Machine Learning at the Operating Room of the Future:

A Comparison of Machine Learning Techniques applied to Operating Room Scheduling

by

Samuel Ingraham Davies

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of Master of Engineering in

Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 26, 2004
[June 2004]

Author_____
Department of Electrical Engineering and Computer Science
May 26, 2004

Certified by_____
Leslie Pack Kaelbling
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Machine Learning at the Operating Room of the Future:

A Comparison of Machine Learning Techniques applied to Operating Room Scheduling

by

Samuel Ingraham Davies

Submitted to the
Department of Electrical Engineering and Computer Science

May 26, 2004

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The quality of an operating room schedule is determined by the accuracy of the surgery duration estimation used. State of the art estimation algorithms consider only three surgery variables—procedure type, surgeon identity, and date of surgery—to predict the length of surgeries. We show that if we can take advantage of a richer set of available information, we can significantly improve estimation accuracy. Additional recorded (but unused) variables include patient age, gender, and morbidity, anesthesiologist identity, and surgery location. We implement and compare the accuracy of four standard machine learning algorithms that take advantage of this richer data set: linear regression, nearest neighbors, regression trees, and support vector regression. We conclude that additional variables can improve the accuracy estimate by as much as 20%. Finally, we discuss the implementation challenges and future work necessary to make machine learning techniques available to the data analyst concerned with implementation.

Portions of this work are sponsored by the U.S. Dept. of the Army, under DAMD 17-02-2-0006. The information does not necessarily reflect the position of the government, and no official endorsement should be inferred.

Thesis Supervisor: Leslie Pack Kaelbling

Title: Assistant Director, Computer Science and Artificial Intelligence Laboratory

Acknowledgements

Infinite thanks to Leslie Kaelbling, my freshman 6.001 recitation instructor in the fall of 1999, for attracting me to computer science at MIT. Infinity-plus-one thanks also to her for suggesting this project to me, and letting me fall into my own dead-end paths but helping me get up, brush off, and continue back on the right path. Working with her has been a wonderful learning experience.

Many thanks also to Dr. David Rattner, a surgeon and professor who served as my co-advisor at MGH. Working with him and his research group on the Operating Room of the Future project has opened my eyes to importance of the "human factors" side of a project like this. And although it made me a little woozy, it is thanks to him that I saw my first live surgery.

Dr. James Stahl has been an extremely and patient resource on this project—he helped me work through some of the difficult technical issues related to hospital information systems. His support and willingness to make time for me is greatly appreciated. Thanks to Bethany Daily for helping me to retrieve data and answering my millions of questions about the MGH database systems. Marie Egan's kindness and warmth was a constant support, and she was invaluable in helping me understand the scheduling problem from a nursing perspective. Warren Sandberg had great suggestions and input on my project and was willing to help give my family a tour of the operating room on a day's notice. Julian Goldman was full of probing questions, as well as some great references to scheduling work from the medical and anesthesia literature.

Finally, I would like to thank my professor Tomaso Poggio for teaching the clear and consistent notation machine learning notation and formalisms underlying this thesis.

Contents

Figures

1 Introduction

People around the country and the world can benefit from more efficient operating rooms; if O.R. schedules were accurate enough to allow for more surgeries per day, then patient satisfaction, staff satisfaction, and productivity would all increase. Machine learning may help make a step in this direction. Researchers have developed a rich set of learning algorithms that they claim can work with rich, complex, high-dimensional datasets like the historical case databases for large hospitals. These algorithms can learn to improve O.R. schedules by observing the past and predicting the future more accurately.

Many machine learning algorithms have been developed and analyzed to a point of theoretical maturity; their strengths and flaws are understood theoretically, but their flexibility and applicability to practical data sets have only begun to be investigated. Despite the development of freely available software learning tools such as YALE [Joachims98] and Weka [Witten99], the implementation and testing of machine learning algorithms on real-world data sets still requires a significant amount of custom coding and dataset-specific parameter tuning, variable selection, and variable manipulation (particularly in the case of categorical variables).

In this thesis, our purposes are:

1. To determine the feasibility of improving O.R. scheduling through the application of standard machine learning algorithms to the estimation of surgery duration, and the use of a richer data set than the state of the art, and

2. To use the representative example of O.R. scheduling to expose and address common technical pitfalls that are likely to arise in future applications of machine learning algorithms.

The first goal aims to help the O.R. manager community by exposing them to a wealth of new analysis tools to aid their practice, and the second aims to help the artificial intelligence community by identifying the most important areas for improvement in existing algorithms to prepare them for widespread practical use.

1.1 Introduction to Supervised Machine Learning

The oldest and simplest form of machine learning was invented in 1875 by Sir Francis Galton, a cousin of Charles Darwin, when he was looking for a way to express the linear correlation he found between the pea sizes of parent and offspring pea plants [Stanton01]. Researchers in the field of artificial intelligence give regression a special name: supervised machine learning.

Regression is "supervised" learning in the sense that a human or an automated sensor collects a large training set S of examples of (uniform-length) n-dimensional input vectors $\mathbf{x}_i \in \mathbb{Z}^n$, and attaches real-valued labels $y_i \in \mathbb{R}$ to them. However, after this initial supervision, the machine is left alone. The machine's job is to find the "best" function $f_S : \mathbb{Z}^N \to \mathbb{R}$ from a hypothesis space $\mathcal{H}$ of possible functions that predicts labels for the data. Note that there are many good ways of defining what the "best" function is—we will discuss one way later in this section. Once the machine has chosen f_S, it may be used to guess labels for new data $y_{pred} = f_S(\mathbf{x}_{new})$.

Note that in the above paragraph, we use the symbol $\mathbb{Z}$ for the types of input variables. We let $\mathbb{Z}$ stand for the union of two variable types: real variables $\mathbb{R}$, and categorical variables $\mathbb{C}$. The first type $\mathbb{R}$ refers to the familiar real, ordered variables. Real variables need not have infinite range or resolution—ASA_class, for example, which measures patient morbidity, has a range of five discrete values. Real values have a total ordering under the $>$ relation.

The second type of variable $\mathbb{C}$ is categorical. Categorical variables always have a finite set of m values, but their values have no ordering relation; that is, we cannot say that one is greater than another, nor can we say that any two values more "similar" than any other two. Examples of categorical variables include surgeon identity or surgery room. Gender is a special case variable because it has exactly two values, meaning that it can be treated as categorical or real (by assigning male to 0 and female to 1 or vice versa).

The supervised learning paradigm may be applied directly to a wide range of important real-world problems. It can be used to model simple one-dimensional ($n = 1$) problems such as predicting the size of offspring peas given the size of their parents; it may also be applied to higher-dimensional ($n \gg 1$) problems such as predicting a person's life expectancy from a corpus of personal data, determining if a collection of pixels depict a face, or figuring out if a piece of email is spam.[1] In this thesis, we consider the particular supervised learning problem of predicting how long a surgery will take given all the data about it that is available in a hospital's database.

---

[1] The problems of determining whether a pattern of pixels is a face and determining if a piece of email is spam are both examples of the *classification* problem in which the learning algorithm chooses a binary function $f_S : \mathbb{R}^n \rightarrow \{-1, 1\}$. The classification problem is, of course, a strict subset of the regression problem.

We now examine the question of what exactly "best" means when it comes to choosing a function to fit the data. To do so, we introduce a loss function $V(f, \mathbf{x}, y) : (\mathbb{Z}^n \to \mathbb{R}), \mathbb{Z}^n, \mathbb{R} \to \mathbb{R}$. V takes three arguments: a learned function f, a data point $\mathbf{x}$ that may be passed to f in order to predict a label, and the actual label y for $\mathbf{x}$. V returns a cost associated with the quality of the predicted label in comparison to the actual label. Generally, higher loss values imply poorer prediction performance. For example, one common loss function is the square loss or L2 loss: $V_{L2}(f, \mathbf{x}, y) = (f(\mathbf{x}) - y)^2$. We will be using the square loss function exclusively throughout this thesis.[2]

We define the empirical error of f for a training set S of d data points and labels as follows $I_S[f] = \frac{1}{n} \sum_{i=1}^{n} V(f, \mathbf{x}_i, y_i)$. While intuition tells us that the best function to choose should be one that gives the lowest empirical error, $f_S = \underset{f \in \mathcal{H}}{\operatorname{argmin}} I_S[f]$, this often turns out not to be the case. For example, consider the case that $\mathcal{H}$ is the space of polynomial functions. If we choose a training set of 11 data points with $n = 1$ dimensions such that none of the points are inconsistent—i.e. a different y is assigned to the same x—then we can always find a 10^{th} degree polynomial function f_{p11} that will give $I_S[f_{p11}] = 0$ such that every training point is hit precisely (see Figure 1). On the other

[2] Other common loss functions include absolute value or L1 loss $V(f, x, y) = | f(x) - y |$, and Huber loss $V(f, x, y) = \begin{cases} (y - f(x))^2, & \text{for } | y - f(x) | \leq \delta \\ \delta(2 | y - f(x) | - \delta), & \text{otherwise} \end{cases}$. [Hastie01]

hand, a 2nd degree polynomial function f_{p2} yields $I_S[f_{p2}]>0$, and yet the fit is intuitively much more satisfying.

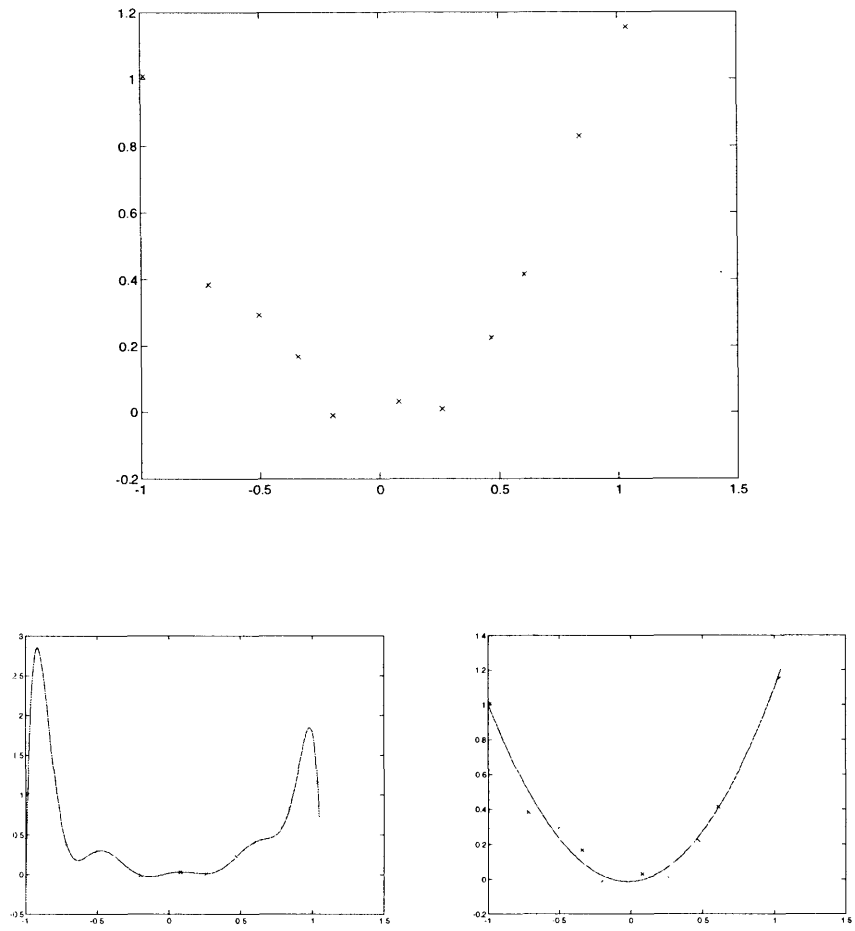

Figure 1 (top) 11 data points, (bottom left) an 11th degree polynomial fit to the points, (bottom right) a 2nd degree polynomial fit to the points

To capture this intuitive notion, we introduce the concept of regularization. Regularization restricts the function space $\mathcal{H}$ using a measure of function complexity $\| f \| : (\mathbb{Z}^n \to \mathbb{R}) \to \mathbb{R}$ called the norm of the function. A norm is a non-negative functional that represents an estimate of the complexity of the parameter function f; a higher norm value represents a more complex function. For example, Ivanov

regularization looks for the function with the best empirical error $f_S = \underset{f \in \mathcal{H}}{\operatorname{argmin}} I_S[f]$ subject to the constraint $\| f_S \| \leq \lambda$. Ivanov regularization forms the basis of our linear regression variable selection technique described in section 4.2.1.

Another type of regularization is Tikhanov regularization. In Tikhanov regularization, the norm term is added as a penalty to the empirical error expression: $f_S = \underset{f \in \mathcal{H}}{\arg\min} I_S[f] + \lambda \|f\|$. This form of regularization is used in support vector regression and regression tree techniques.

Cross validation is a technique used to help predict the quality of generalization of any particular learning algorithm; it is often used, for example, to optimize parameters empirically. Cross validation divides a labeled training dataset S into c equally-sized chunks $S_1,...,S_c$. The learning algorithm is first trained on all the data minus the first chunk $S \setminus S_1$, then tested on the held-out chunk S_1, producing an empirical error value $e_1 = I_{S_1}[f_{S \setminus S_1}]$. This process is repeated c times to calculate $e_1,...,e_c$. The mean and variance of these empirical error values can be used to estimate the performance and variance of the algorithm when applied to new, unlabelled data.

1.2 Introduction to Operating Room Scheduling

The goal of an operating room manager is to keep the OR suite running effectively and efficiently. At the end of the day, success is measured by the overall utilization $\left(= \frac{\text{scheduled hours used}}{\text{total scheduled hours}} \right)$ of the operating rooms, and staff satisfaction [Magerlein78]. Utilization is closely linked to cost efficiency and generally ranges from 60% to 80% at

different institutions [McQuarrie81]. Staff satisfaction is an important factor in the OR suite's effectiveness, although it is much more difficult to measure than utilization. Satisfaction is greatly influenced by hours of overtime—OR utilization that extends beyond the scheduled hours of surgery. Ideally, there would be no frustrating delays or costly dead periods during the scheduled hours of an operating room, but variability is inevitable.

Surgeries are much more highly variable than, say, the clearly defined mechanical steps of a manufacturing plant studied in traditional scheduling applications. OR managers around the country agree that one of the most essential factors in reducing the element of variability in the schedule is to improve the estimate of surgery duration [Hamilton94].

Under the supposition that the labor costs of OR underutilization C_u are different from those of overutilization C_o, [Dexter99] proposes the following asymmetric absolute loss function:

$$V(f, \mathbf{x}, y) = C_u \sum_{i=1}^{n} |\min(0, f(\mathbf{x}_i) - y_i)| + C_o \sum_{i=1}^{n} |\max(0, f(\mathbf{x}_i) - y_i)|$$

While efforts are currently underway at MGH to determine the exact monetary values for C_u and C_o, those values are not yet determined. Instead we assume $C_u = C_o$, and we use the symmetric loss function V_{L2}.

Histograms of surgical duration tend to exhibit a "bell shaped" curve shape with a wide spread, and are usually modeled as Gaussian or log-Gaussian [Strum98,Zhou98]. Variability can be due to any number of real-world causes, some of which are more

difficult to predict than others. Surgeons may employ different techniques to perform a surgery; a particular patient may have unforeseen health complications that require additional procedures; or the support staff may run out of sutures. These are the sorts of delays may be impossible to predict for certain, but rich knowledge about the future surgery can give some hints. For example, one surgeon may be a seasoned expert, while another is still a beginner; a patient with an ASA class of 4 (indicating poor health) is more likely to exhibit health complications than a young and healthy patient in ASA class 1; an operating room with new and up-to-date equipment clearly decreases the chance of delay due to equipment malfunction. While we can not discover the true underlying causation for variance in procedure length from the data alone, machine learning algorithms can help to tease out these important trends.

We can gain considerable information about a surgery by examining its attributes. Certain types of procedures take longer than others; some surgeons are faster than others; and even room location has some influence on how long a surgery will take.

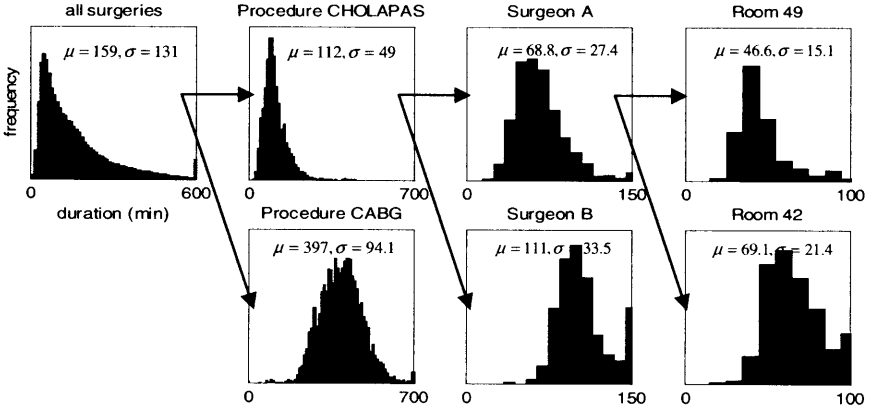

Figure 2 Using more attributes of a surgery, one can reduce the variability of duration estimation. (Left column) the duration histogram of all surgeries in the database. (Left middle column) duration histograms of two common procedure types. (Right middle column) duration histograms for two

surgeons performing the CHOLAPAS procedure. (Right column) Surgeon A performing a CHOLAPAS in two different rooms.

Figure 2 demonstrates that attributes like procedure type, surgeon identity, and room location can be used effectively to decrease variability in prediction. This result is intuitively obvious: the more we know about a future surgery, the better we should be able to predict how long it will take.

2 The Problem

Fortunately, OR suites like the one at the Massachusetts General Hospital have been accumulating rich databases of historical surgical information for years. The MGH dataset includes 304,334 cases from 1993 to 2003. We have extracted some of the more salient variables from the database for use in this study. Variables may be real, with a range of ordered and possibly continuous values, or categorical with discrete and unordered range of possible values. We use the following variables:

Variable	Type	Range	Notes
duration	real	1 – 1385 minutes	We use the other variables to estimate this variable, which is not known until after the surgery occurs. A nurse enters the start and end time of the surgery.
procedure_type	categorical	3,920 procedures	Procedure types are labeled by the OR scheduling and billing staff. Sometimes unscheduled procedures are added to a case, and some error in manual entry is inevitable. Unlabelled surgeries occur more frequently than any other type (74,672 instances), followed by surgeries labeled "OTHER" (36,363 instances). The next most common procedure type is KNEEACSS (5,881 instances).
surgeon	categorical	1,517 surgeons	The primary surgeon
anesthesiologist	categorical	686 anesthesiologists	The primary anesthesiologist
nurse	categorical	2,209 nurses	The head nurse. Some nurse name entries are entered twice with slight misspellings; 2,209 is an overestimate.
team_count	real	0 – 5 support staff	The total nurses and anesthesiologists
surgical_service	categorical	7 services	The department responsible for administering the surgery
date	real	October 10, 1993 to August 8, 2003	The date on which the surgery occurred

Variable	Type	Range	Notes
operating_room	categorical	68 rooms	There are 68 different operating room names
patient_age	real	0 – 195 years	There are a significant number of obviously erroneous ages reported: 47,257 entries report patients older than 170 years.
patient_sex	categorical	F or M	
patient_ASA_class	real	1 – 5	A measure devised by the American Society of Anesthesiologists to classify a patient's morbidity.

Table 1 Summary of the selected variables from the MGH data set.

3 Prior Art

The state of the art algorithm in surgery duration estimation, which we refer to as "proc-surg," proceeds as follows: when scheduling a new surgery, estimate its duration by calculating the average case duration of k "similar" surgeries from the database of historical surgeries. By "similar," we mean that the cases must have the same procedure type and surgeon, and must have occurred recently. Figure 3 gives an implementation of this algorithm in standard query language (SQL).

```
CREATE TEMPORARY TABLE recent_cases (actual_time INT);
INSERT INTO recent_cases SELECT actual_time FROM cases WHERE
  procedure_type=[query procedure type] AND surgeon=[query surgeon]
  ORDER BY date DESC LIMIT [k];
SELECT AVG(duration) FROM recent_cases;
```

Figure 3 The proc-surg duration prediction algorithm written in three lines of standard query language (SQL). The output from the final command is an estimate of the duration for a query case with procedure type and surgeon. *k* is a parameter to to the algorithm determining how many similar surgeries to average.

Proc-surg estimation has a number of nice properties. First, the reasoning behind it is simple and intuitive—no one who has experience with operating room scheduling will deny that the procedure type and surgeon are important factors that influence surgery duration.

Second, proc-surg assigns personal responsibility for scheduling success or failures to the past performance of surgeons. It is generally in a surgeon's best interest to complete their work efficiently in order to minimize underutilization and increase the number of cases she or he may complete in a single day. Past experiments have shown that allowing

surgeons to estimate their own schedules generally leads them to underestimate durations, causing unnecessary case delays and patient waiting.

Third, proc-surg is computationally efficient and simple. A naïve implementation produces an estimate of duration in $O(d + p \log p)$ time for a database of d surgeries, p of which match the query procedure type and surgeon. The look-up step of this algorithm could easily be improved to constant time with a properly-maintained data structure. However, this is would add unnecessary complexity; the naïve algorithm takes only a few seconds to execute with a standard database system—even a large database like the one at MGH. In practice, this algorithm will only need to be run once for each surgery to be scheduled per day; the low computational cost therefore does not necessitate algorithmic optimizations beyond the naïve implementation.

Despite its simplicity, efficiency, and widespread popularity, the proc-surg algorithm, as described above, has three serious flaws. First, the algorithm breaks down when there are insufficient examples of a particular procedure-surgeon combination. At MGH, where schedulers set the parameter $k = 10$ previous surgeries, many of the surgeries on record have fewer than 10 past instances of the same procedure-surgeon combination. A common solution to this problem [Macario99] is to back off on the same-surgeon requirement and examine all procedures of the same type, regardless of the surgeon who performed them. In many cases, this technique is effective, however even some procedure types have few past examples on record.

This second flaw is surprisingly prevalent in practice. Of the 3908 different procedure types in the MGH database, 2348 of them occur fewer than 10 times. It is therefore very

difficult to predict how long such a rare surgery may take. One common suggestion is to assume that similar procedure types have similar case durations. The question, then, is how to define similarity for procedure types. [Dexter99] points out that a vitrectomy procedure takes more than one hour longer than a scleral buckle procedure, even though their standard CPT codes differ by only one (67108 and 67107, respectively). In general, it is very difficult to predict whether two different procedures on the same part of the body, with similar techniques or with similar goals will have the same average duration. This flaw is inherent in the fact that new procedure types enter into practice and there is simply not enough data to estimate their length accurately.

Third, proc-surg does not make use of very much of the data that is available in the database. Consequently, proc-surg is simple to implement and understand, but, as we will see, it is not the most accurate estimator of surgery durations. Despite its limitations, the proc-surg prediction algorithm yields a dramatic improvement over a strategy that uses no specialized information over the surgery and simply reports the mean surgery time for all surgeries (see Table 2).

	Average	**Proc-surg**
square root of mean squared duration estimation error (minutes), ± standard deviation	**127.7 ± 1.6**	**74.0 ± 3.5**

Table 2 Comparison of surgery duration estimation algorithms. Algorithms are applied in 10-way cross-validation on all examples from the 27 procedure types that occur in the database at least 1000 times. "Average" uses no specialized information to estimate surgery duration; it always reports the mean surgery time computed over the whole sample set. Proc-surg works as described above, backing off from matching procedure-surgeon combinations to just procedures when there are not enough (≥10) examples.

4 Learning Algorithms

4.1 Nearest Neighbors

When they invented the proc-surg algorithm, OR managers inadvertently rediscovered a very popular supervised machine learning technique called nearest neighbors. Proc-surg is, in fact, just one particular instance of the more general nearest neighbor algorithm. The intuition behind nearest neighbors is that any particular point is likely to have a similar value to points that are close by; we can estimate a label for an unknown point simply by using the label from the nearest neighbor. To add some robustness, nearest neighbor algorithms often average the labels of the k nearest neighbors.

The effectiveness of the nearest neighbor algorithm is entirely dependent on the definition of "nearness." Proc-surg gave us one particular method for determining how "near" two surgeries are: they are near if they have the same procedure type, they are nearer if they also have the same surgeon, and they are nearer still if they occur temporally close to one another. Nearest neighbors uses a nonnegative metric $\rho(\mathbf{x}_1, \mathbf{x}_2) : \mathbb{Z}^n, \mathbb{Z}^n \rightarrow \mathbb{R}$ in order to judge the similarity of a pair of data points. A metric has three fundamental properties:

1. $\rho(\mathbf{x}_1, \mathbf{x}_2) = 0$ iff $\mathbf{x}_1 = \mathbf{x}_2$

2. symmetry: $\rho(\mathbf{x}_1, \mathbf{x}_2) = \rho(\mathbf{x}_2, \mathbf{x}_1)$, and

3. triangle inequality: $\rho(\mathbf{x}_1, \mathbf{x}_3) \leq \rho(\mathbf{x}_1, \mathbf{x}_2) + \rho(\mathbf{x}_2, \mathbf{x}_3)$

For real-valued input vectors ($\mathbf{x} \in \mathbb{R}^n$), a common similarity metric is Euclidean distance $\rho_E(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{n} (x_{1i} - x_{2i})^2$. Euclidean distance is clearly symmetric and zero for $\mathbf{x}_1 = \mathbf{x}_2$ (and positive otherwise). Furthermore, [Kolmogorov70] proves that ρ_E satisfies the triangle inequality.

If the variables have many different units and types, the Euclidean distance makes an arbitrary assumption: units are scaled to reflect the actual similarity of pairs of data points. Consider, for example, patient Smith of ASA class 2 and age 30. It is difficult to decide whether she should be more similar to patient Jones of ASA class 3 with age 30, or patient Doe of ASA class 2 and age 40. If an ASA class difference of 1 unit is valued as more similar, then Jones is more like Smith; if, on the other hand a difference of 10 years in age is valued as more similar, then Doe is more like Smith. We formalize this type of difference by introducing a vector $\mathbf{w}$ of weights for each variable. The new similarity metric becomes $\rho_w(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{n} w_i (x_{1i} - x_{2i})^2$. As in the example of Smith, Jones, and Doe, it is often unclear exactly what $\mathbf{w}$ should be.

Euclidean distance works for real-valued variables, but two of the three variables used by proc-surg are not real—both procedure type and surgeon are categorical, but dates are real-valued. The proc-surg algorithm uses a special metric for dealing with these categorical variables; they introduce a "difference" function $d_{\mathbb{C}} : \mathbb{C}, \mathbb{C} \to \mathbb{R}$ analogous to the squared difference used in the Euclidean similarity metric: $d_{\mathbb{C}}(x_1, x_2) = \begin{cases} 0 \text{ if } x_1 = x_2 \\ 1 \text{ if } x_1 \neq x_2 \end{cases}$, where $\mathbb{C}$ is a variable with unordered, categorical values. The new categorical metric

function then becomes $\rho_{\mathbb{C}}(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{n} w_i d_{\mathbb{C}}(x_{1i}, x_{2i})$. It is easy to see that this definition satisfies both requirements 1 and 2 for a metric function. Furthermore, we can see that requirement 3 (triangle inequality) holds by breaking each categorical variable v into m different binary-valued variables $\delta_1, ..., \delta_m$, each representing one of the m different values of that v can take on; we call these variables "dummy" variables. When v takes on the ith categorical value, δ_i is set to 1 and the remaining $m-1$ variables are set to 0. If the each variable in the original input vector is replaced with dummy variables, then the new distance metric becomes a Euclidean distance, which we already know satisfies the requirements of a metric function. We will revisit dummy variables later in section 4.2.

Introducing an analogous difference function for real variables, $d_{\mathbb{R}}(x_1, x_2) = (x_1 - x_2)^2$, we write the final mixture real/categorical similarity metric: $\rho_{\mathbb{R},\mathbb{C}}(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{n} w_i d(x_{1i}, x_{2i})$. Now we have the technical tools necessary to treat mixtures of categorical and real variables in the same framework.

4.1.1 Results

	Average	Proc-surg	Proc-surg-ASA-age	Proc-surg-anes	Proc-surg-room	Proc-surg-sex
square root of mean squared duration estimation error (minutes), ± standard deviation	**49.0 ± 3.1**	**40.7 ± 3.0**	**39.1 ± 3.3**	**39.0 ± 5.4**	**38.7 ± 3.3**	**39.1 ± 4.3**

Table 3 Comparison of surgery duration estimation algorithms. Algorithms are applied in 10-way cross-validation on surgeries with procedure type CHOLAPAS.

4.1.2 Discussion

Since there is no learning phase in the nearest neighbor algorithm, it is particularly well-suited to an application where the data is already stored in a convenient database format. Each prediction requires a search through all the entries in the database, which may seem unreasonably slow, but we have found empirically that it takes no more than a few seconds maximum. Despite all of the work to reduce the look-up time for nearest neighbor algorithms, the naïve implementation of this algorithm seems more than sufficient for the frequency that it is used.

4.1.3 Future Work

Table 3 presents a number of extensions to the standard proc-surg algorithm involving other variables. Each of these variants yield a slight improvement over the standard proc-surg algorithm, but the advantage is insignificant. It would be worthwhile to perform a full search over all possible prioritizations of variables.

Until now, we have only considered cases in which one variable has a strictly higher or lower precedence in computing the distance function. In fact, our weighting model is more general than that. It allows for more subtle combinations of weights, mixing the priorities of weights in the similarity metric. This fact makes a full manual search over variable weights infeasible, but [Lowe95] presents a method of learning weights using a conjugate gradient descent to minimize the squared error on a given training set.

4.2 Linear Regression

Historically, the first known supervised learning algorithm was an instance of what we now call linear regression. This is probably because linear models are simple to evaluate and efficient to compute from data. For a data set of real variables, a linear model is of

the form $f_{lin}(\mathbf{x}_i) = \beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}_1$, where β_0 is the value of f_{lin} when $\mathbf{x}_i = 0$, and $\boldsymbol{\beta}_1$ defines the weighted contribution of each variable to f_{lin}; a positive value of $\boldsymbol{\beta}_1$ implies positive correlation for that variable, and vice versa. As usual, we use the L2 (square) loss function to measure prediction error, and we wish choose the values for β_0 and $\boldsymbol{\beta}_1$ that minimize the empirical error for a given data set S: $\arg\min_{\beta_0, \boldsymbol{\beta}_1} I_S[f_{lin}] = \arg\min_{\beta_0, \boldsymbol{\beta}_1} 1/n \sum_{i=1}^{n} V(f_{lin}, \mathbf{x}_i, y_i)$. We reformulate this problem with matrices to reach an analytical solution. First, we combine the parameters into a single vector $\boldsymbol{\beta} = [\boldsymbol{\beta}_1^T \quad \beta_0]^T$. Next, we rearrange the input data into the rows of a single matrix with ones corresponding to the β_0 parameter: $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \dots & \mathbf{x}_n \\ 1 & \dots & 1 \end{bmatrix}^T$. Finally, we put all of the true output labels into a single column vector: $y = [y_1 \quad \dots \quad y_n]^T$. Now our objective problem becomes $\arg\min_{\boldsymbol{\beta}} I_S[f_{lin}] = \arg\min_{\boldsymbol{\beta}} 1/n (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$. To solve this function, we set the derivative with respect to $\boldsymbol{\beta}$ to zero and solve for the optimal value $\hat{\boldsymbol{\beta}}$ that minimizes $I_S[f_{lin}]$: $\frac{\partial}{\partial \boldsymbol{\beta}} I_S[f_{lin}] = \mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = 0$. Finally, we get $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ if $\mathbf{X}^T\mathbf{X}$ is nonsingular. $\mathbf{X}^T\mathbf{X}$ will be singular when there are fewer than $n+1$ independent data points, in which case there is no unique solution; given the amount of available surgical data, the possibility of singular values are extremely unlikely and can be safely ignored. $\hat{\boldsymbol{\beta}}$ can be computed using standard off-the-shelf linear algebra software to calculate the matrix inversion (using singular value decomposition).

As with nearest neighbors, least squares regression is designed only to handle categorical variables. Unlike nearest neighbors, however, linear regression does not rely on a similarity metric between data points. Rather, linear regression requires each variable to have an actual value such that it may be multiplied by the $\hat{\boldsymbol{\beta}}$ coefficients. We use a technique described in section 4.1 to convert each categorical variable into a set of "dummy" variables, one for each value that the categorical variable may take on. We can now compute a linear model for a set of input data that is entirely real.

While nearest neighbors required weighting values to be calculated externally, linear regression explicitly searches for optimal weights. The $\hat{\boldsymbol{\beta}}$ coefficients may be interpreted as the contribution from each variable (positive values increase the output and negative variables decrease it), and $|\hat{\boldsymbol{\beta}}|$ may be used to estimate the overall importance of these variables. The coefficients from an analysis of cases with procedure type "CHOLAPAS" (Figure 4) show that the dummy variable corresponding to one particular surgeon (surgeon "A") dramatically reduces the estimated time for surgery, while a patient with a higher ASA class (morbidity rating) is likely to have a longer surgery.

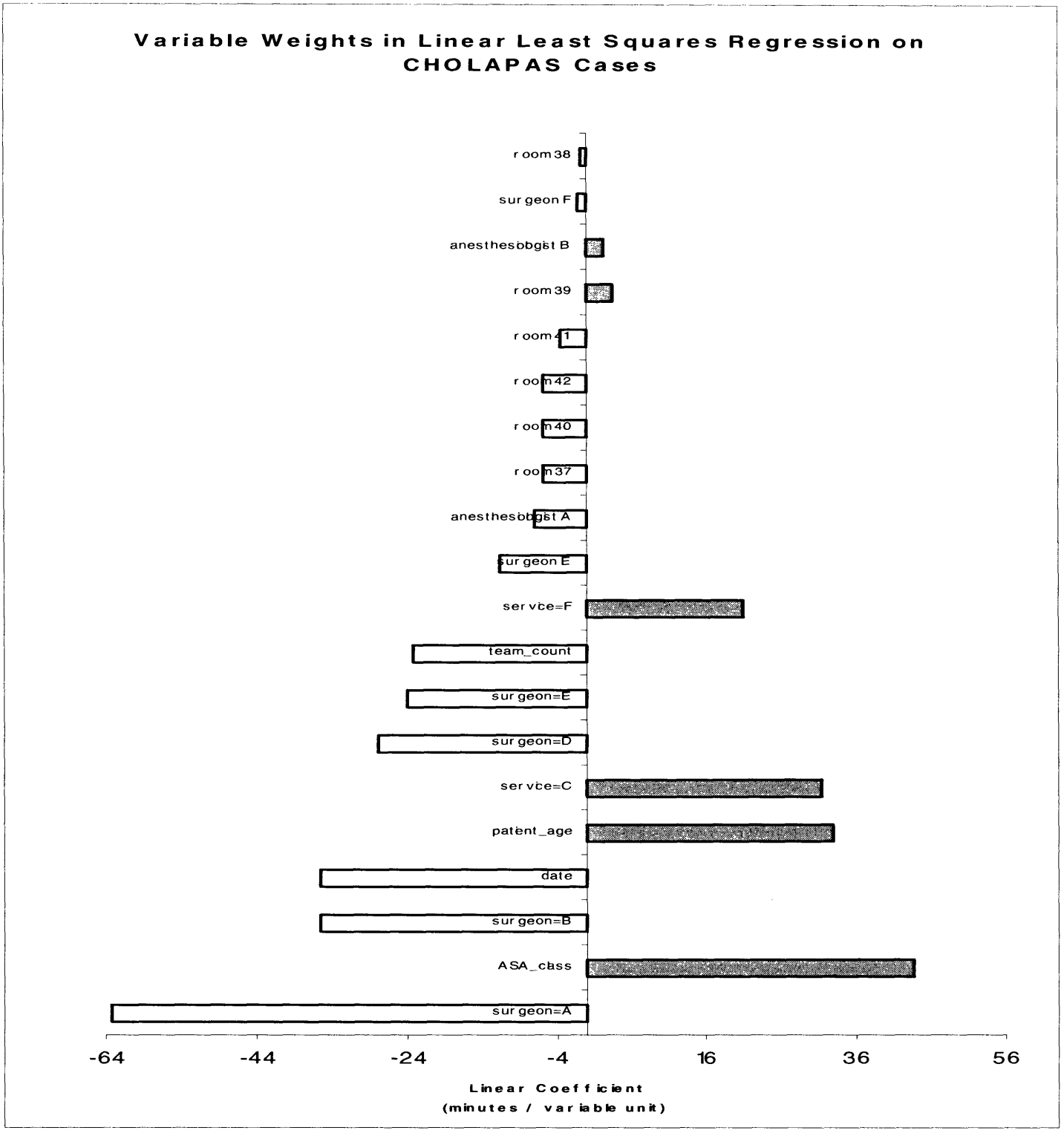

Figure 4 Coefficients for a model the duration of "CHOLAPAS" cases. Note that ASA_class and patient_age behave as we would intuitively expect (a sicker, older patient is likely to have longer surgeries), as does date (surgeries get faster with time and practice), while team_count runs counter to our intuition that "too many cooks spoil the broth." We also see the significant effect that individual surgeons can have on the case time.

4.2.1 Variable Selection

The weights depicted in Figure 4 would not have been possible to compute without some sort of variable selection. When all of the categorical variables in our database are expanded into dummy variables, the dimensionality is increased from a manageable 12 to an unwieldy 8,418, without changing the information content, this increase in dimensions essentially increases the complexity of our model and, unless properly checked, it leads to overfitting.

Variable selection techniques are in commonly used to reduce the number of dimensions in a model, through a process called variable selection. Aha [Aha95] compares the empirical merits of three such techniques: forward selection, backward selection, and beam search. In all three of these techniques, the variable selection task is treated as a search through the space of subsets of variables.

Forward selection is a greedy algorithm that starts with an empty set of variables, and iteratively adds the variable that most increases the algorithm's prediction performance. The process continues until the addition of any of the unused variables degrades performance (i.e., leads to overfitting). Performance is measured in the standard way using cross validation on the training set.

Backward selection works much like forward selection except that it starts with the set of all variables and iteratively removes variables until all of the remaining variables contribute positively to the algorithm's performance. Like forward and backward selection, beam search iteratively adds or removes variables, but beam search is not entirely greedy. Beam search maintains a queue of the q most promising variable

subsets, and iterates on each of the members of the queue. Aha shows that beam search can be significantly better than forward or backward selection.

With the number of dummy variables in our application, these techniques are prohibitively expensive, requiring an estimated 5,000 applications of cross-validation to select a reasonable set of variables. Instead, we employ a new method tailored to the use of dummy variables. We notice that the more frequently-occurring variables are the ones that tend to have the most stable absolute weights. For example, a surgeon who has only performed one surgery before is likely to perform erratically on her or his surgeries until she or he reaches a stable routine. We therefore choose to include only those dummy variables that are true (i.e. they equal 1) with a frequency greater than some minimum threshold. Formally, we define $\|f_{lin}\|$ to be the minimum frequency threshold, and we perform Ivanov regularization under the constraint that $\|f_{lin}\| \geq \lambda$. Figure 5 demonstrates the effect of varying the minimum frequency threshold parameter λ.

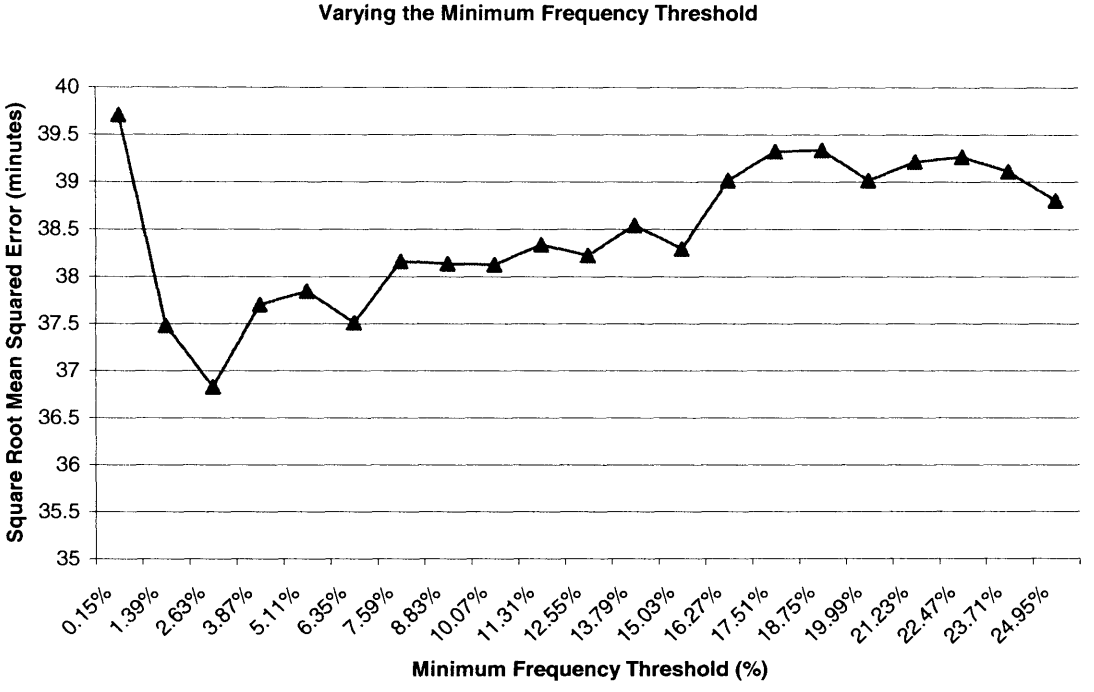

Figure 5 The effect of varying the minimum frequency threshold for dummy variable selection. Linear regression performs best when dummy variables that are true less than around 2.5% are rejected.

4.2.2 Results

	Average	Proc-surg	Linear
square root of mean squared duration estimation error (minutes), ± standard deviation	54.4 ± 4.5	49.8 ± 6.0	41.7 ± 4.9

Table 4 Comparison of surgery duration estimation algorithms. Algorithms are applied in 10-way cross-validation on surgeries with procedure type CHOLAPAS.

4.2.3 Discussion

While linear regression works reasonably well, it has some obvious pitfalls. For example, if doctor A is fast at procedure type B but slow at procedure type C, the linear model is still forced to select a single overall rating for the contribution due to one particular surgeon. To avoid this problem, one could add still more variables representing all the pairwise "ands" between each of the dummy variables. However,

due to the difficulties caused by introducing so many dummy variables already, this would not seem wise. Another possible fix for this problem is to combine linear models with tree-based models (see section 0).

One nice feature of linear models is the fact that the importance of different variables is easy to visualize. Graphs like Figure 4 can actually give O.R. managers some insight as to the main causes of variability in their rooms.

4.2.4 Future Work

An alternative approach to variable selection, whose implementation is left for future work, is principal component analysis (PCA). The idea behind principal component analysis is that data can be projected onto orthogonal vectors that point in the directions of highest variance. Since the first few of these orthogonal vectors capture most of the variance of the data, we can ignore the rest. The principal components of the data are just the eigenvectors of the variance. Principal components with the largest corresponding eigenvalues capture the most of the variance in the data; those with the lowest eigenvalues can be safely discarded. The principal components may be calculated using singular value decomposition. Note that PCA provides a more principled approach to the intuition behind our approach: lower-variance dimensions are less important in learning.

Regularization of linear regression can also be employed to reduce the effects of overfitting. If we penalize large slopes, i.e. $\|f_{lin}\| = \|\boldsymbol{\beta}\|$, in Tikhanov regularization the linear model will be more robust [Rifkin02].

4.3 Regression Trees

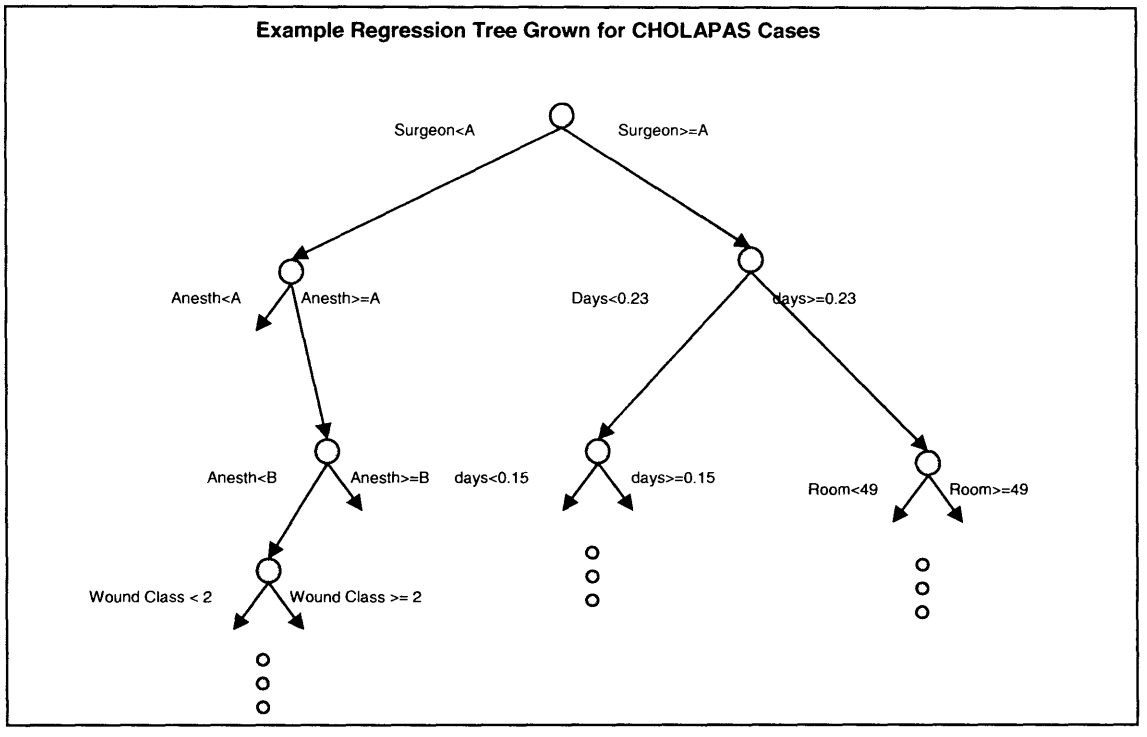

Figure 6 Regression trees recursively split the training data with binary partitions. Each node of a decision tree tests a condition on a single input variable. Each leaf contains a constant prediction of the real-valued output. Nodes are chosen to minimize prediction error.

Regression trees are easy to interpret and efficient to calculate. They are particularly favored in the medical community because they seem to mimic the style of decision-making doctors usually make.

The function of a regression tree is straightforward. Regression trees partition the range of input variables into discrete regions; each region is assigned a constant predictor value. Each unlabelled input data point is labeled according the constant associated with its region.

The structure of a regression tree facilitates the rapid determination of which region a point should be assigned to (see Figure 6). Each non-leaf node in the tree contains a binary condition on a single variable. For real variables, the condition tests whether the

value in the incoming data is below some threshold; for categorical variables, the condition tests whether it equals some particular value. If the condition is true, the incoming data is passed down the left branch; if false, it passes to the right. When the incoming data reaches a leaf node, the constant value of that leaf node is returned.

Theoretically, we may view a regression tree simply as a piecewise-constant function with ℓ regions $R_1,...,R_\ell$ and ℓ constants $c_1,...,c_\ell$ corresponding to each of the ℓ leaves in the tree. When $\mathbf{x} \in R_i$, the regression function is $f_{RT}(\mathbf{x}) = c_i$.

How do we choose a good regression tree? For categorical input variables, the number of trees to consider is exponential. For real variables, the number is infinite. We use a greedy algorithm to make the search computationally feasible.

At each iteration, we choose the binary split that minimizes prediction error. We choose a variable v to split, then we choose a partition c of the range of v on which to condition the data. For real v, c is a real value, and the data is split into the regions $R_{true}(v,c) = \{\mathbf{x} \mid x_v \geq c\}$ and $R_{false}(v,c) = \{\mathbf{x} \mid x_v < c\}$, while for categorical v, c is a set of values, and the data is split into the regions $R_{true}(v,c) = \{\mathbf{x} \mid x_v \in c\}$ and $R_{true}(v,c) = \{\mathbf{x} \mid x_v \notin c\}$). In particular, we wish to find, for leaf regions $R_1,...,R_\ell$:

$$\underset{R \in \{R_1,...,R_\ell\},v,c}{\arg\min} \left(\sum_{x_i \in R \wedge x_i \in R_{true}(v,c)} (y_i - \overline{y}_{x_i \in R \wedge R_{true}(v,c)})^2 + \sum_{x_i \in R \wedge x_i \in R_{false}(v,c)} (y_i - \overline{y}_{x_i \in R \wedge R_{false}(v,c)})^2 \right),$$

where $\bar{y}_R$ is the average label for points in region R. Note that by the definition of expectation, the best predictor for a set of data (i.e. the constant label that gives the lowest mean squared error for all points in that set) is the mean value.

Initially, the number of possible splits appears intractable; for real variables, the number of possible splits to consider is infinite, and for a categorical variable with m possible values, the number is exponential: $2^{m-1}-1$.

We avoid the infinite possible split points of real-valued variables by observing that they need only be split at values covered by the training data; all other splits on real variables are indistinguishable and redundant. Furthermore, we can reduce the number of splits for a categorical variable linear by sorting the categorical values in order of their average corresponding label within the region R of interest. [Breiman84] shows that one can arrive at the optimal split by testing the partitioning of this ordered sequence of categorical values in each of the $m-1$ possible ways.

The only question left is when to stop. We could keep dividing our data up until there is only one data point at each leaf, but this would be nearly the same as calculating k nearest neighbors where $k=1$, which we know is not very robust to error and exhibits overfitting. This is an opportunity to employ a regularization metric $\| f_{RT} \|$. In this case, we wish to find a tradeoff between empirical error $I_S = \frac{1}{n}\sum_{i=1}^{n} V_{L2}(f_{RT}, \mathbf{x}_i, y_i)$ $= \frac{1}{n}\sum_{i=1}^{\ell}\sum_{x_j \in R_i}(y_j - \bar{y}_{R_i})^2$ and tree size $\|f_{RT}\| = \ell$. We then search for the parameter λ that minimizes the Tikhanov regularization expression $I_S + \lambda\|f_{RT}\|$. Each successive split that is added to the tree decreases I_S and increases $\|f_{RT}\|$ such that each tree in a

sequence of n regression trees $T_1,...,T_n$ has a λ associated with it (note that the nth tree has a leaf for each data point). We determine a value for λ that generalizes well and avoids overfitting by cross-validation on the training set, then we use that value for λ to compute the final tree using all of the training data.

4.3.1 Results

	Average	Proc-surg	Linear	Regression Tree
square root of mean squared duration estimation error (minutes), ± standard deviation	**49.9 ± 4.5**	**43.8 ± 6.0**	**36.8 ± 4.9**	**42.9 ± 3.3**

Table 5 Comparison of surgery duration estimation algorithms. Algorithms are applied in 10-way cross-validation on surgeries with procedure type CHOLAPAS.

4.3.2 Discussion

There seem to have been several arbitrary design decisions in creating the regression tree algorithm. First, why does the algorithm use only binary splits? The answer is that larger splitting factors tend to reduce the amount of training data in each branch too quickly; furthermore, they would make our representation over-complete in the sense that any n-way split can be represented by $n-1$ binary splits.

Second, why not allow for more complicated decision boundaries? For example, one could follow the left branch when $\sum_i a_i x_i < c$ and the right branch otherwise. While this has the potential to improve our model, this would require a computationally intensive search over continuous values of a_i with only a finite set of $\mathbf{x}$ points at any given branch point.

Regression trees have several nice properties that make them an attractive option for use in practical applications.

First, unlike linear regression, variable selection is built into the structure of a regression tree. Since one single variable is used at each split in the tree, all other variables are explicitly ignored. Regression trees are one of the only algorithms in which variables are chosen individually on their ability to increase predictive power of the model. Consequently, the input data can have a high proportion of noisy (or completely irrelevant) variables, and they will be automatically ignored.

Second, compared to all other techniques we discuss, regression trees deal elegantly with categorical variables without necessitating a full expansion into dummy variables or a combinatoric search over all combinations of these variables.

Third, as mentioned earlier, trees are particularly favored by medical practitioners because of their intuitive structure and clear meaning. Like the structure of disease-diagnosis manuals, the line of questioning implicit in a regression tree is reminiscent of a doctor's own thought process when confronted with a tricky medical problem. While the actual predictive power of the algorithm does not match linear regression, interpretability can be more comforting to the surgeons whose daily schedule (and pay) is determined by the algorithm.

4.4 SVM Regression

Support vector machines have recently become a favorite among machine learning researchers [Vapnik98]. Usually used for the binary classification task, the support vector machine framework is extended naturally to predict real-valued outputs for

regression. Support vector machines are favored because they are theoretically sound, require few external parameters, and are relatively efficient to compute.

As with binary support vector machines, support vector regression gains its computational efficiency from the observation that only a subset of the data points (called support vectors) are crucial in determining the overall shape of the predicted function. The remaining points are not used in computing the function for future prediction of labels and may be safely discarded.

Support vector regression discriminates between support vectors and unimportant data points using the epsilon-insensitive loss function: $V_\varepsilon(f, \mathbf{x}, y) = \begin{cases} 0 & \text{if } |t| < \varepsilon \\ |t| - \varepsilon & \text{otherwise} \end{cases}$. V_ε discounts points that are within a tube of radius ε around the predicted function. All training data points that lie in this range can be safely discarded. Points outside this tube provide "support" for the shape and position of the regression function.

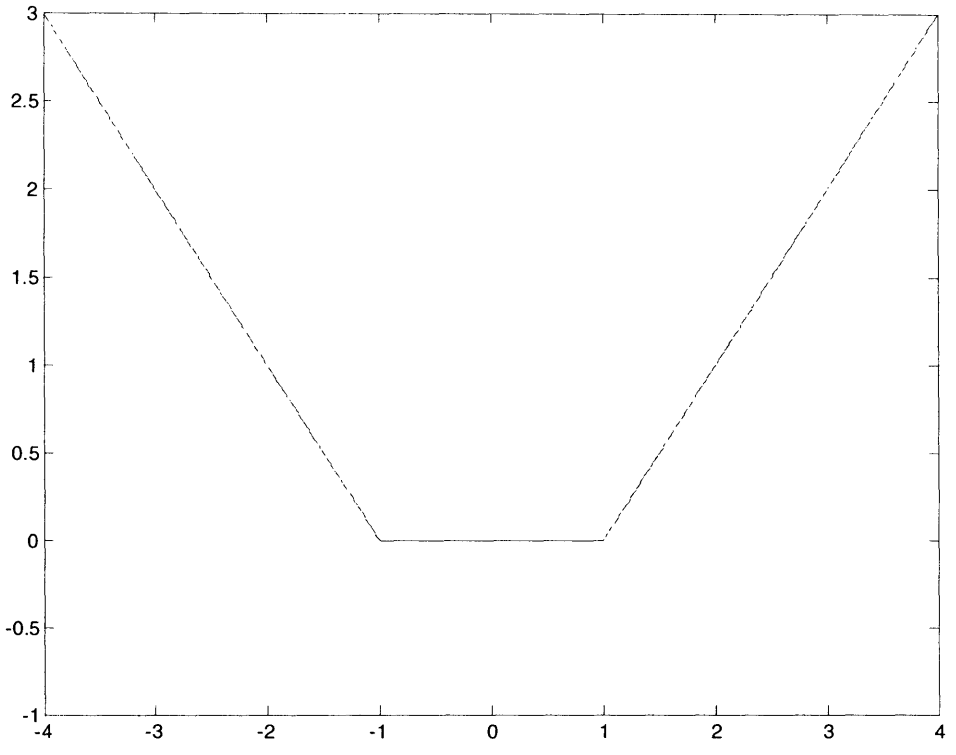

Figure 7 The epsilon-insensitive loss function V_ε plotted over the difference $y - f(\mathbf{x})$ provides support vector regression with its speed.

Initially, we give no explicit formulation of the function space $\mathcal{H}$ other than the constraint that it be a reproducing kernel Hilbert space (RKHS) (defined precisely in [Kolmogorov57]). Support vector regression is essentially a Tikhanov regularization problem in which the function norm is the RKHS kernel $\|f_{SVR}\| = \|f_{SVR}\|_K^2$:

$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} V_\varepsilon(f_{SVR}, \mathbf{x}, y_i) + \lambda \|f_{SVR}\|_K^2$. The representer theorem [Rifkin02] lets us derive the form of the optimal function in the space: $f_{SVM}(\mathbf{x}) = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) K(\mathbf{x}, \mathbf{x}_i) + \beta_0$, where $K(\mathbf{x}, \mathbf{x}_i) : \mathbb{R}^n, \mathbb{R}^n \to \mathbb{R}$ is the kernel product between the input (unlabeled) data point, and the ith training point. In the case where the kernel function is simply the dot product, $K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$, we note that the optimal support vector regression function takes the

same form as the linear regression function $f_{SVM}(\mathbf{x}) = f_{lin}(\mathbf{x}) = \mathbf{x}^T\beta_1 + \beta_0$ (see section 4.2), where $\boldsymbol{\beta}_1 = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)\mathbf{x}_i$. Note, however, that linear regression uses a different loss function (and does not even use a norm for regularization), so the two formulations will generally give different optimal values for $\boldsymbol{\beta} = [\boldsymbol{\beta}_1^T \quad \beta_0]^T$. In particular, support vector regression seeks a solution to the quadratic programming problem

$$\arg\min_{\alpha_i,\alpha_i^*} \varepsilon\sum_{i=1}^{n}(\alpha_i^* + \alpha_i) - \sum_{i=1}^{n} y_i(\alpha_i^* - \alpha_i) + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to the constraints $0 \le \alpha_i \le 1/\lambda$, $0 \le \alpha_i^* \le 1/\lambda$, $\sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0$, and $\alpha_i\alpha_i^* = 0$. These constraints make it likely that $(\alpha_i^* - \alpha_i) = 0$ for many of the training points—points that are not support vectors and may be discarded when applying f_{SVM} to unlabeled data points [Hastie01].

Some kernels for $\mathcal{H}$ effectively cast the input data into a higher-dimensional space. For example, consider a d th degree polynomial kernel $K(\mathbf{x}_1, \mathbf{x}_2) = (1 + \mathbf{x}_1 \cdot \mathbf{x}_2)^d$. For $d = 2$, we calculate $K(\mathbf{x}_1, \mathbf{x}_2) = 1 + 2\mathbf{x}_1 \cdot \mathbf{x}_2 + (\mathbf{x}_1 \cdot \mathbf{x}_2)^2$. Although these terms are not explicitly computed, the kernel product is implicitly treating a higher-dimensional input space. Linear functions in this higher dimensional space are no longer linear when the space is recast to its original input dimensionality. Searching over values for the parameter d, we find that choosing a polynomial degree that is too high can lead to overfitting, but the

basic linear polynomial is not quite expressive enough; $d = 2$ give the minimum the error rate (Figure 8).

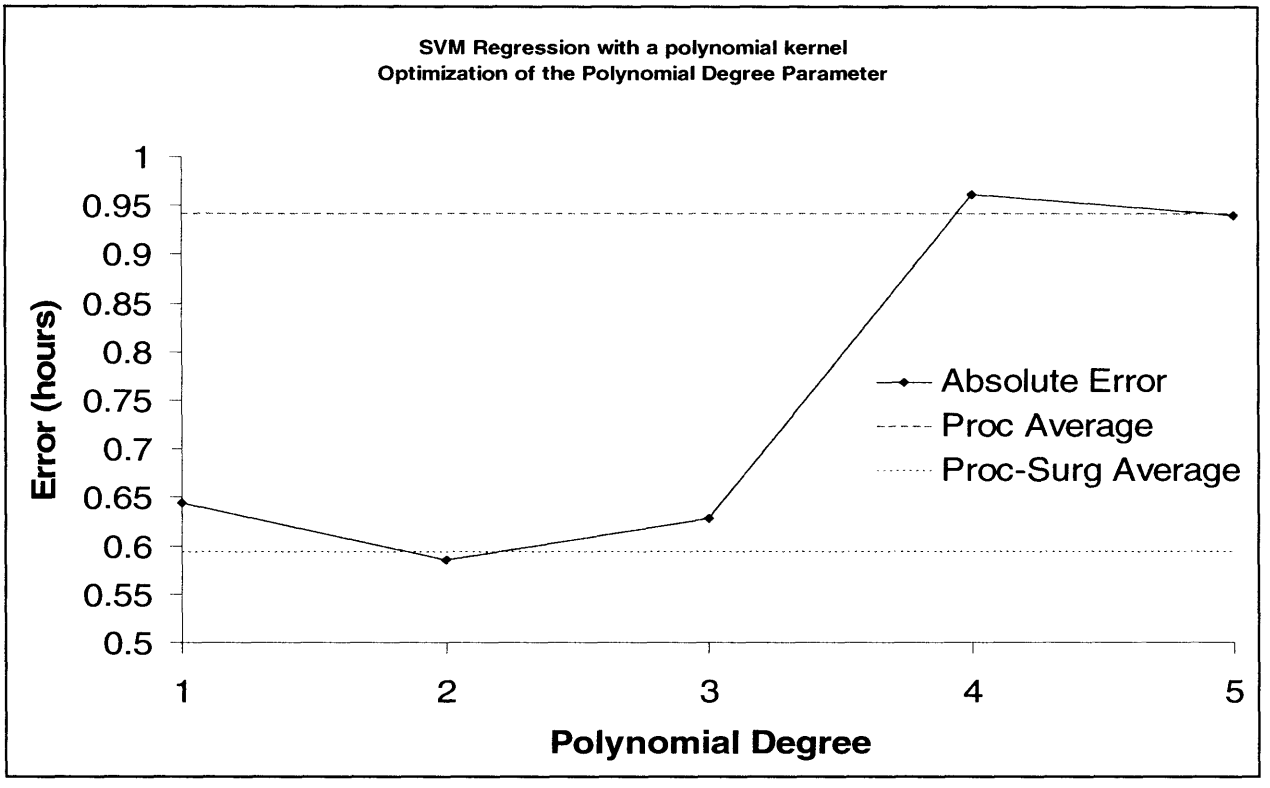

Figure 8 Support vector regression with a polynomial kernel. Error rate vs. polynomial degree.

Another common kernel is the Gaussian or radial basis kernel: $K(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2 / \gamma)$. Considering the series expansion of this function, the Gaussian kernel is often said to cast data points into an infinite dimensional space. Once again, there is a single parameter to optimize over: γ (see Figure 9).

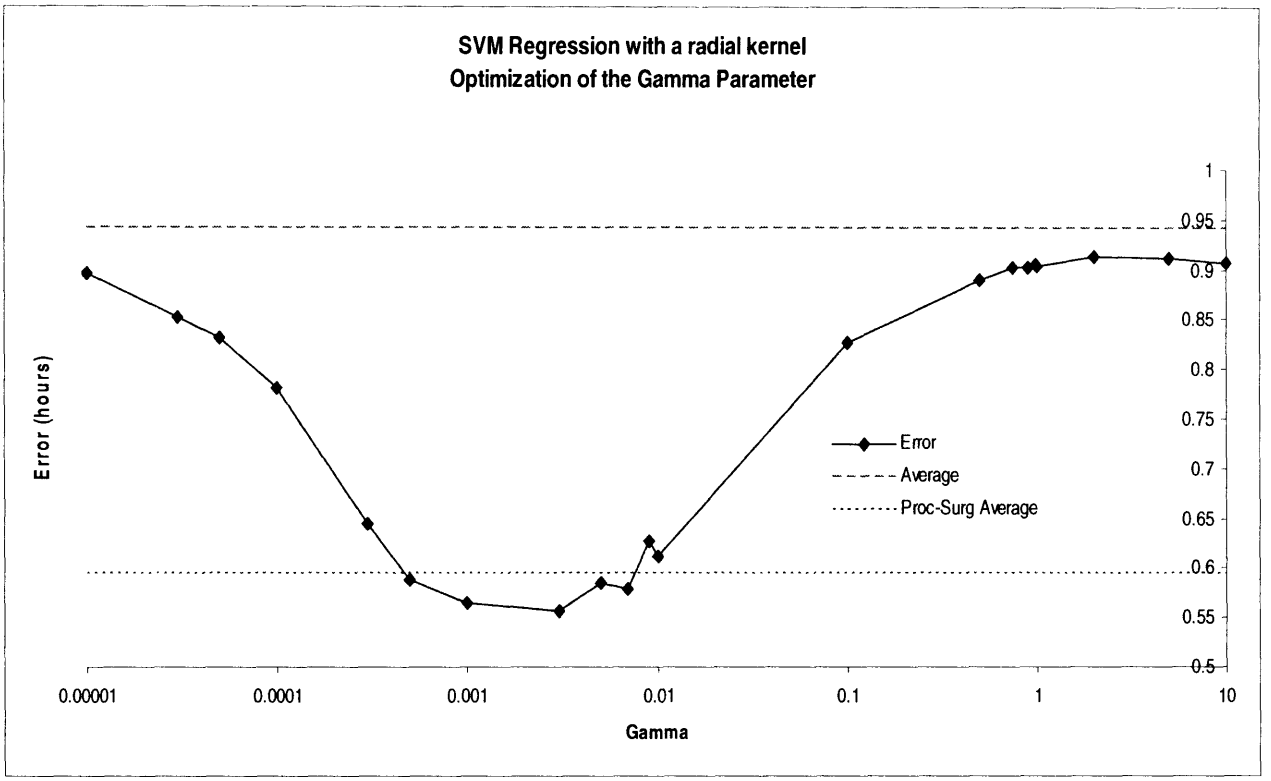

Figure 9 Support vector regression with a Gaussian kernel. Error rate vs. polynomial degree.

4.4.1 Results

	Average	Proc-surg	Linear	Regression Tree	SVM Polynomial	SVM Gaussian
square root of mean squared duration estimation error (minutes), ± standard deviation	49.9 ± 4.5	43.8 ± 6.0	36.8 ± 4.9	42.9 ± 3.3	40.3 ± 5.7	38.0 ± 5.5

Table 6 Comparison of surgery duration estimation algorithms. Algorithms are applied in 10-way cross-validation on surgeries with procedure type CHOLAPAS. As with linear regression, SVM regression trials vary with the minimum "dummy" variable true rate. Polynomial and Gaussian kernel SVM reach their optimal predictive power using minimum frequencies of 23.71% and 6.35%, respectively.

4.4.2 Discussion

Despite the recent excitement they have generated, support vector machines for regression did not predict as accurately as standard least squares linear regression (though

the Gaussian kernel came close). Furthermore, support vector regression functions are notoriously difficult to visualize—the weights combine large numbers of input training points and cast them into high (or infinite) dimensional kernel space. They also suffer from the same problem as linear regression: all inputs must be real, and the input dimensionality becomes saturated when every possible categorical value becomes a new dimension. Though they perform second best of the algorithms we compare, support vector machines are least likely to be adopted by practicing O.R. schedulers and surgeons.

5 Discussion

> *"An 'off-the-shelf' method is one that can be directly applied to the data without requiring a great deal of time consuming data preprocessing or careful tuning of the learning procedure."* [Hastie01]

We have applied several standard supervised machine learning algorithms to the domain of surgical duration prediction. Considering this problem to be one particularly representative example of a real-world machine learning application, we offer criticism of the supervised machine learning state of the art.

Of the six learning algorithms compared in this thesis, two of them may reasonably be considered "off-the-shelf"—linear and SVM regression. Implementations of singular value decomposition for linear least squares are reliable, fast, and freely available [Jmat04]. And several free packages, such as SVM^{light} [SVMlight04] are available for SVM regression Despite the promise of general learning packages such as YALE [Joachims98] and Weka [Witten99], "standard" algorithms were not yet implemented: regression trees and nearest neighbors. Furthermore none of the learning toolkits we investigated had support for automatically treating categorical variables as "dummy" variables.

Though fast computing clusters were available, we opted to train and test on a standard desktop machine comparable to what would be available to most practitioners. We found that processor speed and memory limited training of the data set. The entire dataset was well over the 350 megabytes of available RAM, and expanding categorical variables to "dummy" variables further exacerbated the memory shortage. Future use of this algorithm should make use of a sparse matrix representation to lower the memory

footprint. While we implemented conjugate gradient descent over nearest neighbor weightings [Lowe95], we never finished an entire run because it was too time-intensive. Even using this reduced training set, many of the other algorithms took several days or a week to complete a cross-validation experiment.

6 Future Work

The literature describes many extensions and permutations of the algorithms tested in this thesis. One of the most promising extensions is the multiple additive regression trees algorithm (MART), which combines the output of several small regression trees using boosting. This technique retains the ease of application of regression trees, and proponents claim that it dramatically increases the predictive power.

Another fruitful area of exploration is to continue testing the space of nearest neighbor weightings using Lowe's nearest neighbor weight conjugate gradient descent algorithm [Lowe95]. This algorithm is particularly interesting because it is a direct generalization of the current state of the art proc-surg algorithm.

On the practical side, it would be useful to the schedulers at MGH also to have an accurate prediction of the turnover time before and after each surgery. Our prediction system predicted from the time the patient entered the O.R. to the time she or he left. We believe turnover time prediction would be fairly straightforward to implement because we already have identified many of the important variables (such anesthesiologist, room number, and team count).

We have also begun an initiative at MGH to collect and incorporate the surgeon's estimations of duration and case complexity into the prediction system, as suggested by [Wright96]. The MGH scheduling system used to rely entirely on surgeon's predictions (rather than historical times), but the proc-surg algorithm was introduced to eliminate surgeon's inherent underestimation bias. Zhou demonstrates that incorporating both information sources can improve overall prediction accuracy [Zhou99].

Given that we have found a significant improvement in prediction accuracy when testing on a subset of the data, we should use a faster computing cluster with more memory to verify that this finding applies to the entire data set. After verification, the next step is implementation: this prediction system should replace the old one at MGH.

Currently, we predict the expected time of a surgery, but as we have seen (Figure 2), the surgery durations generally follow a normal or log-normal distribution. Using this knowledge, we can begin to ask more subtle questions than simply the expected start and end time of each surgery of the day. For example, one may ask "What is the likelihood that a given case will finish within a certain period of time?". This requires a generative model of the probability density function (PDF) for any given duration given the knowledge we have about a surgery. While researchers have expressed interest in calculating these values [Dexter96], actual O.R. managers are skeptical of its practical utility, citing the fact that they would prefer an accurate estimate of the expected time to a probabilistic estimate that is often difficult to interpret.

7 Conclusion

In this thesis, we have considered the theoretical and practical aspects of five standard machine learning algorithms as applied to the problem of predicting surgery duration in operating rooms. Table 7 summarizes our comparison of the practical qualities of each algorithm.

Algorithm	Accuracy	Ease of understanding/ Interpretability	External Parameters	Availability/ease of implementation	Training time	Naturally incorporates categorical variables?
Proc-surg	Fair	Excellent	None	Excellent	None	Yes
Nearest Neighbors	Fair	Medium	Many	Poor	Long	Yes (with added tweak)
Linear Regression	Excellent	Good	Some	Excellent	Medium	No
Regression Trees	Medium	Excellent	None	Medium	Medium	No
Support Vector Machines	Medium	Poor	Some	Excellent	Medium	No

Table 7 A qualitative comparison of learning algorithms tested in this thesis.

Given our experience and testing results, we conclude that linear regression is most likely to be accepted as an improved surgery duration estimation algorithm. Linear regression is the most accurate— our tests find an improvement of approximately 20% over the proc-surg algorithm currently in use. It is simple and interpretable, and it is well-understood and trusted by the people who will be relying on it every day.

8 References

[Aha95] Aha, D., and Bankert, R. "A Comparative Evaluation of Sequential Feature Selection Algorithms." in eds, Fisher, D., and Lenz, H. Learning from Data: AI and Statistics V. 1996, Springer-Verlag.

[Breiman84] Breiman, L., et al. "Classification and Regression Trees." Kluwer Academic Publishers: New York. 1984.

[Dexter96] Dexter, F. "Application of Prediction Levels to OR Scheduling." AORN 63(3), 1996. 607-615.

[Dexter99] Dexter, Franklin, Traub, Rodney, and Qian, Fang. "Comparison of Statistical Methods to Complete a Series of Surgical Cases." J. Clin. Monit. 1999; 15: 45-51.

[Hamilton94] Hamilton, D., and Breslawski, S. "Operating Room Scheduling: Factors to Consider." AORN Journal. March 1994, Vol. 59, No. 3.

[Hastie01] Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. The Elements of Statistical Learning.

[Joachims98] http://citeseer.ist.psu.edu/joachims98making.html, YALE

[Jmat04] "JMAT Project: Java MATrix tools package" http://jmat.sourceforge.net/. Visted May 27, 2004.

[Kolmogorov57] Kolmogorov, A., and Fomin, S. "Elements of the Theory of Functions and Functional Analysis." Dover Publications, New York, 1957.

[Kolmogorov70] Kolmogorov, A., and Fomin, S. "Introduction to Real Analysis." Dover Publications, New York, 1970.

[Lebowitz03] Lebowitz, P. "Why Can't My Procedures Start On Time." AORN 77(3), 2003. 594-597.

[Lowe95] Lowe, D. "Similarity Metric Learning for a Variable-Kernel Classifier." Neural Computation, 7(1):72-85, 1995.

[Macario99] Macario, Alex, and Dexter, Franklin. "Estimating the Duration of a Case When the Surgeon Has Not Recently Scheduled the Procedure at the Surgical Suite." Anesthesia and Analgesia, Volum 89(5). November 1999.

[Magerlein78] Magerlein, J., Martin, J. "Surgical Demand Scheduling: A Review." Health Services Research. Winter 1978: 418-433.

[McQuarrie81] McQuarrie, D. "Limits to Efficient Operating Room Scheduling." Archives of Surgery—Vol 116, 1981. 1065-1071.

[Rifkin02] Rifkin, R. "Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning." MIT Ph.D. Thesis, 2002.

[Stanton01] Galton, Pearson, and the Peas: A Brief History of Linear Regression for Statistics Instructors. http://www.amstat.org/publications/jse/v9n3/stanton.html

[Strum98] Strum, D., May, J., Vargas, L. "Surgical procedure times are well modeled by the log normal distribution." Anesthesia and Analgesia. 1998: 86: S47.

[SVMlight04] "SVMlight: Support Vector Machine" http://svmlight.joachims.org/. Updated March 7, 2002. Visited May 27, 2004.

[Vapnik98] Vapnik, V. Statistical Learning Theory. John Wiley & Sons, Inc: New York, 1998.

[Witten99] Witten, I., and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Press, October 1999.

[Wright96] Wright, I., et al. "Statistical Modeling to Predict Elective Surgery Time: Comparison with a Computer Scheduling System and Surgeon-provided Estimates." Anesthesiology 1996; 85:1234-45.

[Zhou98] Zhou, J., Dexter F. "Method to assist in the scheduling of add-on surgical cases – upper prediction bounds for surgical case duration based on the log normal distribution." Anesthesiology 1998; 89: 1228-1232.

[Zhou99] Zhou, J., et al. "Relying Solely on Historical Surgical Times to Estimate Accurately Future Surgical Times is Unlikely to Reduce the Average Length of Time Cases Finish Late." J. Clinical Anesthesia. 11:601-605, 1999.