CANCER DIAGNOSIS AND PROGNOSIS VIA LINEAR-PROGRAMMING-BASED MACHINE LEARNING

By
W. Nick Street

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy
(Computer Sciences)

at the
UNIVERSITY OF WISCONSIN – MADISON
1994

Dedication

This document is dedicated to the people who made it possible, my parents: Caryl, who preferred that I write a mystery novel, and Nick, who thinks maybe I did.

Acknowledgements

My greatest thanks go to my family, Margaret and Kelly, for their unwavering love and support during these years of little time and less money. Thanks also to the faculty members who provided all of the context and much of the insight for this work: Olvi Mangasarian, who is not only a brilliant mathematician but also a great mentor and a good friend; Bill Wolberg, who has provided both direction and raw data for my work and trusts the results enough to incorporate them into his practice; and Jude Shavlik, whose shared expertise in the field of machine learning has improved every aspect of this research.

I also wish to single out some of my colleagues for their contributions: Kristin Bennett, whose path I have followed; the machine learning group, particularly Rich Maclin and Mark Craven; and the math programming group, particularly Chunhui Chen and Steve Dirkse. The exchange of ideas with these people has been the best part of being a graduate student.

This research was partially supported by Air Force Office of Scientific Research Grants 89–0410 and F49620–94–1–0036 National Science Foundation Grants CCR–9101801, CCR–93222479 and CDA–9024618.

Abstract

The purpose of this research is twofold. The first purpose is the development of machine learning methods based on linear programming. These advances come in the form of both novel learning algorithms and generalization-improvement approaches that are applicable to a wide range of learning algorithms. The second aim is the application of these algorithms, along with ideas from the fields of statistics and image processing, to problems arising in the diagnosis and treatment of breast cancer. The first chapter provides background into the relevant areas of research, most importantly machine learning and breast cancer diagnosis and prognosis, and describes previous work that has united the two fields.

The first major contribution of this research is an automated cytological analysis system that we call **Xcyt**. This system includes a partially automatic image segmentation facility for isolating cell nuclei from a digital image. **Xcyt** allows computation of various size, shape and texture features of these nuclei. A linear separator distinguishes benign cases from malignant cases based on 569 training examples. **Xcyt** also provides an estimate of the probability of malignancy for each case. The **Xcyt** system is currently in use at the University of Wisconsin Hospitals.

We also address the problem of cancer prognosis, that is, determining when a cancer is likely to recur. A new learning method, the recurrence surface approximation (RSA), is introduced for predicting the time of recurrence. This

procedure uses linear programming to predict recurrence from right-censored input data. Various extensions and variations of RSA are also explored, including a separation-based procedure that we call implicit RSA (IRSA).

The topic of improving the generalization of a learning system is first addressed in the context of RSA, where we introduce a procedure for choosing the most relevant input features for use the the predictive model. The final two chapters also describe generalization-enhancement methods. The first is mock generalization, which incorporates a tuning set into the optimization process. The second is banded approximation, which explicitly defines a tolerance band around the predictive surface in order to avoid overfitting the training data.

Contents

List of Tables

List of Figures

Chapter 1

Review of relevant technologies

The research presented in this dissertation draws from a number of different fields of study, the relevant areas of which will be briefly reviewed in this chapter. We are primarily concerned with inductive machine learning, that is, the automatic construction of general concepts from specific examples. The tools with which we build learning algorithms come from the field of mathematical optimization, specifically linear programming. We begin with an overview of the application area which unifies this research: the diagnosis and prognosis of breast cancer.

1.1 Breast cancer

Despite a great deal of public education and scientific research, breast cancer continues to be the most common and the second most deadly form of cancer among women [59]. An increased incidence of the disease has offset improving survival rates, resulting in a nearly constant mortality from breast cancer over the past twenty years [61]. Two major areas of breast cancer research will be addressed in this work: early detection through timely diagnosis, and appropriate planning of treatment through accurate prognosis.

1.1.1 Diagnosis

The diagnosis of breast tumors has traditionally been performed by a full biopsy, an invasive surgical procedure. Fine needle aspirations (FNA's) – needle biopsies, easily performed on an outpatient basis, which maintain cell integrity at the cost of architectural information – provide a way to examine a small amount of tissue from the tumor. By carefully examining both the characteristics of individual cells (or cell nuclei) and important contextual features, such as the size of cell clumps, physicians at some specialized institutions have been able to diagnose breast cancer successfully using FNAs.

However, diagnosis with this procedure has met with mixed success; see for instance the summary of Frable [29]. Many different features are thought to be correlated with malignancy, and may interact with one another. Thus the process remains highly subjective, depending upon the skill and experience of the physician. The review of Giard and Hermans [33] particularly emphasized the need for performance standards in FNA diagnosis. Further, few institutions have individuals with the necessary cytological training to perform this type of diagnosis.

1.1.2 Prognosis

A more difficult problem is that of breast cancer prognosis, the prediction of when (or if) the cancer will recur following surgery. Since the 1950's, the standard method for prognosis has been the TNM (tumor size, lymph node, metastasis) staging system [6, 36]. Patients with distant spread of the disease (metastasis) at time of surgery have the worst prognosis. Among other patients, both increased tumor size and an increased number of cancerous lymph nodes (near the tumor) have been shown to be negative prognostic factors [19]. Nuclear features observed at the time of diagnosis have also been shown to be correlated with prognosis [12, 68].

Prognosis in the TNM model is expressed as survival curves, that is, plots of time vs. probability of survival. Values of tumor size and lymph node status are discretized into a small number of intervals, and bins are created by plotting them against each other. Survival curves are computed based on a large sample of patients which fall into each bin. Treatment plans for new patients can then be determined based on their expected probability of survival at two years, five years, etc. Although it is a standard in clinical practice, the TNM model has a number of shortcomings:

- Survival probabilities are a coarse description of prognosis and do not allow individualized predictions.
- The coarse discretization of prognostic factors groups dissimilar patients together.
- The model is fixed and inflexible, not allowing the addition of other prognostic features.
- Most importantly, the removal of lymph nodes for prognostic analysis carries the risk of long-term swelling and loss of use in the arm. An accurate prognostic method which does not need lymph node status as an input feature would remove the need for this procedure.

1.2 Machine learning

1.2.1 Overview

The solutions to the problems of diagnosis and prognosis investigated in this thesis lie within the field of *inductive learning* (also known as *supervised* or *similarity-based learning*). Intuitively, the task is to learn ("induce") general concepts from specific examples. More formally, the learning system constructs a function f that attempts to map given points $x^1, x^2, ..., x^m$ in some space $\mathcal{X}$ into given points $y^1, y^2, ..., y^m$ in some other space $\mathcal{Y}$. Each x^i consists of n input features which describe the particular example, that is, x^i is a point in the n-dimensional real space $\Re^n$. The corresponding y^i is the dependent variable to be learned. In the special case of *classification*, y^i denotes a class membership for each example, e.g., the class of benign cases vs. the class of malignant cases, and hence $y^i \in \{0, 1\}$.

There exist many different approaches for learning the function f, and many different ways to represent it. In recent years, one of the most popular of these approaches has been artificial neural networks (ANN's), or connectionist methods. In particular, feed-forward ANN's trained with algorithms such as backpropagation [78] have been shown to be an extremely flexible and powerful approach to function approximation [39, 50, 79]. Historically, the field of statistical pattern recognition, as summarized in Duda and Hart [25], contributes much to our understanding of such problems. Methods such as linear discriminant analysis, logistic regression, and nearest-neighbor classification, as described in [25], are widely used. Another popular concept representation which is more relevant to our work is that of decision trees. This paradigm has been used extensively in work from both machine learning [70, 73, 87] and statistics [14]. Each of these methods carries with it a particular *inductive bias*; that is, a tendency to form one type of concept over another.

Another important issue in inductive learning is evaluating the resulting classifier in terms of how well it *generalizes*, that is, how well it performs on new data which was not used during training. The observed accuracy on the training set is nearly always a poor estimate of how well the system will generalize to unseen data, since most learning algorithms can reproduce the training data exactly (as an extreme example, consider memorizing the training data by simply building a lookup table). One effective estimation method is leave-one-out testing [46], wherein the generalizer is trained with all but one of the samples and tested on the "left-out" sample. This procedure is repeated m times until each training point has been used for testing. The result is a very accurate and nearly unbiased, but computationally expensive, accuracy estimate. A more tractable re-sampling method is cross-validation [84], which successively removes, say, 10% of the data for testing, again using each point in exactly one test set. Other statistical techniques, such as jackknifing and bootstrap [27], are also available.

Generalization performance on real-world data sets is often improved if some sort of smoothing procedure is applied to the learned concept, preventing the method from memorizing the training data at the expense of accuracy on unseen examples. For instance, a *tuning* or *validation set* (a portion of the training examples set aside to simulate unseen data) can be used to stop training a backpropagation network before it reaches optimality on the training set. Connectionist methods such as optimal brain damage [48] and cascade correlation [28] prevent overfitting by searching for an appropriate neural network architecture. Similarly, decision trees learned by the algorithms CART [14] or C4.5 [70, 73] are pruned to remove potentially extraneous decision branches. While biasing a learning system toward *simpler* concepts makes the accuracy more reliably measurable [13] and often improves generalization on real-world data, it should not be interpreted as a universally desirable bias [80, 98]. The key to good generalization is to find a generalizer with the *right* complexity, where complexity may be measured by the number of parameters in the predictive model (e.g., the number of separating planes in a decision tree), the number of training iterations, the number of features used in training, etc.

For much of the work presented here, choosing an appropriate subset of the available input features will be particularly important for good generalization. In higher dimensional problems a much larger number of points is needed in order to cover the space adequately. Even for reasonably sized data sets, this can result in sparse, hard-to-fit data, or in the case of a classification task, deceptively easy-to-separate data. However, finding the optimal feature set is a combinatorial search problem. If framed as an integer programming problem, the search can be approximated with methods such as branch and bound [65]. Most methods, including those used in statistical regression, focus on a greedy, directed search through the space of possible feature sets [24, 44, 60], where only a small manageable set of variable subsets are examined.

1.2.2 Medical applications

Machine learning approaches – particularly ANN's – have been used for a number of disparate clinical diagnosis tasks; diagnosing, for example, skin lesions [101], appendicitis [26], and myocardial infarction [5]. In breast cancer, different researchers have applied neural networks to diagnosing from mammograms [100], ultrasound images [34], and pathological markers [2]. Still, few of these systems have received general clinical usage.

Machine learning methods have also been applied to prognosis. Burke [16] uses a traditional backpropagation network to predict 10-year survival in breast cancer patients. A self-organizing neural architecture was used by Schenone *et al.* [81] to predict recurrence time. Ravdin and Clark [74] include time as one of the input features of an ANN to predict a survival curve for each patient (see Chapter 4). This method has also been used to verify the value of new prognostic features [75].

1.2.3 Linear programming approaches

Mathematical optimization approaches, in particular linear programming [22], have long been used in problems of pattern separation. Highleyman [37] first found that showing linear separability is equivalent to to finding a nonnegative solution to a set of linear equalities. Both Charnes [20] and Mangasarian [52] developed linear programs which construct planes to separate linearly separable point sets. Mangasarian [52] also showed how to separate sets by a nonlinear surface using linear programming whenever the surface parameters appeared linearly, e.g. a quadratic or polynomial surface. However, these formulations were prone to fail on sets that were not separable by a surface linear in its parameters, and were therefore of limited usefulness for learning general patterns.

Mangasarian's multisurface method (see next section) avoided this difficulty by constructing a piecewise-linear separation surface. More recently, Roy *et al.* [77] and Mukhopadhyay *et al.* [63] used linear programming to generate convex "covers" which are combined to classify general patterns.

1.3 Review of previous work at Wisconsin

The application of machine learning techniques to problems in breast cancer diagnosis at the University of Wisconsin began in 1989 with the collaborative work of W. H. Wolberg (Surgery and Human Oncology) and O. L. Mangasarian (Computer Sciences). The Multisurface Method (MSM) [53, 54] of pattern separation was first applied to a collection of cases represented by nine subjectively evaluated cytological features [57, 58, 92]. The MSM procedure uses a linear programming model to place successive pairs of separating planes in the feature space of the input examples, building a piecewise-linear separating surface. The procedure can also be considered a neural network training algorithm [9]. While successful [93], the diagnostic results still depended on the ability to subjectively assign values to input features, and were therefore difficult to replicate. Further, the classifier employed was relatively complex, employing four pairs of planes in 9-space. Subsequent work by Mangasarian with K. P. Bennett [7, 10], including the development of the MSM-Tree (MSM-T) algorithm (described in Chapter 2), resulted in improved diagnostic results. The problem of cancer prognosis was also cast as a classification problem [91, 96], separating those patients who experienced recurrence in less than two years from those known to have been disease free for at least two years.

Chapter 2

Breast cancer diagnosis: Xcyt

2.1 Introduction

The first problem we address is that of diagnosing breast masses as benign or malignant. The diagnosis process begins by taking a fine needle aspirate (FNA) from the patient's breast. This tissue sample is then stained and mounted on a slide. Digital images from the FNA are transferred to a workstation by a video camera mounted on a microscope. These images are the input to the computerized diagnostic system.

The platform for the image processing and diagnosis portion of this work is an X Window System based user interface called **Xcyt**. In addition to being an effective research tool, the UNIX version of **Xcyt** is also being used in clinical decision making, and is in current use for breast cancer diagnosis on real patients at the University of Wisconsin Hospitals.

2.2 Image analysis (snakes)

Many studies, some of which were summarized by Frable [29], have shown that various features of cell nuclei can be used to distinguish benign from malignant breast tumors. The first step in automatic diagnosis is to isolate and precisely specify the nuclei in a captured digital image. However, properly segmenting an image into its constituent parts is an important and largely unsolved problem in itself. This is made worse by the fact that the cells are very heterogeneous, with widely varying sizes, shapes, and image intensity properties. We tried several different approaches, including thresholding, region growing [3] and contour following [18]. Various commercial and public domain image analysis systems utilize these methods.[1]

Our chosen solution [85, 86, 95] is a semi-automatic segmentation procedure [43] known as "snakes." Beginning with a user-defined approximate boundary as an initialization (see Figure 2.1) the snake locates the actual boundary of the cell nucleus. A snake is a deformable spline which seeks to minimize an energy function defined over the arclength of a closed curve. The energy function is defined in such a way that the minimum value occurs when the curve accurately corresponds to the boundary of a cell nucleus.

To achieve this, the energy function to be minimized is defined as the following function of arclength s:

$$E = \int_s (\alpha E_{cont}(s) + \beta E_{curv}(s) + \gamma E_{image}(s))ds$$

Here E represents the total energy integrated along the arclength s of the snake. The energy computation is a weighted sum of energy terms E_{cont}, E_{curv} and E_{image} with respective weights α, β and γ. To simplify the necessary processing, the energy function is computed at a number of discrete points along the curve, and the sum of these values is minimized. The component energy terms measure the following quantities:

- Continuity: E_{cont}

  This term is constructed to penalize discontinuities in the curve. In the discrete case, this term measures how evenly spaced the snake points are.

[1]Some examples are: **CAS 200**, Cell Analysis Systems, Inc., Elmhurst, IL 60126;
Forcyte, Dianon Systems, Inc., Stratford, CT 06497;
IMPATH, Baltimore VA Medical Center, Baltimore, MD 32301.

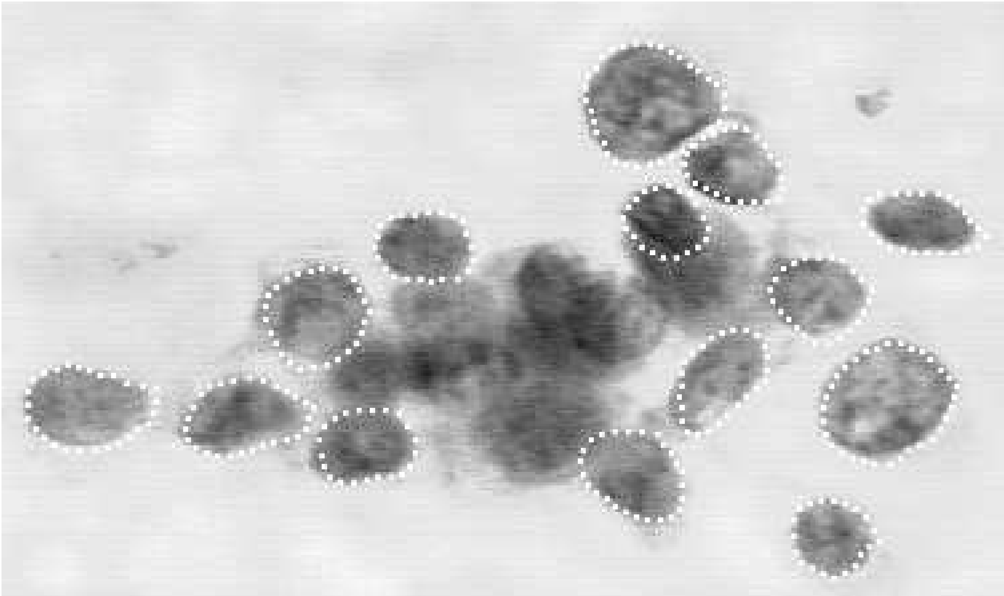

Figure 2.1: Digital image of breast lump cells, showing the initial approximate boundaries of the cell nuclei, outlined using a computer mouse.

Note that this is a geometric property of the snake itself, and does not depend on the nucleus boundary that is being determined. The distance from a snake point to one of its neighbors is found and compared to the average distance between adjacent points. The magnitude of this difference is then E_{cont}.

- Curvature: E_{curv}

  This geometric term measures discontinuities in the curvature of the snake. Cell nuclei are more or less ellipsoidal; hence, points with abnormally high or low curvature, compared to a circle, are penalized. The following method was adopted by taking advantage of this knowledge about the nuclear shape. First, the 'center' of the snake (center of mass of the snake points) is located. The distance from a snake point to the center (i.e., length of radial line) is then compared to the average of such distances in a neighborhood of the point. The magnitude of the difference is this energy term E_{curv}.

- Image: E_{image}

  This is the term that ties the snake's performance to the underlying image. In our case E_{image} measures the gray-level discontinuity along the snake. To quantify this discontinuity we convolve the area of the image corresponding to the snake point with a Sobel [3] edge detector and observe the resulting edge magnitude. This term is customized by taking advantage of the fact that cell nuclei are generally darker than the surrounding material. Hence, the edge detection template is rotated so that the expected edge is perpendicular to the radial line of the nucleus at that point. For instance, for a snake point directly above the center of the nucleus, the following edge template would be applied:

1	2	1
0	0	0
-1	-2	-1

  In this way, gray scale discontinuities which are perpendicular to the radial line produce the highest edge score. E_{image} is defined so a sharp discontinuity minimizes the energy value.

The weights α, β and γ are empirically derived constants. For best performance on these images, γ is set somewhat higher than the others to ensure that the snake converges to any visible boundary; typically, $\gamma = 1.5$, $\alpha = \beta = 1.0$. The curvature term determines the snake's shape in cases of low contrast or partial occlusion. The continuity term does not determine shape, but does prevent snake points from bunching together near areas of sharpest gray scale contrast.

In order to control computation time, the optimal local value of the energy function is approximated using a greedy algorithm due to Williams and Shah [90]. If the function value at a particular snake point can be lowered by moving the point to an adjacent pixel, then it is moved, thus possibly affecting the energy computation at other points. The process is repeated for each point until all points settle into a local minimum of the energy function. The results of a typical image are shown in Figure 2.2. After a little practice, anyone comfortable with using a mouse can process an image (20 to 30 cells) in less than 5 minutes.

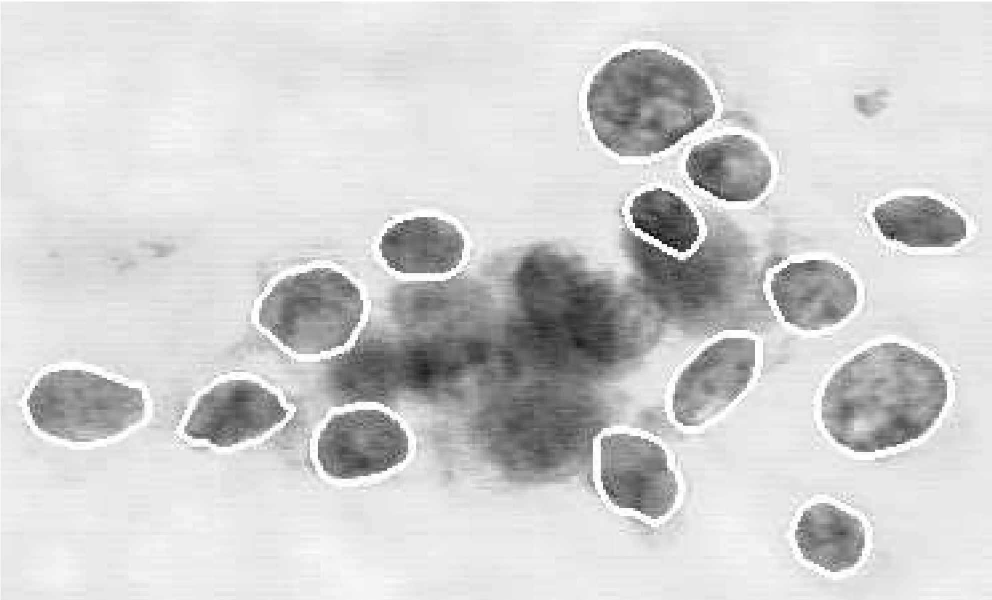

Figure 2.2: Snakes after convergence to cell nucleus boundaries.

In contrast to other segmentation methods, snakes can approximate boundaries even in areas of little or no gray-scale contrast. Further, both region growing and thresholding methods also tend to leave gaps in the object which must then be corrected by the user. The snake approach provides a fast, robust segmentation procedure which gives the precision necessary for evaluating the relevant nuclear features. By customizing the energy function, the snakes have been adapted to this particular application in two ways: the directional edge detector and the elliptical curvature assumption.

2.3 Extraction of digital nuclear features

In order to evaluate the size, shape and texture of the cell nuclei, we arrived at a set of ten features [86, 94] which are computed for each cell. Some of these replicate or approximate features previously evaluated subjectively by Wolberg [92], while others are unique to this work. All of the size and shape features were verified using idealized phantom cells [97]. The computed features are as follows.

1. *Radius*

   Radius is computed by averaging the length of radial line segments, that is, lines from the center of mass of the snake to each of the snake points. See Figure 2.3.

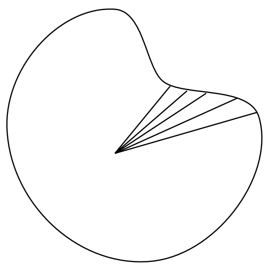

Figure 2.3: Line segments used to compute radius.

2. *Perimeter*

   Perimeter is measured as the sum of the distances between consecutive snake points.

3. *Area*

   Nuclear area is measured by counting the number of pixels on the interior of the snake and adding one-half of the pixels in the perimeter, to correct for the error caused by digitization.

4. *Compactness*

   Perimeter and area are combined [3] to give a measure of the compactness of the cell nuclei using the formula $perimeter^2/area$. This dimensionless number is minimized for a circle and increases with the irregularity of the boundary. However, this measure of shape also increases for nuclei which are elongated, which is not necessarily an indication of malignancy. The feature is also biased upward for small cells because of the decreased accuracy imposed by digitization of the sample.

5. *Smoothness*

   The smoothness of a nuclear contour is quantified by measuring the difference between the length of a radial line and the mean length of the two radial lines surrounding it. If this number is small relative to the distance between consecutive snake points, c, then the contour is smooth in that region. To avoid the numerical instability associated with small divisors, the numerators and denominators were summed first, leading to the following equation:

$$smoothness = \frac{\sum_{points} |r_i - \frac{r_{i-1}+r_{i+1}}{2}|}{n\bar{c}} \tag{2.1}$$

$$= \frac{\sum_{points} |r_i - \frac{r_{i-1}+r_{i+1}}{2}|}{perimeter} \tag{2.2}$$

   where the radii r_i and inter-snake point distances c_i are as shown in Figure 2.4, and $\bar{c}$ is the mean of the c_i.

6. *Concavity*

   Concavity is captured by measuring the size of any indentations in the cell

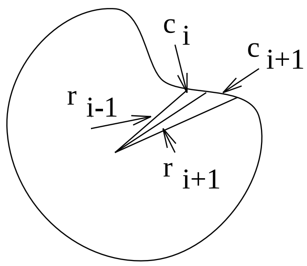

Figure 2.4: Line segments used to compute smoothness. The difference in length of the radial segments r_i and r_{i+1} are compared in magnitude to the distances between snake points c_i and c_{i+1}.

nucleus. See Figure 2.5. Chords are drawn between non-adjacent snake points, and the extent to which the actual boundary of the nucleus lies on the inside of each chord is measured. The average of such distances over the nuclear boundary is the final value. Note that a circular or elliptical nucleus would show no concavity.

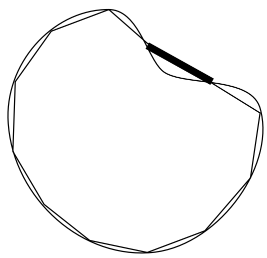

Figure 2.5: Line segments used to compute concavity and concave points. The bold line segment indicates a concavity.

7. *Concave Points*

This feature is similar to concavity but counts only the number of snake points which lie on concave regions of the contour, rather than the magnitude of such concavities.

8. *Symmetry*

   Symmetry is measured by finding the relative difference in length between pairs of line segments perpendicular to the major axis of the cell nucleus contour. See Figure 2.6. The major axis is determined by finding the longest chord which passes from a snake point through the center of the nucleus. The segment pairs are then drawn at regular intervals. To avoid numerically unstable results based on extremely small segments, the sums are again divided, rather than summing the quotients:

   $$symmetry = \frac{\sum_i |left_i - right_i|}{\sum_i left_i + right_i} \tag{2.3}$$

   Care is taken to avoid special cases, such as when a concavity causes both contour points along the symmetry line to be on the same side of the major axis.

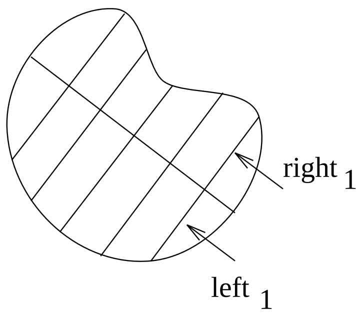

Figure 2.6: Line segments used to compute symmetry. The lengths of perpendicular segments on the right of the major axis are compared to those on the left.

9. *Fractal Dimension*

   The fractal dimension of a cell is approximated using the "coastline approximation" described by Mandelbrot [51]. The perimeter of the nucleus is measured using increasingly larger 'rulers.' As the ruler size increases, decreasing the precision of the measurement, the observed perimeter decreases. See Figure 2.7. Plotting these values on a log scale and measuring the downward slope gives the negative of an approximation to the fractal dimension.

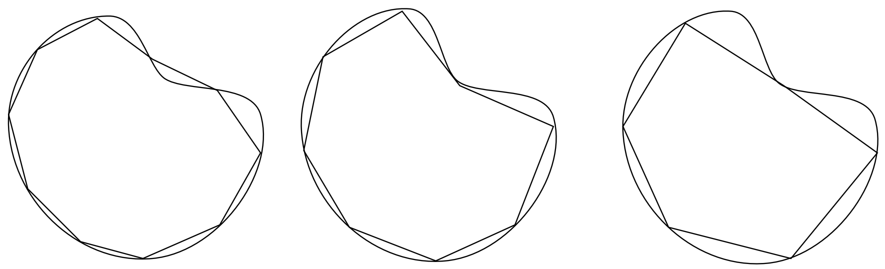

Figure 2.7: Sequence of perimeter measurements for computing fractal dimension. The apparent perimeter of the contour decreases as it is measured more coarsely. The rate of this decrease is used to approximate fractal dimension.

10. *Texture*

    The texture of the cell nucleus is measured by finding the variance of the gray scale intensities in the component pixels.

The mean value, extreme (largest) value and standard error [2] of each feature are computed for each image. To reduce possible noise, the three largest values are averaged in computing the extreme values. The extreme value features are the most intuitively useful for the problem at hand, since only a few malignant cells may occur in a given sample. These 30 features were computed for each of 569 images, yielding a database of 569 30-dimensional points representing 357 benign and 212 malignant cases. The means and standard deviations of

[2] For a sample of n points with variance σ^2, the standard error is defined as $\frac{\sigma}{\sqrt{n}}$.

each of these 30 features, separated by benign and malignant cases, appears in Appendix A.

2.4 Application to breast cancer diagnosis

2.4.1 Classification method: Multisurface Method-Tree (MSM-T)

The classification procedure used to separate benign from malignant samples is a variant on the Multisurface Method (MSM) [53, 54] known as MSM-Tree (MSM-T) [7, 10]. This method uses a linear programming [22] model to iteratively place a series of separating planes in the feature space of the examples. If the two sets of points are linearly separable, the first plane will be placed between them. If the sets are not linearly separable, MSM-T will construct a plane which minimizes the average distance of misclassified points to the plane, thus nearly minimizing the number of misclassified points. The procedure is recursively repeated on the two newly created regions. Figure 2.8 shows an example of the types of planes MSM-T might construct. The resulting planes can then be used in the manner of a decision tree to classify new points.

MSM-T has been shown to learn concepts as well or better than more traditional learning methods such as C4.5 [71, 72] and CART [14]. It also has an advantage over artificial neural network (ANN) methods such as backpropagation [78] in that the training proceeds much faster [8]. The linear programming in the current version of MSM-T is implemented using the MINOS numerical optimization package [64].

2.4.2 Exhaustive feature selection

The 30-dimensional diagnosis data of benign and malignant points are linearly separable. Thus finding one MSM-T plane in this 30-dimensional feature space

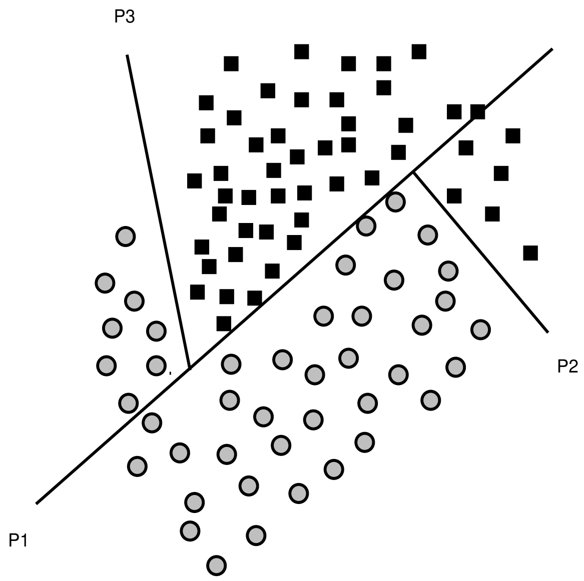

Figure 2.8: MSM-T separating planes.

is already an example of overfitting and leads to poor generalization. In order to find a single classifier with good generalization performance, the space of possible feature subsets was searched. Using a small number of features (1, 2, or 3) and a small number of separating planes (1 or 2), the training set separation accuracy closely approximates the estimated predictive accuracy. Taking advantage of the speed of MSM-T, the training performance of all of the above combinations (1, 2, or 3 of the 30 features and 1 or 2 planes) was observed, and generalization on the most promising feature sets was estimated using 10-fold cross-validation. The best results were obtained with *one plane and three features*: extreme area, extreme smoothness and mean texture. Applied to all the data, the training separation was 97.5%; the estimated accuracy was also 97.5%. In actual clinical practice, this classifier achieved 100% chronological correctness on 92 new cases over the nine months ending April 30, 1994.

2.5 Probability density approximation

Most machine learning applications evaluate performance with nothing more than accuracy results like those above. However, to be useful in a clinical setting, the physician needs some idea of "how benign" or "how malignant" a new case is. By using some ideas from non-parametric statistics, we constructed a method of computing the approximate probability of malignancy. All of the training points were projected onto the normal to the separating plane by simply computing xw, the scalar product of the normal w to the separating plane with the feature value vector x. A graphical depiction of this process is shown in Figure 2.9.

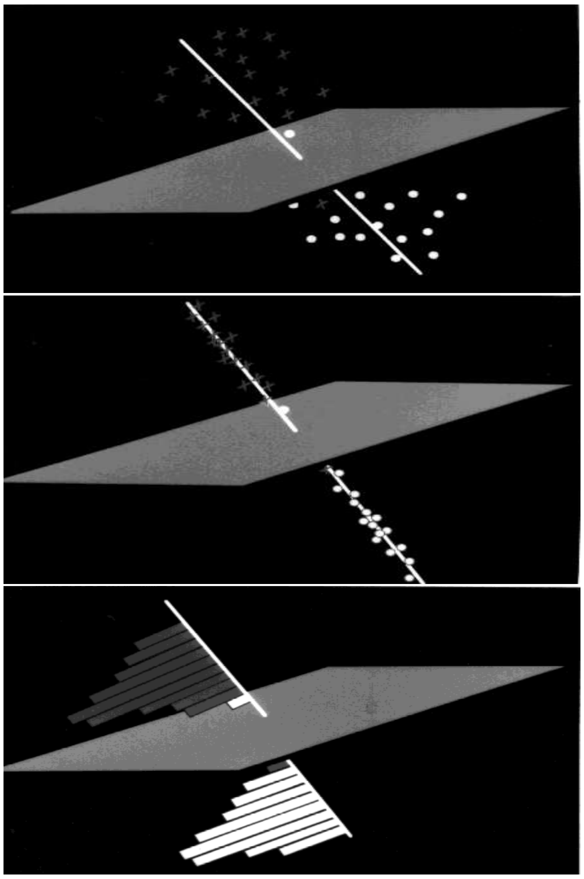

Figure 2.9: Construction of estimated probability density functions. All the training points are projected onto the normal to the separating plane. Separate histograms are then formed for the benign and malignant points along the normal.

Two probability density functions (relative to the classifier) were then approximated using a Parzen window [67, 69] or kernel technique. In general, kernel techniques assign each sample a probability mass centered on the point and falling away gradually in both directions. All of these probability masses are summed and normalized to produce a density function (i.e., the area under the curve is forced to equal one). For this application, the kernel was simply a triangle, with the width chosen empirically to smooth the resulting densities somewhat without filtering out potentially relevant aspects. The resulting densities for the training samples of 357 benign and 212 malignant points are shown in Figure 2.10. The probability of malignancy for a new case can be computed with a simple Bayesian computation, taking the height of the malignant density divided by the sum of the two densities at that point. Note that this assumes equal prior probabilities of the two classes, which was preferred even though roughly 60% of the cases in the training set were benign.

Because of the smoothing property of the kernel technique, the separating plane does not cut the two densities exactly at their intersecting point. In practice, one ambiguous point was classified as benign, but given a probability of malignancy of 0.57. This case was ultimately diagnosed as being atypical, but not yet malignant.

2.6 Clinical application

As mentioned earlier, the **Xcyt** system has been used by Dr. W. H. Wolberg in his clinical practice for the past nine months, and through collaborative studies will soon be used at other institutions. Once the FNA slide from a new patient has been analyzed, the patient is shown a density diagram similar to Figure 2.10 along with the value of xw for the sample. In this way, the patient can easily appraise the diagnosis in relation to hundreds of other cases, in much the same way that an experienced physician takes advantage of years of experience. Thus the patient has a better basis on which to base a treatment decision. Tumors

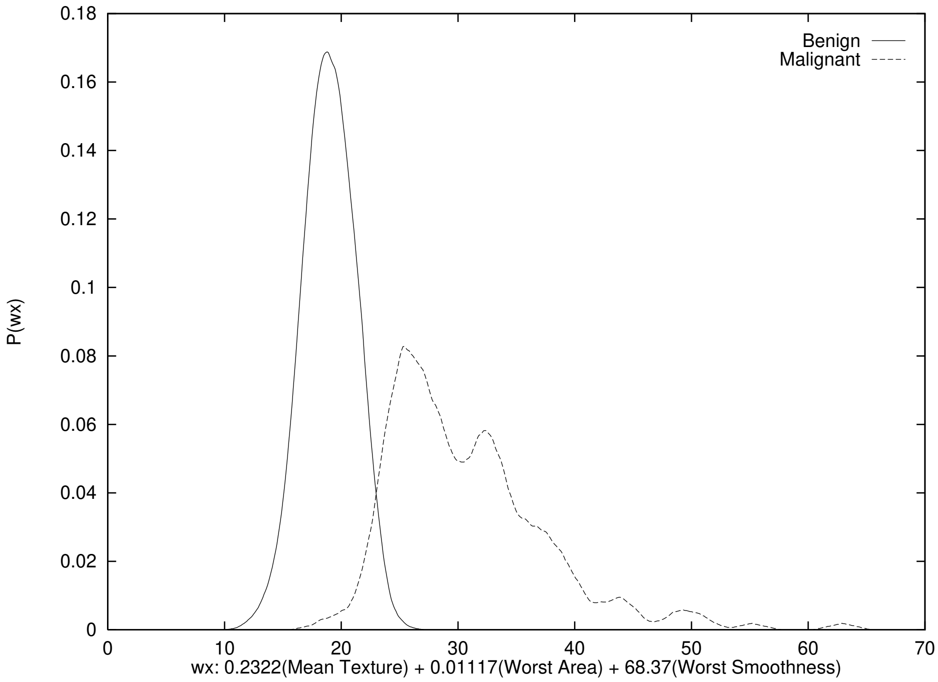

Figure 2.10: Approximated densities of benign and malignant points.

from patients who opt for surgical biopsy have their diagnosis histologically confirmed. Patients who choose not to have the biopsy done are followed for a year at one-month intervals to check for changes in the tumor.

2.7 Discussion and extensions

While the methodology of applying machine learning techniques to features extracted from medical images is certainly not new [4, 89], this work is unique in several respects. First is the precise quantification of nuclear shape, using several different features. This should make the **Xcyt** system more generally applicable to new situations where different representations of shape are relevant to outcome. Secondly, the linear programming-based learning procedure allowed us to find an extremely simple, and therefore reliable, decision surface. By estimating the densities of benign and malignant points relative to this classifier, we have a concise graphical representation of the confidence associated with each diagnosis. Most importantly, the result has been deemed reliable enough to be incorporated into actual clinical practice. Based on numerous correspondence from other clinicians, we expect this use to become more widespread in the near future.

We are currently preparing to distribute the **Xcyt** system for use in multi-institutional clinical trials. As more people use the system, many potential improvements and extensions will undoubtedly come to light. Among the most important extensions are the following:

- Automatic selection of cells:
  One of the key unanswered questions about **Xcyt** is whether or not the selection of cells to be digitized constitutes a significant bias. All of the images were created by a physician who is experienced in the diagnosis of breast FNA. His results have been repeated on a small number of samples by untrained operators, simply by selecting for digitization the area containing the largest observed cell nuclei. Still, the effect of cell selection on eventual diagnosis remains unclear. A more objective approach would be to automate the process, possibly with a motor-driven slide reader which digitizes many different areas of the slide. Each of these images would then be diagnosed in the usual manner, with the final diagnosis corresponding to the worst of these results.

- Improved image segmentation:
  The snake procedure has proven to be an effective solution to the problem of identifying cell boundaries. However, it is currently limited by the local nature of the convergence algorithm, requiring a close initial approximation. One solution to this limitation would be to smooth the image to successive levels of reduced detail, as in the Gaussian pyramid scheme

[17]. The initialized snake could be translated to the lowest, most highly filtered level and run to convergence on the filtered cell nucleus. The resulting snake would then be translated back and used as the initialization at the next more detailed level. In this manner, the snakes might be better able to avoid converging to local features which are not really part of the object boundary.

- Database interfaces:
  Another improvement would be to provide interfaces to popular database programs. One of the key components of a successful application of machine learning techniques is the availability of a quality database of examples. Providing a built-in interface from the image processing system to commonly used commercial database packages would help ensure the quality of the resulting database, without which even the best machine learning approach will be unsuccessful.

- System calibration:
  Lacking a purely automatic method for outlining the cell nuclei, the **Xcyt** system is subject to inter-observer variation. This variation lies along two dimensions: the nature of the cells selected by a particular operator, and how demanding the operator is regarding the converged snakes. We have examined the extent of this variation [96] and found that computation of shape features can vary by up to 10% among different users. One way to avoid this type of variation is to calibrate the system for each new user. We will take into account the bias of the operator by training a learning system to recognize and consequently to counter the bias of the operator (if it is incorrect, e.g. a selection by an inexperienced novice), or to enforce it (if correct, e.g. a selection by an experienced practitioner). This training will be done on samples for which the feature values are known, using those values which were used in constructing the classifier. Thus before the computer system is used by an individual to digitize and diagnose a

sample, the system will calibrate itself by training, say, a neural network that will recognize the bias of the operator and hence take it into account in future diagnosis. Of course as the individual gains experience, his bias may change and therefore the system can be re-calibrated periodically to take that into account.

- Other diseases:
  There is some evidence that the classification system built into **Xcyt** may be applicable to other forms of cancer, even without modification. Twenty slides of FNA's taken from thyroid tumors at UCLA hospitals were successfully diagnosed, indicating that there may be significant structural similarities between the two types of tumors. Other types of cancer cells will be explored in the same way, including present work on brain tumors. In any case, the methodology used to create the **Xcyt** classifier is sound, and can be repeated on for new diseases using large numbers of samples.

From a machine learning perspective, one interesting research direction is extending the idea of *a posteriori* density estimations to a general decision surface. The key here is to define a suitable distance metric, analogous to "distance from the separating plane," which applies to piecewise-linear or non-linear decision surfaces. Initially, we will use "distance from the classifier" as our metric, measuring how close a point was to falling in a region which would have classified it differently. This allows us to collapse the high-dimensional space into one dimension, and use simple counting or kernel techniques to estimate the distributions for each class. In addition, it is possible that better results would be obtained from constructing the distributions in a local fashion, by observing only the points in a neighborhood of the newly observed point. This neighborhood could be customized to reflect the nature of the classifier; for instance, in the case of separating planes, a cylinder might be a more useful neighborhood than a ball. Hence, we are looking to incorporate ideas from non-parametric statistics, together with locality concepts similar to k-nearest-neighbor classification, to arrive at a general method for estimating the probabilities associated with classifications.

Chapter 3

Breast cancer prognosis: the recurrence surface approximation

We next address the more difficult question of the long-term prognosis of patients with cancer. Several researchers, beginning with Black *et al.* [12], have shown evidence that cellular features observed at the time of diagnosis can be used to predict whether or not the disease will recur following surgery. However, with the widespread use of the TNM (tumor size, lymph node, metastasis) staging system [6, 36], nuclear grade is now rarely used as a prognostic indicator. For instance, the breast database from the Surveillance, Epidemiology, and End Results (SEER) Program of the National Cancer Institute [19], which contains follow-up data for over 24,000 breast cancer patients, includes a three-valued input representing nuclear grade for only about 17% of the cases.

This chapter introduces a new, linear programming-based prognosis prediction method. This method is widely applicable to problems involving recurrence or survival data. By applying it to the set of features described in Chapter 2, we show that use of digital morphometry increases the utility of nuclear features for prognosis, with the end result of lessening the reliance upon lymph node status.

3.1 Censored data problem definition

In applying machine learning approaches to prognosis, we first note that this problem does not fit into either of the classic paradigms of classification or function approximation. While a patient can be classified 'recur' if the disease is observed, there is no real cutoff point at which the patient can be considered a non-recurrent case. The data are therefore *censored* [49], in that we know a time to recur (TTR) for only a subset of patients; for the others, we know only the time of their last check-up, or disease free survival time (DFS). In particular, recurrence or survival data is *right censored*, i.e., the right endpoint (recurrence time) is sometimes unknown. Previous work at Wisconsin [91, 96] and elsewhere [16] has framed prognosis as a classification problem by choosing a particular endpoint, such as two years or five years. The more sophisticated work of Ravdin *et al.* is discussed in Chapter 4.

Problems involving censored data are common to several fields; see for instance [41, 49]. In engineering, one might be interested in the survival characteristics of electronic components, while sociologists might consider what factors lead to long-lasting marriages. However, the application of machine learning methods to these problems has been rare. We hope that inductive learning will provide an effective means of prediction using censored data.

3.2 Statistical approaches

There is a significant body of work concerning censored data and survival estimation within the statistics community. While we do not directly compare our approaches with these either quantitatively or qualitatively, brief descriptions of some of the more relevant methods are presented for completeness.

The *survival curve* is a standard representation of survival data in the medical literature. A survival curve plots the probability of survival against time, as time increases from zero. Since these probabilities are not directly available due to the right-censoring of the samples, a standard approximation known as the Kaplan-Meier or product limit plot can be used [42]. Product limit curves can be seen as a special case of the traditional *life-table* estimate [11], in which a large number of samples are grouped into bins based on survival time. The survival probabilities $s(t_i)$ in the Kaplan-Meier curve are computed as follows: For the time of first recurrence, $s(t_1)$ is simply the percentage of non-recurs among all cases with follow-up of at least t_1. At each successive time t_j, the previous survival estimate is multiplied by the instantaneous survival rate. Defining n_j to be the number of cases still at risk at time t_j and r_j to be the number of recurrences at time t_j, the Kaplan-Meier approximation is thus recursively defined as $s(t_1) = \frac{n_1 - r_1}{n_1}$, $s(t_j) = s(t_{j-1})\frac{n_j - r_j}{n_j}$. For an example, see Figure 3.8, where this representation will be used to examine predictive results.

Survival curves can also be constructed in a parametric fashion by fitting a known distribution to the observed cases. For instance, the survival probability can be estimated with an exponential distribution: $s(t) = e^{-\lambda t}$. The parameter λ can then be estimated in a standard maximum likelihood fashion [31]. More relevant are parametric regression methods, in which the distributional parameters are themselves functions of the available input features. See for example Lee [49] and the references therein.

Non-parametric statistical techniques include the Cox proportional hazards model [21] which estimates the *hazard function*, $h(t)$, of time, that is, the instantaneous risk at time t. This regression method assumes that hazard functions for different patients are all proportional, and therefore that the survival functions are powers of one another. A prediction for time to recur (TTR) can then be obtained by finding the median of the expected survival function. By contrast, Buckley and James [15] devised a statistical method which directly estimates TTR. This non-parametric, iterative regression method is the most directly comparable to the approaches we propose.

3.3 Linear programming approach

We approach the prediction of time to recur as a function estimation problem, a mapping of an n-dimensional real input to a one-dimensional real output. The most obvious approach to predicting TTR is to use only the uncensored cases – those for which a TTR is known – and use the input features to fit them with least squares (or least one-norm) procedure. However, by exploiting the straightforward manner in which inequalities are handled in linear programming, we are able to include all available cases to build a more accurate, robust predictive model.

Our solution to this estimation problem is termed the recurrence surface approximation (RSA) technique. RSA uses linear programming to determine a linear combination of the input features which accurately predicts TTR. The intuitive motivation for the RSA approach is that:

- Recurrences actually took place at some point in time previous to their detection. However, the difference between the time a recurrence is detectable (actual TTR) and the time it is actually detected (observed TTR) is assumed to be small.
- Observed disease free survival time (DFS) is a *lower bound* on the recurrence time of that patient.

These assumptions can be formulated into the following linear program for a given training set:

$$
\begin{array}{llrcll}
 & \underset{w,\gamma,v,y,z}{\text{minimize}} & \multicolumn{4}{l}{\frac{1}{m}e^T y + \frac{1}{k}e^T z + \frac{\delta}{m}e^T v} \\
(3.1) & & -v \leq & Mw + \gamma e - t & \leq & y \\
 & \text{subject to} & & -Nw - \gamma e + r & \leq & z \\
 & & & v, y, z & \geq & 0
\end{array}
$$

The purpose of this linear program is to learn the weight vector w and the constant term γ. These parameters determine a recurrence surface $s = xw + \gamma$, where x is the n-dimensional vector of measured features and s is the surface (in this case, a plane in the feature space) which fits the observed recurrence times. Here M is an $m \times n$ matrix of the m recurrent points, with recurrence times given by the m-dimensional vector t. Similarly, the k non-recurrent points are collected in the $k \times n$ matrix N, and their last known disease free survival times in the k-dimensional vector r. The vectors y and z represent the errors for recurrent and non-recurrent points, respectively; overestimating the TTR of recurrences is considered an error, while predicting a TTR which is shorter than an observed DFS is also an error. The objective averages the errors over their respective classes. (Note: e is a vector of 1's of appropriate dimension.) As with MSM-T, the linear program is implemented using the MINOS numerical optimization software [64].

Because recurrences take place at some unknown time prior to their detection, we do not consider underestimated recurrent points to be as serious of an error as overestimated ones. To reflect this, the v term in the objective is weighed by an appropriately small positive parameter δ, forcing underestimated recurrent points closer to the surface. Based on a perturbation theorem [55], for a sufficiently small positive δ, that is $0 < \delta \leq \bar{\delta}$ for some $\bar{\delta}$, the objective minimizes the weighted term conditionally, i.e., of those possible variable values which minimize the first two terms of the objective, those values which minimize the third term are chosen. In this work, the "sufficiently small" value of δ was chosen empirically, by lowering it until further reductions had no effect on the training objective.

One possible variation of the RSA linear formulation is to vary the relative importance of the recurrent and non-recurrent points. The above approach considers the two *sets* of points to be equally important by averaging them separately. One straightforward modification which has been tested is to weigh each example equally, resulting in the following *pooled error* objective:

$$\frac{1}{m+k}e^T y + \frac{1}{m+k}e^T z + \frac{\delta}{m+k}e^T v \tag{3.2}$$

In fact, varying the relative weight applied to the recurrent and non-recurrent points is an easy way to change the predictive characteristics of the resulting generalizer, depending on the application. For instance, if most cases are uncensored, the censored cases may add only slightly more information and could be appropriately weighted with a small multiplier.

3.4 Feature selection

As mentioned in Chapter 1, overfitting avoidance can sometimes be achieved by training on only a subset of the given features. The diagnosis results in Chapter 2 performed feature selection externally to the learning procedure, using a combinatorial search to find a single best classifier. This sections describes a more elegant and automatic solution, which incorporates feature selection into the learning algorithm as a means of parameter elimination. Thus we view feature selection as a means of finding the optimal number of parameters, in the spirit of neural network algorithms such as pruning [82], weight decay [38] and optimal brain damage [48].

Essential to the feature-selection algorithm is the fact that the features have all been normalized to lie in approximately the same range. All ordered features – whether real valued (as with the breast cancer data) or integer valued (as the SEER data) – are "z-transformed" to have mean zero and standard deviation one. This allows the assumption that the relative importance of an individual feature in the predictive model is approximated by the magnitude of the feature's corresponding linear coefficient. Note that this scaling has two useful side effects:

- Outlier feature values (that is, individual values that are extremely large or small relative to the mean value of the feature) are easily identified, since the value reflects the number of standard deviations the original value is from the mean. Outliers may have some semantics; for instance, extremely large values of the "standard error" features computed by **Xcyt** might indicate that a very small number of cells was processed.

- Missing feature values can be safely set to zero. Setting missing values to the mean is a common practice which is particularly appropriate here. Since we are building linear models, a missing value has no effect on the prediction for that case.

The feature-selection procedure is a variation of the heuristic sequential backward elimination method [45], a top-down, greedy search through the space of feature sets. The procedure begins by setting aside a tuning or validation set, that is a surrogate testing set, in our case a randomly selected 10% of the training cases. The regular RSA procedure is then applied to the training set, finding the global optimum of the training objective using all the available features. We then examine the resulting weight vector w, and force the following weights to zero: all those which are non-basic (zero) variables in the LP solution, and the smallest in magnitude among the (usually non-zero) weights represented by basic variables. By bounding the weight value above and below by zero, the corresponding variable is removed from the predictive model. The RSA LP is then re-formulated and a new solution found in this reduced dimension. This elimination of variables is repeated until only one variable remains. For the final predictive model, we choose the feature set which demonstrated the best performance on the tuning. Since setting aside a tuning set can cause significant degradation when using only a small number of samples, we add the tuning set back into the training set, and perform one final optimization step.

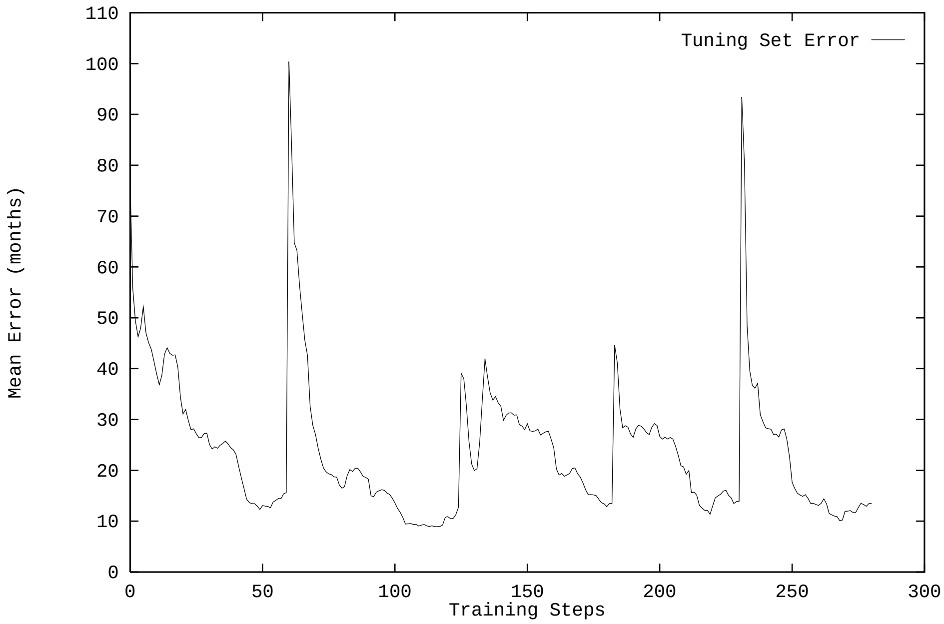

Figure 3.1: The tuning-set error during training with feature-set selection. Each spike corresponds to a dropped feature, while the subsequent drop in error corresponds to re-optimization in the reduced feature space.

Figure 3.1 shows the observed error on the tuning set for a typical run. Each reduction of the problem dimension results in a jump in the error, which is then reduced through the optimization process. However, by carefully utilizing the "hot start" capability of the MINOS linear programming package, these error spikes can be eliminated. Eliminating a variable can be accomplished by setting its upper and lower bounds both to zero. Since only the bounds have changed, the new linear program can be solved starting at the solution to the previous LP. Since the solution in the new space typically lies very close to the previous solution, these re-optimizations are very fast. As shown in Figure 3.2, the number of training iterations is substantially reduced using hot starts.

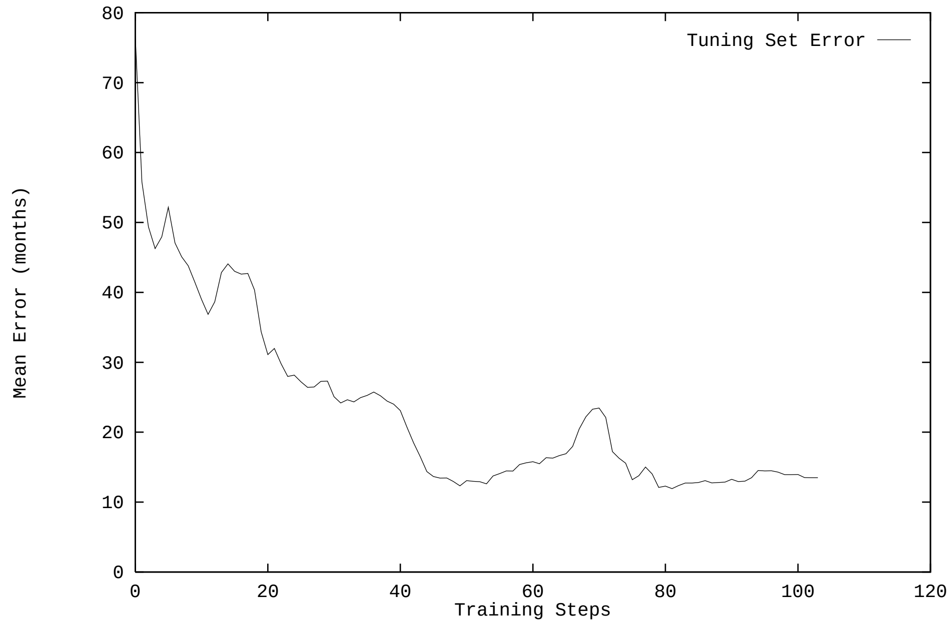

Figure 3.2: The tuning-set error during training with feature-set selection when using hot starts. The error spikes associated with dropping a feature have been eliminated and the number of training steps has been reduced significantly.

We also built this method of choosing an effective feature space into the MSM-T classification procedure. Table 3.1 shows the classification improvement on a closely related task: the two-year prognosis problem, where the positive class includes those cases which recurred within two years and the negative class those which were observed disease-free for at least two years. Note that new tuning sets are allocated for each subproblem in the MSM-T decision tree building procedure.

The feature-selection procedure forces MSM-T to build larger trees, since

	Correctness ± sd	Features	Planes
MSM-T	67.99 ± 3.42	32	2.77
MSM-T with Feature Selection	71.15 ± 2.81	6.44	9.74

Table 3.1: MSM-T results with and without feature selection. Testing results are the average of 50 cross-validation tests, and are reported as correctness plus or minus one standard deviation.

the initial splits do not separate the points as completely as was possible in the higher dimension. However, a more precise measure of the complexity of a decision tree is the total number of parameters, in this case, the number of planes times the average number of features used for the planes. Using this metric, the feature selection causes a 29% reduction in the complexity of the resulting trees when applied to this problem. This results in a 4.6% increase in accuracy and a 32% drop in the observed variance of the test-set correctness measurements. Regarding each cross-validation as a separate trial, the difference of the mean accuracy rates is statistically significant at the 99% confidence level, while the difference in variances is significant at the 90% level.

3.5 Application to prognosis problem

3.5.1 Wisconsin breast cancer prognostic data

We tested the RSA procedure on a data set consisting of 187 malignant breast cancer patients, 44 of which had been observed to recur. These cases represent all those patients for which any follow-up data was available, with the following exceptions:

- Patients with distant metastasis at time of surgery, for whom prognosis is poor.

- Patients with *in situ* cancer, for whom prognosis is very good. *In situ* cancers have not yet invaded the surrounding tissue, and rarely spread.

The input consisted of the 30 real-valued features computed with **Xcyt** along with tumor size and number of positive lymph nodes.

The exact results of a leave-one-out test run using RSA on the Wisconsin data are shown in Figure 3.3. This figure plots the observed TTR or DFS vs. the predicted TTR depending on the type of the example, recur (+) or non-recur ($\Diamond$), along with the line $y = x$ for reference. Ideally, all the recurrent points should lie above but close to the $y = x$ line, and all non-recurrent points would lie below. Figure 3.4 shows the results of using the pooled error RSA shown in Equation 3.2.

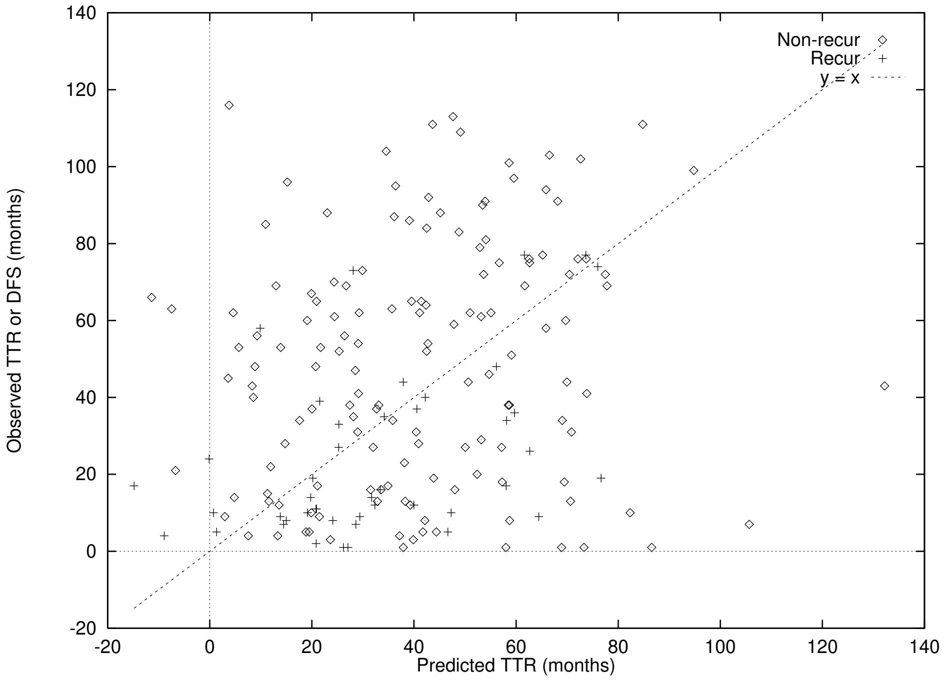

Figure 3.3: Observed TTR or DFS vs. predicted recur time: Original RSA, Equation 3.1.

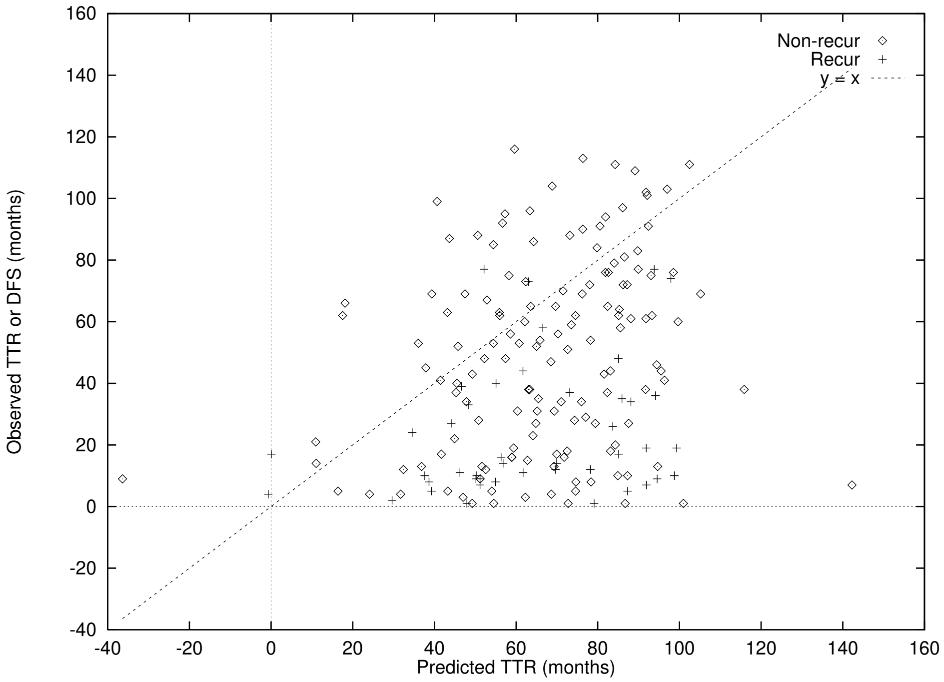

Figure 3.4: Observed TTR or DFS vs. predicted recur time: Pooled error RSA, Equation 3.2.

The tradeoff between these two methods is shown in these two plots. Since the original RSA objective function weighs the recurrent points relatively more, they are fit more closely, at the cost of underestimating the unknown TTR of many of the non-recurrent cases. Conversely, equal weighting of the points fits the non-recurs well (thus lowering the average error of the test points) but generates consistently optimistic predictions for the recurrent points. In this case the less “accurate” predictive method, the original RSA, is probably preferable in a clinical setting.

Further, consider the biases caused by the two classes, recurs and non-recurs, on this predictive method. The unavoidable fact that not all cases were followed to recurrence means that those which have been observed to recur exhibit a bias toward early recurrence. Conversely, forcing the predicted recurrence times to be

	Mean Error	Non-recur Error	Recur Error
Original RSA	21.8	20.1	27.5
Pooled RSA	18.6	10.3	45.7
Original: Selection	18.3	19.9	13.0
Pooled: Selection	13.9	6.1	39.3
Original: Select 4	17.7	19.0	13.6
Pooled: Select 4	14.3	6.2	40.9

Table 3.2: Average error (in months) of RSA formulations: Wisconsin prognostic data.

above the DFS time of the non-recurrence cases biases the predictions upward. In the absence of other information, it is reasonable to weight the two sets of cases equally, thus hopefully canceling these competing biases.

Table 3.2 summarizes computational results of the RSA procedure on this data, using both RSA formulations. The cumulative error is divided into errors in predicting recurrent and non-recurrent cases to show the difference caused by weighting of the cases in the objective function. In addition to the different objectives, 3 different styles of feature selection were tested: no selection, tuning-set selection as described above, and a variation which eliminates features down to a certain predetermined number. In this case, a cross-validation determined that the average number of features required was four. All results reflect leave-one-out testing. Note that in the reporting of these test results, underestimated recurrences are *not* counted as errors, although they are treated as discounted errors in the training objective.

In all cases, eliminating redundant or irrelevant features improved the results significantly. Using cross-validation to determine the number of features in each model may also improve performance, by reducing the variation allowed by tuning with a small subset.

The feature-selection procedure also provides insight into which of the input features is most important for prognosis prediction. Consider the "select four" test run with the original RSA formulation, in which leave-one-out testing results in 187 separate predictive models. The mean radius feature was used in 155 (83%) of those models, followed by worst radius (145), worst area (132) and mean perimeter (129). In fact, the nine size-related variables (of the original set of 32 variables) accounted for 91% of the total feature usage. Clearly, nuclear size is significantly more important for prognostic estimation than any of our representations of shape. It is also interesting to note that the traditional tumor size and lymph node status features were used in none of the 187 tests.

3.5.2 SEER data

Table 3.3 reports similar results for different subsets of the SEER database. These cases contain values five integer-valued input features. Each feature value represents a certain property of the observed specimen. Although the values are coded, they typically follow the "bigger means worse" ordering which was coded into the Wisconsin features.

1. Histological Grade: integer in {1, 2, 3}
2. Tumor Size: integer in {1, 2, ..., 11}
3. Tumor Extension: integer in {1, 2, ..., 5}
4. Number of Axillary Lymph Nodes Positive: integer in {1, 2, ..., 10}
5. Number of Axillary Nodes Examined: integer in {1, 2, ...}

As this database of 44,135 cases is too large for the memory of our workstations, a subset of 1,000 cases was selected by choosing every 44th record. After again eliminating cases with distant metastasis and *in situ* cancer, 890 remained for testing. Because of the small number of features, the feature selection had

	Mean Error	Non-recur Error	Recur Error
RSA: Random Cases	17.2	17.0	18.0
RSA: Histo. Grade Cases	17.6	18.1	16.4

Table 3.3: Average error (in months) of RSA: SEER data.

little effect on the results, so tests without it will not be reported. A second test was performed with just the 7,438 cases which included a value for histological grade. Note that the SEER data is recorded slightly differently, since survival time is recorded rather than recurrence time.

The errors on this larger data set confirm the effectiveness of the RSA method. We have also shown that a precise, detailed description of nuclear grade is enough information to predict prognosis as well as the more traditional features recorded in the SEER study. However, histological grade seems to have little effect on the predictive power of the SEER model. This we attribute to the small amount of information available in an objectively graded, three-valued feature.

3.5.3 Gutenberg data

We also evaluated the RSA procedure on a third data set from the group at Gutenberg, Germany [62]. These cases contain 13 integer and coded features: age at diagnosis, menopausal status, histologic grading, number of lymph nodes removed, number of lymph nodes involved, tumor size, estrogen receptor status, progesterone receptor status, immunohistologic overexpression of erbB-2, urokinase-like plasminogen activator, proliferation rate via ki67 antigen, plasminogen activatou inhibitor (PAI-1), and Cathepsin D. Computational results with these cases are summarized in Table 3.4.

First, these positive results further validate the use of the RSA method for

	Mean Error	Non-recur Error	Recur Error
Original RSA	6.36	5.56	9.08
Pooled RSA	5.70	2.79	15.61
Original: Selection	8.01	7.88	8.46
Pooled: Selection	5.68	2.43	16.73
Original: Select 5	7.43	7.28	7.91
Pooled: Select 5	5.91	2.56	17.28

Table 3.4: Average error (in months) of RSA formulations: Gutenberg data.

prognosis prediction. However, adding the feature selection procedure did not help in this case, as the risk of eliminating useful features outweighed the benefit of overtraining avoidance.

The average errors are significantly lower here than with the other data sets. However, we attribute this less to the utility of the Gutenberg features than to the relative homogeneity of the cases. No non-recurrent cases with follow-up time less than 24 months were included. Further, 66% of the recurrence times and 94% of the follow-up times are between two and six years. Hence, this prediction task was significantly easier than the other two explored.

The feature usage on the "select five" case also supports the above conclusion. The most prevalent feature in the 141 testing cases was histological grade, which was selected 97% of the time. Other relevant features were involved lymph nodes (chosen in 87% of the tests) and estrogen receptor status (72%). Since the other two data sets also involved some indication of nuclear grading and lymph node status, we conclude that the new features such as the hormone-related measurements were not the source of the improved results.

3.6 Comparative results

In order to evaluate the effectiveness of the recurrence surface approximation method, it was compared to other learning methods on the Wisconsin data set. Figure 3.5 shows the mean generalization errors of the original RSA compared against the following prediction methods:

- **Pooled error RSA:** All of the feature examples were weighted equally in the objective. See Equation 3.2.

- **Modified backpropagation:** We also evaluated an artificial neural network (ANN) using a modified version of back-propagation [78]. The output unit for our ANN used the identity function as its response rather than the familiar sigmoid, allowing any real-valued output. The error function was also changed to the one-sided errors as used in the RSA; learning took place only on underestimated non-recurrent cases and overestimated recurrent cases. A tuning set was used for early stopping to avoid possible overtraining. Several different network architectures were tested, and the reported results represent the best results, from a network with two hidden layers of ten nodes each. We do not consider the ANN results a "strawman" used solely for comparison purposes but rather an extension of the RSA concept to a different prediction method.

- **Least 1-norm error on recurs:** An obvious method for predicting recurrence is a regression procedure which minimizes the average error on just the recurrent cases. For compatibility, this test was run using the greedy feature-selection method.

- **k-nearest neighbors:** The nearest neighbor procedure [1, 35] is another effective and intuitive method for generalization. In this application, the recurrence time for the k recurrent cases closest to the given test point in Euclidean space were averaged to give a prediction. We tested values of k between one and ten, with both scaled and unscaled features. In order to simulate feature selection, the k-nearest neighbor algorithm was also tested using only the six features found most relevant by RSA (see previous section). The best results, which are reported in Figure 3.5, were obtained using all of the unscaled features, and k equal to seven.

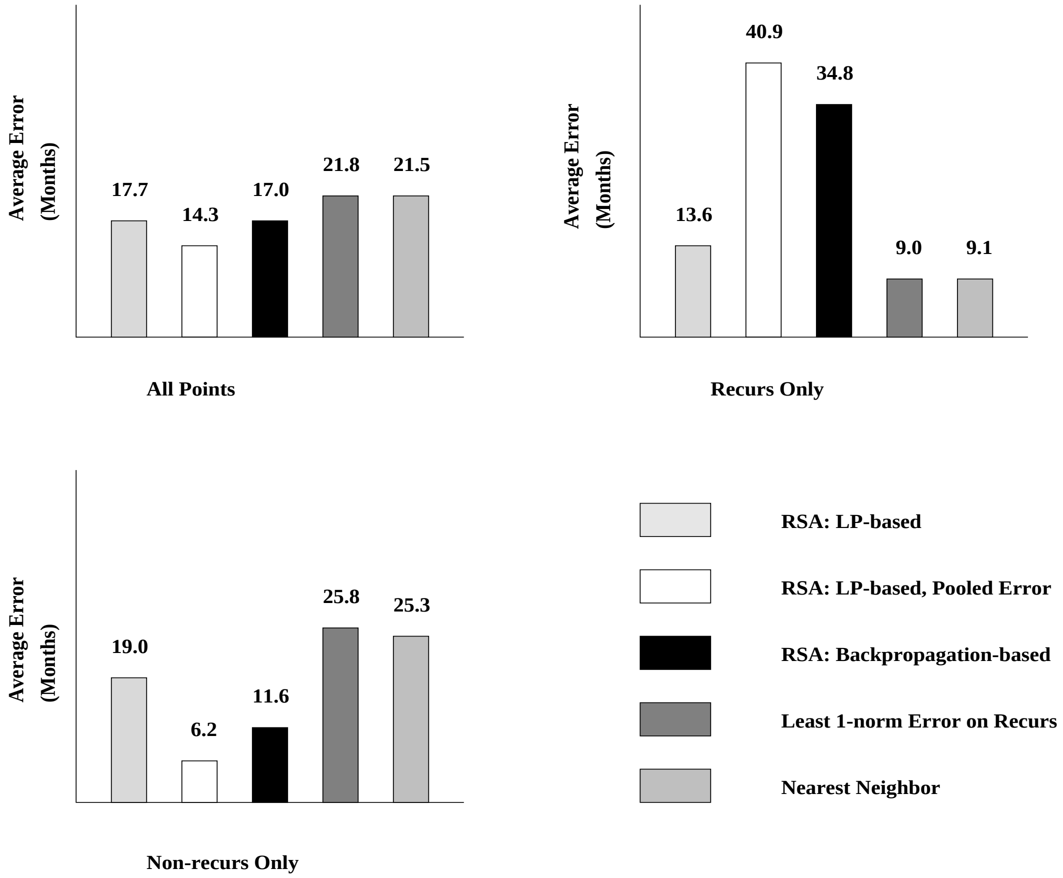

Figure 3.5: Average errors for different prognostic predictors.

Comparative results on all points, recurrent cases only and non-recurrent cases only are reported for the various prediction methods. We note again that the original RSA formulation performs comparably on both recurrent and non-recurrent cases, while the pooled error and backpropagation methods favor the majority non-recurrent class, thereby lowering their mean error. The highly nonlinear artificial neural network predictions were comparable in quality to those made the the RSA method, indicating that nonlinearity adds little to predictive ability on this particular problem. Still, nonlinear extensions to RSA are explored in the next chapter. The results for least-1-norm estimation are improved by the relatively short time of the study. Most of the observed recurrences were at relatively short times, therefore a regression method which uses only the recurrent cases skews the predictions downward. The nearest-neighbor predictor also benefitted from this bias.

3.7 Medical interpretation

Individual predictions of recurrence (or survival) time can be a significant contribution to the planning of treatment for cancer patients. However, because of the limited amount of predictive power in the available prognostic features, the predictions shown in Figures 3.3 and 3.4 exhibit a significant amount of variance; the recurrence prediction problem has certainly not yet been solved. Therefore, it is important to present the group characteristics of patients with a particular predicted recurrence time. Two approaches to this representation problem are discussed.

3.7.1 Fixed-time prediction densities

The first method addresses recurrence probabilities at a particular point in time. For instance, what were the RSA predictions for patients who recurred in less than two years, compared to those who were disease-free for more than two years? To answer this question, we first divide the training examples into the two classes *recur* and *non-recur* using a particular endpoint (e.g., two or five years), as is done when treating prognosis as a separation problem. Histograms are then built to count the number of cases in each class which were predicted to recur at all possible times. For clearer presentation, we again use a Parzen window kernel technique [67] as in the diagnosis probability density estimations in Chapter 2. Results for endpoints of two years and five years are shown in Figures 3.6 and 3.7.

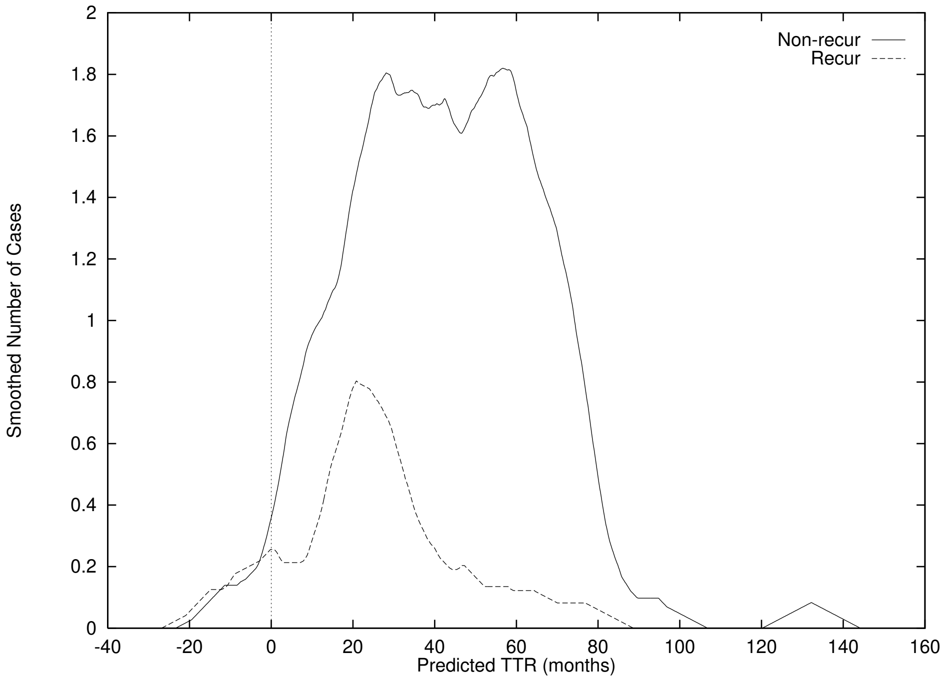

Figure 3.6: Number of non-recur and recur cases at two years versus the RSA feature-based prediction of time to recur $xw + \gamma$.

Among those patients who will recur in less than two years, most of the RSA predictions were suitably pessimistic, clustering around the two year mark. However, among all patients, the majority will not recur by two years, no matter what the RSA prediction.[1] For the five year recurrence case, note that for nearly

[1]Negative prediction for time to recur are not disallowed by the RSA model. This seeming

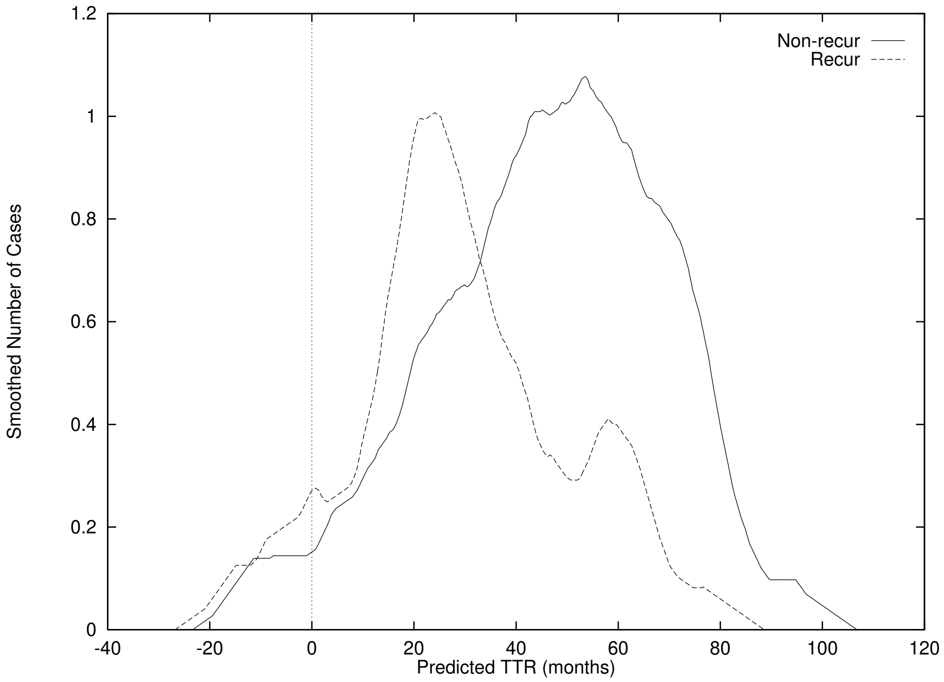

Figure 3.7: Number of non-recur and recur cases at five years versus the RSA feature-based prediction of time to recur $xw + \gamma$.

all predictions of less than three years, recurrences are the predominant class. Hence, an RSA prediction of three years or less translates to a poor prognosis at five years. There is also a smaller peak of recurrent cases clustered around the five year prediction, indicating that several of the RSA predictions were very nearly accurate.

irregularity could have been prevented with an extra non-negativity constraint in the RSA linear program. However, it was deemed more important to emphasize the relatively poor prognosis of these patients.

3.7.2 Survival curves

The recurrence predictions made by RSA can also be interpreted using survival curves. Again using the leave-one-out testing data from Figure 3.3, the cases were divided into three groups based upon their predicted time of recurrence: predictions of less than two years (54 cases), predictions from two to five years (98 cases), and predictions greater than five years (35 cases). The actual recurrence probabilities of these three groups were then constructed with the Kaplan-Meier approximation, as shown in Figure 3.8.

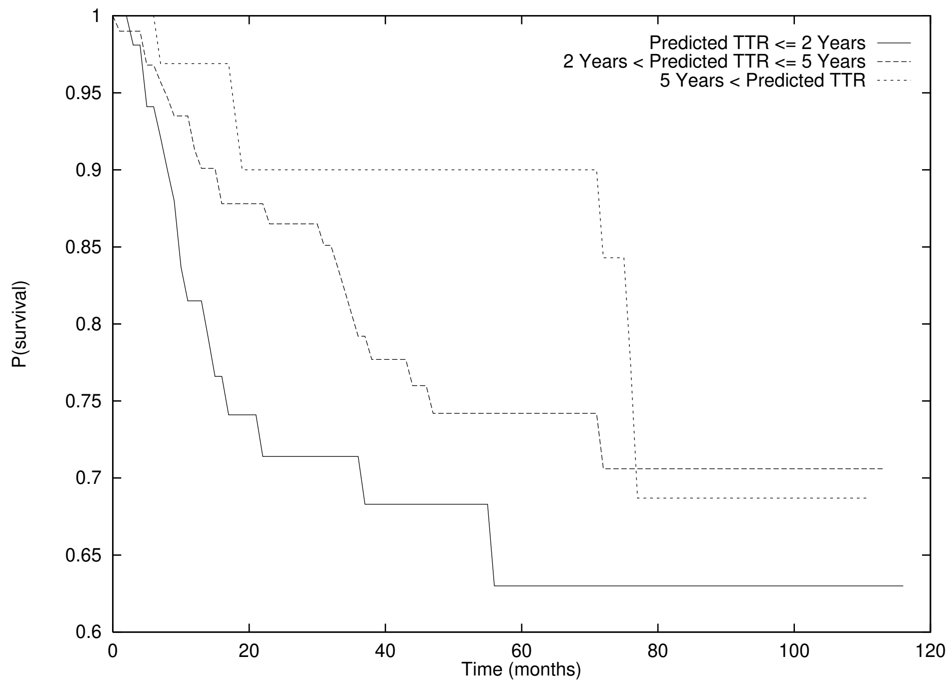

Figure 3.8: Survival probabilities based on RSA predicted TTR.

This plot clearly shows that the RSA predictions for these cases was consistent with the actual prognosis. Those examples for which the prediction was less than two years have the worst prognosis for the duration of the study, and a recurrence rate of nearly 30% at two years. Those with predicted TTR's of greater than five years have an extremely good prognosis, with only 10% recurrence up to five years, but with more recurrences being observed (as expected) after that time.

There are a number of points to be made about these different representations of the RSA method. First we note that the two styles of plots show similar yet complementary information. The fixed-time density curves show all of the predictions for a given "snapshot" in time, while the survival curves follow time throughout the study and group similar predictions together. The survival curves could in fact be generated from a series of density plots by integrating both the recur and the non-recur curves over the appropriate interval and taking the ratio of the non-recur result to the sum of the two.

Figure 3.8 also suggests the ability to generate customized survival curves for each new patient. Once the input features have been collected and a prediction has been made, a survival curve can be plotted using those training cases for which a "similar" recurrence prediction was made. This addresses the issue of specific, customized recurrence probability estimates for each patient, by making comparisons relative to the predictive model. This could be easily done by including only those cases in a certain interval around the current prediction – e.g., for a prediction of two years, a survival curve could be constructed using those points predicted to recur between one and three years. This subset would consist of all cases between 12 and 36 months in Figure 3.3. Another approach would be to construct survival curves based on some number of nearest neighbors of the current prediction.

This capability could have a significant impact on treatment planning. One widespread use of survival curves is that of comparing different types of treatment: randomized tests are performed on the different treatment modalities and the various survival expectations examined. Consider an RSA predictive model which includes the presence or absence of a particular type of treatment as an input feature. A patient could then be shown two individualized survival curves based on his or her expected recurrence with or without the treatment. By combining all the available features in the predictive model, the RSA approach is able to specifically address interactions which may not become obvious in randomized testing. As a hypothetical example, a large tumor might be shown to neutralize the effectiveness of extensive chemotherapy, giving both the doctor and the patient more information to use in quality-of-life vs. expected-survival decisions.

3.8 Extensions

A thorough comparison of the RSA method with the standard statistical practice, the Cox proportional hazards model, will be performed. As mentioned, recurrence time predictions are obtained from the Cox survival curves by choosing the median time, that is, the time at which the survival probability equals 0.5. This comparison will utilize simulated data in order to resolve questions about error computation (all recurrence times will be known accurately) and to remove any biases which may be present in observed data.

The predictive power of the recurrence surface approximation method is limited somewhat by the fact that the resulting predictive surface is linear. Several methods for adding nonlinearities are discussed in the next chapter. Another possibility is the addition of nonlinear features to the predictive model, in the form of reciprocals ($\frac{1}{x}$), cross-terms ($x_1 x_2$), etc.

Chapter 4

RSA variations

The recurrence surface approximation (RSA) method described in the previous chapter is a robust, effective way to predict recurrence or survival times using right-censored data. However, linear predictive models – that is, models in which the output is a linear combination of the inputs – are inherently limited in their predictive power. In many cases, the underlying function being approximated by the learning technique has nonlinear components. Several methods of "de-linearizing" the RSA procedure have been implemented and tested and are described in this chapter. These methods include the following:

- **Feature-space partitioning methods:** These methods divide the feature space based on the the predictive performance of the RSA on the various training points. New cases can then be handled differently based on whether we expect the RSA to predict their time to recur (TTR) well. We examine two different methods of using this information:

  1. **Multiple RSA:** In the first method, a new RSA is trained using only the points expected to be in error using the first RSA.
  2. **Augmented RSA:** The second method adds an auxiliary feature to each example, and sets the value of this feature based on the RSA's expected performance. A new RSA is then trained in this augmented

space.

- **Stacked RSA:** This method is an adaptation of Wolpert's Stacked Generalization algorithm [99] and uses cross-validation to estimate the errors of the RSA procedure and attempts to correct them with a second-level predictor. The first level may consist of a single RSA or several separately trained RSA's.

- **Implicit RSA:** This approach adds time as an input feature and builds a separation surface in this augmented feature space. This surface can then be used to predict TTR for new cases.

4.1 Partitioning the Input Space: Multiple RSA and Augmented RSA

The first set of methods uses the idea of partitioning the set of training examples by constructing separating surfaces in the feature space. In this manner, examples with different, identifiable properties can be handled separately. Specifically, those points which are expected to be predicted poorly by the RSA method are treated differently. We consider several variations of this idea in this section.

4.1.1 Multiple RSA (MRSA)

This method divides the feature space based on the results of a preliminary RSA fit. We then evaluate the training examples in the context of the approximating surface, and construct a decision tree to separate points that are well fit by the RSA from those that are not. The feature space is then partitioned via MSM-T into two categories and a new, separate RSA is constructed for the class of examples that were poorly fit by the original RSA. This training procedure is depicted in Figure 4.1. This effectively adds a nonlinearity, in the form of a linear threshold or step function, into the predictive model. This approach can also be seen as a way of learning the errors made by the original RSA, as a new predictive surface is constructed specifically for hard-to-fit cases.

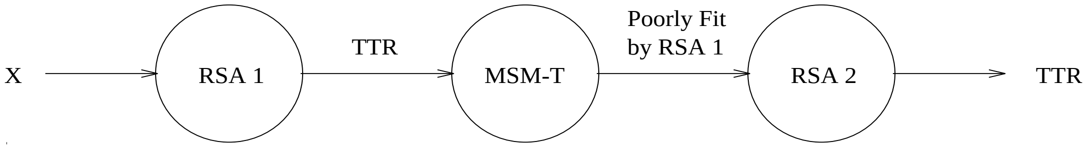

Figure 4.1: MRSA training. Both multiple RSA variations begin training by constructing an RSA. A decision tree is then constructed to recognize points which are this RSA predicts poorly. A new RSA is built using only these points.

Results will be reported on two variations of this approach.

1. First, a classifier is trained to learn the set of points that were fit poorly by the RSA, that is, underestimated non-recurrent cases and overestimated recurrent cases. A separate RSA is then trained for points classified in this region. When generalizing, we use the prediction of one of the two RSAs, depending on the classification of the new case. Figure 4.2 is a graphical depiction of the prediction process.

2. This second variation is the same as the first, except that the second RSA is treated as an error correction. New cases classified in the "poorly fit" region have the original prediction averaged with the second one. See Figure 4.3.

Computational results for these two approaches are shown in Table 4.1. All of these errors are the average of five tenfold cross-validation tests. Results of the original RSA using this testing methodology are somewhat different than those reported in the previous chapter, and are shown here for reference. While

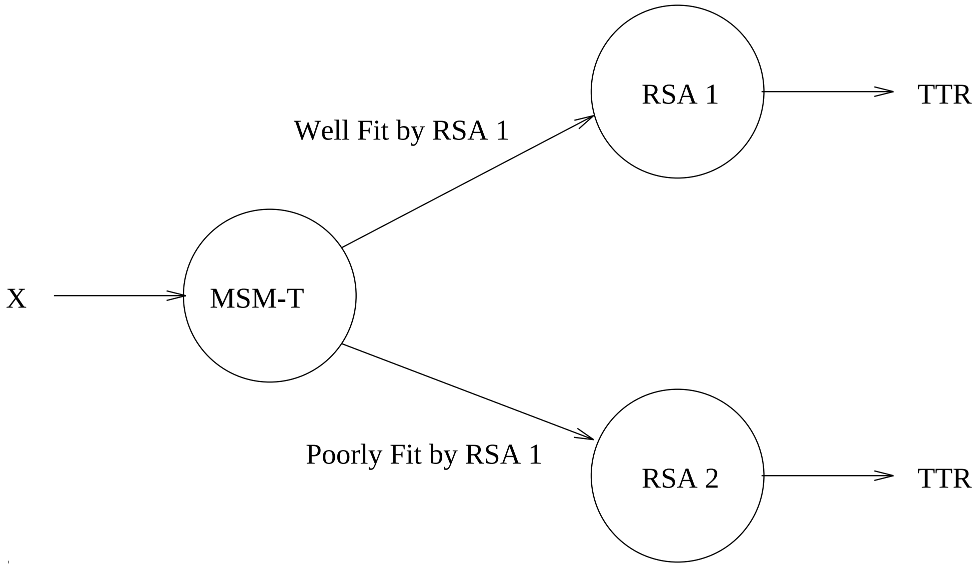

Figure 4.2: The first variant of the Multiple RSA method uses a decision tree to separate out cases which are unlikely to be predicted well by the original RSA surface.

some of the individual cross-validation runs show significant improvement, the generalization performance in general did not improve over RSA. There are two factors contributing to this performance:

1. Very small learning sets were being used for the auxiliary generalizers, giving the RSA too few cases to make good predictions. Better performance can be expected using larger training sets.

2. These methods depend on the implicit assumption that "predictable" and "unpredictable" points can be differentiated in the feature space. The generalization correctness of the classifiers used in the architectures of Figures 4.2 and 4.3 was only around 55%, about the same as the size of the majority class. Hence, points were often "corrected" when no correction

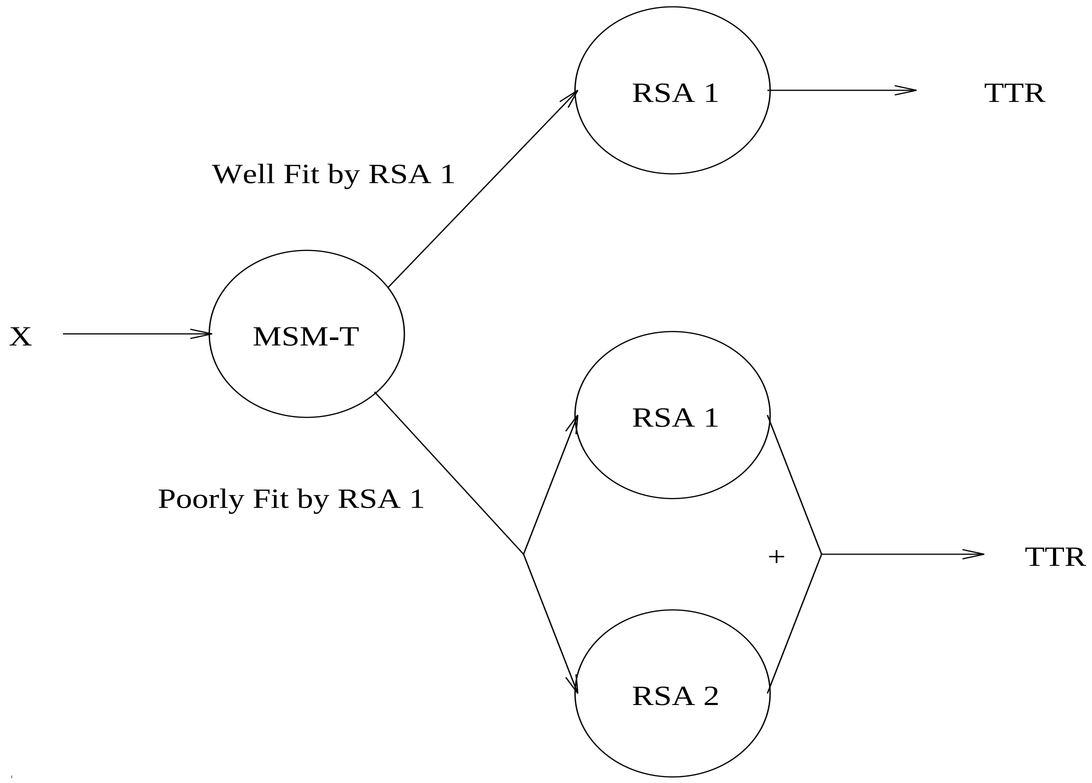

Figure 4.3: The second variant again attempts to separate out the cases for which the RSA predictor will perform poorly. However, the second RSA is this time used as an error corrector, with the final prediction being a linear combination of the two RSA predictions.

was necessary.

A third method of partitioning the feature space was also tested. In order to improve classification, the two error cases (overestimated recurrent cases and underestimated non-recurrent cases) were learned separately, using two different decision trees. New predictive surfaces were then trained on each of the two error classes. While the classification was improved, the overall predictive performance of the method was significantly worse than RSA alone. This emphasizes the importance of having both recurrent and non-recurrent

	Mean Error	Non-recur Error	Recur Error
MRSA 1, Eq. 3.1	23.1	21.6	27.8
MRSA 1, Eq. 3.2	27.6	22.2	54.9
MRSA 2, Eq. 3.1	20.3	19.9	21.4
MRSA 2, Eq. 3.2	21.4	14.5	43.8
RSA, Eq. 3.1	24.9	22.6	32.4
RSA, Eq. 3.2	20.6	11.7	49.6
RSA + Feat. Select., Eq. 3.1	19.3	19.7	18.3
RSA + Feat. Select., Eq. 3.2	15.4	7.2	42.0

Table 4.1: Average errors for multiple RSA formulations, using both original error function (Equation 3.1) and pooled error (Equation 3.2) with feature selection. Reported results are the average of five 10-fold cross-validation tests. Results from the original RSA are somewhat different using this testing methodology and are therefore included for reference.

cases in the RSA training set. The errors of the RSA objective are defined to force a trade-off between the quality of predictions for the two classes. This trade-off is lost when training with only one type of example. If an RSA is trained with only recurrent points, the surface is severely biased downward in the time dimension, resulting in overly pessimistic predictions. Conversely, training with only non-recurrent cases forces the surface up, hence biasing the predictions to be overly optimistic.

4.1.2 Augmented RSA (ARSA)

Another approach is to add auxiliary binary features to each of the of the examples. A value of 1 for an auxiliary feature will designate a difficult case for an RSA and 0 a simple case. Three tasks are performed to train this type of predictor, as shown in Figure 4.4:

1. An initial RSA is trained to predict TTR for all the examples.

2. Based on the performance of the RSA, the feature space is separated by one or more MSM-T decision trees.

3. A new RSA is trained to predict the TTR with the added auxiliary feature(s).

Once the above procedures are carried out, a new case is processed first by the MSM-T to determine the value of the auxiliary feature, then the example is processed by RSA 1 to give a predicted TTR.

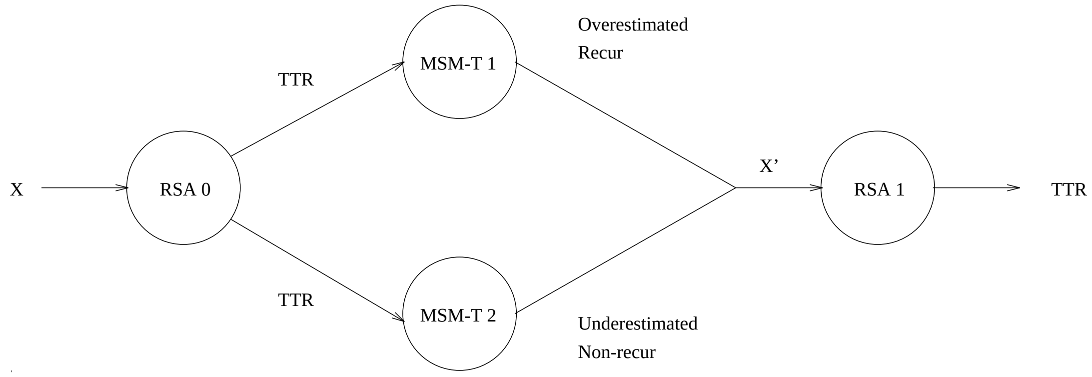

Figure 4.4: Augmented RSA training. Cases are again categorized into classes based on whether or not we expect RSA 0 to predict their TTR well. The two types of possible errors, overestimated recurrent cases and underestimated non-recurrent cases, are learned separately. This information is then coded as two binary input features which are added to the original features, producing augmented feature vector X'. A final RSA 1 is then constructed in this augmented space.

The main purpose of this procedure is to avoid the possibility of poor generalization due to very small training sets, at the cost of slightly reduced generality. Adding an extra binary feature effectively lets the RSA learn two parallel planes in the original feature space, with orientation determined by the coefficients of the original features and an offset determined by the coefficient of the auxiliary binary feature.

We also note that learning the two error classes (overestimated recurs and underestimated non-recurs) separately is more useful here than learning them together. The relatively poor generalization achieved when learning the error cases together renders the extra feature irrelevant. Conversely, by separating the overestimated recurrent cases from all others, and the underestimated non-recurrent cases from all others, we obtain better generalization and hence more relevant auxiliary features. The prediction method is depicted in Figure 4.5.

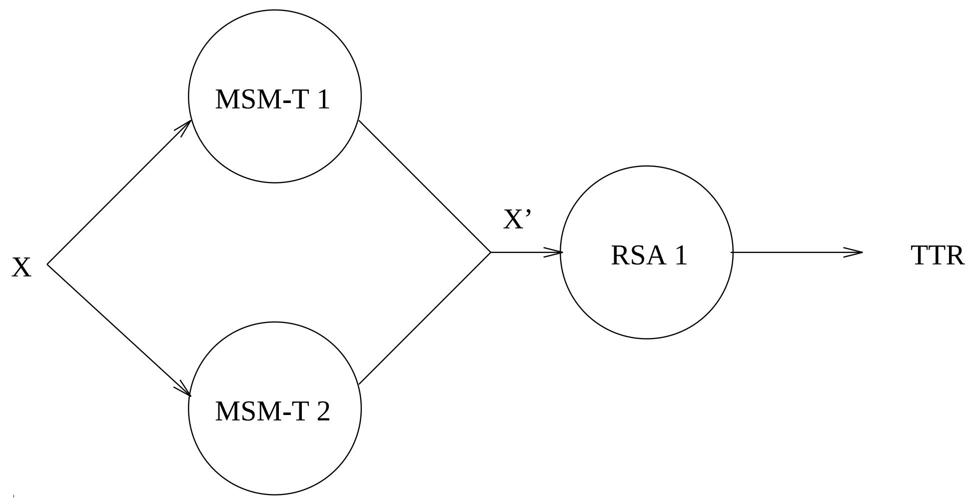

Figure 4.5: Augmented RSA testing. The new case is tested with both of the classifiers to determine values for the auxiliary features. The final TTR prediction is then performed on the augmented feature vector X'.

This method adds a slight complication: Should the newly created features be subject to the feature-elimination procedure? One could leave them at the mercy of the tuning set, or at the other extreme, insist that they be used in the final generalizer. The results reported in Table 4.2 take a compromise position; if k features are used in the original RSA, the feature space of the augmented RSA is pared down to k + d features, where d is the pre-specified number of new features added. In practice, this method always included at least one, and usually both, of the auxiliary features.

The reported results use two binary features, one for each type of possible error. This improves the classification performance to between 3 and 7 percentage points above the majority class. Despite the improved generalization, the average error is again no better than the original RSA formulation, with most of the degradation occuring because of extremely poor predictions on just a few points.

	Mean Error	Non-recur Error	Recur Error
ARSA, Eq. 3.1	26.6	21.0	44.6
ARSA, Eq. 3.2	19.8	12.6	43.6
RSA, Eq. 3.1	24.9	22.6	32.4
RSA, Eq. 3.2	20.6	11.7	49.6
RSA + Feat. Select., Eq. 3.1	19.3	19.7	18.3
RSA + Feat. Select., Eq. 3.2	15.4	7.2	42.0

Table 4.2: Average errors for the augmented RSA formulation, using both original error function (Equation 3.1) and pooled error (Equation 3.2) with feature selection. Reported results are the average of five 10-fold cross-validation tests. Results from the original RSA are included for reference.

4.2 Stacked RSA (SRSA)

This experiment evaluates several variations of Wolpert's Stacked Generalization method [99]. The idea is to use a second-level generalizer (level 1, in Wolpert's terminology) to learn the errors being made by one or more initial generalizers (at level 0). To accomplish this, each training phase consists of cross-validating the original level 0 training set and recording the predictions on the left-out points. These predictions are then used to construct a training set for the level 1 generalizer.

In the first version of stacked generalization, there is only one level 0 generalizer, in this case an RSA. The errors made at level 0 are then used as a level 1 training set. When predicting new points, some percentage of the error predicted at level 1 is added to the level 0 prediction of TTR. In this domain, the error signal (level 1 training signal) was computed as predicted TTR - (DFS or TTR). If the level 1 generalizer is able to correct this error, it improves the predictions of the recurrent cases while having no detrimental effect on the non-recurrent cases. The generalization errors were estimated at level 1 with a least-1-norm linear fit. Prediction is done on new points by adding half of the predicted error to the predicted TTR. Figure 4.6 depicts the prediction process.

Feature selection was performed in the following manner. During the individual runs of the training cross-validation, the number of features (f) chosen by the tuning set method was recorded. At the end of the cross-validation, the whole training set is used to create the final level 0 generalizer. This RSA was pared down to exactly $\overline{f}$ features, where $\overline{f}$ is the average of the individual f's observed during the cross-validation.

Computational results for this approach are shown in Table 4.3. When feature selection is done on the individual RSA runs, we see no improvement in the generalization performance. However, with no feature selection (i.e., all 32 features used for each RSA) the stacked approach shows fairly substantial error reduction. This approach to learning the generalization errors apparently

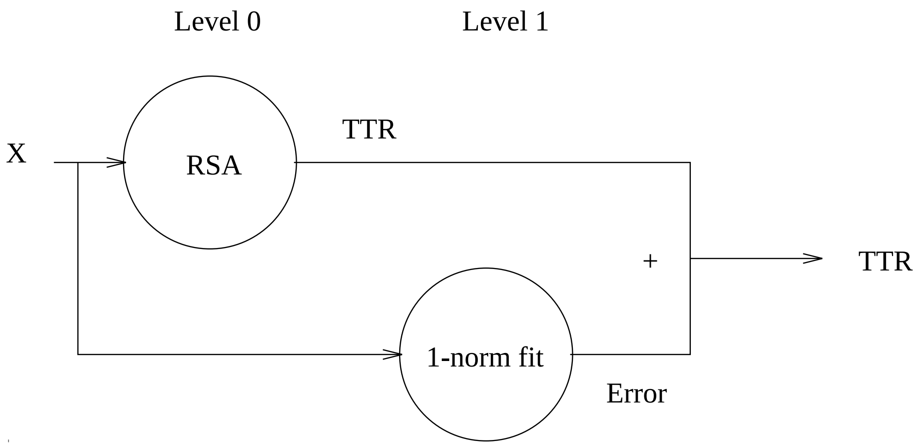

Figure 4.6: Version 1 of stacked RSA (SRSA). Level 0 consists of an RSA predictor constructed in the usual manner. The input cases are cross-validated and the prediction errors recorded for use as the level 1 training outputs. The input for level 1 consists of the same features used at level 0, and the training method is a simple least 1-norm error fit. New cases are predicted by adding one-half of the predicted error to the predicted TTR.

eliminates about the same amount of error as the feature-elimination process.

The second version of stacked generalization uses a number of predictive models at level 0. These can be constructed in several ways, for instance, they could reflect different training methods (neural networks, regression trees, etc.), or they could use the same training method on different subsets of the training examples. During the initial cross-validation, each level 0 generalizer records the predictions of the testing cases. These predictions are collected and used as the input features to the level 1 generalizer, which uses them to predict TTR. Hence, each level 1 training example has one input feature for each level 0 predictor.

	Mean Error	Non-recur Error	Recur Error
SRSA, Eq. 3.1	20.3	18.6	26.0
SRSA, Eq. 3.2	18.2	13.0	34.8
SRSA + Feat. Select., Eq. 3.1	20.3	19.7	22.2
SRSA + Feat. Select., Eq. 3.2	17.5	12.7	33.2
RSA, Eq. 3.1	24.9	22.6	32.4
RSA, Eq. 3.2	20.6	11.7	49.6
RSA + Feat. Select., Eq. 3.1	19.3	19.7	18.3
RSA + Feat. Select., Eq. 3.2	15.4	7.2	42.0

Table 4.3: Average errors for the single-generalizer version of stacked RSA (SRSA). Reported results are the average of five 10-fold cross-validation tests. Results from the original RSA are included for reference.

The implementation of stacked generalization reported here creates the training sets for the level 0 RSA predictors by dividing up the available input features. Each RSA is presented with a random, fixed-size set of features from all of the training examples, with each feature being allocated to exactly one generalizer. This method was preferred over dividing the training examples because of the relatively small size of the training set. The level 0 predictions are used to predict TTR at level 1 in the usual RSA fashion. Experiments were also run using a slight variation of RSA in the level 1 generalizer, restricting the predictive model to be a convex combination of the inputs, i.e., the sum of the feature weights w_i was forced to equal one. See Figure 4.7. Put another way, the level 0 RSA 1, RSA 2, etc. produce their own, local predictions; these local predictions are then combined by the level 1 RSA k + 1 to do a global prediction. Because of the small size of the feature sets, the feature-selection mechanism was not used for these experiments.

Computational results for this second version of stacked RSA are shown

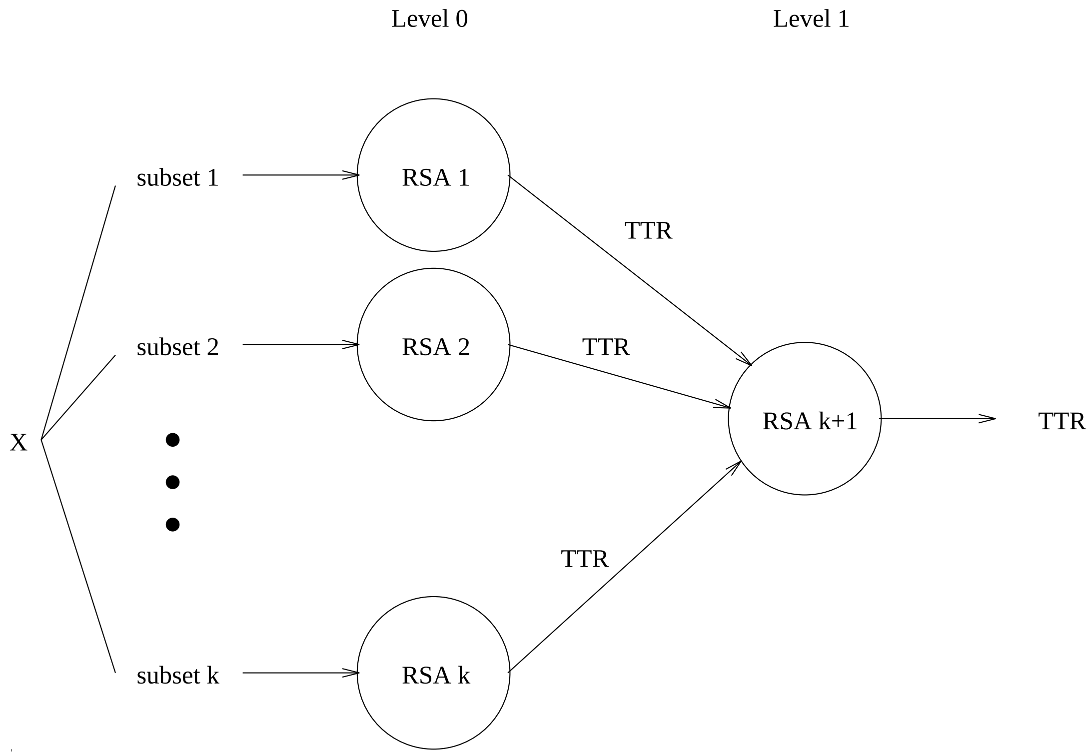

Figure 4.7: Version 2 of stacked RSA (SRSA). The input features are randomly partitioned into sets of three. Each level 0 RSA uses one set to predict TTR. These predictions are gathered and used as input features to the level 1 RSA, which provides the final prediction of TTR.

in Table 4.4. By itself, the stacking approach did not improve the observed generalization. However, significant error reduction was obtained by restricting the final prediction to be a convex combination of the level 0 predictions. This simple restriction reduces the possibility of extremely large errors, and was also more consistent across the various test runs. However, the predictions also tended to cluster around the mean, making it harder to predict the extreme cases (i.e., very early recurrence or long disease-free time).

	Mean Error	Non-recur Error	Recur Error
SRSA 2, Eq. 3.1	22.9	21.7	26.8
SRSA 2, Eq. 3.2	18.5	10.1	45.6
SRSA 2, $\sum_i w_i = 1$, Eq. 3.1	18.5	18.9	18.4
SRSA 2, $\sum_i w_i = 1$, Eq. 3.2	13.9	5.8	40.0
RSA, Eq. 3.1	24.9	22.6	32.4
RSA, Eq. 3.2	20.6	11.7	49.6
RSA + Feat. Select., Eq. 3.1	19.3	19.7	18.3
RSA + Feat. Select., Eq. 3.2	15.4	7.2	42.0

Table 4.4: Average errors for the multiple-generalizer version of stacked RSA (SRSA). Reported results are the average of five 10-fold cross-validation tests. Results from the original RSA are included for reference.

4.3 Implicit RSA (IRSA)

Another interesting approach to predicting time to recur is termed the implicit recurrence surface approximation (IRSA). Here a separation methodology is used in the space of $features \times time$, with the result again being a plane which can be used to predict recurrence. The data manipulation used here builds upon the work of Ravdin and colleagues [23, 74, 75], who use a neural network to learn survival curves.

Before constructing the predictive surface, the training examples must be pre-processed in order to successfully frame the prognosis problem as one of separation. Time is added as an input feature, and given values along the range of follow-up times in the study. Each case thus produces a number of new training examples which are identical except for the time feature. In this case, a new training case is created for each six month interval, producing cases with time equal to 6 months, 12 months, etc. up to the longest known follow-up time. For any particular time, the training case can be given a classification of recur (R) or non-recur (N), based on the patient's status at that point in time. This is depicted graphically in Figure 4.8 for two samples, 1 and 2. The non-recurrent sample 1 with DFS time in the fifth time interval would produce four new training examples, all with the N classification. Nothing is known about this case for any later times, so no further examples are created. The recurrent case similarly produces five N cases up to the recorded TTR. For all time intervals beyond that, the same features produce a case with classification R.

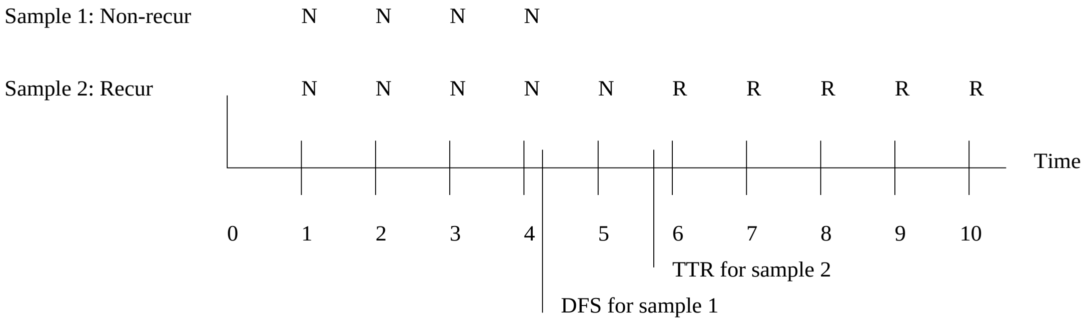

Figure 4.8: Pre-processing used to create the training set for the implicit RSA classification. Samples that have not recurred generate non-recur (N) points up to the DFS time. Samples that have recurred again produce non-recur points up to their TTR, and recurrent points (R) for all later times.

As a result of this pre-processing, there are now a significantly increased number of cases which can be separated using any classification procedure. Ravdin uses these cases (after some further pre-processing) as input to a neural network which learns to separate recur (with output training signal 1) from non-recur (trained with 0). For new cases, the response of the ANN output unit is interpreted as a probability of recurrence. Survival curves can then be constructed by varying the time feature. We use instead the MSM-T separation procedure to directly predict the time of recurrence. First, a single separating plane (the "implicit" recurrence surface) is constructed between R and N cases in the $features \times time$ space. For a new case, consider holding all the features fixed and varying time. The point at which this line meets the separating plane is the value of time for which this particular case would go from being classified "non-recur" to being classified "recur". Hence, that time can be interpreted as a predicted TTR. See Figure 4.9. Specifically, we solve the equation

$$TTR = \frac{\gamma - \sum_i w_i x_i}{w_{n+1}}. \tag{4.1}$$

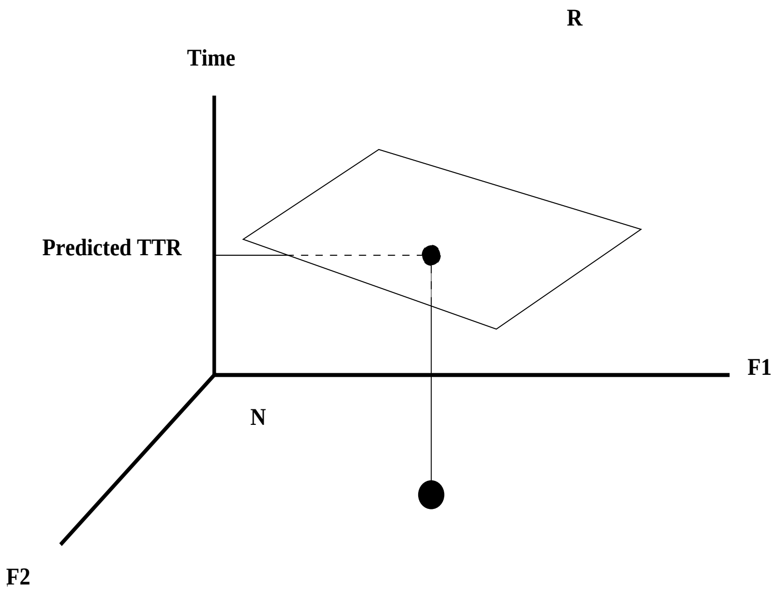

Figure 4.9: Predicted TTR's are generated by first constructing a separating plane in the augmented $features\times time$ space, here represented by features F1, F2 and time. For a new case with particular F1 and F2 values, we solve for time in the planar equation. This is the predicted time of recurrence.

The implicit RSA procedure can be given more predictive power by adding nonlinear functions of time as new input variables. In this experiment, we added $time^2$, $\frac{1}{time}$, and $\frac{1}{time^2}$. This adds several complications:

- The predicted TTR now requires finding the root of a nonlinear set of equations. This was handled with a safe Newton-Raphson method, as implemented in [69]. However, the resulting root may or may not be unique. This was handled in the following experiments by beginning the search for roots near the median recurrence time and working outward to both extremes. The first root located in such a fashion was used at the predicted TTR.
- The feature-selection method is compromised by the fact that none of the time variables are guaranteed to be kept. This suggests that the time features should be exempted from possible elimination. However, in practice we found that at least one, and often more, of time features were retained, indicating that they are indeed providing the added predictive power that was expected.

Computational results for the two methods are shown in Table 4.5. Because of the way the MSM-T method handles errors, these results are comparable only to the RSA results using the objective in Equation 3.2. We note that these results are qualitatively very similar to those produced by RSA, and that the nonlinear extension significantly improved the predictive power.

4.4 Discussion and extensions

This chapter presents a number of different methods for adding nonlinearity to the recurrence surface. The first two methods, multiple RSA and augmented RSA, divide the training data based on the expected accuracy of a linear RSA. While this approach seems sound, these approaches did not improve the overall accuracy on the Wisconsin prognostic data, partially due to the small size of the

	Mean Error	Non-recur Error	Recur Error
IRSA	19.2	11.5	44.3
Nonlinear IRSA	15.3	7.1	41.8
RSA, Eq. 3.2	20.6	11.7	49.6
RSA + Feat. Select., Eq. 3.2	15.4	7.2	42.0

Table 4.5: Average errors for the implicit RSA (IRSA) procedure. Reported results are the average of five 10-fold cross-validation tests. Results from the original RSA are included for reference.

example subsets. They also suffered from the difficulty of accurately predicting the class of new examples. This suggests one avenue for improvement, that is, more easily learnable divisions of the training data. Application of an unsupervised learning or clustering technique to the input data might find non-obvious divisions which would benefit from separate training.

More successful was the application of stacked generalization to the prognosis prediction problem, particularly the combination of several level 0 generalizers, each using a subset of the input features. While our feature subsets were generated randomly, the sets could chosen algorithmically, either as a manual pre-processing step or with a principal components analysis [40]. In this manner the level 0 predictors could operate on sets of features which "belong together."

Of all the RSA extensions described in this chapter, the implicit RSA demonstrated the most improvement in predictive power. One possible extension of this approach, again similar to Ravdin and Clark [74], is to screen the generated examples to reduce the large majority of non-recurrent cases for small values of time and the similar preponderance of recurrent cases for later times.

Chapter 5

Mock generalization

5.1 Introduction

A recurrent theme throughout this work has been increasing the generalization ability of the various predictive systems, that is, to improve their performance on previously unseen cases. Some methods, such as the feature-selection algorithm used in previous chapters and early stopping in backpropagation, utilize a tuning or validation set [47], which is a part of the training data that is not explicitly used by the training algorithm. Rather, this set is used to tune the parameters of the training method, for instance, to choose the set of features for training. The tuning set represents a sample of unseen data, and can therefore be used to avoid overfitting the training data.

Having reserved a set of examples for tuning, it is reasonable to question the exact role played by such a set. We would like to use it not just to tune parameters, but to directly help train the generalizer, since it serves as a surrogate testing set and test-set accuracy is the ultimate goal. Again viewing training as a linear programming problem, one approach would be to add a constraint that limits the tuning error to a level below a pre-determined set rate. However, setting such an *a priori* threshold is problematic, as the error varies widely from one application to the next. Our approach is therefore to put the tuning error into the objective function and minimize it directly, and hence the name "mock generalization." This active use of the tuning set utilizes all of the available information from the tuning set, in contrast to the passive use of the tuning set in parameter-choosing methods.

This chapter describes the results of such an approach applied to the MSM-T decision-tree building program [7, 10]. As described in Chapter 2, each step of the MSM-T algorithm creates a separating plane that minimizes the average distance of misclassified points from the plane. The original MSM-T separation LP is

$$\underset{w,\theta}{\text{minimize}} \quad \frac{1}{m}\|(-Aw + e\theta + e)_+\|_1 + \frac{1}{k}\|(Bw - e\theta + e)_+\|_1 . \tag{5.1}$$

The result is a plane $xw + \theta$ that attempts to separate the m points in matrix A from the k points in B. To incorporate the mock generalization procedure, this is changed to

$$\underset{w,\theta}{\text{minimize}} \quad \begin{array}{r} (1-\lambda)[\frac{1}{m^R}\left\|(-A^R w + e\theta + e)_+\right\|_1 + \frac{1}{k^R}\left\|(B^R w - e\theta + e)_+\right\|_1] + \\ \lambda[\frac{1}{m^S}\left\|(-A^S w + e\theta + e)_+\right\|_1 + \frac{1}{k^S}\left\|(B^S w - e\theta + e)_+\right\|_1]. \end{array} \tag{5.2}$$

Here the training points are collected in the matrices A^R and B^R while the matrices A^S and B^S contain the tuning points. The positive parameter λ controls the relative importance of the training and tuning sets and can take on values between 0 and 1. The percentage of training examples reserved for the tuning set is denoted by ρ. Only those values $0 < \rho \leq 0.5$ are relevant, since $\rho = 0$ reverts the problem to the original MSM-T and the values $\rho > 0.5$ are symmetric, with the roles of the training and tuning sets reversed.

5.2 Relationship to Pareto optimality

Considering the tuning and training errors separately in the objective function suggests the field of multi-objective mathematical programming [83]. This area of optimization considers problems of the form

$$\text{(5.3)} \qquad \operatorname*{vector\,min}_{x \in \mathcal{S}} \quad \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_k(x) \end{bmatrix}$$

Here, multiple, possibly competing objectives are to be optimized. The possible solutions to such a problem are known as *efficient* or *Pareto optimal* [66] points. At a Pareto optimal point, none of the $f_i(x)$ can be further lowered without increasing the value of some other $f_j(x)$. Pareto optimal points can be obtained by solving

$$\text{(5.4)} \qquad \begin{array}{ll} \displaystyle\operatorname*{minimize}_{x \in \mathcal{S}} & \displaystyle\sum_{i=1}^{k} \lambda_i f_i(x) \\ \text{subject to} & \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1 \end{array}$$

While mock generalization does not explicitly define two objective functions, there are clearly two distinct parts to the objective, one each for the training and tuning sets.

In this context, the trade-off between accuracy on the training and tuning sets is also of interest. An experiment was performed in which the tuning set size was set to 30% of the training data and the weight λ was varied to explore this trade-off. The results are shown in Figure 5.1. For any set value of λ, the solution can be considered the single "Pareto optimal" point for that problem.

5.3 Computational results

Moving the tuning set into the objective function adds two operational parameters to the problem: the relative size ρ of the tuning set, and the relative weight λ applied to the tuning-set error. While there is little theoretical basis for setting these, a few heuristics are available:

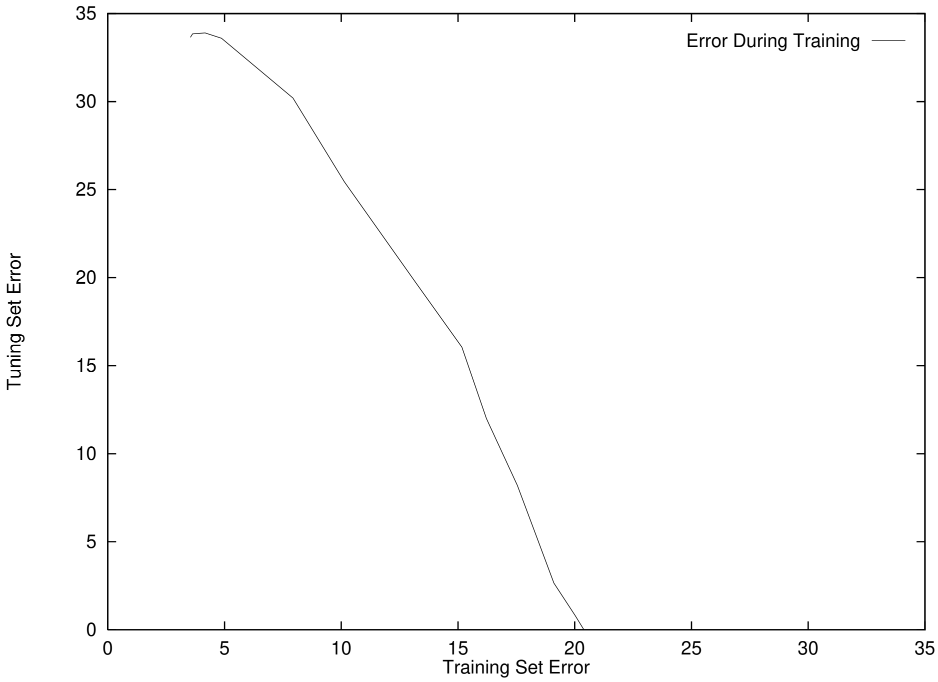

Figure 5.1: Training error for training and tuning set for various values of the relative weight parameter λ. Tuning set size was set to 30%.

- Methods that use a tuning set for parameter choosing, such as early stopping in backpropagation, typically set aside a relatively small percentage of the training set, such as 10%. Since little information from the tuning set is being utilized, it is reasonable to maximize the size of the training set. Other methods which take fuller advantage of the tuning set [76] split the data evenly between training and tuning. Mock generalization shares more with these latter approaches, as it attempts to use all the information available in both training and tuning sets. The proper size is further influenced by the fact that we are working with relatively small data sets. Hence, setting aside a small percentage may result in a tuning set which is not representative of the larger training set.

- If λ is very small, the resulting solution essentially uses the tuning set as a tie-breaker among optimal solutions to the training objective. As discussed in Chapter 3, there exists some sufficiently small $\bar{\lambda}$ such that any positive $\lambda \leq \bar{\lambda}$ results in the LP choosing the solution with best tuning set performance among all those solutions which minimize the training-set error. While this might be seen as an improvement, the space of possible solutions (those which minimize training error) is significantly constrained. Further, the ultimate goal of the procedure is to improve performance on unseen data, which the tuning set is simulating. Larger values of λ which give more credence to the tuning set should probably be favored.

In order to substantiate these heuristics, various values of the two parameters were tested. Tests were performed varying the tuning-set size from 10% to 50% of the training-set size, and varying λ from 0.1 to 0.9, averaging five cross-validation runs at each parameter setting. The mean and variance of MSM-T generalization error are plotted in Figures 5.2 and 5.3. These plots support the above intuition. Both errors and variances appear to be lowest for tuning set sizes of at least one third of the training set. The error is minimized for λ values between 0.3 and 0.7, although the variances are less for smaller λ values.

More extensive tests were then applied to promising values of the parameters. Results of fifty cross-validation tests are summarized in Table 5.1. The mock generalization results used 50% of the training data for tuning, and weighted the training and tuning sets equally. Tests were performed both with and without a pruning procedure which is implemented in MSM-T to remove spurious subtrees. This pruning is borrowed from the C4.5 decision tree algorithm [73], and involves a statistical test of significance for each decision node.

Mock generalization improved the MSM-T testing results both with and without the pruning procedure. The most noticeable improvement is in the reduction of the variance of the estimated correctness. With pruning turned off, the differences in both mean error rate and variance were statistically significant at the 95% confidence level, treating each cross-validation as an independent

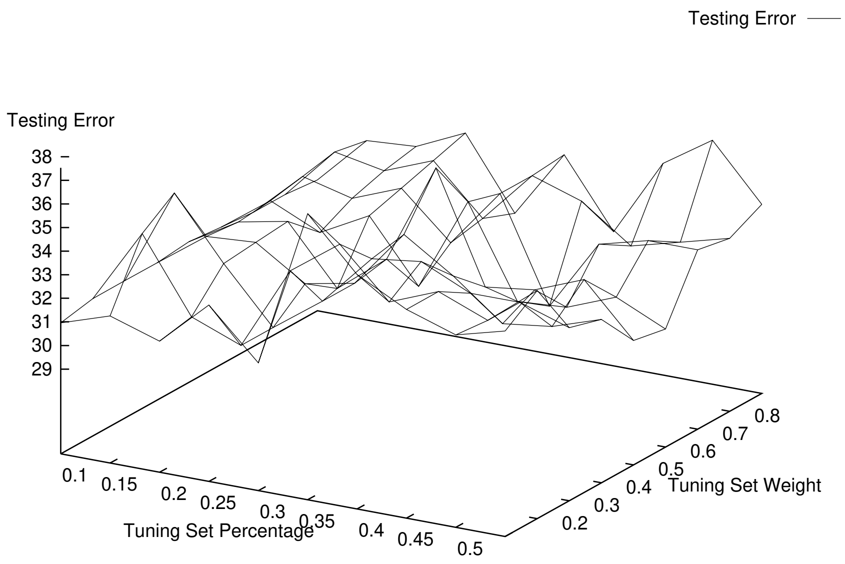

Figure 5.2: Generalization errors with various sizes and weights assigned to the tuning set during mock generalization. In all cases, MSM-T was trained to full training-set separation, and insignificant branches were pruned. Reported results are the average of five cross-validation runs.

trial. While the size of the resulting decision trees was not significantly changed, the splits were more reliable and more consistent. This was also true to a lesser extent for other values of the mock generalization parameters.

The effect of mock generalization can be viewed as analogous to that of the pruning procedure, that is, as an alternate overtraining avoidance technique. This is particularly true on data such as the prognostic data used here which contains relatively few samples and many features. Decision surfaces formed "down the tree" are particularly susceptible to overtraining, as they are being

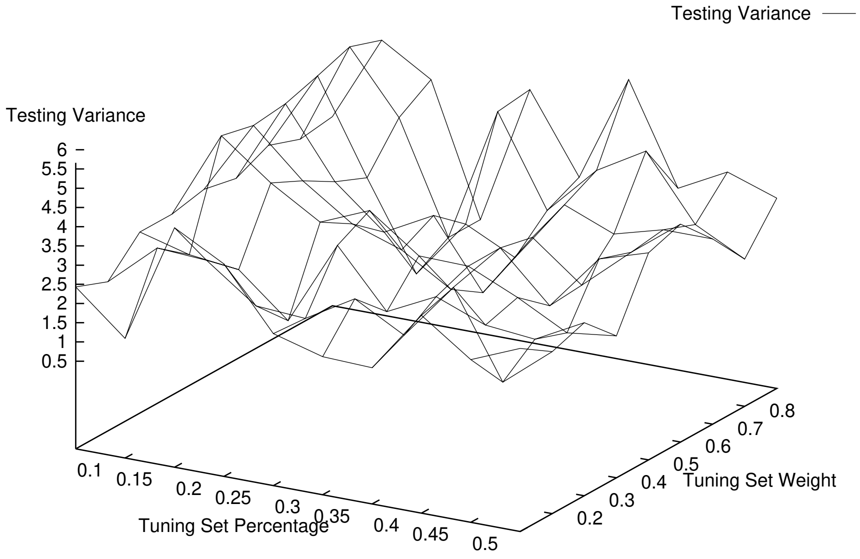

Figure 5.3: Generalization variances with various sizes and weights assigned to the tuning set during mock generalization.

formed using fewer and fewer examples. In our tests, mock generalization mitigates this problem to about the same extent as the pruning of insignificant nodes.

5.4 Extensions

The ideas and experiments described in this chapter lay the groundwork for mock generalization, a potentially significant method for improving the generalization of a learning system. The next phase in exploring this approach involves three steps:

	Correctness ± sd
MSM-T	65.29 ± 3.87
MSM-T with mock gen.	67.38 ± 2.99
MSM-T + pruning	67.99 ± 3.42
MSM-T + pruning with mock gen.	68.48 ± 2.98

Table 5.1: MSM-T correctness results on the two-year prognosis problem with and without mock generalization. Testing results are the average of 50 cross-validation tests, and are reported as correctness plus or minus one standard deviation.

- First, the combination of mock generalization with the MSM-T decision tree learner should be further tested on different learning tasks. Exploratory analysis like that of the previous section should be performed to further explore the relationship between the parameters of the system.

- The application of mock generalization is certainly not confined to decision tree building. It will also be added to the RSA prognosis prediction method. Further, we believe mock generalization can improve the generalization of any optimization-based machine learning method. This will be tested by adding the procedure to backpropagation.

- The most important extension to mock generalization is the automatic setting of the various parameters. It is possible that larger or smaller tuning sets with larger or smaller weights will improve performance on different types of data; the procedure would become significantly more useful if this could be detected automatically. The possibility of predicting generalization performance based on the relative correctness of training and tuning sets (as shown in Figure 5.1) should also be investigated.

Chapter 6

Banded approximation

6.1 Introduction

This chapter describes "banded" approximation, another method for improving the generalization of a learning system. Here we consider learning explicitly in terms of fitting a function from a set of noisy input data. The goal of such a learning system is to capture the important aspects of the output in relation to the input features, while filtering out the irrelevant noise. Of course, without *a priori* knowledge or assumptions about the characteristics of the various features, it is difficult to distinguish noise from signal.

In trying to fit noisy data, we attempt to obtain better generalization by assuming a "band" of errors around the measured observations. This is implemented with a parametric tolerance band of width τ surrounding the predictive surface, as illustrated in Figure 6.1. Points that fall into this band during training will not be considered to be in error; points outside the band have their error reduced by the width of the band. When $\tau = 0$, this is classical least error fit and one might expect poor generalization if the data is noisy. On the other hand, if a large tolerance τ is used, then an inaccurate fit would again result in poor generalization. Somewhere between $\tau = 0$ and large values of τ, one might expect an "optimal" τ that generalizes in a more satisfactory way. Indeed our computational results indicate precisely that is true.

In addition to reducing the effect of outliers, banded approximation also frees the learning method from trying to fit exactly the artifacts of the training data. Thus we conjecture that banded approximation allows a sufficiently powerful learning system to extract more easily important aspects of the data without overfitting or memorizing less important variations.

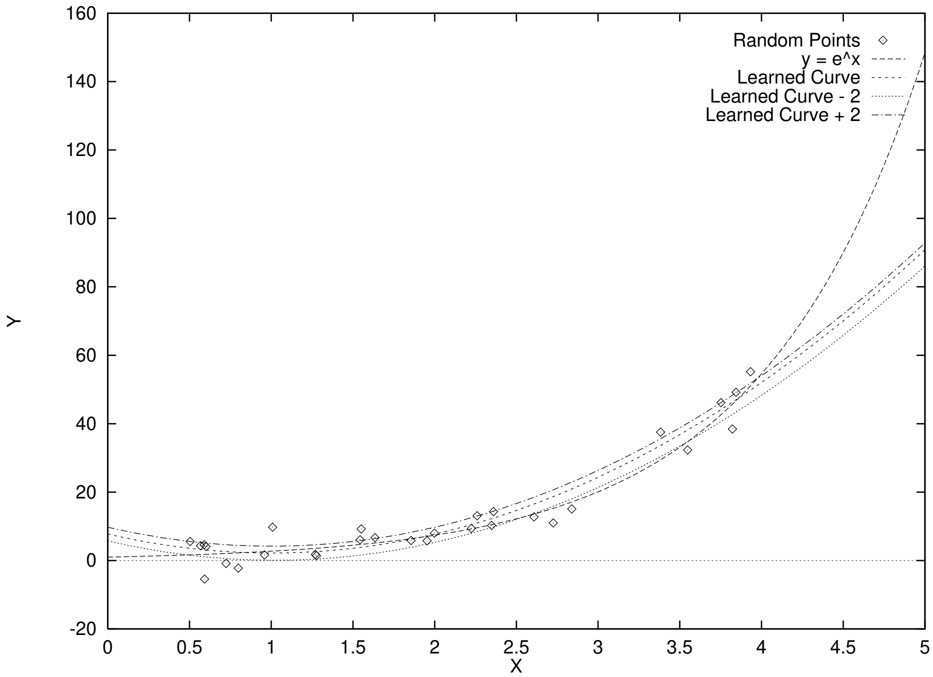

Figure 6.1: Typical data-fitting result. The exponential curve $y = e^x$ was fit with a quadratic. The tolerance bands used in training are shown above and below the learned curve.

This approach to function estimation can be seen as the inverse of the statistical practice of constructing confidence bands after approximating the function [24, 30, 88]. We also note that in the field of approximation theory, similar ideas have been used by letting equality constraints of the problem be replaced by inequalities [56].

6.2 Simulation results

The banded approximation idea was first tested with a simple curve fitting application, using simulated data. The use of simulated data allows the predicted curve to be compared directly to the known function that generated the input points. For the first experiment, sample points were generated from the exponential curve $y = e^x$ sampled uniformly on the interval [0,4]. Additive Gaussian noise with mean zero and standard deviation 4 was then added to the y value. The points were fit with a quadratic curve $f(x) = ax^2 + bx + c$, using a least 1-norm fit to learn the coefficients a, b and c with various values of τ. To evaluate the results, the learned curves were compared to the generating curve $y = e^x$ at intervals of 0.1 along [0,4], and the average error computed. The results of a typical run are shown in Figure 6.1, in which 30 points were fit using a tolerance band of width 4. The value of the tolerance parameter τ was varied from 0 to 6, with fifty tests performed for each value of τ. The results of four such experiments using training sets with different numbers of points are shown in Figure 6.2.

In each of these experiments, some value of τ greater than zero improved the resulting fit. This effect was most noticeable for the smaller training sets, as the improvement ranged from 12.5% for the 10 point set to 2% for 50 points. The diminished return of banded approximation for larger training sets makes sense in terms of the bias/variance tradeoff [32]. Since the same type of surface is being constructed either with or without the tolerance band, the methods have equal predictive power and hence the expected bias part of the error remains the same. However, banded approximation helps prevent overfitting, thereby reducing the variance part of the error, that is, the dependence on the exact distribution of the training data. Since larger training sets also reduce the variance, the effect is not as apparent as the number of training cases increases.

A similar experiment is summarized in Figure 6.3. Here, a second input variable was added, and given random values unrelated to the output. The

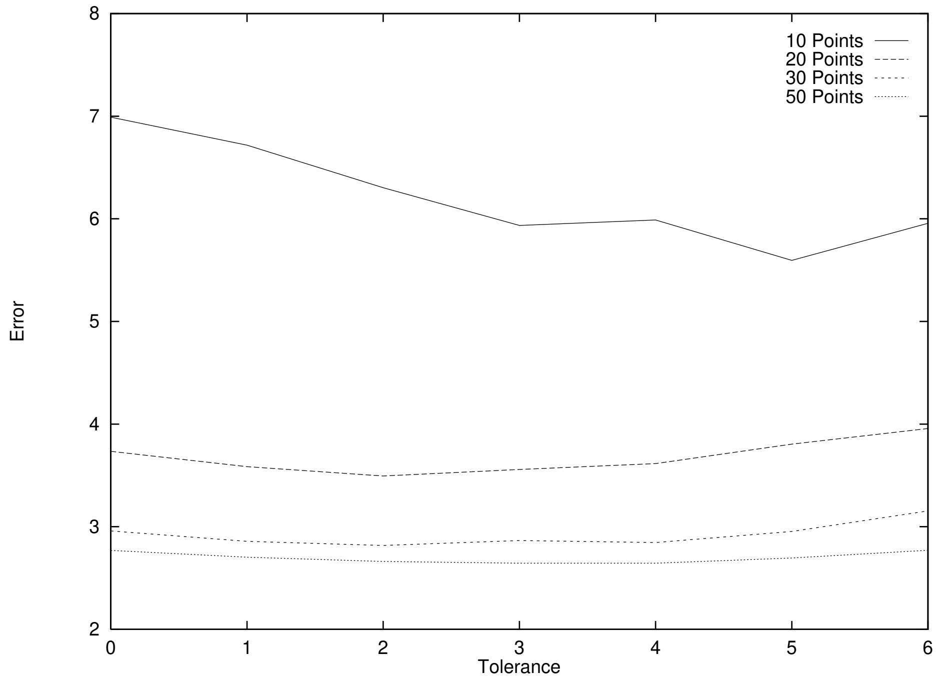

Figure 6.2: Total error when fitting e^x with a quadratic on the interval [0,4]. Each data set contains points with Gaussian noise (mean 0, standard deviation 4) added. For each set of points, the value of the tolerance variable τ was varied from 0 to 6. The errors shown here were averaged over fifty runs.

ability of banded generalization to filter noise is shown even more clearly in this case, with error reductions varying from 20% to 4.5%. This again follows from a bias vs. variance argument, as the addition of an irrelevant feature significantly increases the variance of small data sets. Much of this variance was filtered out by banded approximation, as indicated by the nearness of the best test results in the two experiments.

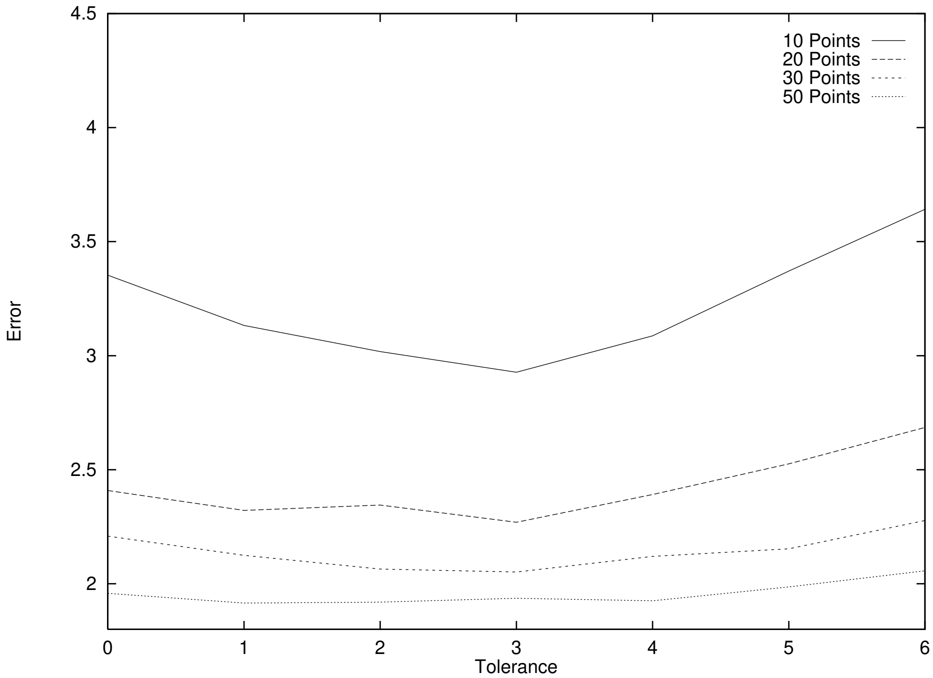

Figure 6.3: Total error when fitting e^{x_1} with a quadratic on the interval [0,4]. In this experiment a random x_2 feature was added as input.

6.3 Prognostic prediction

The banded approximation method was built into the recurrence surface approximation (RSA) prognosis prediction system. A slight change was made to RSA in that underestimated recurrent cases were counted as errors, both in training and in testing. Hence, this version of RSA attempts to fit recurrence times exactly (within the limits of the tolerance band), and as usual tries to overestimate the disease free times of the non-recurrent cases. The resulting linear program is the following:

$$
(6.1) \qquad \begin{array}{ll}
\underset{w,\gamma,v,y,z}{\text{minimize}} & \frac{1}{m}e^T y + \frac{1}{k}e^T z \\
\text{subject to} & \begin{array}{rcccl} -y - \tau e & \leq & Mw + \gamma e - t & \leq & y + \tau e \\ & & -Nw - \gamma e + r & \leq & z + \tau e \\ & & y, z & \geq & 0 \end{array}
\end{array}
$$

Testing results for various values of τ are shown in Figure 6.4. Again we note that increasing the value of τ produces a significant reduction in the error. The best result was observed at $\tau = 24$, representing a band of 2 years on either side of the predictive surface. The average error in this case was more than 12% lower than when no tolerance was allowed. A more detailed description of this test is shown in Table 6.1. Using a pairwise t-test to compare the results, the drop in average error is statistically significant at the 90% confidence level.

	Train	Test	Recur	Non-recur
RSA, $\tau = 0$	24.3	19.6	14.8	34.9
Banded Approx., $\tau = 24$	32.7	17.2	14.2	27.0

Table 6.1: Breakdown of the errors when predicting the Wisconsin prognostic data with banded approximation, $\tau = 24$ months. Both positive and negative errors were measured for the recurrent points.

6.4 Extensions

As is the case with the mock-generalization procedure in the previous chapter, the next key extension to the the banded approximation is the automatic setting of the parameter. This could easily be accomplished with tuning or cross-validation procedure. However, the fact that the parameter τ can be viewed as an error measurement suggests that a good value may be available with a

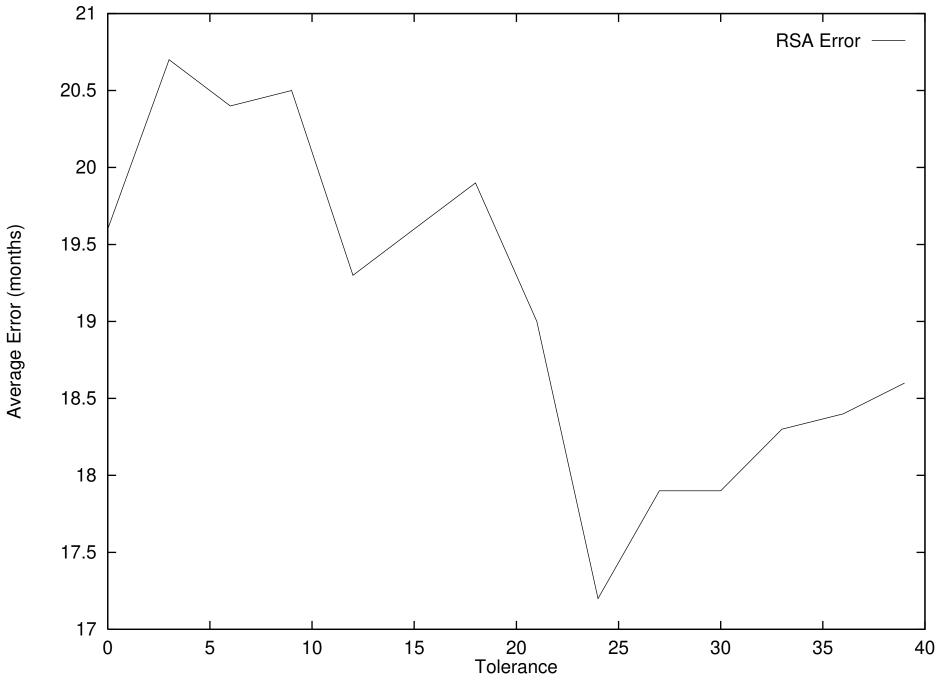

Figure 6.4: Average RSA error on the Wisconsin prognostic data for various values of the tolerance parameter τ.

minimum of extra processing. Note in Table 6.1 that the observed training error with no tolerance was around 2 years. This was also the optimal value for the tolerance band, suggesting that training error may be a good choice for τ, thereby requiring only two optimization steps. This hypothesis was partially supported by the simulation experiments, as shown in Table 6.2, as that setting of τ always resulted in a reduced testing error. From these experiments, it seems clear that banded approximation is an effective method for improving the generalization of a learning method, particularly with small, noisy data sets. Future work will explore exactly how the error is being captured and how the method can be optimized.

		Train Error, $\tau = 0$	Optimal τ
Exp. 1,	10 Points	2.46	3
	20 Points	3.02	3
	30 Points	3.24	3
	50 Points	3.40	1
Exp. 2,	10 Points	1.96	5
	20 Points	2.85	2
	30 Points	2.94	2
	50 Points	3.24	3

Table 6.2: This table shows the value of the training objective with $\tau = 0$ for each of the simulation experiments, along with the value of τ that minimized the observed testing error. Except for two cases, the two values are very close.

Another interesting direction for future work is to combine banded approximation with other ideas presented in previous chapters. In particular, experiments combining banded approximation with mock generalization will be performed. The tolerance band will be applied only to the training set, and the tuning set evaluated with no tolerance, as were the testing cases in this chapter. Banded approximation may also be enhanced by including the feature selection procedure, as a tolerant fit may allow a smaller set of features, thereby further reducing the possibility of overfitting.

Chapter 7

Conclusions

The research presented here has resulted in a number of novel machine learning approaches, as well as the successful application of these approaches to the type of small, noisy data sets prevalent in medical domains. The **Xcyt** program combines learning via linear programming with image processing and statistics to form an automatic diagnosis system with a so far perfect track record on 92 cases in clinical practice. The recurrence surface approximation method and its variations predict time of recurrence in malignant patients with an average error of less than two years, even on small data sets. If verified, the accuracy of this approach could have a significant effect on treatment decisions, such as eliminating the necessity of lymph node removal during mastectomies for prognostic purposes. The RSA methodology is applicable to any problem involving censored data. The three proposed methods for improving generalization – feature selection, mock generalization and banded approximation – are shown to reduce the error rate and/or the variance of the learning systems to which they were added. Linear programming-based machine learning is shown to be a successful methodology for addressing the diagnosis and prognosis of breast cancer.

Bibliography

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] M. L. Astion and P. Wilding. Application of neural networks to the interpretation of laboratory data in cancer diagnosis. *Clinical Chemistry*, 38:34–38, 1992.

[3] D. Ballard and C. Brown. *Computer Vision.* Prentice–Hall, Inc, Englewood Cliffs, NJ, 1982.

[4] P. H. Bartels, G. F. Bahr, M. Bibbo, and G. L. Wied. Objective cell image analysis. *Journal of Histochemistry and Cytochemistry*, 30(4):280–354, 1973.

[5] W. G. Baxt. Use of an artificial neural network for data analysis in clinical decision making: The diagnosis of acute coronary occlusion. *Neural Computation*, 2(4):480–489, 1990.

[6] O. H. Beahrs, D. E. Henson, R. V. P. Hutter, and B. J. Kennedy. *Manual for Staging of Cancer.* J.B. Lippincott, Philadelphia, 4th edition, 1992.

[7] K. P. Bennett. Decision tree construction via linear programming. In *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference*, pages 97–101, 1992.

[8] K. P. Bennett. *Machine Learning via Mathematical Programming.* PhD thesis, University of Wisconsin-Madison, Computer Sciences Department, July 1993.

[9] K. P. Bennett and O. L. Mangasarian. Neural network training via linear programming. In P. M. Pardalos, editor, *Advances in Optimization and Parallel Computing*, pages 56–67. North–Holland, Amsterdam, 1992.

[10] K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[11] J. Berkson. The calculation of survival rates. In W. Walters, H. K. Gray, and J. T. Priestley, editors, *Carcinoma and Other Malignant Lesions of the Stomach.* W. B. Saunders, Philadelphia, 1942.

[12] M. M. Black, S. R. Opler, and F. D. Speer. Survival in breast cancer cases in relation to the structure of the primary tumor and regional lymph nodes. *Surgery, Gynecology and Obstetrics*, 100:543–551, 1955.

[13] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[14] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Inc., Pacific Grove, CA, 1984.

[15] J. Buckley and I. James. Linear regression with censored data. *Biometrika*, 66:429–436, 1979.

[16] H. B. Burke. Artificial neural networks for cancer research: Outcome prediction. *Seminars in Surgical Oncology*, 10:73–79, 1994.

[17] P. J. Burt. Fast filter transforms for image processing. *Computational Graphics, Image Processing*, 16:20–51, 1981.

[18] J. D. Cappellitti and A. Rosenfeld. Three-dimensional boundary following. *Computer Vision, Graphics, and Image Processing*, 48:80–92, 1989.

[19] C. L. Carter, C. Allen, and D. E. Henson. Relation of tumor size, lymph node status, and survival in 24,740 breast cancer cases. *Cancer*, 63:181–187, 1989.

[20] A. Charnes. Some fundamental theorems of perceptron theory and their geometry. In J. T. Tou and R. H. Wilcox, editors, *Computer and Information Sciences*. Spartan Books, Washington, D. C., 1964.

[21] D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society*, B 34:187–202, 1972.

[22] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton NJ, 1963.

[23] M. De Laurentiis and P. M. Ravdin. A technique for using neural network analysis to perform survival analysis of censored data. *Cancer Letters*, 77:127–138, 1994.

[24] N. R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley and Sons, New York, 1966.

[25] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. John Wiley and Sons, New York, 1973.

[26] R. C. Eberhart, R. W. Dobbins, and L. V. Hutton. Neural network paradigm comparisons for appendicitis diagnosis. In *Proceedings of the Fourth Annual IEEE Symposium on Computer-Based Medical Systems*, pages 298–304, 1991.

[27] B. Efron. *The Jackknife, the Bootstrap and Other Resampling Plans*. Number 38 in CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, 1982.

[28] S. E. Fahlman and C. Lebiere. The cascade–correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 524–532, San Mateo, CA, 1990. Morgan Kaufmann.

[29] W. J. Frable. Thin-needle aspiration biopsy. In *Major Problems in Pathology 14*. W.B. Saunders Co., Philadelphia, 1983.

[30] R. Fraiman and G. Perez Iribarren. Conservative confidence bands for nonparametric regression. In G. Roussas, editor, *Nonparametric Functional Estimation and Related Topics*, pages 45–66. Kluwer Academic Publishers, 1991.

[31] J. E. Freund. *Mathematical Statistics*. Prentice Hall, Englewood Cliffs, NJ, fifth edition, 1992.

[32] S. Geman, E. Bienestock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[33] R. W. M. Giard and J. Hermans. The value of aspiration cytologic examination of the breast. A statistical review of the medical literature. *Cancer*, 69:2104–2110, 1992.

[34] V. Golberg, A. Manduca, D. L. Ewert, J. J. Gisvold, and J. F. Greenleaf. Improvement in specificity of ultrasonography for diagnosis of breast tumors by means of artificial intelligence. *Medical Physics*, 19:1475–1481, 1992.

[35] P. Hart. The condensed nearest neighbor rule. *Transactions on Information Theory*, IT-14:515–516, 1967.

[36] P. Hermanek and L. H. Sobin, editors. *TNM Classification of Malignant Tumors*. Springer-Verlag, Berlin, 4th edition, 1987.

[37] W. Highleyman. A note on linear separability. *IRE Transactions on Electronic Computers*, pages 777–778, 1961. Correspondence Section.

[38] G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, Hillsdale, 1986. Erlbaum.

[39] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[40] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[41] R. G. Miller Jr. *Survival Analysis*. John Wiley and Sons, New York, 1981.

[42] E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53:457–481, 1958.

[43] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[44] K. Kira and L. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 129–134, San Mateo, CA, 1992. Morgan Kaufmann.

[45] J. Kittler. Feature selection and extraction. In Young & Fu, editor, *Handbook of Pattern Recognition and Image Processing*. Academic Press, New York, 1986.

[46] P. Lachenbruch and P. Mickey. Estimation of error rates in discriminant analysis. *Technometrics*, 10:1–11, 1968.

[47] K. Lang, A. Waibel, and G. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(23–43), 1990.

[48] Y. Le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, San Mateo, CA, 1990. Morgan Kaufmann.

[49] E. T. Lee. *Statistical Methods for Survival Data Analysis*. John Wiley and Sons, New York, 1992.

[50] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, April 1987.

[51] B. B. Mandelbrot. *The Fractal Geometry of Nature*, chapter 5. W. H. Freeman and Company, New York, 1977.

[52] O. L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1965.

[53] O. L. Mangasarian. Multi-surface method of pattern separation. *IEEE Transactions on Information Theory*, IT-14:801–807, 1968.

[54] O. L. Mangasarian. Mathematical programming in neural networks. *ORSA Journal on Computing*, 5:349–360, 1993.

[55] O. L. Mangasarian and R. R. Meyer. Nonlinear perturbation of linear programs. *SIAM Journal on Control and Optimization*, 17:745–752, 1979.

[56] O. L. Mangasarian and L. L. Schumaker. Splines via optimal control. In *Approximation with Special Emphasis on Spline Functions*, pages 119–156. Academic Press, New York, 1969.

[57] O. L. Mangasarian, R. Setiono, and W. H. Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis.

In *Proceedings of the Workshop on Large-Scale Numerical Optimization*, pages 22–31, Philadelphia, Pennsylvania, 1990. SIAM.

[58] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23:1 & 18, 1990.

[59] E. Marshall. Search for a killer: Focus shifts from fat to hormones in special report on breast cancer. *Science*, 259:618–621, 1993.

[60] J. Michel, G. Mirchandani, and S. Wald. Prognosis with neural networks using statistically based feature sets. In *Fifth Annual IEEE Symposium on Computer-Based Medical Systems*, pages 695–702, Los Alamitos, CA, June 1992. IEEE Computer Society Press.

[61] B. A. Miller, E. J. Feuer, and B. F. Hankey. Recent incidence trends for breast cancer in women and relevance of early detection: An update. *CA Cancer Journal for Clinicians*, 43(1):27–41, 1993.

[62] M. Mitze, 1992. Personal communication.

[63] S. Mukhopadhyay, A. Roy, L. S. Kim, and S. Govil. A polynomial time algorithm for generating neural networks for pattern classification: Its stability properties and some test results. *Neural Computation*, 5:317–330, 1993.

[64] B.A. Murtagh and M.A. Saunders. MINOS 5.0 user's guide. Technical Report SOL 83.20, Stanford University, December 1983. MINOS 5.4 Release Notes, December 1992.

[65] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9):917–922, September 1977.

[66] V. Pareto. *Manuel d'Economie Politique*. Giard, Paris, 1909.

[67] E. Parzen. On estimation of a probability density and mode. *Annals of Mathematical Statistics*, 35:1065–1076, 1962.

[68] K. J. Pienta and D. S. Coffey. Correlation of nuclear morphometry with progression of breast cancer. *Cancer*, 68:2012–2016, 1991.

[69] W. L. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, 1986.

[70] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[71] J. R. Quinlan. Decision trees as probabilistic classifiers. In *Proceedings of Fourth International Workshop on Machine Learning*, Los Altos, CA, 1987. Morgan Kaufmann.

[72] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27, 1987.

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

[74] P. M. Ravdin and G. M. Clark. A practical application of neural network analysis for predicting outcome of individual breast cancer patients. *Breast Cancer Research and Treatment*, 22:285–293, 1992.

[75] P. M. Ravdin, A. K. Tandon, D. C. Allred, G. M. Clark, S. A. W. Fuqua, S. H. Hilsenbeck, G. C. Chamness, and C. K. Osborne. Cathepsin D by western blotting and immunohistochemistry: Failure to confirm correlations with prognosis in node-negative breast cancer. *Journal of Clinical Oncology*, 12:467–474, 1994.

[76] A. Röbel. The dynamic pattern selection algorithm: Effective training and controlled generalization of backpropagation neural networks. Technical Report 93/23, Technische Universität Berlin, 1994.

[77] Asim Roy, Lark Sang Kim, and Somnath Mukhopadhyay. A polynomial time algorithm for the construction and training of a class of multilayer perceptrons. *Neural Networks*, 6:535–545, 1993.

[78] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8. MIT Press, Cambridge, MA, 1986.

[79] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing*. MIT Press, Cambridge, MA, 1986.

[80] C. Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10:153–178, 1993.

[81] A. Schenone, L. Andreucci, V. Sanguinetti, and P. Morasso. Neural networks for prognosis in breast cancer. *Physica Medica*, IX(Supplement 1):175–178, June 1993.

[82] J. Sietsma and R. J. F. Dow. Neural net pruning – why and how. In *IEEE International Conference on Neural Networks*, volume 1, pages 325–333, New York, 1988. IEEE Press.

[83] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley and Sons, 1986.

[84] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, 36:111–147, 1974.

[85] W. N. Street. Toward automated cancer diagnosis: An interactive system for cell feature extraction. Technical Report 1052, Computer Sciences Department, University of Wisconsin, October 1991.

[86] W. N. Street, W. H. Wolberg, and O. L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *IS&T/SPIE 1993 International*

Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861–870, San Jose, California, 1993.

[87] P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.

[88] G. Wahba. Bayesian confidence intervals for the cross-validated smoothing spline. *Journal of the Royal Statistical Society (Series B)*, 45:133–150, 1983.

[89] G. L. Wied, P. H. Bartels, M. Bibbo, and H. E. Dytch. Image analysis in quantitative cytopathology and histopathology. *Human Pathology*, 20(6):549–571, 1989.

[90] D. J. Williams and M. Shah. A fast algorithm for active contours. In *Proceedings of the Third International Conference on Computer Vision*, pages 592–595, Osaka, Japan, December 1990.

[91] W. H. Wolberg, K. P. Bennett, and O. L. Mangasarian. Breast cancer diagnosis and prognostic determination from cell analysis. Manuscript, Departments of Surgery and Human Oncology and Computer Sciences, University of Wisconsin, 1992.

[92] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences, U.S.A.*, 87:9193–9196, 1990.

[93] W. H. Wolberg and O. L. Mangasarian. Computer-designed expert systems for breast cytology diagnosis. *Analytical and Quantitative Cytology and Histology*, 15(1):67–74, February 1993.

[94] W. H. Wolberg, W. N. Street, D. H. Heisey, and O. L. Mangasarian. Computer-derived nuclear features distinguish malignant from benign breast cytology. Cancer, submitted, 1994.

[95] W. H. Wolberg, W. N. Street, and O. L. Mangasarian. Breast cytology diagnosis via digital image analysis. *Analytical and Quantitative Cytology and Histology*, 15(6):396–404, December 1993.

[96] W. H. Wolberg, W. N. Street, and O. L. Mangasarian. Computerized breast cancer diagnosis and prognosis from fine-needle aspirates. Analytical and Quantitative Cytology and Histology, submitted, 1994.

[97] W. H. Wolberg, W. N. Street, and O. L. Mangasarian. Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates. *Cancer Letters*, 77:163–171, 1994.

[98] D. H. Wolpert. On overfitting as bias. Technical Report TR 92-03-5001, The Santa Fe Institute, 1992.

[99] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[100] Y. Wu, M. L. Giger, K. Doi, C. J. Vyborny, R. A. Schmidt, and C. E. Metz. Artificial neural networks in mammography: Application to decision making in the diagnosis of breast cancer. *Radiology*, 187:81–87, 1993.

[101] Y. O. Yoon, R. W. Brobst, P. R. Bergstresser, and L. L. Peterson. A desktop neural network for dermatology diagnosis. *Journal of Neural Network Computation*, pages 43–52, Summer 1989.

Appendix A

Xcyt features on diagnosis data

The following table lists the means and standard deviations, for both benign and malignant cases, of each of the **Xcyt** nuclear features as described in Chapter 2. All results are shown to four significant digits, and were computed on the 569 original training cases (357 benign, 212 malignant). Note: SE = standard error, W = worst (largest).

	Benign		Malignant	
	Mean	Std. Deviation	Mean	Std. Deviation
Radius	12.15	1.778	17.46	3.196
Texture	17.91	3.99	21.6	3.771
Perimeter	78.08	11.79	115.4	21.8
Area	462.8	134.1	978.4	367.1
Smoothness	0.09248	0.01343	0.1029	0.01258
Compactness	0.08008	0.0337	0.1452	0.05386
Concavity	0.04606	0.04338	0.1608	0.07484
Concave Pts	0.02572	0.01589	0.08799	0.03429
Symmetry	0.1742	0.02477	0.1929	0.02757
Fractal Dim	0.06287	0.006738	0.06268	0.007555
SE Radius	0.2841	0.1124	0.6091	0.3442
SE Texture	1.22	0.5884	1.211	0.482
SE Perimeter	2.0	0.7701	4.324	2.562
SE Area	21.14	8.831	72.67	61.21
SE Smoothness	0.007196	0.003056	0.00678	0.002884
SE Compact	0.02144	0.01633	0.03228	0.01834
SE Concavity	0.026	0.03287	0.04182	0.02155
SE Concv Pts	0.009858	0.005701	0.01506	0.005504
SE Symmetry	0.02058	0.006989	0.02047	0.01004
SE Fractal Dim	0.003636	0.002934	0.004062	0.002037

	Benign		Malignant	
	Mean	Std. Deviation	Mean	Std. Deviation
W Radius	13.38	1.979	21.13	4.273
W Texture	23.52	5.486	29.32	5.422
W Perimeter	87.01	13.51	141.4	29.39
W Area	558.9	163.4	1422	596.6
W Smoothness	0.125	0.01999	0.1448	0.02182
W Compactness	0.1827	0.09205	0.3748	0.17
W Concavity	0.1662	0.1402	0.4506	0.1811
W Concave Pts	0.07444	0.03575	0.1822	0.0462
W Symmetry	0.2702	0.04169	0.3235	0.07451
W Fractal Dim	0.07944	0.01378	0.09153	0.0215