10] William D. Dupont. *Statistical Modeling for Biomedical Researchers.* Cambridge University Press, 2002.

[11] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[12] Gavin J. Gordon, Roderick V. Jensen, Li-Li Hsiao, Steven R. Gullans, Joshua E. Blumenstock, Sridhar Ramaswamy, William G. Richards, David J. Sugarbaker, and Raphael Bueno. Translation of microarray data into clinically relevant cancer diagnosis tests using gene expression ratios in lung cancer and mesothelioma: Supplemental information. Found at: http://chestsurg.org/microarray.htm.

[13] Gavin J. Gordon, Roderick V. Jensen, Li-Li Hsiao, Steven R. Gullans, Joshua E. Blumenstock, Sridhar Ramaswamy, William G. Richards, David J. Sugarbaker, and Raphael Bueno. Translation of microarray data into clinically relevant cancer diagnosis tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Research*, 62:4963–4967, 2002.

[14] Gavin J. Gordon, Roderick V. Jensen, Li-Li Hsiao, Steven R. Gullans, Joshua E. Blumenstock, William G. Richards, Michael T. Jaklitsch, David J. Sugarbaker, and Raphael Bueno. Using gene expression ratios to predict outcome among patients with mesothelioma: Supplemental information. Found at: http://chestsurg.org/geneexpress.htm.

[15] Gavin J. Gordon, Roderick V. Jensen, Li-Li Hsiao, Steven R. Gullans, Joshua E. Blumenstock, William G. Richards, Michael T. Jaklitsch, David J. Sugarbaker, and Raphael Bueno. Using gene expression ratios to predict outcome among patients with mesothelioma. *Journal of the National Cancer Institute*, 95(8):598–605, 2003.

[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, New York, 2001.

[17] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation.* Addision-Wesley, Redwood City, CA, 1991.

[18] Hiroshi Hirano, Shinichi Takeda, Yoshinori Sawabata, Yoshitomo Okumura, Hajime Maeda, Masaki Hanibuchi, Masami Ito, Masaru Nakagawa, and Kunio Uematsu. Localized pleural malignant mesothelioma. *Pathology International*, 53:616–621, 2003.

[19] Jonathan Jackson. Extensible neural network software: Application in gene expression analysis. Master's thesis, Massachusetts Institute of Technology, 2004.

[20] Thorsten Joachims. Estimating the generalization performance of an svm efficiently. Found at: http://www.joachims.org/publications/joachims_00a.pdf.

[21] Steen Knudsen. *A Biologist's Guide to Analysis of DNA Microarray Data.* Wiley-Interscience, New York, 2002.

[22] Kyeong Eun Lee, Naijun Sha, Edward R. Dougherty, Marina Vannucci, and Bani K. Mallick. Gene selection: a "bayesian" variable selection approach. *Bioinformatics*, 19(1):90–97, 2003.

[23] Simon M. Lin and Kimberly F. Johnson, editors. *Methods of Microarray Data Analysis.* Kluwer Academic, Boston, 2002.

[24] Christopher B. Manning, Val Vallyathan, and Brooke T. Mossman. Diseases caused by asbestos: mechanisms of injury and disease development. *International Immunopharmacology*, 2:191–200, 2002.

[25] Wei Pan. A comparative review of statistical methods for discovering differentially expression genes in replicated microarray experiments. *Bioinformatics*, 18(4):546–554, 2002.

[26] Harvey I. Pass. Malignant pleural mesothelioma: surgical roles and novel therapies. *Clinical Lung Cancer*, 3(2):102–117, 2001.

[27] Harvey I. Pass, Zhandong Liu, Anil Wali, Raphael Bueno, Susan Land, Daniel Lott, Fauzia Siddig, Fulvio Lonardo, Michele Carbone, and Sorin Draghici. Gene expression profiles predict survival and progression of pleural mesothelioma. *Clinical Cancer Research*, 10:849–859, 2004.

[28] Scott L. Pomeroy, Pablo Tamayo, Michelle Gaasenbeek, Lisa M. Sturla, Michael Angelo, Margaret E. McLaughlin, John Y.H. Kim, Liliana C. Goumnerova, Peter M. Black, Ching Lau, Cynthia Wetmore, Jaclyn A. Biegel, Tamaso Poggio, Shayan Mukherjee, Ryan Rifkin, Andrea Califano, Gustavo Stolovitzky, David N. Louis, Jill P. Mesirov, Eric S. Lander, and Todd R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415:436–442, 2002.

[29] Scott L. Pomeroy, Pablo Tamayo, Michelle Gaasenbeek, Lisa M. Sturla, Michael Angelo, Margaret E. McLaughlin, John Y.H. Kim, Liliana C. Goumnerova, Peter M. Black, Ching Lau, Cynthia Wetmore, Jaclyn A. Biegel, Tamaso Poggio, Shayan Mukherjee, Ryan Rifkin, Andrea Califano, Gustavo Stolovitzky, David N. Louis, Jill P. Mesirov, Eric S. Lander, and Todd R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression: Supplemental information. Found at: www.genome.wi.mit.edu/MPR/CNS.

[30] John Quackenbush. Microarray data normalization and transformation. *Nature Genetics Supplement*, 32:496–501, 2002.

[31] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[32] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[33] Bernhard Schölkopf, Christopher J.C. Burges, and Alexander J. Smola, editors. *Advances in Kernel Methods: Support Vector Learning.* MIT Press, 1999.

[34] Sunil Singhal, Rainer Weiwrodt, Liliana Malden, Kimberly Matzie, Joseph Friedberg, Kunjlata M. Amin, John C. Kucharczuk, Leslie A. Litzky, Steven W. Johnson, Larry R. Kaiser, and Steven M. Albelda. Gene expression profiling of malignant mesothelioma. *Clinical Cancer Research*, 9:3080–3097, 2003.

[35] David H. Sterman, Larry R. Kaiser, and Steven M. Albelda. Advances in the treatment of malignant pleural mesothelioma. *Chest*, 116(2):504–520, 1999.

[36] Duncan J. Steward, John G. Edwards, W. Roy Smythe, David A. Waller, and Kenneth J. O'Byrne. Malignant pleural mesothelioma–an update. *International Journal of Occupational and Environmental Health*, 10:26–39, 2004.

[37] Gustavo Stolovitzky. Gene selection in microarray data: the elephant, the blind men and our algorithms. *Current Opinion in Structural Biology*, 13:370–376, 2003.

[38] Virginia Gross Tusher, Robert Tibshirani, and Gilbert Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences of the United States of America*, 98(9):5116–5121, 2001.

[39] Lambros S. Zellos and David J. Sugarbaker. Multimodality treatment of diffuse malignant pleural mesothelioma. *Seminars in Oncology*, 29(1):41–50, 2002.