**[ITS Center designation: Traffic forecasting: non-parametric regressions]**

## UVA Center for Transportation Studies

**A Research Project Report**

**For the Center for ITS Implementation Research**

**A U.S. DOT University Transportation Center**

# Traffic Flow Forecasting Using Approximate Nearest Neighbor Nonparametric Regression

*Authors*

R. Keith Oswald
Dr. William T. Scherer
Dr. Brian L. Smith

Center for Transportation Studies
University of Virginia
Thornton Hall
351 McCormick Road, P.O. Box 400742
Charlottesville, VA 22904-4742
804.924.6362

December 2000
UVA-CE-ITS_01-4

**Disclaimer**

**Abstract**

The purpose of this research is to enhance nonparametric regression (NPR) for use in real-time systems by first reducing execution time using advanced data structures and imprecise computations and then developing a methodology for applying NPR. Due to the nature of the enhancements to nonparametric regression, each application of NPR will be specific for each system. This research, therefore, provides general guidelines for deploying nonparametric regression, similar to how Box and Jenkins (1970) provided a methodology for conducting time series analysis.

Nonparametric regression is a forecasting technique similar to case-based reasoning that does not make any rigid assumptions about the data. In short, the method searches a collection of historical observations for records similar to the current conditions and uses these to estimate the future state of the system. Unfortunately, the data that are so vital to the accuracy of nonparametric regression also restrict its use in real-time systems, where the success of the system depends on the accuracy of the forecast as well as the time required to compute the estimate. Whereas parametric models compress all training data into a set of equations through the process of parameter fitting, nonparametric regression retains the data and searches through them for past similar cases each time a forecast is made. Examination of the nonparametric regression algorithm reveals that a majority of execution time is spent searching for past similar cases, especially when the number of historical observations is large.

Furthermore, the lack of a systematic approach for applying nonparametric regression is another problem restricting its widespread use. Prior to 1970, time series analysis techniques were not widely used because a structured method for applying them did not exist. Box and Jenkins (1970) spurred the general use of time series analysis by combining their personal experiences with existing techniques such as exponential smoothing, double exponential smoothing, and ARIMA to provide a systematic methodology for applying them to data.

The purpose of this research, therefore, is to enhance nonparametric regression for use in real-time systems by first reducing execution time and then developing a methodology for applying NPR. The results presented indicated that advanced data structures such as KD trees can reduce execution time by as much as 1,000 times.

Imprecise computations such as approximate nearest neighbors may be used to further reduce execution time, if needed, but at the expense of forecast accuracy. Based on the complexity added to NPR so that it may be used in real-time systems, a systematic, iterative methodology for applying nonparametric regression is also developed.

Table of Contents

# List of Figures

# List of Tables

# Glossary of Terms and Abbreviations

**ANN-NPR** – approximate nearest neighbor nonparametric regression

**Approximate Nearest Neighbors (ANN)** – the $k^{th}$ $(1+\varepsilon)$-approximate nearest neighbor of $q$ is a data point whose relative error from the true $k^{th}$ nearest neighbor of $q$ is $\varepsilon$

**Big $O(\bullet)$ Notation (i.e. $f(n) = O(g(n))$)** – set of all functions for which there are constants $c > 0$ and $n_o \geq 0$ such that $f(n) \leq c * g(n)$ for $n \geq n_o$; $g(n)$ is the smallest upper bound on $f(n)$ for all $n \geq n_o$

**Forecast Function** – component of nonparametric regression that takes (approximate) nearest neighbors as inputs and estimates the future state of the system

**Historical Database** – collection of historical observations defined by a state vector for use in nonparametric regression

**NN-NPR** – nearest neighbor nonparametric regression

**NPR** – nonparametric regression

**Real-Time System (RTS)** – applications in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced

**Search Procedure** – component of nonparametric regression that finds (approximate) nearest neighbors in the historical database for use in the forecast function

# List of Symbols

| | |
|---|---|
| $\varepsilon$ | maximum relative error between the $k^{th}$ true nearest neighbor and the $k^{th}$ approximate nearest neighbor |
| *AET* | average execution (query) time |
| *D* | number of lagged observations used in the state vector |
| *K* | number of elements in the state vector |
| *k* | number of observations from the historical database used to generate a forecast in nearest neighbor nonparametric regression |
| $L_m$ | Minkowski distance metric |
| *MAPE* | mean absolute percent error |
| *N* | number of records in the historical database |
| *p* | observation from the historical database defined by a state vector |
| *pop_size* | number of individuals in a population in a genetic algorithm |
| *q* | current conditions of the system defined by a state vector |
| $\hat{V}(t+1)$ | estimated state of the system at time $t$+1 |
| $V(t)$ | univariate measure of the system at time $t$ |
| $V_{hist}(t)$ | historical average univariate measure of the system at time-of-day and day-of-week associated with time $t$ |
| $v_i$ | collection of specific values for decision variables representing a solution to an optimization problem in a genetic algorithm; an individual |
| $X(t)$ | state of system at time $t$ defined by $K$ univariate measures |

# 1 Introduction

Forecasting is a central component in many organizations. For example, one of the most critical aspects of inventory and supply chain management is the ability to accurately predict demand. Inventory levels, reorder points, probability of stock outs, and production levels are all based on forecasted demand. Forecasting is so important that businesses in virtually every industry are deploying data warehouses[1] to collect and retain data over time so that decision support systems may identify and predict changes and trends to improve business operations.

Numerous prediction techniques exist to model a wide variety of situations. Linear regression is perhaps the most well known method but other techniques such as time series analysis, nonlinear regression, neural networks, classification, clustering, and polynomial networks are commonly utilized in decision support systems. Each method has strengths and weaknesses that are designed to handle a specific class of problems. However, during the modeling process, assumptions about the data are made that may or may not be appropriate, thus affecting forecast accuracy. For example, “parametric algorithms assume that the data to be modeled takes on a structure that can be described by a known mathematical expression with a few free parameters” (Kennedy et. al., 1998). Specifically, parametric regression models assume that the random errors associated with observations are independent and normally distributed with a mean of 0 and constant variance. “If the assumptions are flagrantly violated, any inferences derived from the regression analysis are suspect” (Mendenhall and Sincich, 1996).

As data warehouses continue to collect data and increase in size, the forecasting techniques employed must be able to effectively use the additional information as it is gathered. For methods such as parametric regression, new data are incorporated into the model by manually re-computing parameters. Unless parameters are recalculated every time new data become available, there will be a lag from the time new data are recorded to when they are incorporated into the model. Furthermore, as more data are collected, the effect of an individual observation on the overall model is diluted because parametric models fit a function to the data by smoothing the observations in the training set. As a

[1] Data warehouses bring together large volumes of quantitative business data obtained from transaction processes, enterprise resource planning systems, and legacy systems.

result, abnormal operating conditions, which some argue are the most interesting and important situations of a process, tend to get neglected.

Nonparametric regression (NPR) is an alternative forecasting method that does not impose restrictive assumptions on the data. Similar to case-based reasoning, NPR relies on the data to determine a relationship between input and output states. Newly observed data can be easily added to a nonparametric model without the expensive re-computation of parameters needed with parametric techniques. Furthermore, and most importantly in automated data mining applications, no prior knowledge about the system being modeled is required, only enough data to sufficiently describe the underlying process. Finally, nonparametric regression maintains the uniqueness of every observation because it does not smooth cases, making NPR especially useful in modeling unusual situations.

## *1.1 Nonparametric Regression and Case-Based Reasoning*

Nonparametric regression is a data-driven heuristic forecasting technique. “The nonparametric estimation of [a function] is often called ‘distribution-free’, implying that no assumptions about the statistical structure [of the data] are made” (Schaal, 1994), which allows NPR to model a wide variety of situations. Nonparametric regression does not require any prior knowledge about the process being modeled, only sufficiently large quantities of data representing the underlying system. NPR relies on past data to describe the relationship between input and output states rather than a modeler who imposes a (possibly incorrect) model upon the data.

Nonparametric regression is similar to case-based reasoning (CBR). A system that employs case-based reasoning attempts to solve a new problem by making an analogy to an old problem and adapting its solution to apply to the current situation. Case-based reasoning is advantageous in domains where a well-defined theory does not exist but large amounts of data are readily available. Some domains, such as law and medicine, are inherently case-based.

Previously experienced situations are recorded as cases where each episode consists of a specific problem, its solution, and a set of indices. An index is a feature of a

case that distinguishes it from other cases. Indexing is central to the representation, storage, and retrieval of cases because it guides the process for selecting experiences relevant to the current problem situation.

When presented with a new problem, CBR searches a library of past cases for those experiences with features similar to the current problem situation. Cases are ranked by a similarity metric, which compares a case to the current problem situation with regard to various features of the index. The most similar case or cases are selected and adapted to the current problem situation to develop a solution. Traditional systems that employ parametric models or rules tend to be brittle because they cannot learn from new experiences without expensive re-computations of parameters. Systems employing CBR can adapt to changes in the underlying process by simply acquiring new cases.

Nonparametric regression is not an appropriate modeling technique to use in all situations, however. When a well-defined theory exists and the assumptions of a given modeling technique prove true, it is often desirable to use parametric models due to their statistical properties. However, because nonparametric regression does not smooth historical observations, NPR may be used to improve the accuracy of parametric models, especially during abnormal operation conditions. On the other hand, when the process being modeled is not well understood but large quantities of data are readily available, nonparametric regression is an ideal modeling technique.

Given a situation where nonparametric regression is applicable, there are two significant obstacles hampering the widespread use of nonparametric regression. First, the runtime execution of NPR tends to be much slower than traditional parametric models. Second, there is no systematic approach for deploying nonparametric regression comparable to those available for parametric techniques.

### *1.1.1 Problems in Real-Time Systems*

Unfortunately, the data that are so vital to the accuracy of nonparametric regression (and case-based reasoning) restrict its use in real-time systems (RTS), where the success of the system depends on the accuracy of the forecast as well as the time required to compute the estimate. Whereas parametric models compress all training data into a set of

equations through the process of parameter fitting, nonparametric regression retains the data and searches through it for past similar cases each time a forecast is made. Examination of the nonparametric regression algorithm reveals that a majority of execution time is spent searching for past similar cases, especially when the number of historical experiences is large. As Ripley (1999) indicates, though, there are promising methods to overcome slow execution time.

> "One common complaint about both kernel and *k* nearest neighbor [nonparametric regression] methods is that they can take too long to compute and need too much storage for the whole training set. The difficulties are sometimes exaggerated, as there are fast ways to find near neighbors" (Ripley, 1999).

Two general techniques for speeding the execution time of nonparametric regression are readily apparent: advanced data structures and imprecise computations.

The search for past similar cases is analogous to the problem of record retrieval common in the computer science and database communities. A typical solution to speeding record retrievals involves structuring the data in ways that inherently speed nearest neighbor searches. Several researchers have proposed multidimensional search trees to reduce retrieval time.

In the real-time systems literature, imprecise computations are a set of techniques whereby an algorithm can produce an approximate result in less time than is needed to calculate an exact solution. Such methods ensure information flows from one task to another within the constraints imposed by the real-time system. Military targeting systems, for example, use the approximate location of a target's current position rather than where it was a few seconds ago. For nonparametric regression, imprecise computations are achieved through the use of approximate nearest neighbors (ANN), which are cases similar to the current problem situation that may not necessarily be the most similar (Arya and Mount, 1993).

### *1.1.2 Lack of a Systematic Approach*

The lack of a systematic approach for applying nonparametric regression is another problem restricting its widespread use. Prior to 1970, time series analysis techniques were not widely used because a structured method for applying them did not exist. Box and Jenkins (1970) spurred the general use of time series analysis by relating their personal experiences with existing techniques such as exponential smoothing, double exponential smoothing, and ARIMA to provide a systematic methodology for applying them to data.

A systematic approach for applying NPR must be developed. As with parametric modeling techniques, nonparametric regression requires the modeler to make key decisions regarding the performance of the algorithm. A methodology that guides modelers through the process of applying nonparametric regression will allow NPR to be successfully used in areas where it previously has not been applied.

## *1.2 Research Objective*

Assuming that nonparametric regression has been selected as the technique used to model the system under investigation a priori, the purpose of this research is to enhance NPR for use in real-time systems by first reducing execution time using advanced data structures and imprecise computations and then developing a methodology for applying nonparametric regression. Due to the nature of advanced data structures, approximate nearest neighbors, and nonparametric regression, each application of nonparametric regression will be specific for each system. This research, therefore, provides general guidelines for deploying NPR, similar to how Box and Jenkins (1970) provided a methodology for conducting time series analysis.

This research is divided into six distinct sections. Chapter 2 provides a theoretical description of nonparametric regression and decomposes the model into three fundamental components. The two classes of nonparametric regression, kernel and nearest neighbor, are also discussed. Several previous applications of nonparametric regression are described to highlight some of the key advantages and disadvantages.

One specific disadvantage of nonparametric regression is its slow execution time relative to parametric techniques. Chapter 3 addresses the concerns of using

nonparametric regression in real-time systems where the time required to execute an algorithm is as important as the accuracy of the calculations. Chapter 3 also describes the theoretical foundation of advanced data structures and approximate nearest neighbors as methods of reducing the execution time of NPR. Other methods of speeding nonparametric regression are briefly discussed. However, the use of advanced data structures and approximate nearest neighbors increases the complexity of nonparametric regression, furthering the motivation for a systematic methodology for deploying NPR, especially in real-time systems.

Six case studies are examined in Chapter 4, five involving the short-term prediction of highway volume and one estimating maximum daily temperature. The case studies demonstrate the effectiveness of advanced data structures and approximate nearest neighbors to reduce the execution time of nonparametric regression. Because approximate nearest neighbors are a type of imprecise computation, the effects of ANN on forecast accuracy is also examined.

Chapter 5 details a methodology for applying nonparametric regression based on the author's experiences with the case studies. Special attention is given to deploying NPR in real-time systems but the methodology is general enough to apply nonparametric regression in any operating environment. The process, which is iterative like many other model-building methods, offers varying amounts of involvement by the modeler. In other words, the modeler can have as much or as little input about the final model depending on their level of expertise with the system being modeled.

Finally, Chapter 6 concludes with a discussion of the major contributions of this research effort. Several topics for future research are also presented.

## *1.3 Summary*

Nonparametric regression is a forecasting method that relies on the data to determine a relationship between input and output states. NPR is advantageous in situations where the underlying process is not well understood but large amounts of data are readily available because the input-output relationships are derived directly from the data – no prior knowledge about the system being modeled is required.

Nonparametric regression is similar to case-based reasoning and searches through a collection of past experiences or observations when estimating the future state of the system. As a result, nonparametric regression techniques tend to calculate an estimate slower than parametric models, which can be problematic in real-time systems. Another disadvantage of nonparametric regression is that no general methodology for applying the model exists.

The objectives of this research effort are two-fold. First, the nonparametric regression algorithm is enhanced through the use of advanced data structures and approximate nearest neighbors to reduce execution time. Second, a general methodology for applying nonparametric regression in real-time systems is proposed.

# 2 Nonparametric Regression

Nonparametric regression is a forecasting technique similar to case-based reasoning that does not make any rigid assumptions about the data. In short, the method searches a collection of historical observations for records similar to the current conditions and uses these to estimate the future state of the system. Because nonparametric regression does not make strong assumptions about the shape of the true relationship being predicted, it is a "flexible, powerful method for estimating an unknown … function" (Altman, 1992).

In contrast, given a set of input-output training data ($x$, $y$), parametric model fitting develops a relationship

(1) $$y = f(x,\theta)$$

where $\theta$ represents a finite number of parameters. The fitting procedure minimizes an error function $J$ for all data pairs ($x_i$, $y_i$) in the training set. $J$ is often a least squares criterion and when $f$ is linear in the parameters $\theta$, Equation (1) corresponds to linear regression. The fitting procedure is usually performed off-line and, once an acceptable model exists, an estimate can be calculated using only new input data $x_q$, as shown in Figure 1. Because parametric model fitting effectively compresses all data in the training set into one equation, the computational time to estimate $\hat{y}$ for new input is proportional to the complexity of $f$ and is nearly instantaneous.

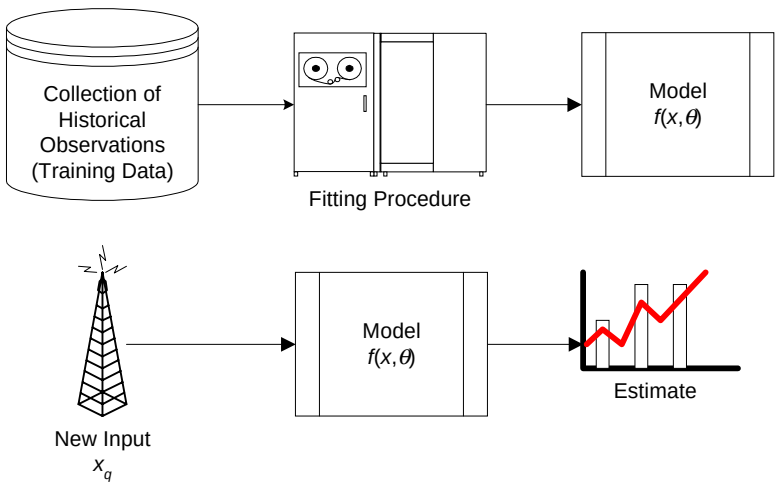


**Figure 1 - Flow of Data through Parametric Models**

Nonparametric regression, on the other hand, can be decomposed into three fundamental components: a database of past observations, a search procedure, and a forecast function. Figure 2 depicts the flow of data through a typical nonparametric regression algorithm. The search procedure finds the nearest neighbors, which are the historical observations that are most similar to the current conditions. The nearest neighbors then become the inputs to the forecast function so that it may generate an estimate. The following section describes each component of nonparametric regression in more detail.

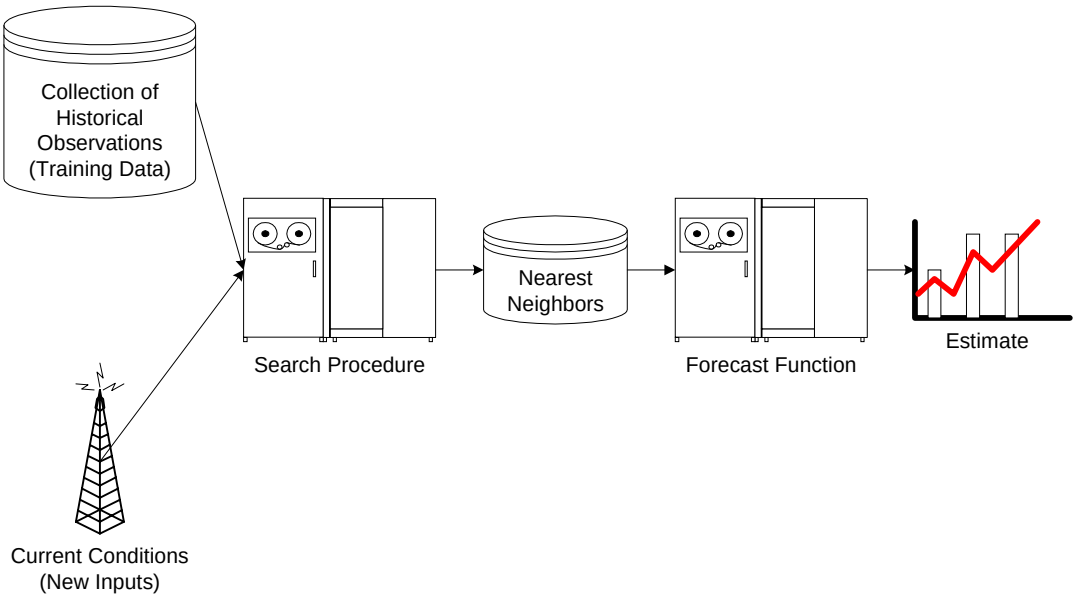


**Figure 2 - Flow of Data through Nonparametric Regression Models**

## *2.1 General Model*

### *2.1.1 Historical Database*

The historical database is a collection of past observations of the system under consideration and is analogous to the training data used in parametric model fitting. Similar to the advantages of having a large training set, nonparametric regression benefits

from a larger database that more accurately and thoroughly represents a broader range of possible system conditions.

A state vector is a set of observations that describe the state of the system. For example, suppose one wants to predict when a person may have diabetes based on measurements such as blood pressure ($V_1$), glucose ($V_2$), insulin ($V_3$), mass ($V_4$), and age ($V_5$). A state vector $X_i$ that describes person $i$ with respect to these attributes may be written as follows.

(2) $$X_i = [V_{i1}, V_{i2}, V_{i3}, V_{i4}, V_{i5}]$$

For univariate data that is time series in nature, a state vector defines the system with a measurement at time $t$, $t$-1, …, $t$-$D$ where $D$ is an appropriate number of lags. For example, a state vector $X(t)$ of measurements collected every ten minutes with $D = 2$ can be written as

(3) $$X(t) = [V(t), V(t-1), V(t-2)]$$

where $V(t)$ is the measurement during the current time interval, $V(t\text{-}1)$ is the measurement during the previous ten-minute interval, and so on. Note that an infinite number of possible state vectors exist. Furthermore, they are not restricted to incorporating only lagged values but may also contain aggregate measures such as historical averages.

### *2.1.2 Search Procedure*

The search procedure finds records in the historical database that are similar to the current conditions and labels them as neighbors. More formally, "given a file of $N$ records (each of which is described by $K$ real-valued attributes) and a dissimilarity measure, [nearest neighbor searching] finds the records closest to a query record $q$" (Friedman et. al., 1977). Nearest neighbor searching is a central component of many applications including knowledge discovery, data mining, pattern recognition, classification, machine learning, data compression, multimedia databases, document retrieval, and statistics (Maneewongvatana and Mount, 1999).

Dissimilarity is usually based on Minkowski distance metrics as defined by

(4) $$L_m = \left[ \sum_{i=1}^{K} |p_i - q_i|^m \right]^{1/m}$$

where $K$ is the dimensionality of the state vector, $p_i$ is the $i^{th}$ element of the historical record currently under consideration, and $q_i$ is the $i^{th}$ element of the current conditions. $L_1$, $L_2$, and $L_\infty$ are the commonly known Manhattan, Euclidean, and max distances, respectively. Referring to the example state vector given in (3), the Euclidean distance from historical record $p$ to the current condition $q$ can be written as follows.

$$(5) \quad dist(p,q) = \sqrt{[V_p(t) - V_q(t)]^2 + [V_p(t-1) - V_q(t-1)]^2 + [V_p(t-2) - V_q(t-2)]^2}$$

Other dissimilarity metrics that employ non-uniform weights may also be used. Such weighted metrics cause the distance calculation to depend on the difference between the values of the variables as well as the variables themselves. Weights may also be used to adjust for different ranges of values for each variable in the state vector. Clearly, an infinite number of dissimilarity metrics are possible.

Once an acceptable dissimilarity metric has been chosen, the search procedure finds the nearest neighbors, which are those historical observations with the smallest values of $dist(p, q)$. The most common implementation of the search procedure found in the literature searches for nearest neighbors sequentially. In other words, distances are calculated from the query point to *every* point in the historical database to determine the closest observations. As the size of the historical database increases, sequentially searching for nearest neighbors becomes an extremely time-consuming process and is largely responsible for the slow execution of nonparametric regression.

### *2.1.3 Forecast Generation*

Associated with each record in the historical database is a dependent variable, which is the variable being estimated. For the example state vector in Equation (2), the dependent variable is a binary number indicating whether or not a person has diabetes. For the example state vector in Equation (3), the dependent variable is the measurement during the next time interval, $V(t+1)$. The most straightforward approach to generating a forecast is to compute a simple average of the dependent variable values of the nearest neighbors. Mathematically, this estimate is calculated according to Equation (6), assuming the state of the system is defined according to Equation (3), where $k$ is the number of neighbors found by the search procedure.

(6) $$\hat{V}(t+1) = \left(\frac{1}{k}\right)\sum_{i=1}^{k} V_i(t+1)$$

The weakness of such an approach is that it ignores all information provided by the distance metric. It is logical to assume that past records closer to the current conditions have higher information content and should have a more significant impact on the forecast.

A number of weighting schemes have been proposed for use within nonparametric regression as discussed by Smith et. al. (2000). In general, these weights are proportional to the distance between the neighbor and the current conditions. An alternative to the use of weights is to fit a linear or nonlinear parametric model to the cases in the neighborhood, and then use that model to forecast the value of the dependent variable as done by Mulhern and Caprara (1994). It should be clear that there are an infinite number of approaches to forecast generation.

## *2.2 Fundamental Classes*

Once a database of adequate breadth has been established, a fundamental challenge of nonparametric regression is to define the number of nearest neighbors used to generate forecasts. The quality of the neighborhood, which is the set of nearest neighbors found by the search procedure, directly impacts the accuracy of the estimate. For example, one would expect to generate a more accurate estimate if a neighborhood was defined containing ten previous cases similar to the input state than if the neighborhood contained ten cases selected at random.

There are two basic approaches to neighborhood definition (Altman, 1992): nearest neighbor and kernel. Nearest neighbor neighborhoods are defined as those with a constant number of samples, *k*, while kernel neighborhoods have constant bandwidth in the state space. In other words, nearest neighbor nonparametric regression (NN-NPR) uses a fixed number of neighbors to generate forecasts while kernel nonparametric regression uses any number of neighbors that are within a fixed distance of the current conditions.

To illustrate the differences between the two neighborhoods more clearly, a simple example that has been adopted and modified from Smith (1995) will be used. For this example, the search procedure defines dissimilarity based on Euclidean distance. Suppose one wants to predict the value of *y* for $x_A = 3$ and $x_B = 10$ given the following data.

**Table 1 - Nonparametric Regression Example Data**

| x | y |
|---|---|
| 0.75 | 6.40 |
| 1.25 | 7.00 |
| 2.00 | 7.75 |
| 3.00 | 5.00 |
| 3.50 | 6.25 |
| 3.75 | 4.25 |
| 5.00 | 8.75 |
| 5.75 | 6.00 |
| 6.25 | 9.00 |
| 7.75 | 9.25 |
| 7.90 | 7.75 |
| 10.25 | 7.75 |
| 10.50 | 7.50 |
| 11.50 | 4.00 |

Using kernel neighborhoods with a bandwidth of 2 (±1 on either side of *x*), the following neighborhoods are defined. In the Figure 3, notice that neighborhood A contains four points while neighborhood B contains only two. Using a straight average of the points within the neighborhoods, the estimate for $x_A$ is $\hat{y}_A = 5.81$ (the average of 7.75, 5.00, 6.25, and 4.25), while the estimate for $x_B$ is $\hat{y}_B = 7.63$ (the average of 7.75 and 7.50).

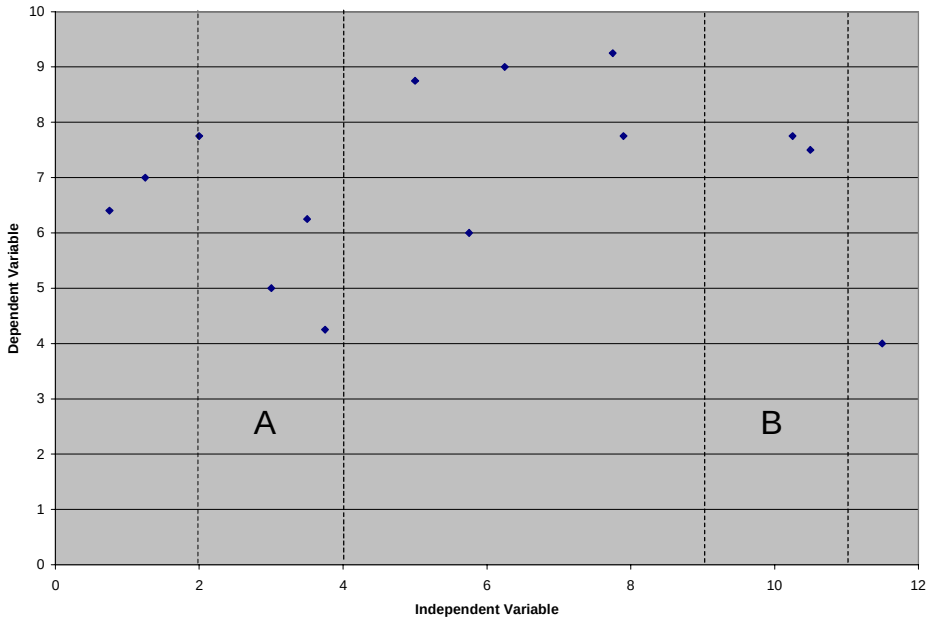


**Figure 3 - Kernel Neighborhoods**

In contrast, the nearest neighbor approach defines different neighborhoods. Suppose $k = 3$. Notice how the neighborhoods have changed in the Figure 4 below. Neighborhood A has shrunken and is slightly offset to the right. Neighborhood B has also shifted right and now includes a point that was not in the kernel neighborhood. In this case, using the straight average of the points within the neighborhoods, $\hat{y}_A = 5.17$ (the average of 5.00, 6.25, and 4.25), while $\hat{y}_B = 6.42$ (the average of 7.75, 7.50, and 4.00).

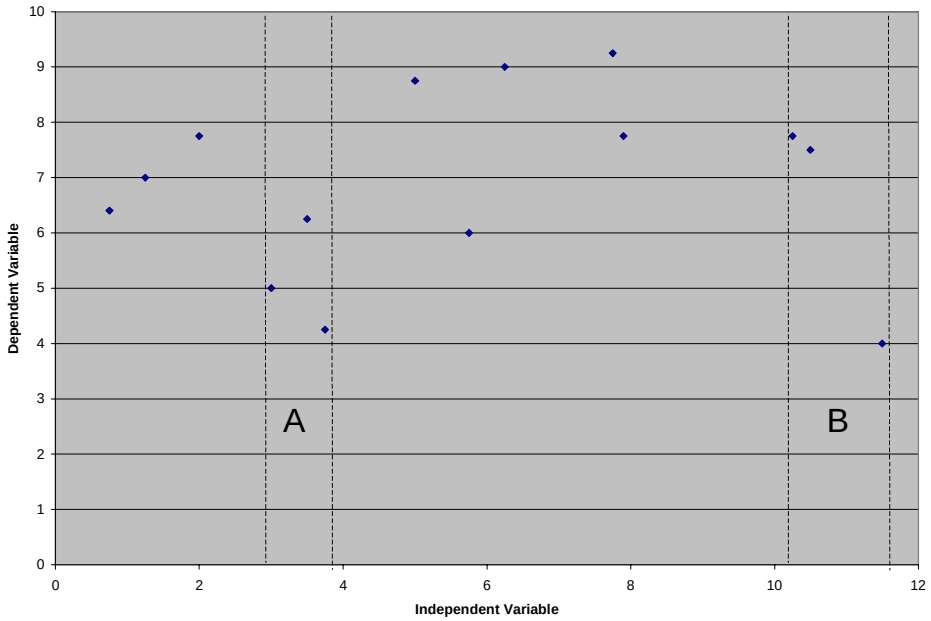


**Figure 4 - Nearest Neighbor Neighborhoods**

Finally, consider the neighborhoods when trying to predict *y* given $x = 9$, as shown in Figure 5. Solid lines define the kernel neighborhood while the nearest neighbor neighborhood is indicated with dashed lines. In this case, the kernel nonparametric regression approach cannot generate a forecast because no points lie within its neighborhood. One advantage of the nearest neighbor approach is that it will always generate a forecast. On the other hand, kernel nonparametric regression does not attempt to generate a forecast that may be grossly in error because no neighbors that are close to the current conditions have been observed in the past. Nearest neighbor nonparametric regression will be used in this research because of its ability to always generate a forecast.

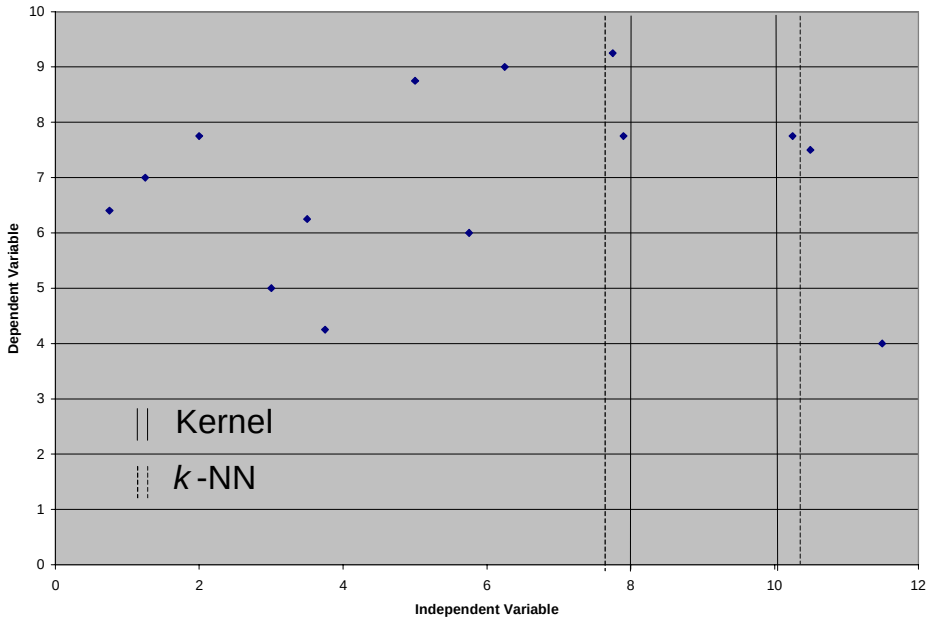


**Figure 5 - Kernel and Nearest Neighbor Neighborhoods**

## *2.3 Previous Applications*

Nonparametric regression techniques have been used in numerous applications including rainfall runoff estimation, utility load forecasting, market prediction, learning models, and freeway flow estimation. The following sections provide a brief description of these efforts to highlight some of the advantages and disadvantages of NPR.

### *2.3.1 Rainfall Runoff Estimation*

Prior to the late 1980's, rainfall runoff was traditionally calculated using instantaneous unit hydrographs (IUH), which applied linear parametric models to small sets of tediously gathered data. However, advances in computer technology quickly made large amounts of data available to hydrologists. The vast quantities of data indicated that current models "may not contain the true rainfall runoff relationship … and may not be asymptotically optimal, no matter how large the database becomes" (Karlsson and Yakowitz, 1987). The surplus of available data caused hydrologists to question the linearity of the rainfall runoff relationship. Despite "the virtues of the IUH and its

[numerous] nonlinear variations, a sizable portion of influential stochastic hydrologists are dissatisfied with them" (Karlsson and Yakowitz, 1987). The doubts that hydrologists expressed about a parametric relationship existing between rainfall and runoff indicated that the experts did not posses a well-defined theory to explain the underlying system. However, large amounts of data were readily available and, therefore, Karlsson and Yakowitz turned to nearest neighbor nonparametric regression.

The authors showed that "nearest neighbor [models] will asymptotically (with increasing number of historical records) be at least as good as any $K^{th}$ order" regression model (Karlsson and Yakowitz, 1987), where $K$ is the dimension of the state vector. In other words, nearest neighbor nonparametric regression can provide forecasts that are at least as accurate as parametric models given a database of adequate size and quality. Furthermore, NN-NPR is not restricted by the assumptions imposed upon parametric models.

The authors note one disadvantage of nearest neighbor nonparametric regression algorithms, however. "Nearest neighbor estimation is not something one does on a IBM-PC (yet) but a VAX is sufficient" (Karlsson and Yakowitz, 1987). While advances in computer technology since 1987 permit nearest neighbor nonparametric regression to run on desktop computers, traditional implementations of the algorithm demand a significant amount of processing time, especially when the historical database is large and continuously increasing in size.

### *2.3.2 Utility Load Forecasting*

Utility load forecasting involves the estimation of future load requirements on power systems up to one week in the future in hourly increments. Accurate forecasts aid in optimizing the operational state of the power system as well as analyzing contingency plans and security settings. Artificial neural networks are commonly employed to forecast demand due to the complex relationship between the required load and factors such as temperature, humidity, and wind speed.

Charytoniuk, Chen, and Olinda (1997) applied a kernel nonparametric regression model using one year of historical data. The state of the system was defined by the time

of day, current temperature, average temperature for the past eight hours, and the price of electricity. The authors obtained results comparable to those of artificial neural networks. However, two neural networks were required to model the demand for the future week – one for weekdays and one for weekends – whereas only one nonparametric regression model was needed to estimate demand for the same period.

The authors note that extending the historical database should improve forecast accuracy by permitting the algorithm to better model abnormal conditions such as holidays, which are situations where other forecasting methods have traditionally generated poor estimates. Because parametric models fit a smooth function to the data, abnormal observations are averaged into the mean, thus decreasing the accuracy of the model when unusual conditions occur again. Nonparametric regression is limited to generating forecasts only over the range of observations present in the historical database. Parametric models are similarly limited. Although forecasts can be extrapolated beyond the range of the training data, the accuracy of such estimates is suspect. Increasing the size of the historical database, especially with observations that cover abnormal operating conditions, allows NPR to model a wider variety of conditions.

### *2.3.3 Market Prediction*

Marketing data, such as that collected by cash registers from checkout lines, are inherently time series in nature. Time series forecasting techniques are well suited to estimating future values. However, some time series exhibit behaviors that cannot be represented with Box-Jenkins formulations, especially when chaos is present in the system.

Chaotic systems are traditionally identified by large changes in the current conditions as a result of small changes in the initial conditions. To illustrate this characteristic, an example from Mulhern and Caprara (1994) will be given. Consider plots of $y_t = 2y_{t-1}^2 - 1$ with two different initial conditions. Figure 6 represents the resulting chaotic time series with a starting value of 0.54321 and exhibits similar, but not identical, patterns of past behavior. Figure 7 shows another time series plot generated from the same function with a slightly larger starting value of 0.54322. While similar

patterns exist in both plots, the specific values of *y* are significantly different for the same value of time. In addition to being extremely sensitive to initial conditions, chaotic systems demonstrate “non-identical but similar patterns of [past] behavior” (Mulhern and Caprara, 1994).

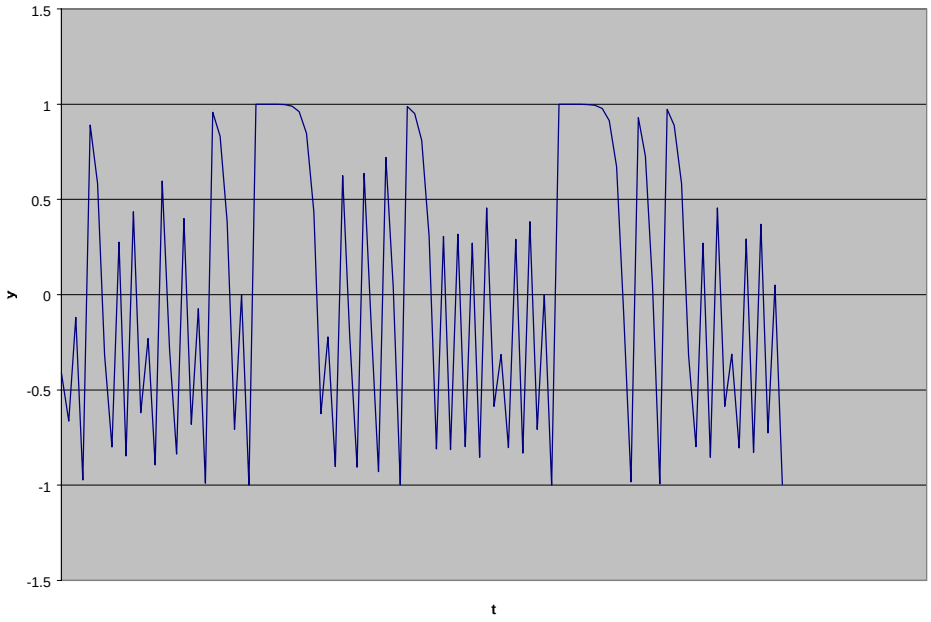


**Figure 6 - Chaos Example, Starting Value = 0.54321**

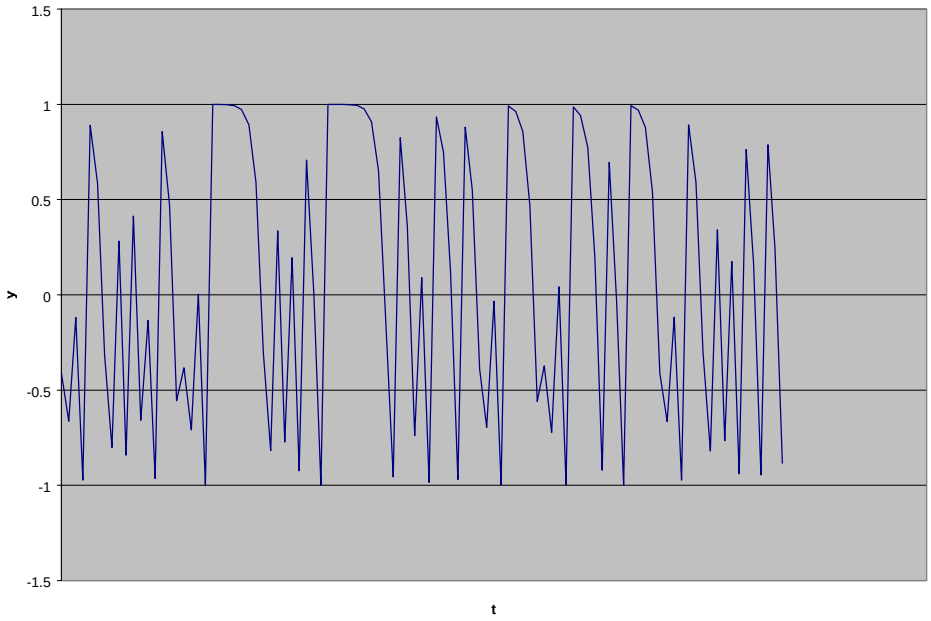


**Figure 7 - Chaos Example, Starting Value = 0.54322**

When modeling chaotic systems, the method must be able to distinguish between truly random behavior and chaotic behavior. The problems with time series approaches applied to chaotic systems can be summarized in the following statement.

> "One criteria for a good Box-Jenkins model is that the residuals of the modeled series are Gaussian white noise. When chaotic behavior exists in a time series, Box-Jenkins procedures falter because the behavior generated by the chaotic process passes the tests of randomness. The result is a time series model that, while satisfying the diagnostic checking criteria in a Box-Jenkins analysis, is unable to make accurate forecasts" (Mulhern and Caprara, 1994).

Nonparametric regression models attempt to reconstruct the relationship in the system geometrically rather than assuming a functional relationship. In other words, a state vector containing *K* variables represents a *K* dimensional vector of the system for a given observation. Nonparametric regression selects historical records that exhibit similar, but not necessarily identical, patterns of past behavior and uses the dependent variables associated with these observations to estimate the future state of the system.

NPR, therefore, avoids the pitfalls of Box-Jenkins models when applied to chaotic systems.

In addition to noting the advantages of a large historical database, Mulhern and Caprara indicate that weighted forecasts reduce the sensitivity of the estimates to the number of nearest neighbors used in the forecast generation process. Furthermore, the authors mention that the need for weighted forecasts increases as the "library of nearest neighbors" decreases, thus compensating for few potential neighbors. However, the authors do not provide a method for selecting weights or the number of neighbors to use in an estimate.

### *2.3.4 Learning Models*

In general, low level learning theory attempts to establish how one "transforms input data to output data under a given performance criterion. In statistics, such a mapping is called regression if the output variables are continuous and it is called classification if the outputs are discrete" (Schaal, 1994).

Schaal makes many striking comparisons between nonparametric and parametric regression techniques. For example, "nonparametric methods should become increasingly local with more experiences which assures the consistency of the learning method, i.e., the decrease of bias" (Schaal, 1994). In other words, assuming the range of the values of the state vector do not change significantly over time, as the size of the historical database increases, the observations become more dense over the state space. In the case of nearest neighbor nonparametric regression, the neighbors selected to create the forecast are nearer to the current conditions than when the historical database contained fewer observations that are sparsely distributed and, therefore, should generate more accurate forecasts. Similar to observations made by previous researchers, Schaal advocates large historical databases.

Schaal identifies the importance of selecting an appropriate neighborhood size. "Too large a neighborhood will introduce too much bias in the predictions of the learning system, while too small a neighborhood will have too much variance and therefore bad generalization properties" (Schaal, 1994). While Schaal's observation has been noted in

other research efforts, the literature does not provide guidance for selecting an appropriate neighborhood size.

Finally, Schaal comments on the computational costs of nonparametric regression. Parametric models effectively compress the training data into a single equation. On the other hand, nonparametric regression techniques store all of the data and search through it to generate each forecast, thus requiring large amounts of storage space and longer computational times than parametric models.

### *2.3.5 Transportation*

Davis and Nihan (1991) were one of the first transportation engineers to use nonparametric regression to estimate short-term traffic flows. Their study concentrated on estimating the transition from a free flow state of traffic to a congested state. However, modern traffic management systems can benefit from accurate forecasts of freeway flows in both states of congestion.

The following statement best summarizes the authors' results. "The nearest neighbor nonparametric regression approach replaces the problem of selecting a class of models and then estimating parameters with the problem of maintaining and sorting an adequately large learning sample" (Davis and Nihan, 1991). The authors identified the need for a large, representative historical database and commented on the long computational times of nonparametric regression techniques.

Smith (1995) conducted another univariate nonparametric analysis to expand upon Davis and Nihan's work. Smith used nearest neighbor nonparametric regression to estimate volumes from two locations on the Capital Beltway near Washington, DC but had a more extensive data set with more than five months of observations. The author defined the state of the system using current and lagged volume measures as well as historical averages. The results show that NN-NPR generated more accurate forecasts than historical average estimates and neural network models.

Smith used Euclidean distance to define the dissimilarity of historical observations to the current conditions but notes that other measures that have proven effective in cluster analysis may improve forecast accuracy in nonparametric regression.

Furthermore, using a small sample of the overall data, he investigated how the number of nearest neighbors affected forecast accuracy. Through enumeration, Smith empirically determined that ten nearest neighbors produced the most accurate predictions.

A more recent study by Clark (1999) investigated multivariate nearest neighbor nonparametric regression. The author defined traffic conditions based on speed, volume, and occupancy measurements and used a weighted dissimilarity metric to reflect the different magnitudes of the measures. The weights selected by the author "specify that a match of 1 percent in the occupancy score is equivalent to a match of 6.67 km/hour in the speed and 66.67 vehicles in the flow measure. If matches against a specific measure were thought to be more critical, then these weights could be varied" (Clark, 1999). However, the author provides no insight for structuring the weights in this manner and one must assume that experience led to the choice of values.

## *2.4 Summary*

Nonparametric regression is a heuristic forecasting technique similar to case-based reasoning that consists of three essential components. The historical database stores past observations of the system under consideration. The search procedure finds observations in the historical database that are geometrically similar to the current conditions and passes these neighbors to the forecast function, which estimates the future state of the system.

Nearest neighbor and kernel nonparametric regression techniques have been applied in many different fields with similar results. Nonparametric regression models are asymptotically at least as good as any $K^{th}$ order parametric model, where $K$ is the dimensionality of the state vector (Karlsson and Yakowitz, 1987). A large historical database that represents a broad range of possible conditions is desired to improve forecast accuracy. Furthermore, nonparametric regression techniques have significant advantages over time series methods when applied to chaotic systems due to their ability to geometrically reconstruct past behavior (Mulhern and Caprara, 1994).

However, there are several obstacles that still need to be overcome to successfully use nonparametric regression in real-time systems.

- Nonparametric regression requires a significant amount of processing time (Karlsson and Yakowitz, 1987; Davis and Nihan, 1991), especially as the size of the historical database increases.
- The size of the neighborhood (in terms of *k* or bandwidth) is critical to forecast accuracy (Schaal, 1994) and must be determined.
- Nonparametric regression requires large amounts of storage because forecast accuracy improves as the size and quality of the historical database increases (Davis and Nihan, 1991; Schaal, 1994; Charytoniuk, Chen, and Olinda, 1997).
- The lack of a systematic method for applying nonparametric regression, especially for online applications where the time required to compute an estimate is just as important as the accuracy of the forecast, hampers its widespread use.

# 3 Nearest Neighbor Nonparametric Regression in Real-Time Systems

“Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced” (Ramamrithan and Stankovic, 1994). Real-time systems are used in many applications including modern traffic management, command and control centers, manufacturing, and inventory and supply chain management.

Scheduling tasks in real-time systems has been extensively studied in the computer science and operations research communities. Scheduling involves the static or dynamic selection of tasks to be executed based on timing constraints, priority, resource requirements, precedence relationships, communication requirements, or some other relevant set of criteria. Ultimately, the metrics that guide scheduling decisions depend on the specific applications of the real-time system.

For this research, consider a real-time system where tasks are scheduled according to timing constraints, which place beginning and ending bounds on the execution of a task. Start time refers to the earliest possible time that a task may begin while the deadline is the latest possible time before which the task must finish. In other words, for a task to meet its timing constraints, it must begin after the designated start time and end before the deadline.

A real-time system is overloaded when one or more tasks cannot meet their deadlines. When such a situation occurs, a common resolution is to shed the load by halting or aborting the task or set of tasks with the lowest priority (or some other criterion based on the specific application of the system). However, in coupled applications where tasks are executed sequentially and each is essential to the overall success of the application, load shedding is not feasible. For example, the three components of nonparametric regression can be considered as individual tasks that execute sequentially where the outputs from the previous task are inputs to the next task. If the search procedure overloads the real-time system and is halted or aborted, the forecast generation function does not receive the neighbors in time to create a forecast and fails to meet the deadline, therefore causing the entire nonparametric regression application to fail.

Slow execution time is a significant constraint to employing nonparametric regression in real-time systems (Karlsson and Yakowitz, 1987; Davis and Nihan, 1991). Examination of the nonparametric regression algorithm reveals that the majority of the execution time is spent searching for past observations similar to the current conditions. This is largely due to inefficient sequential search techniques that find nearest neighbors by computing the distance from the current condition to *every* observation in the historical database. Sequential search techniques are $O(KN)$, where $K$ is the dimensionality of the state vector and $N$ is the number of observations in the historical database. As the size of the historical database or dimensionality of the state vector increases, the expected execution time increases linearly.

Database searching, in general, is a pervasive problem for nonparametric regression. Many of the authors cited in section 2.3 indicate that forecast accuracy increases with a larger database that more accurately and thoroughly represents a broader range of possible system conditions. However, the time to find nearest neighbors also increases for a larger database, which is problematic for nonparametric regression in real-time systems. Thus a trade-off exists: forecast accuracy improves with a larger, more representative database but at the expensive of execution time.

Recent research indicates that there are two promising approaches for reducing execution time, both of which concentrate on speeding the nearest neighbor search process: advanced data structures and imprecise computations.

## *3.1 Advanced Data Structures*

Nearest neighbor searching, also called the post office problem, is a fundamental problem in the computer science community (Arya and Mount, 1993) and research has led to advanced data structures that offer significant time-savings compared to sequential searches. These data structures are generally multidimensional variants of binary search trees such as KD trees (Bentley, 1975; Friedman et. al., 1977; Bentley, 1979), BBD trees (Arya et. al., 1998), or VP trees (Yianilos, 1993). Other approaches such as D(S) or M(S,Q) structures (Clarkson, 1997) have also been proposed. KD trees were applied in this research due to their superior performance characteristics.

Arya et. al. (1998) introduced the concept of a balanced box-decomposition tree (BBD tree) to speed approximate nearest neighbor searches by combining promising characteristics of several previous data structures into one structure. However, the authors empirically determined that “there is little or no significant practical advantage to using the BBD tree over the KD tree” (Arya et. al., 1998).

VP trees are very similar to KD trees but use distances from selected vantage points (hence VP) to partition the space whereas KD trees partition at coordinate points. Unfortunately, VP trees require a distance metric prior to constructing the data structure. KD trees, on the other hand, operate independently of the distance metric. Therefore, VP trees must be rebuilt each time the distance metric is changed while KD trees are constructed only once, thus significantly reducing the overall time spent preprocessing the data when the distance metric changes.

D(S) and M(S,Q) data structures are based on Voroni regions and Delaunay neighbors. During construction, however, the number of nearest neighbors to select during the search process is required. The preprocessing time is exponentially dependent on *k*, thus being less efficient than KD trees, whose preprocessing time is independent of the number of nearest neighbors. Similar to how VP trees must be rebuilt if the distance metric changes, D(S) and M(S,Q) data structures must be rebuilt if *k* changes. The search times for D(S) and M(S,Q) data structures are also exponentially dependent on the number of nearest neighbors. Furthermore, M(S,Q) cannot be used use in online systems where the current conditions are constantly changing because the data structure requires a complete list of points to search for prior to constructing the data structure.

KD trees are summarized below.

### *3.1.1 KD Trees*

KD trees are multidimensional variants of traditional binary search trees. In a state vector with *K* elements, records are represented by *K* keys. Records in a binary search tree, on the other hand, have only one key and are partitioned such that records with keys less than or equal to the key of the parent node reside to the left while keys greater than the parent reside to the right, as shown in Figure 8.

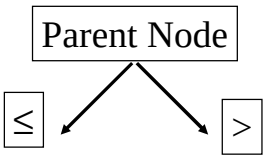


**Figure 8 - Partitioning at a Node**

With multidimensional data, such as the state vector defined by Equation (3), any one of these keys can be used to partition the records. KD trees are constructed by partitioning the records by cycling through the keys in order. In other words, the root node partitions the records based on the first key $V(t)$ of the first record in the historical database, the nodes in the next level use the second key $V(t\text{-}1)$ of the next records encountered, and so on. Once all of the keys have been used to partition the records, the nodes in the next level use the first key and the cycle repeats until all records are placed in the tree. The expected cost of constructing a KD tree with $N$ records is $O(KN \log N)$.

KD trees effectively partition the data into $K$-dimensional boxes "so that given any query point $q$, the $k$ nearest points [in the historical database] to $q$ can be reported efficiently" (Mount, 1998). Every internal node in the tree defines upper and lower bounds on the keys of all points contained in the sub-trees below. The two-dimensional example in Figure 9 graphically shows the resulting partitions of a KD tree. Notice "that as one descends any path in the tree, the geometric size of the associated regions of space (defined, for example, to be the length of the longest size of the associated rectangle) decreases exponentially" (Arya et. al., 1998).

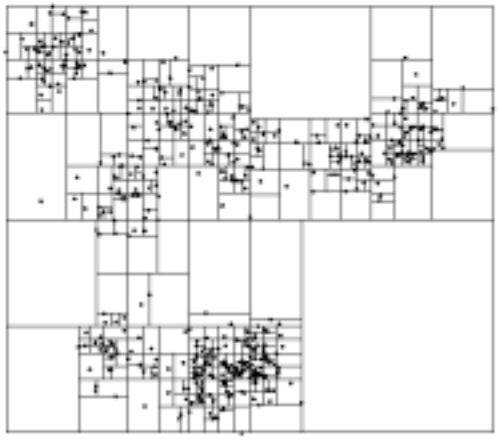

**Figure 9 - KD Tree Partitions (Mount, 1998)**

In addition to describing the tree, Friedman et. al. (1977) provide a nearest neighbor search procedure that involves two tests. The 'Bounds Overlap Ball' test uses the bounds defined by the internal nodes to determine which sub-trees to search while the 'Ball Within Bounds' test determines when to stop searching. Thus, nearest neighbor searches using KD trees examine only a small subset of the available data and significantly reduce the query time to $O(\log N)$ (Friedman et. al., 1977).

The 'Bounds Overlap Ball' test determines if a sub-tree defined by the current internal node may contain points that are closer to the query point than any point currently in the neighborhood. If so, then the sub-tree is searched recursively. As the nearest neighbor search algorithm traverses the tree, it encounters internal and leaf nodes and stores the $k$ nearest neighbors. Let $x_{(k)}$ be the nearest neighbor encountered thus far that is furthest from (i.e. least similar to) the current conditions $x_q$. Therefore, $dist(x_q, x_{(k)})$ is the distance from the query point to the furthest nearest neighbor. The 'Bounds Overlap Ball' test finds the smallest dissimilarity between the bounded region of the box defined at the current internal node and the query record. If the query record's $K^{th}$ key is within the bounds defined for the $K^{th}$ coordinate of the box, then the $K^{th}$ partial distance is set to zero, indicating the best potential match of a point that may exist in the sub-tree. Otherwise, the dissimilarity is incremented by the coordinate distance by which the key falls outside the box. If this dissimilarity is greater than $dist(x_q, x_{(k)})$, then the sub-tree represented by the internal node can contain no points that are closer to the query point

than any of the current nearest neighbors and the sub-tree can be eliminated from consideration (Friedman et. al., 1977).

The ‘Ball Within Bounds’ test, which is also performed at each internal node, determines if sub-trees other than the current need to be searched. If not, the search terminates after the current sub-tree is searched recursively. The coordinate distance from the query record to the closer boundary of the box for each key is compared to the distance to the furthest nearest neighbor. If all such coordinate distances are greater than $dist(x_q, x_{(k)})$, then the distances to all the points in the neighborhood are less than the distance to the furthest point that may exist in the current sub-tree, thus indicating that points in any other sub-tree are still further from the query point and need not be searched (Friedman et. al., 1977).

Pseudo-code for the ‘Bounds Overlap Ball’ and ‘Ball Within Bounds’ tests may be found in Friedman et. al. (1977).

## *3.2 Imprecise Computations*

In real-time systems, one alternative scheduling solution involves the use of imprecise computations. In general terms, “each time-critical task is designed in such a way that it can produce a usable, approximate result in time whenever a failure or overload prevents it from producing the desired, precise result” (Liu et. al., 1994). In other words, some accuracy is sacrificed so that the task can meet its deadline. While tradeoffs between accuracy and timeliness are subject to the specific application of the real-time system, it is essential for some systems to have imprecise results instead of none at all. For example, in military target acquisition and tracking systems, it is clearly advantageous to have timely, rough estimates of target location rather than accurate location information that is late or does not exist at all.

### *3.2.1 Approximate Nearest Neighbors*

Approximate nearest neighbor searching is one method of using imprecise computations with nonparametric regression in real-time systems. ANN searching is employed in

conjunction with advanced data structures and involves the selection of neighbors that are sufficiently close to the query point, but are not necessarily the closest (i.e. exact) neighbors. Approximate nearest neighbors are defined as follows.

> "Given $\varepsilon > 0$, point $p$ is a $(1+\varepsilon)$-approximate nearest neighbor of the query point $q$ if $dist(p, q) \leq (1+\varepsilon)dist(p^*, q)$, where $p^*$ is the true nearest neighbor to $q$. In other words, $p$ is within relative error $\varepsilon$ of the true nearest neighbor. More generally, for $1 \leq k \leq N$, the $k^{th}$ $(1+\varepsilon)$-approximate nearest neighbor of $q$ is a data point whose relative error from the true $k^{th}$ nearest neighbor of $q$ is $\varepsilon$" (Mount, 1998).

The implementation of ANN used in this research employs a modified KD tree data structure that permits searches for approximate nearest neighbors. Specifically, the 'Bounds Overlap Ball' and 'Ball Within Bounds' tests described in section 3.1.1 were altered to use $\varepsilon$. Recall that the 'Bounds Overlap Ball' tests determines which sub-trees to search based on the distance to the furthest nearest neighbor encounter thus far, $dist(x_q, x_{(k)})$, while the 'Ball Within Bounds' test also uses $dist(x_q, x_{(k)})$ to determine when to stop searching. These tests were modified to use $(1+\varepsilon)dist(x_q, x_{(k)})$. Therefore, setting $\varepsilon = 0$ forces the search procedure to find exact nearest neighbors and allows for examination of the performance of the KD tree alone. Setting $\varepsilon > 0$ permits the examination of the performance of approximate nearest neighbors in conjunction with KD trees.

The use of approximate nearest neighbors may result in poorer forecast accuracy due to the decreased quality of neighbors selected for use in the forecast function. This research effort, in part, evaluates the effect of approximate nearest neighbors on forecast accuracy for six case studies.

## *3.3 Tuning Options*

Advanced data structures such as KD trees and imprecise computations such as approximate nearest neighbors employ complex algorithms capable of speeding nearest neighbor searches. The complexity associated with each method is two-fold, however. In addition to requiring more code to implement, both methods have numerous 'tuning'

options, which are used to adjust the performance of the search procedure. In traditional nonparametric regression, only the state vector, dissimilarity measure, number of neighbors, and forecast function need to be determined. The following sections describe several of the options present in the implementations of KD trees and approximate nearest neighbors used in this research.

### *3.3.1 Bucket Size*

The actual implementation of KD trees used in this research does not store records in internal nodes as originally proposed by Bentley (1975). Instead, each internal node records the key used to partition at that level and the splitting value. Therefore, records are stored only at leaf nodes, which are also called terminal nodes. Bucket size refers to the maximum number of records stored at each terminal node. Typically, bucket size is set to one, in which case each terminal node contains at most one record. Note that terminal nodes can hold any number of records less than or equal to the bucket size, as shown in Figure 10.

When the nearest neighbor search algorithm encounters an internal node, it uses the ‘Bounds Overlap Ball’ test to determine which sub-trees to search and the ‘Ball Within Bounds’ test to determine when to stop searching. Lower bucket sizes result in trees with more internal nodes, as indicated in Figure 11, and force the search procedure to run more tests when finding neighbors. On the other hand, larger bucket sizes cause the search procedure to encounter more historical observations in each terminal node and, therefore, make more distance calculations.

For example, consider two KD trees that each contain fourteen records. The tree in Figure 10 has a bucket size of three while the tree in Figure 11 uses a bucket size of one. Figure 10 contains three levels of internal nodes whereas Figure 11 has one more level and eight more splitting nodes. Searching for record 1 in Figure 10 requires traversing from the root node, through two other internal nodes, and examining three records. Using the tree in Figure 11, the same search beginning at the root traverses through three internal nodes but only examines one record.

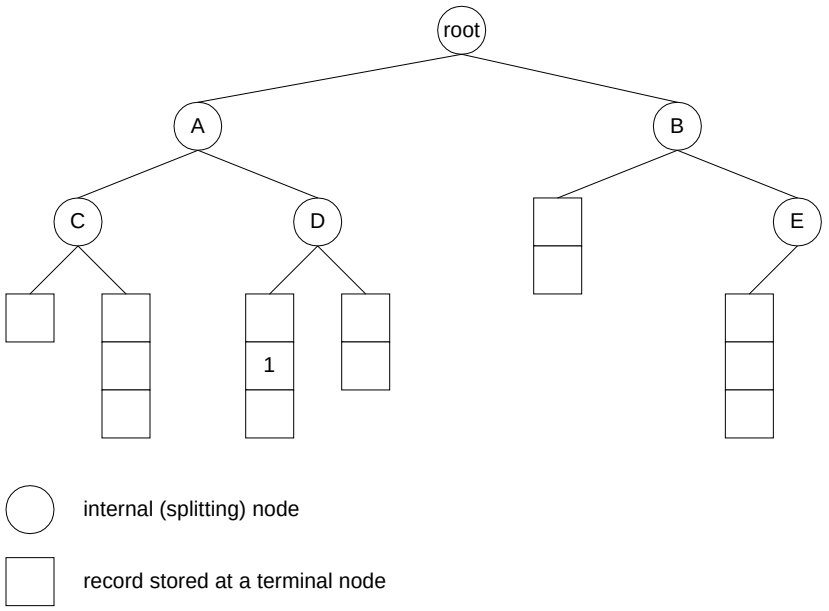


**Figure 10 - Example KD Tree with Bucket Size of Three**

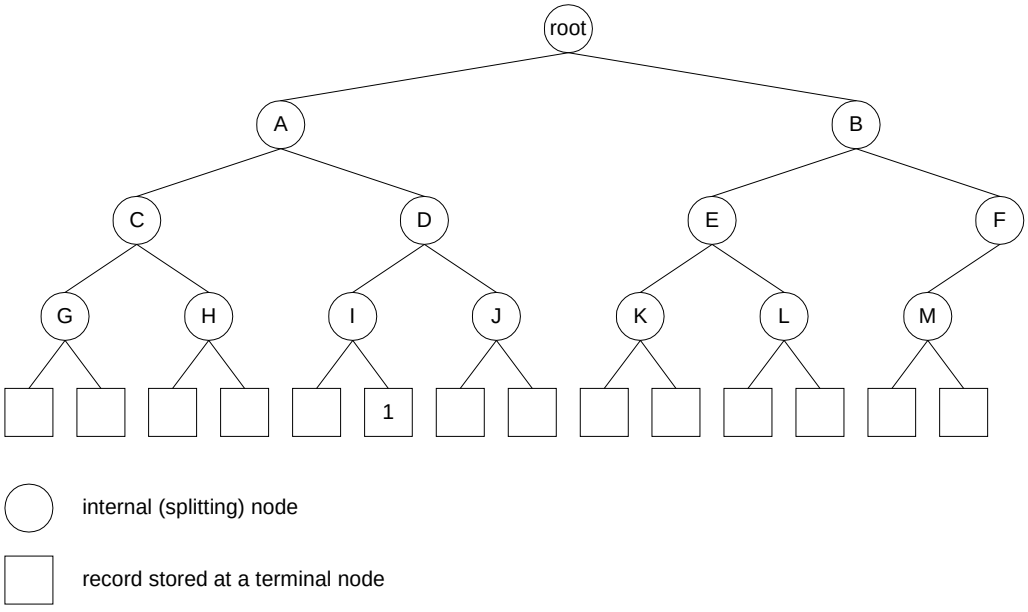


**Figure 11 - Example KD Tree with Bucket Size of One**

### *3.3.2 Splitting Rule*

The shape of the KD tree as originally proposed by Bentley (1975) depends on the order of the records. For example, if records are sorted such that every key in a record is less

than or equal to every key in the preceding record, a linear sub-tree results. In a traffic flow database, such a situation regularly occurs after peak traffic flows (i.e. rush hour) as volume measures return to normal. The search procedure performs less than optimally when the tree is not balanced and highly linear sub-trees should be avoided whenever possible. A tree is balanced if the number of nodes in the left and right sub-trees of every internal node differs by at most one. The trees in Figure 10 and Figure 11 are not completely balanced but the sub-tree defined by node A in Figure 11 is balanced. Figure 12 represents a highly linear tree.

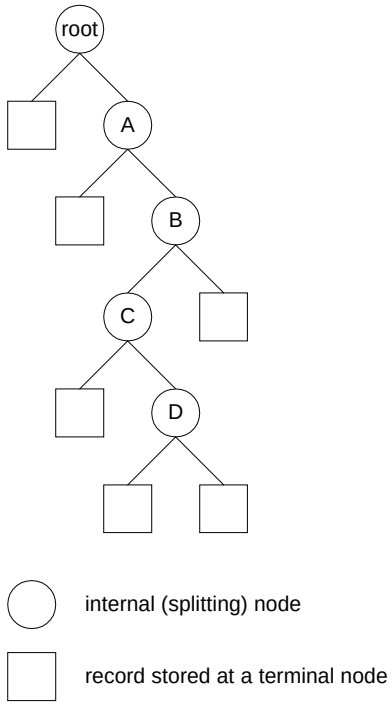


**Figure 12 - Highly Linear KD Tree Example**

Several modifications to the KD tree data structure that eliminate the dependence on record order have been proposed by Mount (1998) and are discussed below.

Let *S* denote the current subset of historical observations to be stored in the tree and let *N* be the total number of observations in the historical database. Each internal node of the tree effectively defines a *K*-dimensional rectangle, also called a box or cell, in which the values for all records in the sub-tree must be within. Define the aspect ratio of a rectangle to be the ratio between its longest and shortest side lengths. Large aspect

ratios indicate that the search procedure may not perform well (Mount, 1998), especially if the distribution of the query points is drastically different than the distribution of historical observations in the tree (Eastman, 1984).

The *standard splitting rule* chooses the splitting dimension as the key with the largest range of values. The splitting value is the median of the splitting dimension. This rule ensures that the final tree height is $\log_2 N$. However, the resulting cells may have arbitrarily high aspect ratio.

The *midpoint-splitting rule* guarantees that cells have bounded aspect ratio. This method examines the current *K*-dimensional box and makes an orthogonal cut along the longest side at its midpoint. If more than one dimension of the current box have equal lengths, the rule splits the dimension whose values have the largest range. Unfortunately, the rule does not consider the distribution of the data with respect to the orthogonal cut and may result in trivial splits such that all of the points in *S* are on one side of the cutting plane. Node F in Figure 11 represents the results of a trivial split because there are no nodes in the right sub-tree. As a result of many trivial splits, the depth of the resulting tree can be large and may even exceed *N* if the historical observations are highly clustered.

The *sliding midpoint rule* is a modification of the midpoint splitting rule such that no trivial splits are allowed. If points in *S* will lie on both sides of a proposed midpoint split, then the algorithm behaves exactly as the midpoint splitting rule. However, if a trivial split were to result, the sliding midpoint rule moves the splitting plane to ensure that points are more evenly distributed on both sides of the splitting plane. Because trivial splits are avoided, the maximum depth of the tree is *N*. "It is possible to generate a cell *C* of very high aspect ratio, but it can be shown that if it does, then *C* is necessarily adjacent to a sibling cell *C'* that is fat along the same dimension that *C* is skinny. It turns out that cells of high aspect ratio are not problematic for nearest neighbor searching if this occurs" (Mount, 1998).

The *fair split rule* is a compromise between the standard splitting rule and the midpoint splitting rule. The fair split rule attempts to construct balanced partitions that do not exceed an arbitrary maximum aspect ratio. For a given cell, the algorithm identifies all possible splits that do not result in cells that exceed the maximum aspect ratio. The

side whose values have the largest range is split such that an even number of points lie on both sides of the partition, subject to maintaining the bound on the maximum aspect ratio. This method provides more robust splits when the data are highly clustered but may result in trivial splits to maintain the bound on maximum aspect ratio, thus permitting the depth of the tree to exceed *N*.

Finally, the *sliding fair split rule* combines the strengths of the fair split rule and the sliding midpoint rule. This splitting rule operates by first identifying the longest side with the largest spread that may be split without violating the maximum aspect ratio. It next identifies the most extreme cuts that would be allowed by the aspect ratio bound. If the midpoint split lies between these extremes, then use it. If not, use the extreme cut that is closest to the median. If a trivial split were to result, slide the cutting plane towards the data until at least one point lies on both sides of the split. The sliding fair split rule results in balanced splits or splits such that every skinny cell has a sibling fat cell as with the sliding midpoint rule.

### *3.3.3 Search Method*

The search procedure used in this research is an adaptation of the search algorithm originally described by Friedman et. al. (1977) but extended to allow for approximate nearest neighbor searching. The two search methods provided control the rate at which the algorithm converges on the nearest neighbors.

The *standard search* procedure is the simpler of the two methods and begins at the root node. At any given non-terminal node, the algorithm first visits the sub-tree that is closest to the query point. When finished, if the other sub-tree may contain an observation within $1/(1+\varepsilon)$ times the distance to the closest point seen so far, then that sub-tree is visited recursively. Note that exact nearest neighbors are found when $\varepsilon = 0$.

On the other hand, the *priority search* method begins searching in the cell where the query point would be located if it were in the tree. Cells are visited recursively in increasing order of distance from the query point. The search ends when the next sub-tree to be searched cannot contain any points that are within $1/(1+\varepsilon)$ times the distance to the closest point seen so far.

## *3.4 Other Approaches*

Certainly there are other methods of reducing the execution time of nonparametric regression. One approach is to use faster equipment. Moore's Law states that computer processor speeds double roughly every eighteen months. However, the sale of hard disks has increased at a rate twice that of Moore's Law, or roughly every nine months. With the majority of data being stored on these disks for use in decision support systems, which involve data-mining and forecasting, computer processor speeds are not keeping pace with the amount of data being collected and analyzed, therefore making the purchase of new equipment to run inefficient algorithms prohibitively expensive.

Limiting the size of the historical database is another method of reducing execution time specific to nonparametric regression. Because nonparametric regression searches the historical database each time a forecast is generated, limiting the size of the database places an upper limit on the number of neighbors to search in the worst case and, therefore, limits execution time. However, as the literature indicates, a larger database that represents a wide variety of conditions is desirable. Heuristically truncating the historical database limits the past conditions (especially observations from abnormal operating conditions) that nonparametric regression may use when generating forecast and, therefore, may reduce forecast accuracy.

Advanced data structures and approximate nearest neighbors hold the most promise for significantly reducing the execution time of nonparametric regression without the requirements of a large budget while maintaining sufficiently large historical databases. Although approximate nearest neighbors may reduce forecast accuracy, moderating the size of the historical database permanently removes observations that may be valuable in estimating future conditions whereas ANN retains these points. Nearest neighbor searching using advanced data structures, on the other hand, returns exact nearest neighbors and does not sacrifice forecast accuracy.

## 3.5 Summary

Slow execution time is a significant constraint to employing nonparametric regression in real-time systems (Karlsson and Yakowitz, 1987; Davis and Nihan, 1991). Because the tasks of the nonparametric regression algorithm are sequentially dependent upon their predecessors, dynamic correction techniques such as load shedding are not practical when the system becomes overloaded. Therefore, alternative methods of reducing the execution time of nonparametric regression are needed.

Advanced data structures such as KD trees can dramatically reduce execution time by significantly reducing the number of calculations required to find the exact nearest neighbors. If needed, imprecise computations are achieved through the use of approximate nearest neighbors to further reduce execution time. However, the use of advanced data structures and approximate nearest neighbors increases the complexity of the nonparametric regression algorithm in terms of code and options, furthering the need for a systematic methodology for applying nearest neighbor nonparametric regression.

# 4 Case Studies

Forecasts using approximate nearest neighbor nonparametric regression were generated for six different data sets to determine a general methodology for deploying the model in real-time systems. Two techniques were used to determine optimal values for the tuning options. The first enumerated all possible combinations of the tuning options while the second utilized multi-objective optimization in an effort to reduce the time required to find near-optimal settings. The data, measures of performance, decision variables, and objectives are described below. The remainder of the chapter discusses the results and sensitivity analysis.

## *4.1 Data*

### *4.1.1 Freeway Flows*

Freeway flow data for this research effort were collected at the Hampton Roads Smart Traffic Center (HRSTC), an urban freeway management system, which covers I-64 from just south of the 64-264 interchange to Norfolk Naval Base and I-264 from just west of the interchange to Virginia Beach, as shown in Figure 13. HRSTC measures traffic flow rate, average speed, and occupancy using inductive loop detectors at two-minute intervals. To support this research, data were collected from June 12, 1998 through the end of January 2000 at five locations and aggregated to ten-minute intervals to reduce the noise inherent in flow measurements using short time intervals (Transportation Research Board, 1997). A variety of highway geometries are represented including an exit ramp, two eastbound mainlines, and two westbound mainlines as indicated on the map in Figure 13.

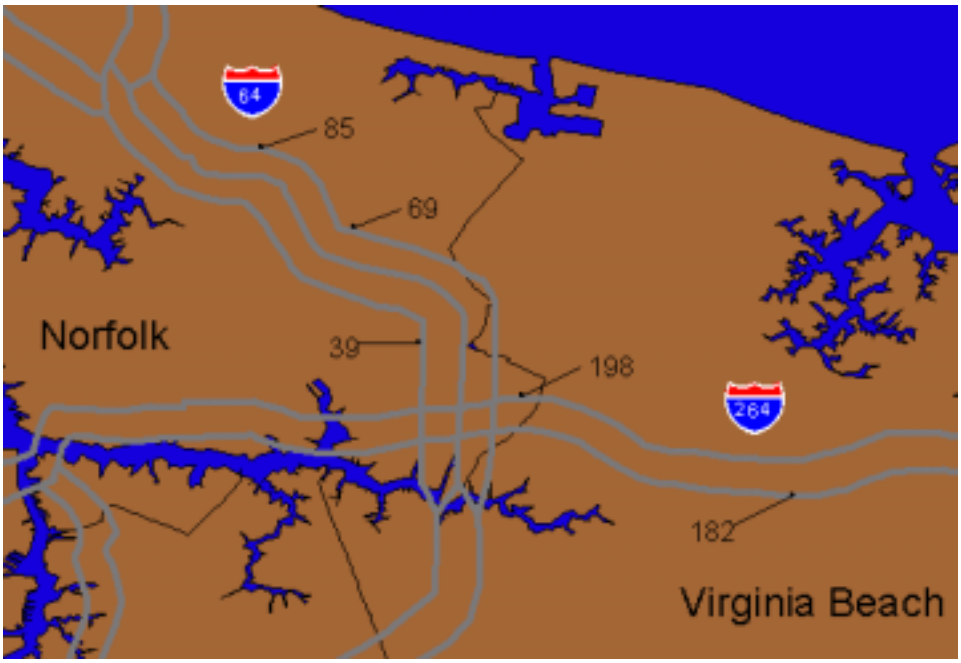


**Figure 13 - HRSTC Coverage Zone with Detector Locations**

The total number of records for each location range in size from 57,000 to 70,000 observations. The large difference in size is due to malfunctioning equipment and removal of erroneous data. The data were divided into two sets for each location with the most recent 4,000-8,000 observations placed in a test set and the remaining observations placed in a historical database. The test sets simulate current conditions for each location on which forecasts were generated. Table 2 describes the locations in more detail.

**Table 2 - Detector Information**

| Station ID (Location) | Highway and Lane Type | Number of Lanes | Training Set Size | Test Set Size |
|---|---|---|---|---|
| 39 | I-64 EB Mainline | 2 | 50,000 | 7,877 |
| 69 | I-64 WB Mainline | 5 | 60,000 | 7,173 |
| 85 | I-64 WB Mainline | 3 | 63,000 | 5,405 |
| 182 | I-264 EB Mainline | 3 | 65,000 | 4,692 |
| 198 | I-264 WB Exit Ramp | 2 | 65,000 | 4,954 |

One week of flow rates for Station 85 are shown in Figure 14. Note that weekend and weekday traffic volumes for this location typically reach 4,000 vehicles per hour during the evening but that the weekends do not demonstrate the pronounced morning

peak that is common during weekdays. Figure 14 also illustrates a gap in observations that began Tuesday evening and ended Wednesday morning. ITS equipment, especially inductive loops that are embedded in roadways, is subjected to extreme weather conditions and occasionally fails, thus causing data loss. Any model employed to estimate future demand must be able to recover from the temporary loss of data and begin generating forecasts once the data connection is restored.

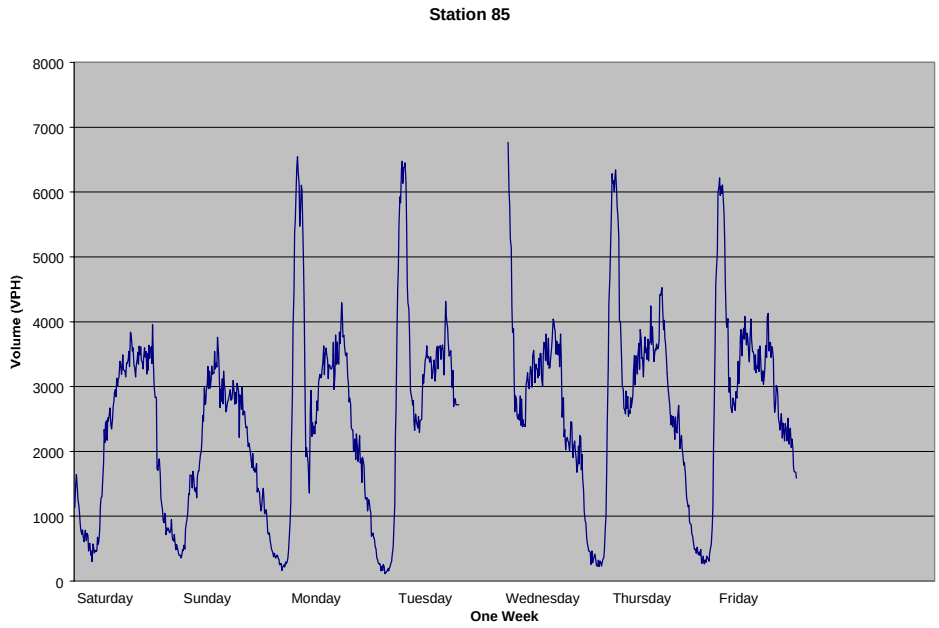


**Figure 14 - Flow Rates for Station 85**

### *4.1.2 Maximum Daily Temperature*

The Australian Bureau of Meteorology collected temperature data for this research for Melbourne from 1981-1990. The maximum daily temperature observations are reported in degrees Celsius and one year is shown in Figure 15. Although there is a noticeable long-term decrease in maximum daily temperature as Melbourne enters the winter season and a subsequent increase as summer approaches, short-term inter-day fluctuations exist and the range of observations appear to vary more widely during the summer than during the winter.

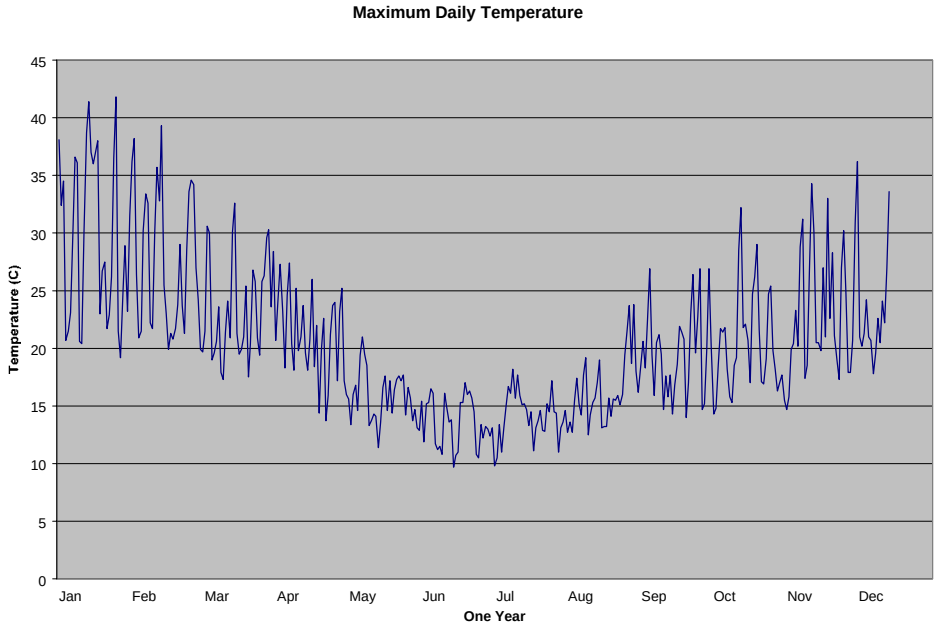


**Figure 15 - Maximum Daily Temperature**

The most recent 2.5 years of data (roughly 950 observations) were placed in a test set and the remaining 2,700 observations were placed in a training set for use in the historical database.

## *4.2 Measures of Performance*

The overall purpose of developing a methodology for deploying nonparametric regression in real-time systems is to generate the best possible forecasts within the constraints of the real-time system by selecting appropriate values for the tuning options. Therefore, forecast accuracy and execution time are the two measure of performance used in this research effort.

Forecast accuracy refers to the mean absolute percent error (MAPE) between estimated and actual observations for all cases in the test set. Since traffic flow observations vary from a few hundred vehicles per hour in the off peak to several thousand vehicles per hour during the peak periods and temperature observations

similarly vary widely over the course of a year, absolute percent error provides the most useful basis for comparison.

Execution time refers to the average time required to find the nearest neighbors for all cases in the test set. The nearest neighbor search component of nonparametric regression is the most time consuming task and the other processes (i.e. receiving the current conditions and forecast generation) occur almost instantaneously. Average query time is a measure of performance that depends on the hardware in the system, operating system, and additional demands placed on the computer. The results presented are for an Intel® Pentium II 350 MHz computer with 128 MB RAM running RedHat® Linux 6.1 in command prompt mode with no other system demands. While individual results may vary, the findings are indicative of general performance.

## *4.3 Decision Variables*

To investigate possible methods of selecting values for the tuning options, this research will consider different values for the number of nearest neighbors ($k$) and relative allowable error ($\varepsilon$) and fix all other tuning options to keep the analysis reasonable. Despite this simplification, in practice, the methods employed in this research effort may be expanded to include all of the other tuning options.

The literature indicates that the choice of $k$ is specific for each application. For example, the most accurate forecasts were generated using $k = 3$ in Karlsson and Yakowitz (1987), $k = 10$ in Smith (1995), and $k = 25$ in Smith et. al. (2000). In general, this research examined cases of $k \leq 100$ in increments of five.

Values for $\varepsilon$ were examined in increments of 0.10 for $\varepsilon \leq 10$. Recall that $\varepsilon$ is the relative error in distance between the true and approximate nearest neighbors. Therefore, a value of $\varepsilon = 1$ indicates that approximate nearest neighbors may be 100%, or twice as far, from the query point as the corresponding exact nearest neighbors. Arya et. al. (1998) used the same range and incremental value for $\varepsilon$ in their investigation of the approximate nearest neighbor search performance of a modified BD tree.

## *4.4 Tuning Option Settings*

Because only two of the possible tuning options are being used as decision variables, acceptable values for the remaining options need to be determined. Table 3 summarizes the values selected and also indicates the total number of possible settings for each.

Several of the settings were chosen based on research by Arya et. al. (1998). The authors investigated the approximate nearest neighbor search performance of a new multidimensional search tree relative to KD trees. Experiments were run using multiple data sets from several different distributions, each consisting of 100,000 data points, 1,000 query points, and 16 dimensions. However, their research investigated the retrieval properties of approximate nearest neighbors and, therefore, the authors can not provide insight about how the various settings may effect ANN-NPR forecast accuracy.

**Table 3 - Tuning Option Settings**

| Tuning Option | Value / Setting | Number of Possible Values / Settings |
|---|---|---|
| State vector | | |
| • freeway | $[V(t), V(t-1), V_{hist}(t), V_{hist}(t+1)]$ | ∞ |
| • temperature | $[V(t), V(t-1), V(t-2), V(t-3)]$ | ∞ |
| Distance metric | Euclidean ($L_2$) | ∞ |
| Forecast function | Straight average | ∞ |
| Bucket size | 1 | $N$ |
| Splitting rule | Standard | 5 |
| Search method | Standard | 2 |

### *4.4.1 State Space*

The state vector for freeway traffic flows used in this research was originally proposed by Smith (1995) and appears in Equation (7). Note that this state space definition includes one lagged measurement ($D$ = 1) and two historical average measures. Because nearest neighbor models "geometrically attempt to reconstruct … a time series" (Mulhern and Caprara, 1994), including historical averages in the state vector further clarifies the position of each observation along the weekly flow-time curve. Therefore, including

historical averages in the state vector should improve forecast accuracy by finding neighbors that are more similar to the current conditions.

(7) $$X(t) = [V(t), V(t-1), V_{hist}(t), V_{hist}(t+1)]$$

For daily temperature data, nearest neighbor nonparametric regression has not been as widely used in predicting future values and a state vector that includes three lagged measures ($D$ = 3) was arbitrarily chosen, as shown in (8). The observations correspond to the current day's temperature and that from the three previous days, respectively. Karlsson and Yakowitz (1987) used a similar state vector for estimating daily rainfall runoff.

(8) $$X(t) = [V(t), V(t-1), V(t-2), V(t-3)]$$

### *4.4.2 Distance Metric*

Nearness from the current conditions $q$ to an observation $p$ in the historical database is defined by Euclidean distance, which is the default distance metric for approximate nearest neighbor searching as implemented by Mount (1998). Euclidean distance is also the most common dissimilarity metric used with nonparametric regression found in the literature. Euclidean distance for the state vector used with the freeway flow data can be computed according to Equation (9) while the distance calculation based on the state vector used for the maximum daily temperature data is defined in Equation (10).

(9) $$dist(p,q) = \sqrt{\begin{aligned}&[V_p(t) - V_q(t)]^2 + [V_p(t-1) - V_q(t-1)]^2 + \\ &[V_{hist\_p}(t) - V_{hist\_q}(t)]^2 + [V_{hist\_p}(t+1) - V_{hist\_q}(t+1)]^2\end{aligned}}$$

(10) $$dist(p,q) = \sqrt{\begin{aligned}&[V_p(t) - V_q(t)]^2 + [V_p(t-1) - V_q(t-1)]^2 + \\ &[V_p(t-2) - V_q(t-2)]^2 + [V_p(t-3) - V_q(t-3)]^2\end{aligned}}$$

The implementation of approximate nearest neighbors used in this research uses squared distances. In other words, the square roots shown in Equations (9) and (10) are not computed to reduce the number of time-consuming floating-point operations performed and speed execution time. "By using squared distances rather than true Euclidean distances, ANN not only saves on the time of computing square roots, but has

the advantage that integer types can be used instead to accurately represent distances when integer type coordinates are used" (Mount, 1998).

### *4.4.3 Forecast Function*

Forecasts were created using a straight average of the dependent variable values of the nearest neighbors, as shown in Equation (6). Smith et. al. (2000) investigated several different forecast functions to improve the accuracy of nearest neighbor nonparametric regression. Their results indicate that forecasts weighted by historical averages and adjusted by distances were more accurate than straight averages. However, the difference in accuracy was less than 1.0% for all values of $k$ investigated. Therefore, this research used straight average forecasts to reduce the number of floating-point operations performed. An implicit assumption is that the use of weighted forecasts will impact the accuracy of alternate approaches equally. In other words, it is assumed that the relative effect of $k$ and $\varepsilon$ on forecast accuracy is independent of the forecast function, which appears to hold true for the results presented in Smith et. al. (2000).

### *4.4.4 Bucket Size*

Friedman et. al. (1977) observed that "to minimize the (upper bound on the) number of records examined, the terminal buckets should contain one record." Recalling the nearest neighbor search procedure described in section 3.1, the search time is proportional to the number of records examined. Therefore, to minimize search time in an effort to meet the timing constraints imposed by the real-time system, bucket size was set to one.

### *4.4.5 Splitting Rule*

Based on the results discussed in Arya et. al. (1998), this research used the standard splitting rule as defined in section 3.3.2. One of their measures of performance was preprocessing time, which is the amount of time required to build the tree. The standard splitting rule used with the KD tree data structure consistently built trees in

approximately 20 CPU seconds, independent of the underlying data distribution. On the other hand, preprocessing time for the proposed multidimensional search tree using the midpoint splitting rule ranged from 20 to 100 CPU seconds. The consistent preprocessing time for the standard splitting rule is a desirable characteristic in real-time systems.

### *4.4.6 Search Method*

Because the priority search technique visits cells in increasing distance from the query point, it should converge on the true nearest neighbors more quickly than the standard search method. In practice, however, "when the error bound [$\varepsilon$] is small, standard search seems to be slightly faster. When large error bounds [$\varepsilon$] are used, … then priority search seems to be superior" (Mount, 1998).

Nonparametric regression assumes that observations in the historical database that are nearer to the query point more accurately represent the current conditions and, as a result, should produce more accurate forecasts. It is logical to assume that values of $\varepsilon$ near zero, which causes the search procedure to retrieve the true nearest neighbors, should produce the most accurate forecasts. Therefore, this research effort used the standard search technique due to the expected value of $\varepsilon$ being small.

## *4.5 Statistical Tests of Significance*

Tests of significance are needed to assess the statistical difference in forecast accuracy between two or more models. Because the distribution of absolute percent errors is not normal, the Wilcoxon signed-rank test for paired observations was used in place of a more traditional analysis of variance approach. The Friedman test evaluates the null hypothesis that three or more related samples are from the same population.

### *4.5.1 Wilcoxon Signed-Rank Test*

The null hypothesis of the Wilcoxon signed-rank test is that the means of two populations $\mu_1$ and $\mu_2$, in this case defined as the absolute percent errors for two models, are

equivalent as indicated in Figure 16. This hypothesis is tested by computing the difference for each of the paired observations and then ranking the differences by absolute value. The null hypothesis is not rejected when the sum of the positive and negative difference ranks are roughly equal. If the sums of the positive and negative difference ranks are substantially different, the alternative hypothesis may be accepted.

$$H_o : \mu_1 - \mu_2 = 0$$
$$H_a : \mu_1 - \mu_2 \neq 0$$

**Figure 16 - Wilcoxon Signed-Rank Test Hypothesis**

### *4.5.2 Friedman Test*

The null hypothesis for the Friedman test is that all models come from the same distribution and, therefore, have equal means as indicated in Figure 17. The observations are first ranked separately within each model and the rank average is then computed for each of the models. When the null hypothesis is true, the rank averages should be relatively close to one another, indicating that within the models the assignment of ranks are equally likely. The alternative hypothesis may be accepted when the rank averages are not close to each other.

$$H_o : \mu_1 = \mu_2 = ... = \mu_I$$
$$H_a : \mu_1 \neq \mu_2 \neq ... \neq \mu_I$$

**Figure 17 - Friedman Test Hypothesis**

## *4.6 Enumeration Results*

Nonparametric regression employing searches with KD tree data structures and approximate nearest neighbors was used to generate forecasts by enumerating possible combination of $k$ and $\varepsilon$ over the ranges discussed in section 4.3. The results illustrate the trade-off between forecast accuracy and execution time, which is a typical dilemma faced

by administrators of real-time systems using imprecise calculations to achieve desired execution times.

### *4.6.1 Forecast Accuracy*

#### 4.6.1.1 Freeway Flow Data

Figure 18 through Figure 22 show how the number of nearest neighbors ($k$) effects forecast accuracy for different values of $\varepsilon$ for the freeway flow data sets. As expected, setting $\varepsilon = 0$, which is equivalent to selecting the exact nearest neighbors, yields the most accurate forecasts for all choices of $k$. For cases where few neighbors are used to create the forecast, $\varepsilon$ has very little effect on forecast accuracy. For example, using the Wilcoxon signed-rank test, there is no statistical difference at the 0.05 significance level in the mean absolute percent error at Station 69 when $k = 5$ and $\varepsilon = 0$ relative to $\varepsilon = 1$.

Note that for exact nearest neighbors ($\varepsilon = 0$), forecast error is a minimum when 20-40 nearest neighbors are used to create the forecast. However, for more than 40 nearest neighbors, forecast accuracy starts to decrease. As $k$ increases, neighbors that do not accurately represent the current conditions are found by the search procedure and used to generate forecasts, thus explaining the decreased forecast accuracy. Similarly, using large values of $\varepsilon$ also selects neighbors further from the current conditions and compromises forecast accuracy. These results are consistent with observations made by Schaal (1994) regarding the importance of appropriate neighborhood size on forecast accuracy as discussed in section 2.3.4.

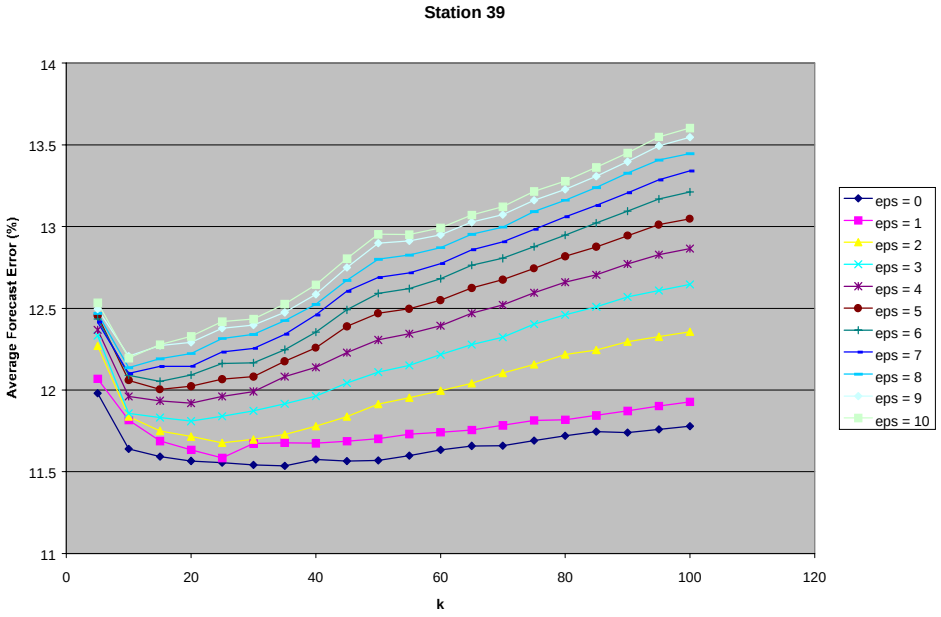


**Figure 18 - Effects of *k* on Forecast Accuracy for Station 39**

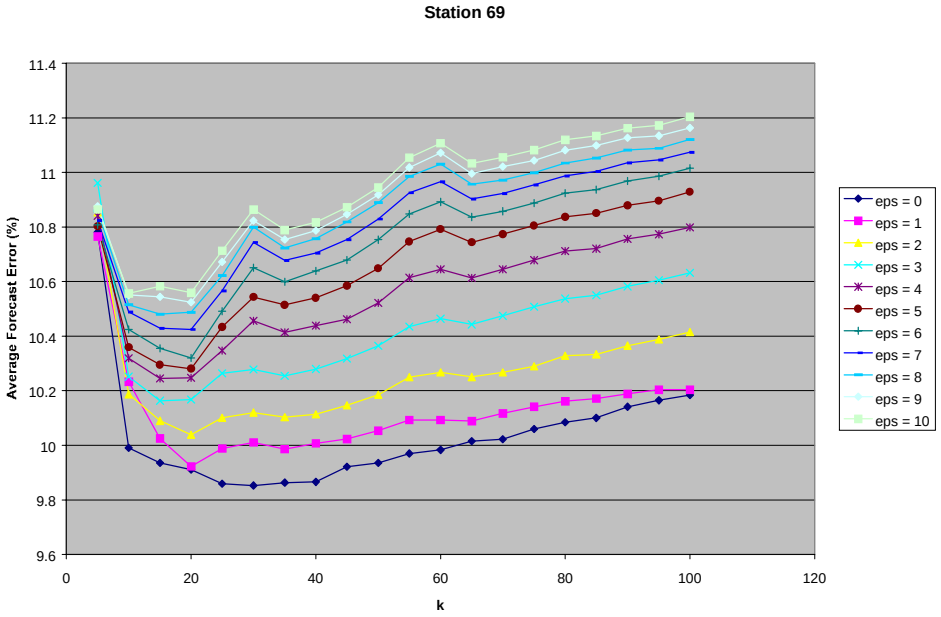


**Figure 19 - Effects of *k* on Forecast Accuracy for Station 69**

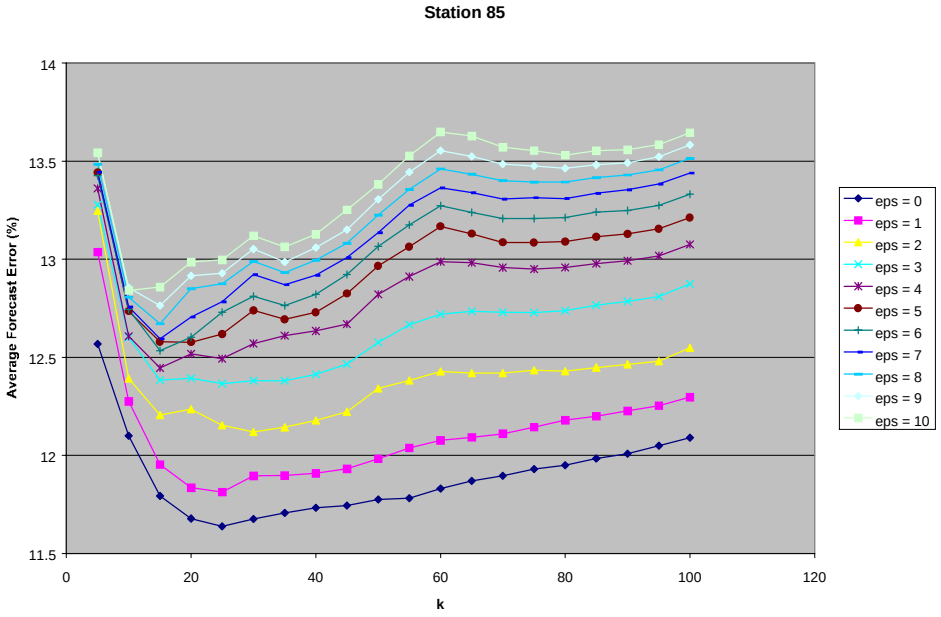


**Figure 20 - Effects of *k* on Forecast Accuracy for Station 85**

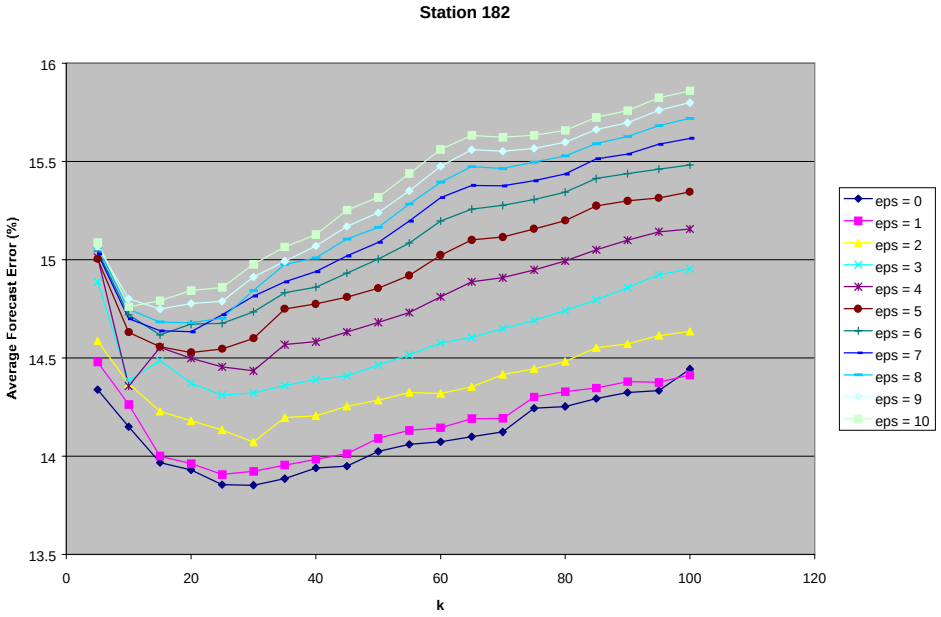


**Figure 21 - Effects of *k* on Forecast Accuracy for Station 182**

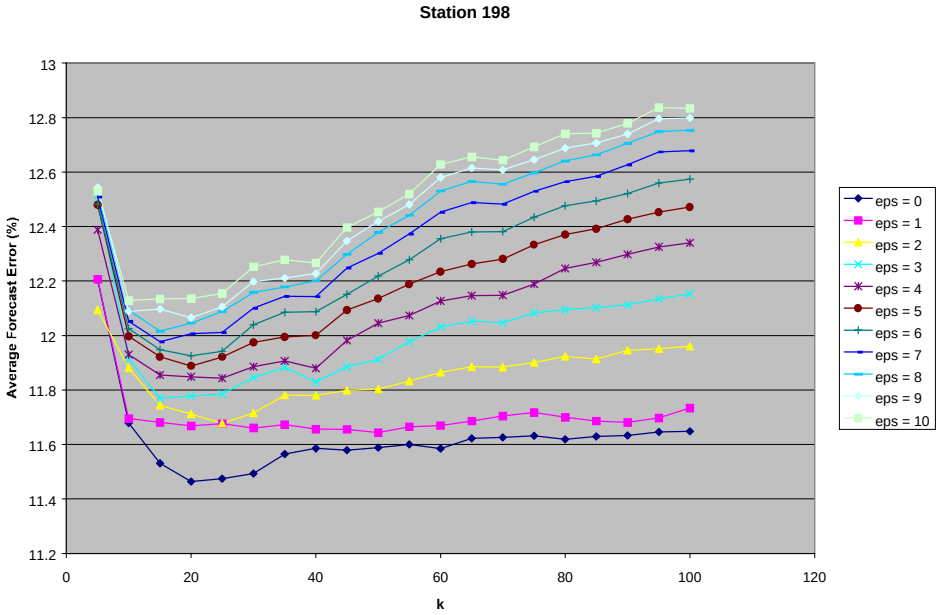


**Figure 22 - Effects of *k* on Forecast Accuracy for Station 198**

Table 4 shows the Wilcoxon Signed-Rank test statistics for each of the settings that minimized forecast accuracy using exact nearest neighbors relative to the other forecasts that used 20-40 exact nearest neighbors. Test statistics less than a given level of significance indicate that the mean absolute percent errors between two settings are statistically different. For example, at the 0.05 level of significance, forecasts calculated using 20 exact nearest neighbors for Station 69 were statistically different than forecasts generated using 30 exact nearest neighbors. The difference in MAPE for Station 198 was also significant for forecasts calculated using 20 and 25 exact nearest neighbors.

**Table 4 - Wilcoxon Signed-Rank Test Statistics for Most Accurate Forecast Relative to 20-40 Exact Nearest Neighbors**

| ***k*** | **Station 39 (*k* = 35)** | **Station 69 (*k* = 30)** | **Station 85 (*k* = 25)** | **Station 182 (*k* = 30)** | **Station 198 (*k* = 20)** |
|---|---|---|---|---|---|
| **20** | 0.462 | 0.023 | 0.220 | 0.308 | --- |
| **25** | 0.080 | 0.171 | --- | 0.420 | 0.026 |
| **30** | 0.743 | --- | 0.743 | --- | 0.082 |
| **35** | --- | 0.584 | 0.346 | 0.149 | 0.715 |
| **40** | 0.531 | 0.682 | 0.485 | 0.745 | 0.863 |

#### 4.6.1.2 Maximum Daily Temperature Data

Figure 23 shows how the number of nearest neighbors ($k$) effects forecast accuracy for different values of $\varepsilon$ for the maximum daily temperature data. Unlike the results from the freeway data sets where $\varepsilon = 0$ consistently produced the most accurate forecasts, there is little difference in forecast accuracy between $\varepsilon = 0$ and $\varepsilon = 1$ for the temperature data. At the 0.05 significance level, the Wilcoxon Signed-Rank test indicates that there is no statistical difference in mean absolute percent error when using 10-100 exact nearest neighbors compared to approximate nearest neighbors with $\varepsilon = 1$. Unlike the freeway flow data, the most accurate forecasts are calculated using 50-60 nearest neighbors.

The non-monotonic behavior of forecast accuracy occurs for the same reasons as with the freeway data. Small values of $k$ cause the search procedure to find a limited set of historical cases that, on average, do not adequately represent the system. Conversely, large values of $k$ result in a neighborhood containing cases that are dissimilar from the current conditions and, as a result, increase forecast error.

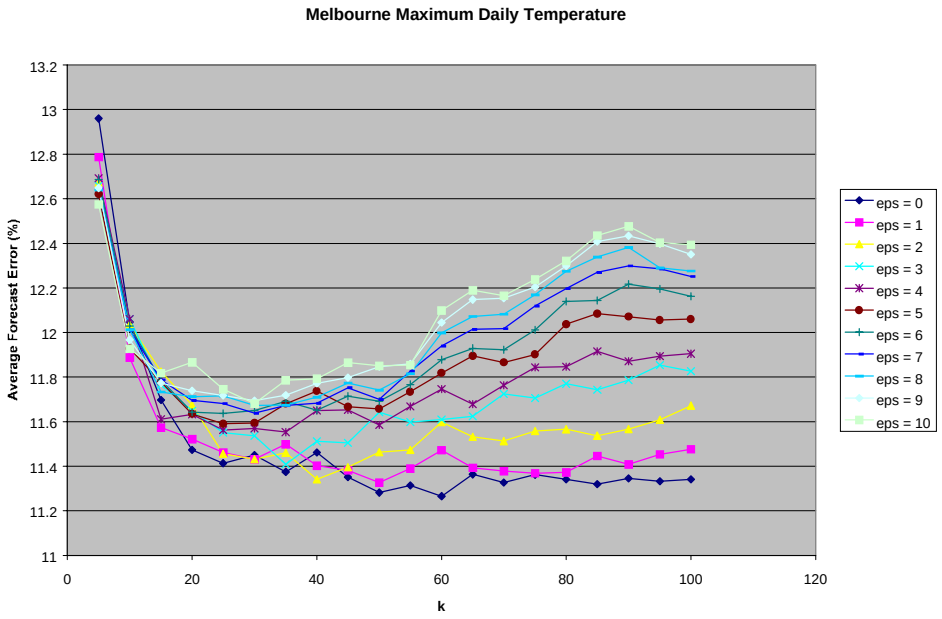


**Figure 23 - Effects of *k* on Forecast Accuracy for Maximum Daily Temperature**

### 4.6.2 *Execution Time*

The decision variables also dictate the amount of time required to generate a forecast. The two figures below compare the time required to find exact nearest neighbors ($\varepsilon = 0$) for two nonparametric regression algorithms using the same historical database with less than 5,800 observations. Figure 24 shows the average query times achieved when using an algorithm that searched for neighbors sequentially while the average query times obtained when using KD trees are presented in Figure 25. Note especially the differences in the vertical scales between the two plots. The algorithm that used sequential search took three orders of magnitude longer per query. In other words, nonparametric regression using KD trees is roughly 1,000 times faster per query than nonparametric regression with sequential searching. Furthermore, average query time increases linearly with the number of nearest neighbors when searching is aided by KD trees, but increases exponentially when sequential searching is used. As the number of nearest neighbors or the size of the historical database increases, nonparametric regression using sequential search techniques quickly becomes impractical in real-time systems. Advanced data structures such as KD trees efficiently overcome this obstacle.

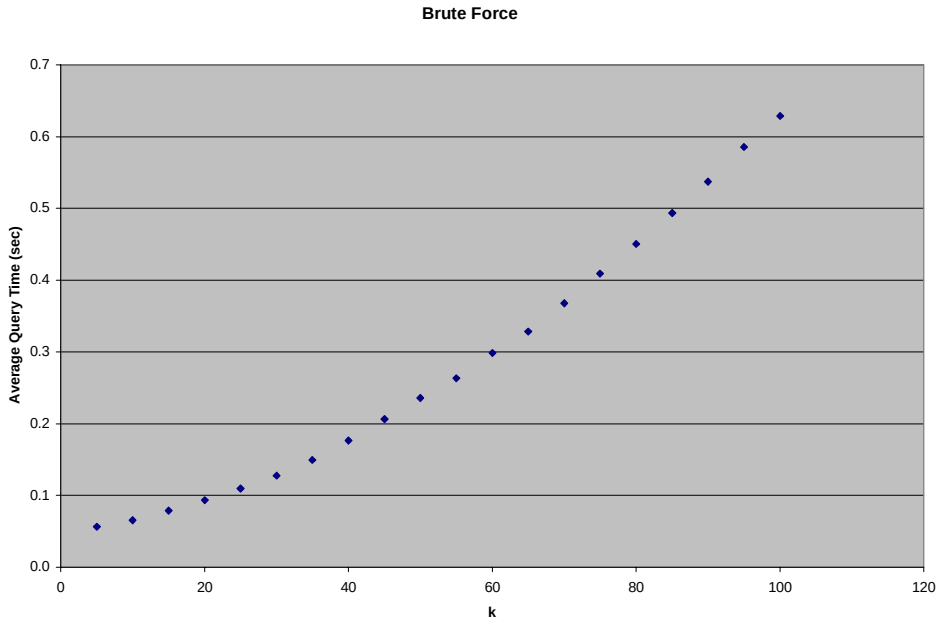


**Figure 24 - Average Query Times for NPR Using Brute Force Searches**

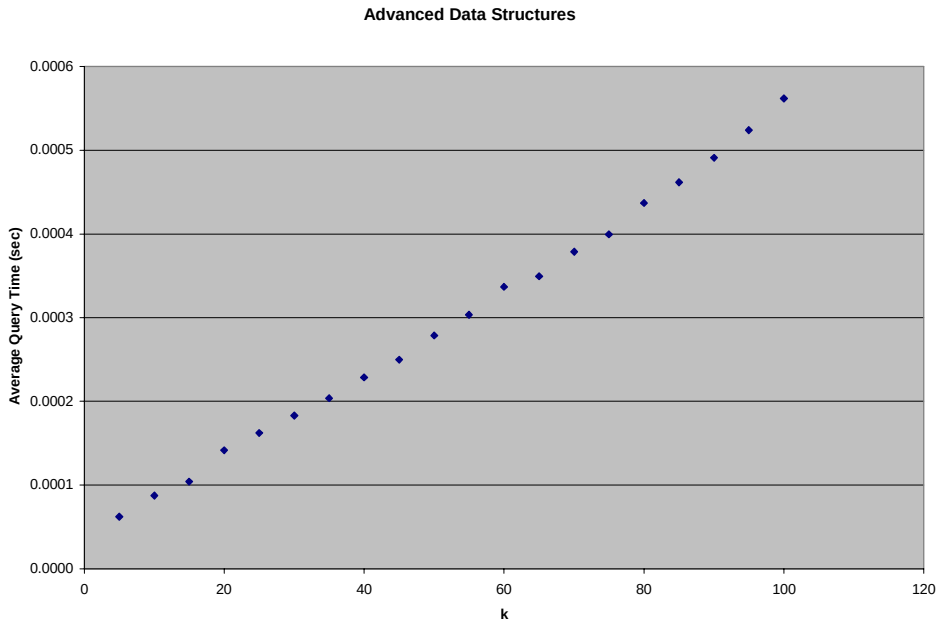


**Figure 25 - Average Query Times for NPR Using Searches with Advanced Data Structures**

The amount of time saved by using advanced data structures varies with the number of nearest neighbors, though, as seen in Figure 26. The vertical axis represents the ratio of average query times of the brute force search compared to the method using advanced data structures. In other words, the vertical axis is the number of queries that can be executed using the search method with advanced data structures in the same amount of time that the brute force search conducts one query. As the number of nearest neighbors ($k$) increases, the search using advanced data structures becomes increasingly more efficient and can execute more queries in the same amount of time required by the brute force search to execute one query, which is represented by the positive slope of the graph for $k \geq 20$.

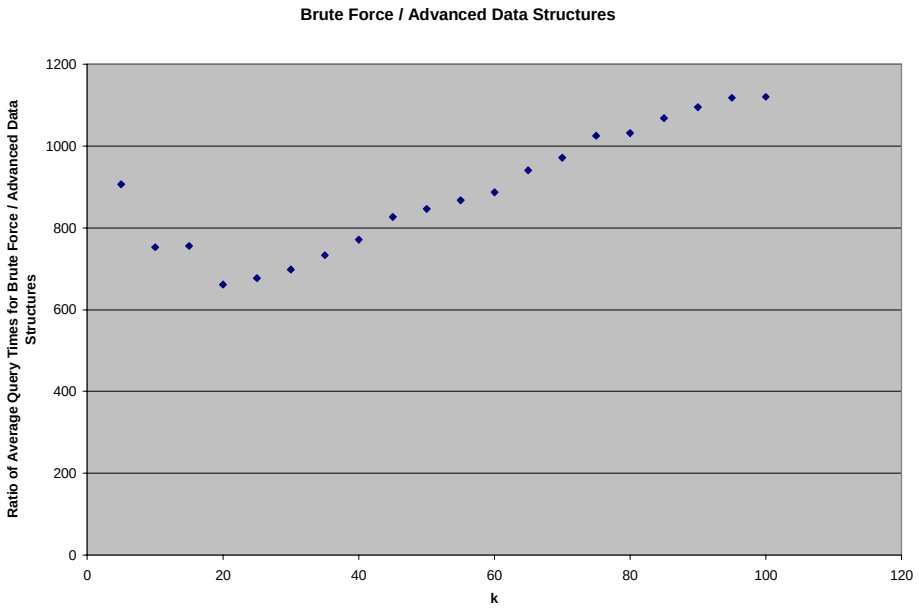


**Figure 26 - Ratio of Average Query Times**

#### 4.6.2.1 Freeway Flow Data

Figure 27 through Figure 31 show how the number of nearest neighbors ($k$) effects average execution time for different values of $\varepsilon$ for both data sets. Clearly the most time consuming search involves finding the largest number of exact nearest neighbors (i.e. $k$ = 100, $\varepsilon = 0$). As $\varepsilon$ increases, average query time decreases nonlinearly and appears to reach an asymptote near $\varepsilon = 10$. In other words, the most significant decrease in average execution time is achieved by using approximate nearest neighbors with $\varepsilon = 1$ compared to exact nearest neighbors ($\varepsilon = 0$). Furthermore, improvements in average query time appear negligible as $\varepsilon$ approaches 10.

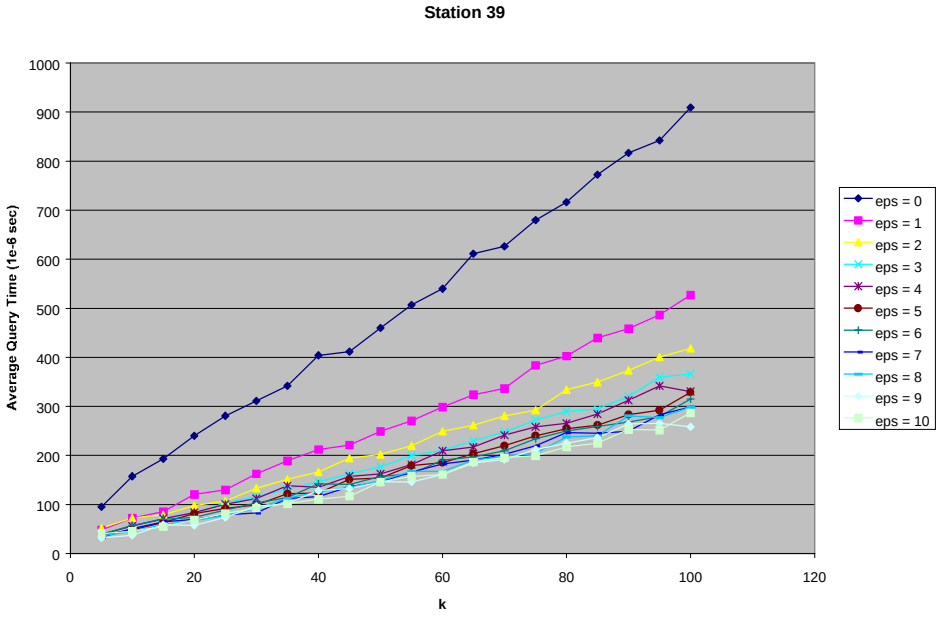


**Figure 27 - Effects of *k* on Query Time for Station 39**

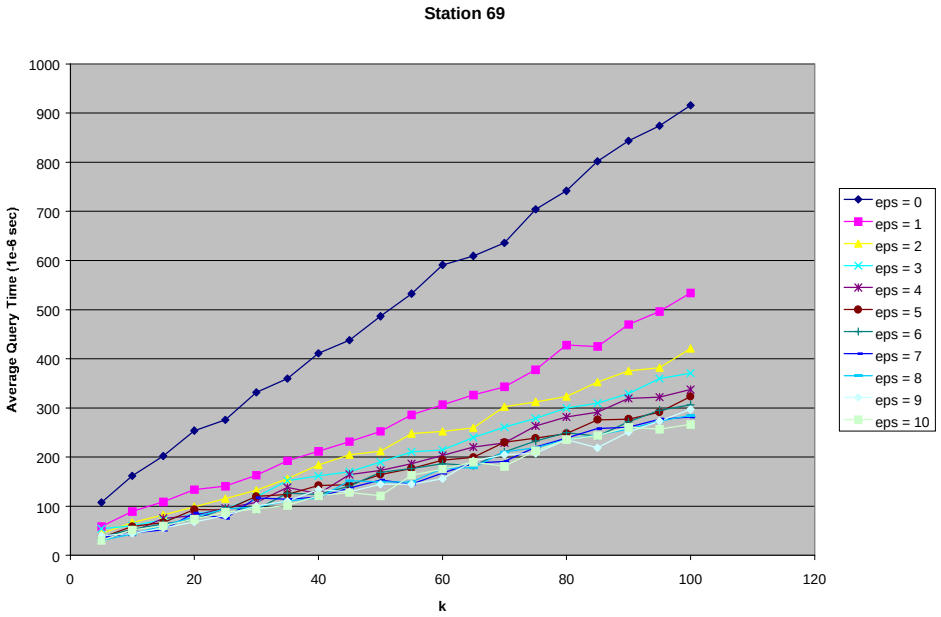


**Figure 28 - Effects of *k* on Query Time for Station 69**

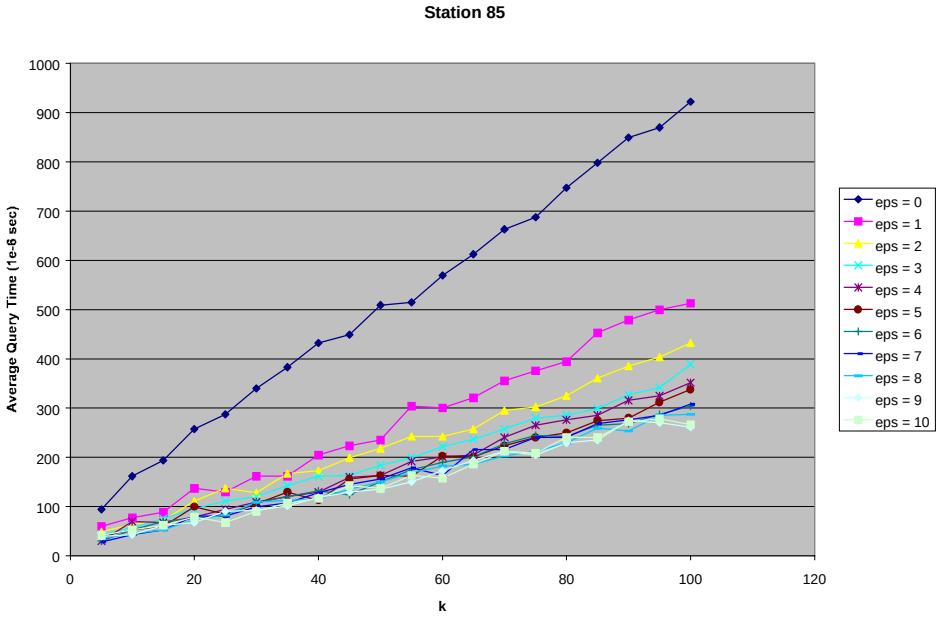


**Figure 29 - Effects of *k* on Query Time for Station 85**

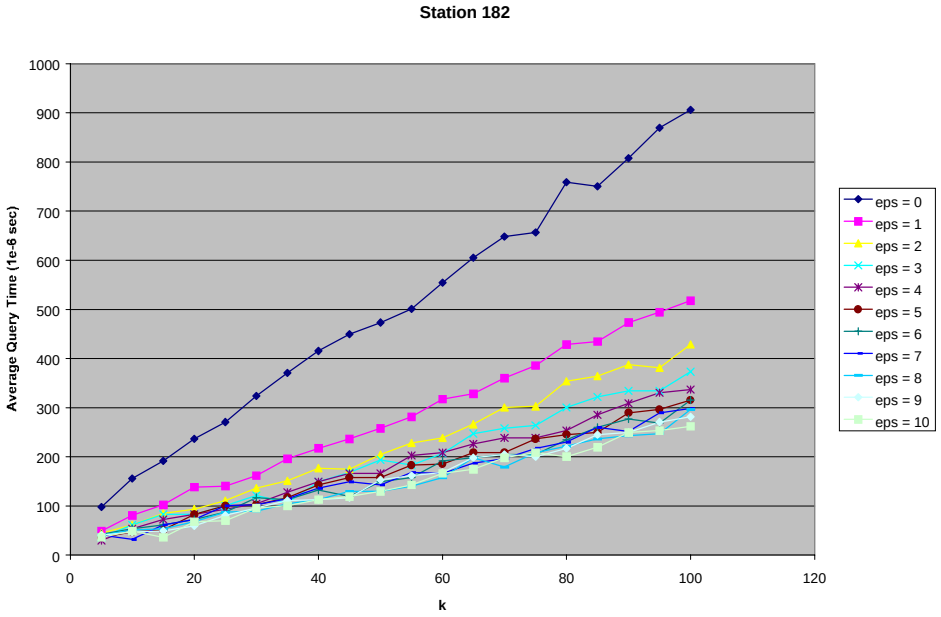


**Figure 30 - Effects of *k* on Query Time for Station 182**

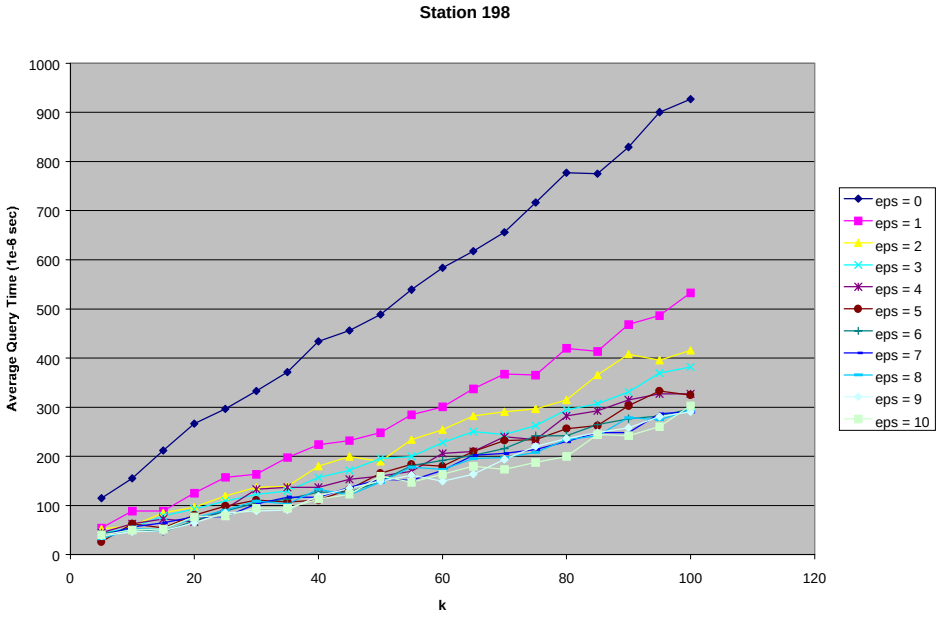


**Figure 31 - Effects of *k* on Query Time for Station 198**

Figure 19 indicated that the most accurate forecasts for Station 69 were generated using exact nearest neighbors ($\varepsilon = 0$) and 25-40 neighbors. However, using the Wilcoxon signed-rank test, there is no statistical difference in mean absolute percent errors for any of the forecasts using 25-40 exact nearest neighbors at the 0.01 significance level. Figure 28 revealed that for a given value of $\varepsilon$, smaller values of *k* take less time to compute a forecast. Therefore, 25 exact nearest neighbors yields the most accurate forecast in the least amount of time. However, the Wilcoxon signed-rank test also indicates that there is no statistical difference between mean absolute percent error between forecasts using $\varepsilon = 0$ and $\varepsilon = 1$ for 25 nearest neighbors at a 0.05 significance level. Therefore, it is possible to reduce execution time almost in half by using approximate nearest neighbors with $\varepsilon = 1$ without significantly sacrificing forecast accuracy.

#### 4.6.2.2 Maximum Daily Temperature Data

The temperature data set behaved similarly to the freeway flow data set. The most time consuming searches involve finding the largest number of exact nearest neighbors. Average query time decreases nonlinearly as $\varepsilon$ increases and appears to reach an

asymptote at $\varepsilon = 10$. However, the temperature data appears slightly noisier than the freeway flow data, which could be a result of the relatively small size of the test set.

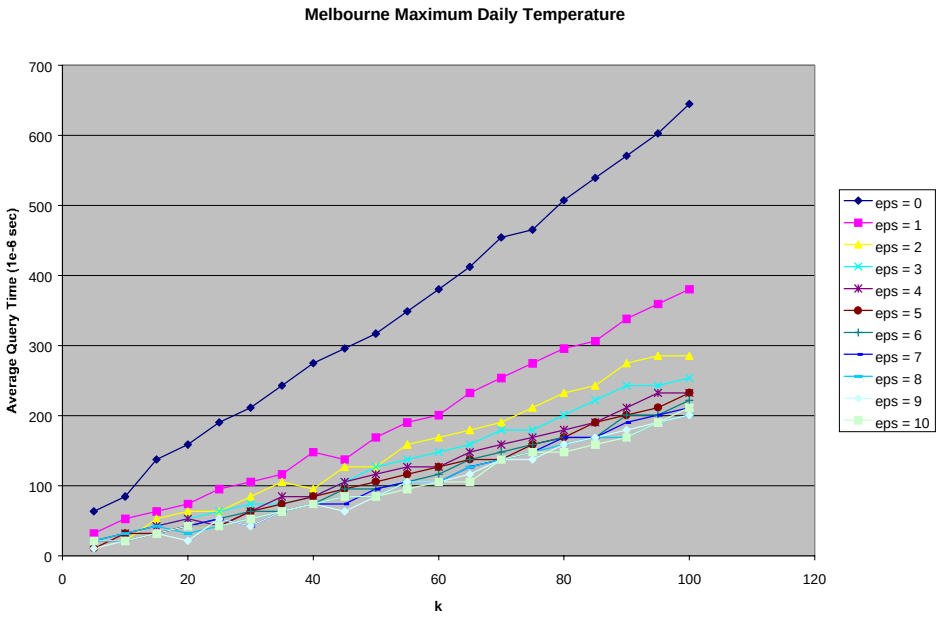


**Figure 32 - Effects of *k* on Query Time for Maximum Daily Temperature**

### *4.6.3 Tradeoff Between Forecast Accuracy and Execution Time*

Combining the plots from the previous sections, it is possible to directly examine the tradeoff between forecast accuracy and execution time. Each colored dot represents a specific set of values for $k$ and $\varepsilon$. The dark lines on Figure 33 through Figure 38 represent Pareto optimal frontiers. All points above and to the right of the frontier are dominated. In other words, another set of values for the tuning options exists that are more accurate or execute more quickly or both.

The Pareto frontier represents the optimal set of tuning option values for an ANN-NPR algorithm operating in a real-time system. In most of the plots, the left edge of the frontier declines quickly, indicating that large reductions in average forecast error may be achieved with a small increase in average execution time. Conversely, the bottom edge of the frontier tends to be relatively flat, indicating that large reductions in average query time are possible by sacrificing small amounts of forecast accuracy. The set of tuning

options that balances forecast accuracy with execution time lies along the Pareto optimal frontier between these two extremes.

#### 4.6.3.1 Freeway Flow Data

For the freeway flow data, there are relatively clear distinctions among the forecasts generated with different values of $\varepsilon$. Furthermore, points using large numbers of nearest neighbors tend to be furthest from the Pareto optimal frontier, indicating that they are dominated by solutions using fewer nearest neighbors.

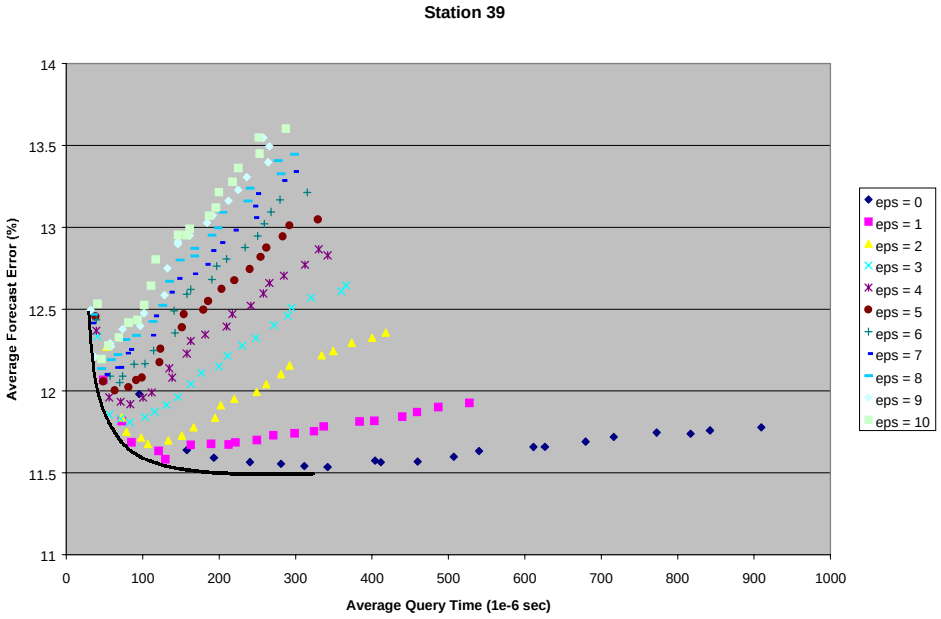


**Figure 33 - Relationship Between Query Time and Forecast Accuracy for Station 39**

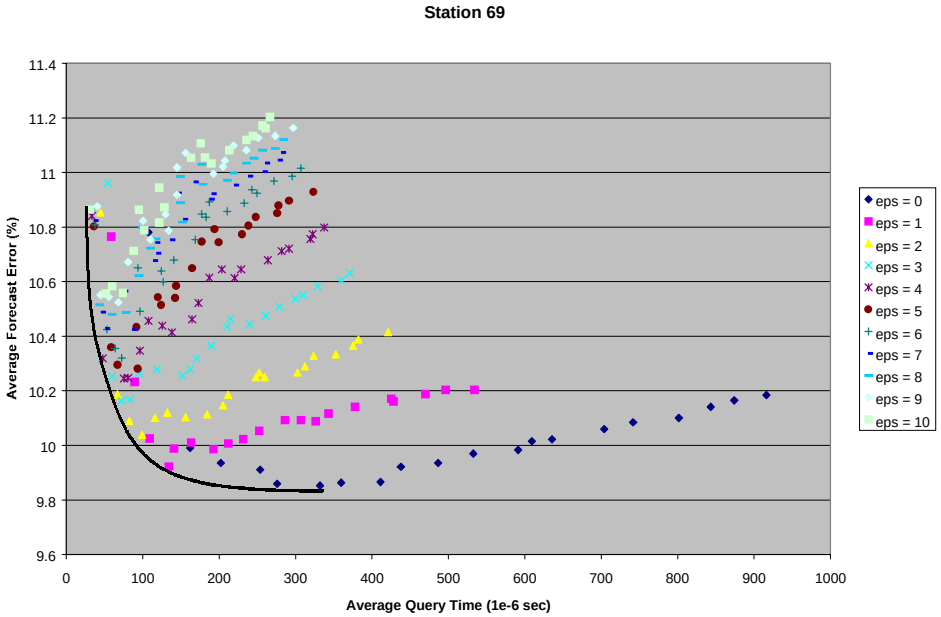


**Figure 34 - Relationship Between Query Time and Forecast Accuracy for Station 69**

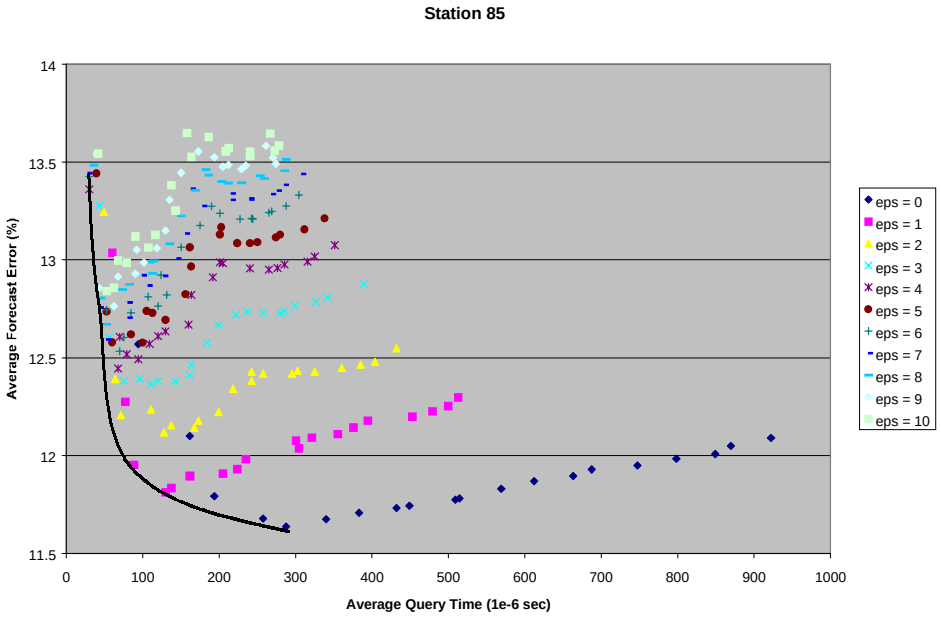


**Figure 35 - Relationship Between Query Time and Forecast Accuracy for Station 85**

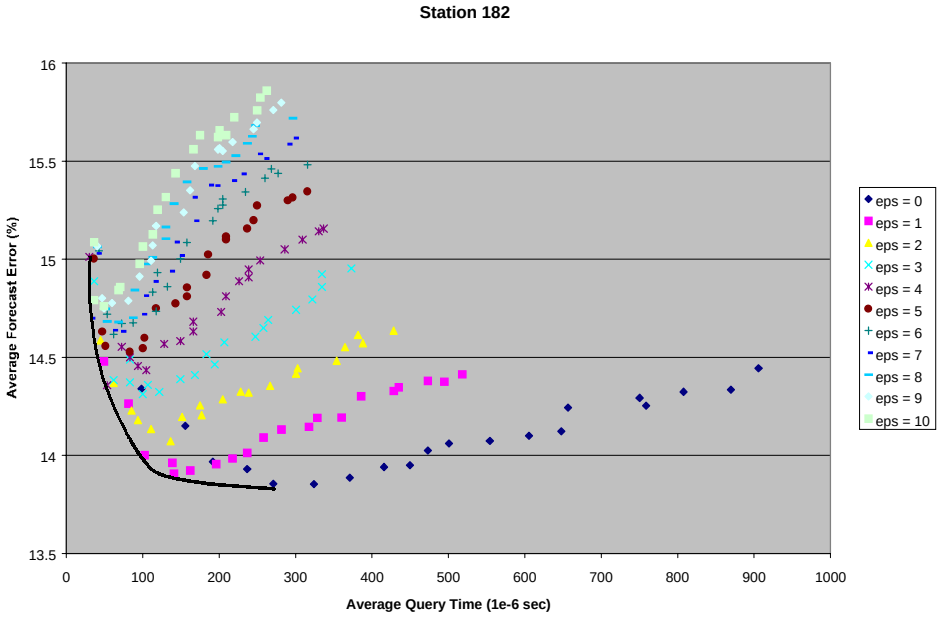


**Figure 36 - Relationship Between Query Time and Forecast Accuracy for Station 182**

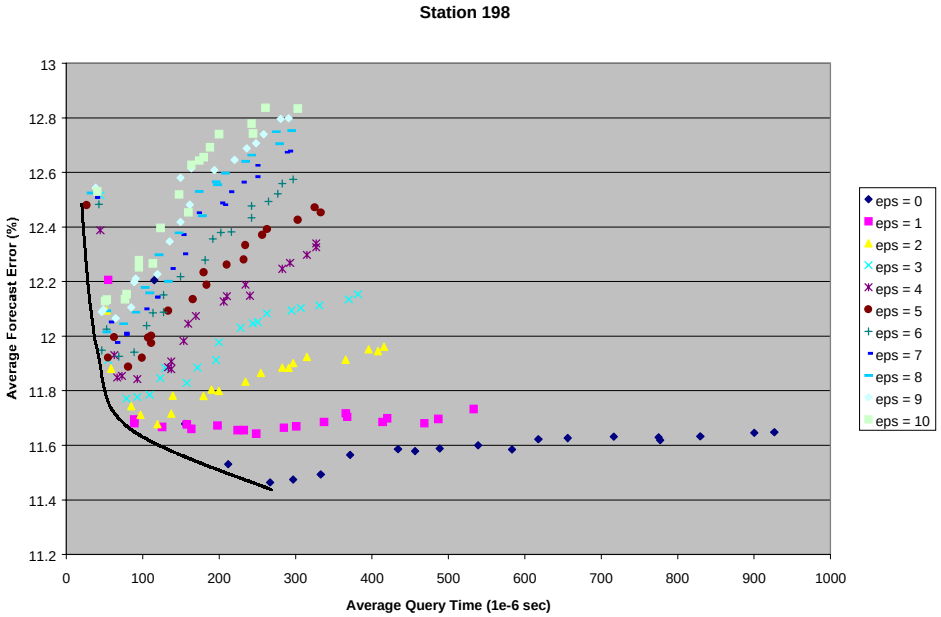


**Figure 37 - Relationship Between Query Time and Forecast Accuracy for Station 198**

#### 4.6.3.2 Maximum Daily Temperature Data

The plot for the maximum daily temperature data set resembles the graphs for the freeway flow data.

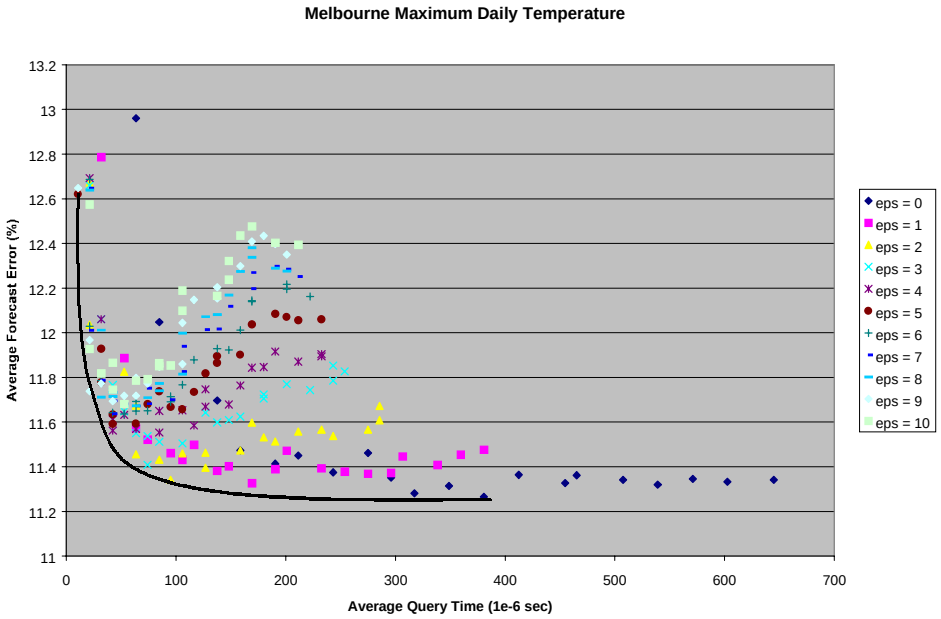


**Figure 38 - Relationship Between Query Time and Forecast Accuracy for Maximum Daily Temperature**

### *4.6.4 Summary*

Table 5 summarizes the values for $\varepsilon$ and $k$ that minimize the mean absolute percent error for each data set within the constraints of the real-time system. The average execution time per query is also reported in microseconds. Despite the large differences between the data sets, the optimal tuning options values are remarkably similarly with $\varepsilon \leq 0.50$ and $25 \leq k \leq 50$.

**Table 5 - Decision Variable Values that Maximize Forecast Accuracy within the Constraints of the Real-Time System (Enumeration)**

| Data Set | $\varepsilon$ | $k$ | MAPE | Average Execution Time ($1^{-6}$ sec) |
|---|---|---|---|---|
| 39 | 0.00 | 35 | 11.53 | 341.71 |
| 69 | 0.40 | 30 | 09.85 | 221.66 |
| 85 | 0.10 | 25 | 11.63 | 242.29 |
| 182 | 0.50 | 25 | 13.83 | 176.89 |
| 198 | 0.10 | 25 | 11.45 | 256.35 |
| Temperature | 0.30 | 50 | 11.23 | 243.13 |

## *4.7 Genetic Algorithm Results*

Section 4.6.3 presented two-dimensional graphs for analyzing the tradeoff between forecast accuracy and execution time. When only two objectives are being optimized simultaneously, enumerating possible combinations of decision variables and plotting the results may be used to graphically determine an acceptable set of tuning options. However, when more than two simultaneous objectives are considered, it becomes difficult or impossible to graphically determine an acceptable set of tuning option values. Furthermore, and more significantly, enumerating all possible combinations of every tuning option described in section 3.3 would require more than 10 iterations for each combination of state vector definition, distance metric, $k$, $\varepsilon$, and bucket size. Depending on the range and increments of the decision variables investigated, a prohibitively large number of iterations may be required to enumerate all possible combinations. As a result, a second approach to determining optimal tuning option settings involves the use of multi-objective optimization techniques such as genetic algorithms (GA). Multi-objective optimization methods intelligently search the decision variable space for values that achieve a set of objectives to minimize the time required to determine the optimal settings.

The following statement describes genetic algorithms and the more general class of evolutionary programs.

> "The evolution program is a probabilistic algorithm which maintains a population of individuals. Each individual represents a potential solution to the problem at hand. Each solution is evaluated to give some measure of

> its ‘fitness’. Then, a new population is formed by selecting the more fit individuals. Some members of the new population undergo transformations by means of ‘genetic’ operators to form new solutions. There are unary transformations [called mutations], which create new individuals by a small change in a single individual, and higher order transformations [called crossovers], which create new individuals by combining parts from several individuals. After some number of generations the program converges – it is hoped that the best individual represents a near-optimum solution” (Michalewicz, 1996).

Genetic algorithms consist of several distinct components. For any given optimization problem, there will be a function to be minimized or maximized, called an objective function, and one or more variables that the modeler can control, called decision variables. The characteristics of the system are defined by values of the decision variables and can be measured by the objective function. Let an individual $v_i$ denote a specific combination of decision variables, i.e. a specific solution. A population, then, is a collection of *pop_size* individuals.

The selection process chooses the individuals from the current population that ‘survive’ to the next generation based on fitness, which is a measure of how well the solution represented by an individual achieves the objective. The remaining individuals die and are removed from the simulation. The individuals that survive mate and adapt with given probabilities of occurrence. Mating involves the combination, or crossover, of characteristics from two (or more) surviving individuals to form a new solution to the problem. Adaptation is achieved through the random mutation of a surviving individual or offspring. Thus a new population is born from the survivors of the previous generation.

The evaluation, selection, mutation, and crossover processes repeat until the genetic algorithm converges to a near-optimal solution. More formally, the heuristic search over the space of decision variables ends after a predetermined number of generations or when the objective function is improved by a negligible amount after several consecutive generations.

A rudimentary genetic algorithm based on code provided by Michalewicz (1996) was implemented to test the fitness of a population. Genetic algorithms have their own settings that determine how the algorithm performs and, subsequently, require values. The most fit member of the population of ten individuals at the end of 100 generations

was chosen as the final (near-optimal) solution. The probabilities of mutation and crossover were initially set to the default values of 0.80 and 0.15, respectively, as shown in Table 6. The sensitivity of these settings is examined in section 4.7.1.

**Table 6 - Initial GA Settings**

| GA Setting | Value |
|---|---|
| Population size | 10 |
| Generations | 100 |
| Probability of mutation | 0.80 |
| Probability of crossover | 0.15 |

For this research effort, a real-time system was used where the average execution time of ANN-NPR must be less than 350 microseconds per forecast. The Hampton Roads Smart Traffic Center currently monitors 200 detector locations but will soon expand to more than 1200 locations. Similarly, the Australian Bureau of Meteorology collects temperature readings at more than 1,000 locations across Australia. The strict timing constraint of an average execution time of 350 microseconds ensures that forecast may be generated for every location in less than 0.50 seconds.

The genetic algorithm attempted to determine values for the tuning options that minimize absolute percent error while meeting the strict timing constraint. The fitness of each solution was used to determine the individuals that survive to the next generation. For this research, fitness was simply the mean absolute percent error associated with the tuning option settings for each test set as long as the average execution time was less than the timing constraint. For solutions where the average execution time exceeded this threshold, the fitness was the mean absolute percent error plus the average execution time, which severely penalizes the solution for not meeting the timing requirements of the real-time system. Equation (11) represents the fitness calculation where $AET_\mu$ is the average execution time measured in microseconds. Lower fitness values are desired since the genetic algorithm attempts to minimize forecast error subject to meeting the timing constraints of the real-time system.

$$fitness = \begin{cases} MAPE + AET_\mu & AET_\mu > 350.0x1^{-6}\,s \\ MAPE & otherwise \end{cases} \tag{11}$$

The values of $k$ and $\varepsilon$ that minimize the mean absolute percent error while adhering to the timing constraint for the trials involving genetic algorithms appear in Table 7. Similar to the results obtained using enumeration, $\varepsilon \leq 0.50$ and $25 \leq k \leq 50$ for all case studies. Comparing Table 7 to Table 5 reveals that there is less than 0.1% difference in the mean absolute percent error for the optimal settings of the tuning options as found using genetic algorithms and enumeration (see Table 10 below). Note that Station 182 and the maximum daily temperature data sets have the longest average execution time due to the very small values of $\varepsilon$ and relatively large number of nearest neighbors used.

**Table 7 - Decision Variable Values that Maximize Forecast Accuracy Accuracy within the Constraints of the Real-Time System (Genetic Algorithms)**

| Data Set | $\varepsilon$ | $k$ | MAPE | Avg. Execution Time ($1^{-6}$ sec) |
|---|---|---|---|---|
| 39 | 0.48 | 38 | 11.57 | 250.25 |
| 69 | 0.35 | 29 | 09.85 | 234.21 |
| 85 | 0.17 | 31 | 11.68 | 272.35 |
| 182 | 0.04 | 32 | 13.82 | 317.56 |
| 198 | 0.28 | 28 | 11.46 | 246.26 |
| Temperature | 0.02 | 52 | 13.89 | 348.84 |

### *4.7.1 Sensitivity Analysis*

The implementation of genetic algorithms used in this research has four primary options to determine how quickly a population converges to a near-optimal solution. The options also effect how easily the genetic algorithm converges to a local optimum instead of the global optimum. Table 6 above summarized the initial settings used for each option. Values for each GA setting were varied independently of the others to determine the sensitivity of the results and appear in Table 8. The settings associated with trial 1 were used for all data sets whereas trials 2-9 investigate the individual effects of the genetic algorithm settings.

**Table 8 - Sensitivity Analysis Settings**

| Trial | Population Size | Generations | Probability Mutation | Probability Crossover |
|---|---|---|---|---|
| 1 | 10 | 100 | 0.80 | 0.15 |
| 2 | 10 | 100 | 0.80 | 0.30 |
| 3 | 10 | 100 | 0.80 | 0.07 |
| 4 | 10 | 100 | 0.90 | 0.15 |
| 5 | 10 | 100 | 0.65 | 0.15 |
| 6 | 10 | 150 | 0.80 | 0.15 |
| 7 | 10 | 50 | 0.80 | 0.15 |
| 8 | 50 | 100 | 0.80 | 0.15 |
| 9 | 5 | 100 | 0.80 | 0.15 |

The sensitivity analysis was conducted for Station 85 and the resulting values of $k$ and $\varepsilon$ that minimize mean absolute percent error subject to the timing constraint appear in Table 9. Note that the mean absolute percent error differs less than 0.1% for all sensitivity trials and that $k$ and $\varepsilon$ do not differ substantially from the base case (trial 1). Furthermore, $\varepsilon \leq 0.60$ and $25 \leq k \leq 35$, which is similar to the results obtained for the other transportation data sets using both enumeration and genetic algorithms. Testing the alternative hypothesis that the mean absolute percent error for trials 2-9 are statistically different than the MAPE for trial 1 using the Wilcoxon Signed Ranks test, there was no statistical difference at the 0.05 significance level. Only trial 7 differed at the 0.10 significance level. Furthermore, testing the null hypothesis that the trials come from the same distribution using the Friedman test, there was no statistical difference at the 0.10 significance level.

**Table 9 - Sensitivity Analysis for Station 85**

| Trial | $\varepsilon$ | $k$ | MAPE | Avg. Execution Time ($1^{-6}$ sec) |
|---|---|---|---|---|
| 1 | 0.17 | 31 | 11.68 | 272.35 |
| 2 | 0.14 | 32 | 11.68 | 306.16 |
| 3 | 0.24 | 29 | 11.66 | 238.54 |
| 4 | 0.08 | 25 | 11.63 | 259.20 |
| 5 | 0.04 | 26 | 11.63 | 291.13 |
| 6 | 0.14 | 25 | 11.63 | 231.02 |
| 7 | 0.60 | 25 | 11.71 | 174.68 |
| 8 | 0.01 | 27 | 11.63 | 319.30 |
| 9 | 0.53 | 27 | 11.69 | 176.55 |

The results indicate that the mean absolute percent error determined by the values for the tuning options found by the genetic algorithm are relatively insensitive to the GA tuning options. Therefore, when using genetic algorithms to determine near-optimal values for the tuning options, the modeler need not be burdened with selecting appropriate values for the GA settings. The settings recommended by Michalewicz (1996) are acceptable.

## *4.8 Summary*

Two methods were used to determine optimal values of the decision variables for approximate nearest neighbor nonparametric regression. The first enumerated all possible combinations of the decision variables over a predetermined range while the second employed genetic algorithms. The mean absolute percent errors associated with the values of the decision variables found by the two methods were different by less than 0.1%, as seen in Table 10.

**Table 10 - Comparison of MAPE for ANN-NPR using Enumeration and Genetic Algorithms**

| Station | MAPE | | |
|---|---|---|---|
| | Enumeration | Genetic Algorithm | Absolute Difference |
| 39 | 11.04 | 11.08 | 0.03 |
| 69 | 9.36 | 9.37 | 0.01 |
| 85 | 11.14 | 11.19 | 0.05 |
| 182 | 13.34 | 13.34 | 0.00 |
| 198 | 10.96 | 10.98 | 0.02 |
| Temperature | 10.99 | 11.08 | 0.09 |

The genetic algorithm used in this research had four options for tuning the performance of the heuristic search. The sensitivity of the values for the population size, maximum number of generations, probability of mutation, and probability of crossover was determined through eight different trials for Station 85. At the 0.05 significance level, the Wilcoxon signed-rank test revealed that the mean absolute percent errors found for each trial were not statistically different. Therefore, the genetic algorithm settings do not appear to effect the solution found by a significant amount.

The results presented in sections 4.6 and 4.7 demonstrate that nonparametric regression enhanced by advanced data structures and imprecise computations may be successfully used in real-time systems.

# 5 Data-Driven Methodology for Applying ANN-NPR

In Chapter 2, the definition of nonparametric regression was not discussed. Strictly speaking, the word nonparametric means “not involving the estimation of parameters of a statistical function” (Neufeldt, 1990). A parameter, on the other hand, is “a property whose value determines the characteristics or behavior of something” (Neufeldt, 1990). Clearly $\varepsilon$, $k$, and the other tuning options described in section 3.3 meet the definition of a parameter because they determine the neighbors found in the search procedure, which ultimately effect the estimate of the future state of the system.

Parameters are often associated with forecasting techniques where a functional relationship between the input and output states is derived from the data. Nonparametric regression, on the other hand, uses a functional relationship with a *subset* of the overall data to calculate a forecast. That relationship is defined by $k$, $\varepsilon$, and the other tuning options, especially the forecast function.

Perhaps the only difference between parameters such as those used in multiple regression and the tuning options found in nonparametric regression is the methods by which their values are estimated. Multiple regression uses formal statistical techniques such as the method of least squares whereas nonparametric regression uses heuristic methods. However, there are no formal techniques for selecting values for the tuning options. This chapter, therefore, presents a systematic methodology for applying nearest neighbor nonparametric regression in real-time systems. The methodology is adapted from more general modeling techniques, such as those proposed by Box and Jenkins (1970), but modified based on the author’s experiences with nonparametric regression. The methodology is demonstrated in the case studies presented in Chapter 4 and specifically addresses the complexity introduced by the modifications required to successfully use nonparametric regression in real-time systems.

Figure 39 summarizes the iterative approach to NN-NPR model building employed in this research effort. The individual steps are discussed in the following sections.

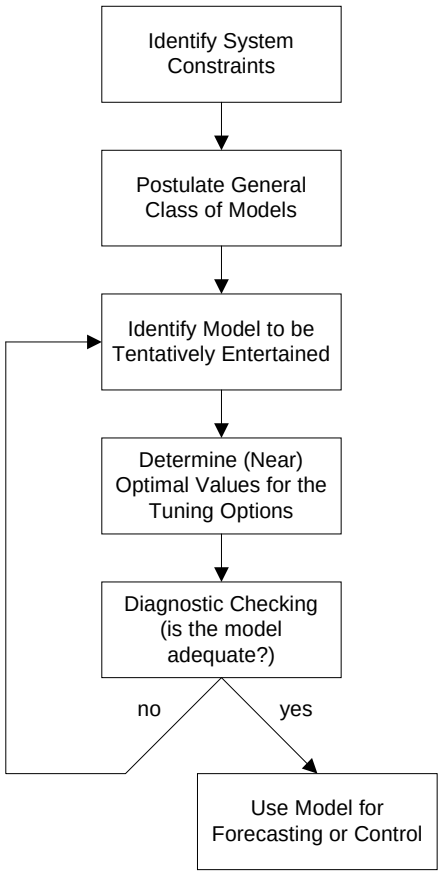


**Figure 39 - Nearest Neighbor Nonparametric Regression Methodology**

## *5.1 Assumptions*

The methodology presented in this chapter assumes that a sufficient amount of historical data is available for use in the historical database and that a portion of the data has been placed in a test set. Furthermore, the methodology assumes that the goals and purpose of the forecasting system are understood, as described below.

### *5.1.1 Data*

A database of adequate depth and breadth must exist before applying nearest neighbor nonparametric regression. As discussed in section 2.1.1, nonparametric regression benefits from a larger database that more accurately and thoroughly represents a broader range of possible system conditions. The quality of forecasts is dependent on the quality

of nearest neighbors selected by the search procedure and the historical database should cover a wide range of possible scenarios including normal and abnormal operating conditions. Box and Jenkins (1970) recommend using a training set containing “at least 50 and preferably 100 [historical] observations” (Box and Jenkins, 1970) to fit time series models. However, these numbers suppose that any seasonal and cyclical trends are accurately represented. For example, a process that repeats daily and is measured every ten minutes requires at least 144 observations to fit a time series model, which is clearly more than the amount suggested by Box and Jenkins. For nearest neighbor nonparametric regression, the historical database should contain enough observations to adequately represent the underlying process and, intuitively, should be at least several times larger than *k*.

Section 4.6.2 demonstrated that advanced data structures speed nearest neighbor searches by as much as 1,000 times. Given this drastic reduction in average query time, ANN-NPR may take advantage of extremely large data sets. Furthermore, approximate nearest neighbor NPR may use databases that are continuously updated with new data.

The data should be divided into a training set for use in the historical database and a separate test set. If the observations are primarily time series in nature, the most recent cases should be used in the test set to accurately simulate the system as it would operate in a production environment. As a rule of thumb, the test set should contain enough samples to be able to draw statistical conclusions about the usefulness of various models. When the overall data set is small, roughly 10%-25% of all the data should be in the test set. For larger data sets, less than 10% of the data may reside in the test set. For example, the five transportation data sets used in this research were quite large and, as a result, the test sets contained slightly less than 10% of the overall data but still had more than 4,600 observations each. On the other hand, the temperature data was relatively small and the test set contained approximately 25% of the available data, which was less than 1,000 observations.

### *5.1.2 Goals*

The goals of the forecasting system should be determined by identifying acceptable and desired performance requirements. In the case studies presented in Chapter 4, the accuracy of the forecasting algorithm in terms of mean absolute percent error was of primary importance. Box and Jenkins (1970), on the other hand, sought to minimize the root mean square error of the forecasts. When determining optimal values for the tuning options, this research effort considered all forecast errors equally whereas Box and Jenkins were concerned with minimizing worst-case performance. Clearly, the objective of the forecasting application will vary from one system to the next and must be fully understood. These measures will be used later when optimizing the tuning options as discussed in section 5.5.

## *5.2 Identify System Constraints*

Although nearest neighbor nonparametric regression may be applied to non-real-time systems, even those systems have physical constraints such as memory size, storage space, and network bandwidth. Therefore, the parts of the methodology presented in this section that are specific to real-time systems actually apply to any system using nonparametric regression. The terms ‘system’ and ‘real-time system’ will be used interchangeably throughout this section.

The discussion of real-time systems in Chapter 3 indicates that a task may be limited by timing constraints, priority, resource requirements, precedence relationships, communication requirements, or other relevant criteria. The limitations imposed by the real-time system must be fully understood and identified. For example, the real-time system employed in the case studies presented in Chapter 4 imposed a rigid timing constraint of an average of 350 microseconds per forecast to ensure that forecasts may be calculated for all locations in the system in less than 0.50 seconds. Any model that did not meet this constraint was rejected, regardless of the forecast accuracy. However, the constraints of some real-time systems are not always so rigid. These systems exhibit some flexibility in terms of the allowable frequency and severity of violations and must

be treated accordingly when optimizing values for the tuning options as discussed in section 5.5.

The implementation of approximate nearest neighbors used in this research imposes a constraint on the size of the historical database because it "is designed for data sets that are small enough that the search structure can be stored in main memory" (Mount, 1998). Parametric models do not have the same storage concerns because the fitting process is done offline and all of the training data is compressed into one function, thus eliminating the need to retain the historical observations online. Virtual memory and swap space allow many operating systems to store data in main memory that is larger than the amount of physical RAM in a computer but care must be taken when considering extremely large training set sizes. The largest data set used in this research contained 65,000 observations in 4 dimensions and easily fit into the memory of the test machine. Mount et. al. (1998) tested historical data sets as large as 100,000 observations in 16 dimensions on a Sun Sparc 20 running Solaris. For significantly larger data sets residing on secondary storage, alternative, more efficient methods for retrieving records exist and should be used when appropriate.

## *5.3 Postulate General Class of Models*

As discussed in section 2.2, there are two fundamental classes of nonparametric regression. Nearest neighbor NPR has the advantage of always generating an estimate whereas kernel nonparametric regression will not generate a forecast if no historical observations are close enough to the current conditions. On the other hand, kernel nonparametric regression ensures that all neighbors used to calculate an estimate are within a predetermined distance from the current conditions.

Advanced data structures can be used with both kernel and nearest neighbor nonparametric regression to significantly reduce execution time. Approximate nearest neighbors, unfortunately, are not easily applied to kernel NPR because any number of neighbors may be used to calculate an estimate.

While this research only considers nearest neighbors nonparametric regression, kernel NPR is a viable modeling approach in many situations. Kernel nonparametric

regression should be used in scenarios where the lack of a forecast provides valuable information about the state of the system. For example, in transportation systems, kernel NPR will not calculate an estimate when the current traffic conditions have not been observed in the past, thus indicating a potential incident requiring special attention from traffic management engineers.

Hybrid models that incorporate characteristics of kernel and nearest neighbor NPR may also be used. One such hybrid model may select nearest neighbors using a kernel search procedure if more than *k* neighbors were within the predetermine distance, otherwise it would use a nearest neighbor search, thus ensuring that at least *k* nearest neighbors are used in the estimate.

## *5.4 Identify Model to be Tentatively Entertained*

Once an appropriate class of models (i.e. kernel or nearest neighbor) has been selected as described in section 5.3, Box and Jenkins (1970) recommend "employing data and knowledge of the system to suggest an appropriate parsimonious subclass of models which may be tentatively entertained" (Box and Jenkins, 1970). In other words, any practical experience with the system being modeled may be used to intelligently restrict the range of values for the tuning options considered.

A model to be tentatively entertained consists of specific values or ranges of values and increments to consider for every tuning option. For example, the tentative models examined in this research effort for the freeway flow data can be described according to Table 11. Based on traffic flow theory and experience forecasting traffic flows, the state vector, distance metric, forecast function, bucket size, splitting rule, and search method were all set to specific values. The number of nearest neighbors was restricted to 100 or fewer and examined in increments of five. The relative error in distance for approximate nearest neighbors was set to a maximum of ten and considered in increments of 0.10. The final model was specified when values for *k* and $\varepsilon$ were determined through optimization.

**Table 11 - Tentative Model for Freeway Flow Data**

| Tuning Option | Value, Setting, or Range | Increment |
|---|---|---|
| State vector | $[V(t), V(t-1), V_{hist}(t), V_{hist}(t+1)]$ | Fixed |
| Distance metric | Euclidean ($L_2$) | Fixed |
| Forecast function | Straight average | Fixed |
| Bucket size | 1 | Fixed |
| Splitting rule | Standard | Fixed |
| Search method | Standard | Fixed |
| $k$ | [5,100] | 5 |
| $\varepsilon$ | [0,10] | 0.10 |

The following sections provide general guidelines for pre-specifying values for the tuning options, when appropriate. Sections 5.4.4 – 5.4.6 are specific for KD trees.

### *5.4.1 State Space*

Identifying an appropriate state space for nonparametric regression is analogous to the problem of selecting which variables (and any nonlinear combinations of variables) to include in multiple regression. Inappropriate state space definition in any modeling technique will lead to poor fit and inaccurate forecasts. According to Box and Jenkins (1970), "from the interaction of theory and practice, a useful class of models for the purposes at hand is considered." In other words, Box and Jenkins (1970) recommend using both theoretical understanding and practical experience with the process being modeled to define an appropriate state space.

For the transportation data used in the case studies presented in Chapter 4, the state vector chosen was developed by Smith (1995). Smith's theoretical knowledge of highway traffic flows combined with his practical experience forecasting future volume levels led to the development of the state vector in Equation (7). Specifically, Smith realized that the inclusion of historical average flow rates would further clarify the state definition and allow the search procedure to select neighbors more representative of the current conditions.

In regression, several heuristic techniques exist for selecting variables for use in models. For example, stepwise regression, backward elimination, and polynomial networks systematically try different combinations of predictor variables to develop a

regression model that fits the data. These methods test the statistical significance of the resulting parameters to determine the predictor variables that remain in the model and those that are rejected. Because comparable tests do not apply to nonparametric regression, other methods of choosing an appropriate state space must be explored.

However, several techniques that may be used when developing regression and other similar models also apply to nonparametric regression. For example, "one of the most important steps in choosing a [state space] may be to exclude features which have little or no relevance. The features selected by classification trees can be a very useful guide" (Ripley, 1999). Classification trees use rules to partition the test set to accurately classify the response variable. A usable byproduct of classification trees is that only the variables most relevant to classifying the response appear in the tree structure, thus providing insight into the variables that should be used to define the state of the system.

### *5.4.2 Distance Metric*

As mentioned earlier, Euclidean distance ($L_2$) is the most common distance metric used with nonparametric regression. However, other dissimilarity metrics, including weighted distances, may be used. For example, if the similarity between records is defined by the one element of the state vector furthest from the current conditions, then the max distance metric ($L_\infty$) should be used. Records found by the search procedure using the $L_\infty$ metric minimize the worst single-element deviation from the current conditions.

### *5.4.3 Forecast Function*

There are an infinite number of forecast functions that may be used with nonparametric regression. Smith et. al. (2000) attempted to "improve upon the straight average forecast by taking into account the relative distance of the neighbors from the forecast point and/or the relation of key neighbor state elements to the corresponding forecast point elements. These methods were founded on the notion that the forecast can be further informed by considering the overall nearness of the neighbors to the forecast state, the

correlation of each neighbor to the current conditions, and the correlation of the neighbors to the historic conditions at the time interval of the forecast."

An alternative to the use of weighted forecast functions is to fit a linear or nonlinear parametric model to the cases in the neighborhood, and then use that model to forecast the value of the dependent variable as done by Mulhern and Caprara (1994). Such techniques, often called spline smoothing, use the search component of nonparametric regression to select a subset of the available historical data close to the current conditions for use as the training data for a parametric model.

### *5.4.4 Bucket Size*

Friedman et. al. (1977) observed that "to minimize the (upper bound on the) number of records examined, the terminal buckets should contain one record." However, the best case or average number of records examined may not be minimized using a bucket size of one. Arya et. al. (1998) point out that "adjusting bucket size affects the search time." Unfortunately, the literature does not describe the relationship between bucket size and execution time.

### *5.4.5 Splitting Rule*

Several splitting rules proposed by Mount (1998) are described in section 3.3.2. The splitting rules attempt to minimize the time required to find (approximate) nearest neighbors by structuring the data in the multidimensional search tree in such a way as to reduce the number of records visited during the search. The standard and sliding midpoint splitting rules ensure that the tree height is $\log_2 N$ but do not bound the aspect ratio, which may adversely effect search times. The other splitting rules bound the aspect ratio or partition the data such that every skinny cell has a sibling fat cell, which is not problematic for (approximate) nearest neighbor searching, but do not limit the height of the tree.

When selecting a splitting rule, the resources required to preprocess the data may be important. For example, this research effort used the standard splitting rule because

the preprocessing “running time is independent of the data distribution” (Arya et. al., 1998). For highly clustered data, however, other splitting rules may find (approximate) nearest neighbors more quickly than the standard splitting rule but have varying preprocessing requirements.

### *5.4.6 Search Method*

Two search methods proposed by Mount (1998) are described in section 3.3.3. If the expected value of $\varepsilon$ is small, Mount (1998) recommends using the standard search method. For large values of $\varepsilon$, priority search should find neighbors more quickly.

## *5.5 Optimization*

Values for the remaining tuning options that were not assigned in section 5.4 must be determined through optimization. The results presented in Chapter 4 concentrate on two methods for determining the optimal values: enumeration and genetic algorithms. While other optimization techniques may be employed, the two investigated in this research effort demonstrate drastically different levels of involvement by the modeler and are presented as guidelines.

### *5.5.1 Enumeration*

Enumeration involves a brute force approach for determining optimal values for the tuning options. Given the tuning options to consider and acceptable ranges of values, enumeration generates forecasts using every possible combination of settings. If many tuning options are being optimized or the options vary over large ranges, a large number of iterations will be needed to exhaustively search for optimal values. For example, this research required more than 2,000 iterations to examine every possible combination of $5 \le k \le 100$ in increments of 5 and $0 \le \varepsilon \le 10$ in increments of 0.10.

Once forecasts are calculated for every possible combination of the tuning options, an optimal set of values is still not immediately apparent. The modeler must

analyze the forecasts to determine the optimal settings. Recalling the goals of the system determined in section 5.1.2, plots may be created to graphically show the relationship between the objectives and the tuning options similar to those presented in sections 4.6.1, which plotted forecast accuracy versus *k*. The constraints of the real-time system may also be graphed against the tuning option values as done in section 4.6.2. In both cases, the plots show the relationship to the tuning options to aid in selecting optimal values.

Finally, the objectives may be graphed against the constraints of the real-time system, similar to the plots in section 4.6.3, to quickly determine optimal values of the tuning options that meet the constraints imposed by the real-time system. The real-time system used in this research imposed a strict timing constraint of an average execution time of 350 microseconds. Therefore, a vertical line may be drawn on the plots at 350 microseconds similar to Figure 40 where all points to the left of the line represent feasible settings. The modeler then may select optimal values for the tuning option from the points that lie along the Pareto optimal frontier to the left of the vertical line.

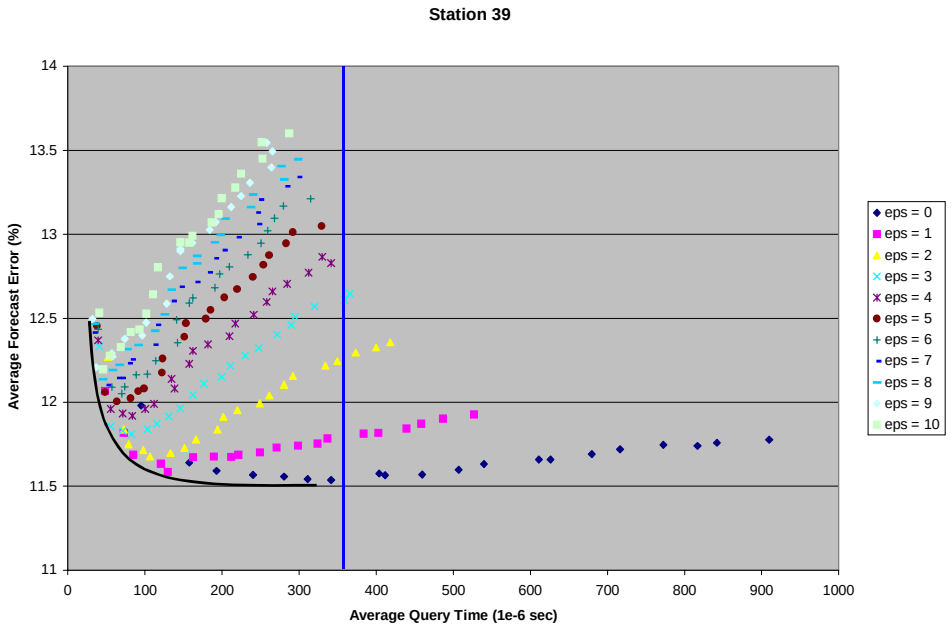


**Figure 40 - Relationship Between Average Query Time and Forecast Accuracy for Station 39 with Constraints of the Real-Time System**

If the modeler can determine an acceptable tradeoff between forecast accuracy and average query time, an optimal set of tuning options may be quickly determined. Parallel lines may be drawn on the plots starting at the origin and expanding outward with slope equal to the ratio of mean absolute percent error divided by the average query time until one of the lines is tangent to the Pareto optimal frontier. The point where the line is tangent to the frontier represents the optimal set of tuning options for the given tradeoff between forecast accuracy and average query time. For example, suppose it has been determined that a 3% decrease in mean absolute percent error is worth an increase in average query time of 50 microseconds. The green line in Figure 41 represents a line with slope = -3/50 = -0.06. The point where this line is tangent to the Pareto optimal frontier corresponds to $\varepsilon = 1$ and $k = 15$ with an average query time of 85.11 microseconds and 11.69% error. Modelers concerned with minimizing average query time should use near-vertical lines while systems attempting to maximize forecast accuracy should use near-horizontal lines.

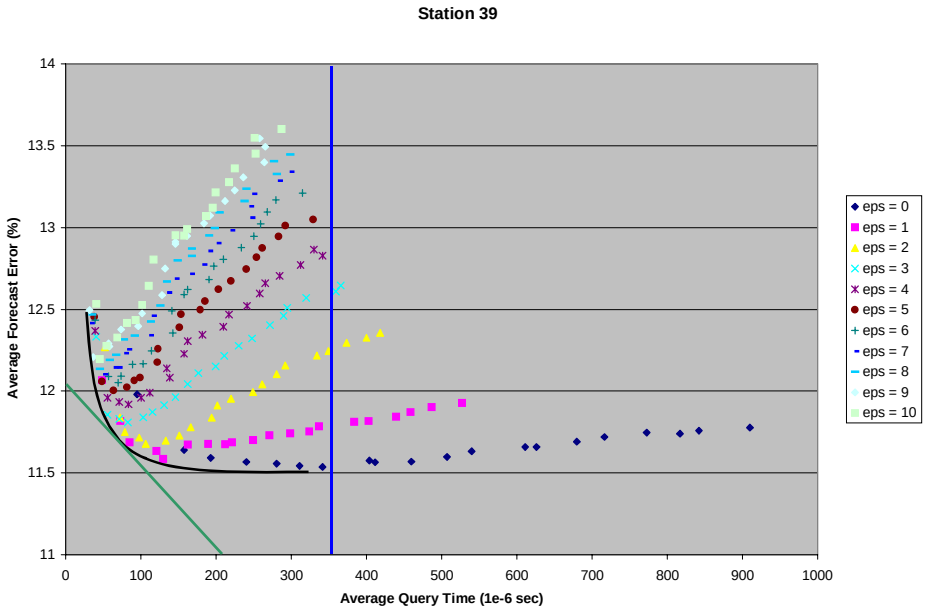


**Figure 41 - Relationship Between Average Query Time and Forecast Accuracy for Station 39 with Constraints of the Real-Time System and Tradeoff Relationship**

#### 5.5.1.1 Advantages and Disadvantages

There are several advantages and disadvantages of enumerating every possible combination of the tuning options to determine optimal values. In most cases, enumeration requires more iterations than an intelligent search process such as genetic algorithms. Enumeration also requires a fair amount of effort from the modeler in creating and analyzing the resulting forecasts to determine optimal values for the tuning options.

On the other hand, because enumeration exhaustively examines all combinations of tuning options, no additional iterations are required if the constraints of the real-time system change. The modeler may use the graphs already created to determine a new set of optimal settings, assuming the rest of the system remains unchanged. For example, suppose the constraints of the real-time system were modified such that ANN-NPR forecasts must be calculated within 150 microseconds per forecast. The vertical line that represents the constraints of the real-time system may be adjusted accordingly as seen in Figure 42 and new optimal values may be selected from the points along the Pareto optimal frontier to the left of the vertical line.

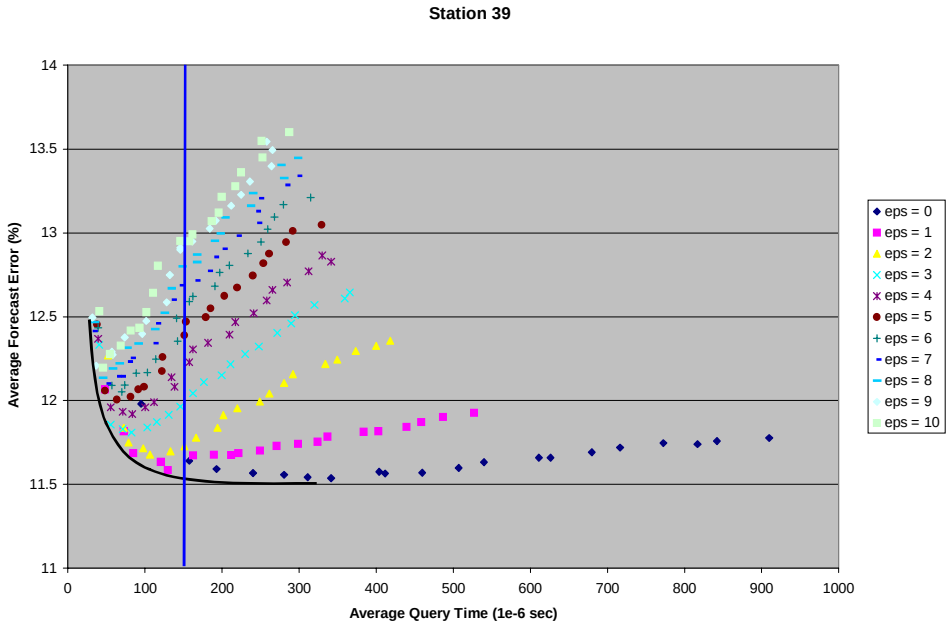


**Figure 42 - Relationship Between Query Time and Forecast Accuracy for Station 39 with Constraints of the Modified Real-Time System**

Perhaps the most useful benefit of enumeration is the ability to test the statistical significance of surrounding tuning option settings to potentially reduce execution time without statistically decreasing forecast accuracy similar to the analysis conducted in section 4.6.2.1. In situations where the use of advanced data structures and imprecise computations do not reduce execution time enough to meet the constraints of the real-time system, statistical tests may be used to find an acceptable tradeoff between execution time and forecast accuracy to meet the constraints of the real-time system.

### *5.5.2 Genetic Algorithms*

One significant drawback of enumeration is that a very large number of iterations may be needed to explore every possible combination of tuning options. The case studies presented in Chapter 4 only considered values for $k$ and $\varepsilon$ over reasonable ranges but still required more than 2,000 iterations. Genetic algorithms, on the other hand, used only 1,000 iterations to find nearly identical values for the tuning options.

Prior to using genetic algorithms, the modeler must first define the objective function and, second, specify values for the GA options.

#### 5.5.2.1 Create Objective Function

The first step in using a genetic algorithm defines the function to be minimized or maximized. The function should account for all objectives defined in section 5.1.2 while also considering the constraints of the real-time system defined in section 5.2. For example, recall the objective function used in this research effort in Equation (11). The fitness of an individual was simply the mean absolute percent error associated with the tuning option settings as long as the average execution time was less than the timing constraint. For solutions where the average execution time exceeded this threshold, the fitness was the mean absolute percent error plus the average execution time, which severely penalized the solution for not meeting the constraints of the real-time system. In terms of optimization, the problem of optimizing tuning options addressed by this research may be written as

(12)
$$\begin{aligned} \text{minimize:} \quad & MAPE \\ \text{subject to:} \quad & AET_{\mu} \leq 350 \\ & k \geq 0 \\ & k \leq 100 \\ & \varepsilon \geq 0 \\ & \varepsilon \leq 10 \end{aligned}$$

where $MAPE$ and $AET_{\mu}$ are determined by $k$, $\varepsilon$, and the tuning options predetermined in section 5.4.

There is a large amount of literature in the optimization community about defining appropriate objective functions. Optimization of a single objective is rather straightforward and may be accomplished via enumeration or genetic algorithms, as done in this research effort. Other techniques such as integer programming (if appropriate) and goal programming may also be used.

In the case of multiple conflicting objectives, the definition of an objective function is critical. There are many philosophies that have been proposed for handling multi-objective models, most of which convert the model to a single-objective format. Several such techniques are presented below. The penalty applied to the objective function in Equation (11) for exceeding the execution time constraint is not unique to genetic algorithms and may be similarly used with the methods presented below.

"One way to force a problem into a single objective format is to select one of the objectives, use it as *the* single objective, and then either ignore the other objectives or treat them as constraints" (Ignizio and Cavalier, 1994). Objectives may be transformed to constraints by assigning aspiration levels, which are values assigned by the modeler that the model must now meet in addition to optimizing the objective. Minimization objectives, for example, may be transformed to inequality constraints where the objective is less than or equal to the aspiration level. If this research desired to minimize execution time in addition to forecast error, then $AET_{\mu} \leq 350$ would be the transformed objective function where 350 microseconds is the aspiration level. Care must be taken when converting objectives to constraints because mathematically infeasible solutions may result (i.e. it may not be possible to calculate a forecast in less than 350 microseconds).

Also beware of subjectivity involved in selecting the single objective function to use and the assignment of aspiration levels.

Theoretically, it is possible to combine any number of objectives into a single, equivalent objective (Ignizio and Cavalier, 1994). This approach assumes that each of the individual objective functions may be expressed in terms of a proxy, which is a common measure of effectiveness. Dollars are a common measure of effectiveness to use but if no proxy exists, utility theory may be used to elicit from the decision-maker the relationship between the multiple objective functions through a series of carefully considered questions. While the discussion of utility theory is beyond the scope of this research effort, assuming that the decision maker's utility function has been determined, and that the utilities are additive, the objectives may be measured in utiles and then combined (Ignizio and Cavalier, 1994), thus reducing the problem to one of optimizing a single objective.

Chebyshev optimization attempts to minimize the deviations of the objectives from the best possible values using several iterations of single-objective optimization. The optimization model is solved using only one objective function at a time while ignoring the other objectives. Once the tuning option values have been optimized, the best possible value for the current objective function has been determined. The tuning options may also be used to determine the values of the other objective functions when this particular objective is optimized. Next, transform each of the objective functions into constraints such that the function minus an error term $\delta$ is less than the best value found using optimization (for minimization objectives). Finally, single-objective optimization may be used to minimize $\delta$ subject to the original and transformed constraints.

For example, supposed this research desired to simultaneously maximize forecast accuracy and minimize execution time. Optimizing the tuning options for only forecast accuracy for Station 85 resulted in a mean absolute percent error of 11.63%. Optimizing for average execution time produced forecasts in 20.66 microseconds. After the transforming the objectives to constraints, the Chebyshev formulation of the optimization problem can be written as follows.

(13)

$$\begin{aligned} &\text{minimize:} && \delta \\ &\text{subject to:} && MAPE - \delta \le 11.63 \\ & && AET_{\mu} - \delta \le 20.66 \\ & && k \ge 0 \\ & && k \le 100 \\ & && \varepsilon \ge 0 \\ & && \varepsilon \le 10 \end{aligned}$$

Certainly there are many possible methods of formulating multi-objective optimization problems. Other methods use weights, lexicographic minimums of ordered vectors, or fuzzy programming. The most straightforward approach for creating a single objective from a multi-objective decision space is to choose one objective and convert the rest into constraints with appropriate aspiration levels. Proxies may be used if a common measure of effectiveness if readily apparent, such as dollars. Ultimately, though, the technique to use is left to the modeler to decide.

#### 5.5.2.2 Select Values for GA Settings

Genetic algorithms can be extremely complex. Even the rudimentary GA code used in this research effort had four different parameters that defined how the algorithm performed, all of which required values from the modeler. As mentioned in section 4.7, each individual of a population represents a set of values for the tuning options. The individuals are evaluated based on how well the values achieve the objective function, which for this research involved minimizing the mean absolute percent error of the forecasts. The most fit members of a population survive to the next generation and mutate and adapt to form a new population.

Population diversity and selective pressure are the two driving forces of genetic algorithms. Population diversity is an indication of how different members of the population are from each other. Members of a very diverse population are not very similar and, as a result, explore a large area of the state space. On the other hand, the members of a homogeneous population are confined to a small area of the state space. Selective pressure determines the likelihood that the most fit members of a population survive to the next generation. If only the most fit members survive, the genetic algorithm

may converge on a local optimum rather than the true global optimum because there are no members to explore other alternatives.

The mutation and crossover probabilities directly determine the population diversity and selective pressure of a genetic algorithm search. Mutation, which is a slight alteration of a surviving individual, is necessary to fine-tune a set of values that truly optimize an objective function because only small changes to an existing individual are made. A large mutation probability indicates that many of the surviving individuals will be slightly altered, thus causing the population in the next generation to resemble the previous generation.

Crossovers, which combine two (or more) surviving individuals from the previous generation, simulate mating. The individuals resulting from a crossover tend to be much more different than their parents compared to an individual resulting from a mutation. A large crossover probability will cause the population in the next generation to be very different than the previous generation. Population diversity and selective pressure, as controlled by the probabilities of crossover and mutation, are inversely proportional to each other.

> "These factors are strongly related: an increase in the selective pressure decreases the diversity of the population, and vice versa. In other words, strong selective pressure 'supports' the premature convergence of the GA search; a weak selective pressure can make the search ineffective. Thus it is important to strike a balance between these two factors" (Michalewicz, 1996).

The GA options recommend by Michalewicz (1996) appeared in Table 6. The mutation and crossover probabilities are 0.80 and 0.15, respectively. These values indicate that a large number of mutations are needed to accurately fine-tune the individuals to the true optimal values but that crossovers are still necessary to ensure that the GA converges on the global optimum instead of a local optimum.

Fortunately, the sensitivity analysis conducted in section 4.7.1 indicated that the mean absolute percent errors determined by the values for the ANN-NPR tuning options found by the genetic algorithm are relatively insensitive to the GA options. Therefore, the default GA options recommended by Michalewicz (1996) should find near-optimal

values for the ANN-NPR tuning options comparable to those found using enumeration while requiring fewer iterations.

### *5.5.3 Recommendation*

Based on the author's experiences modeling the case studies, it is recommended to (initially) use enumeration to determine the optimal values for the tuning options. Although enumeration usually requires more iterations than genetic algorithms, the ability to conduct statistical tests of significance may be extremely powerful. Furthermore, no additional iterations are needed if the system requirements change and plots similar to those developed in section 4.6 are extremely useful for understanding the system.

If the model found using enumeration is inadequate, genetic algorithms may be used. GA's are useful for quickly and efficiently searching a large state space that is infeasible to examine through enumeration. Genetic algorithms may also be used to fine tune the optimal settings found through enumeration to a much higher precision.

## *5.6 Diagnostic Checking*

Given that the model has been identified and values for the tuning options determined through optimization, diagnostic checks are applied to the model to determine its adequacy. Most modeling techniques, including time series analysis and parametric regression, rely on an analysis of the residuals to determine how well a model fits the data. If the model is found to be inadequate, diagnostic checks should indicate how the model may be improved during successive iterations.

Box and Jenkins provide the following insight about diagnostic checks.

> "No model form ever represents the truth absolutely. It follows that, given sufficient data, statistical tests can discredit models which could nevertheless be entirely adequate for the purpose at hand. Alternatively, tests can fail to indicate serious departures from assumptions because these tests are insensitive to the types of discrepancies that occur. The best policy is to devise the most sensitive statistical procedures possible but be

prepared, for sufficient reason, to employ models which exhibit slight lack of fit. Know the facts as clearly as they can be known – then use judgment" (Box and Jenkins, 1970).

### *5.6.1 Plot of the Residuals*

Plotting the residuals versus the actual values observed may provide insight about any model inadequacies. For example, the residual or percent error may be graphed on the vertical axis against the independent variable or estimate on the horizontal axis. If the model appropriately represents the underlying process, the "residual plots [should] have no trends, no dramatic increased or decreases in variability, and only a few residual (about 5%) more than two estimated standard deviations above or below 0" (Mendenhall and Sincich, 1996).

Figure 43 is a plot of forecast error versus the actual values observed for Station 39. The average of the residual is –17 vehicles per hour, which is very close to zero considering the flow measurements range from 0 to 7,500 VPH. Less than 5% of the residuals are more than two estimated standard deviations above or below the mean.

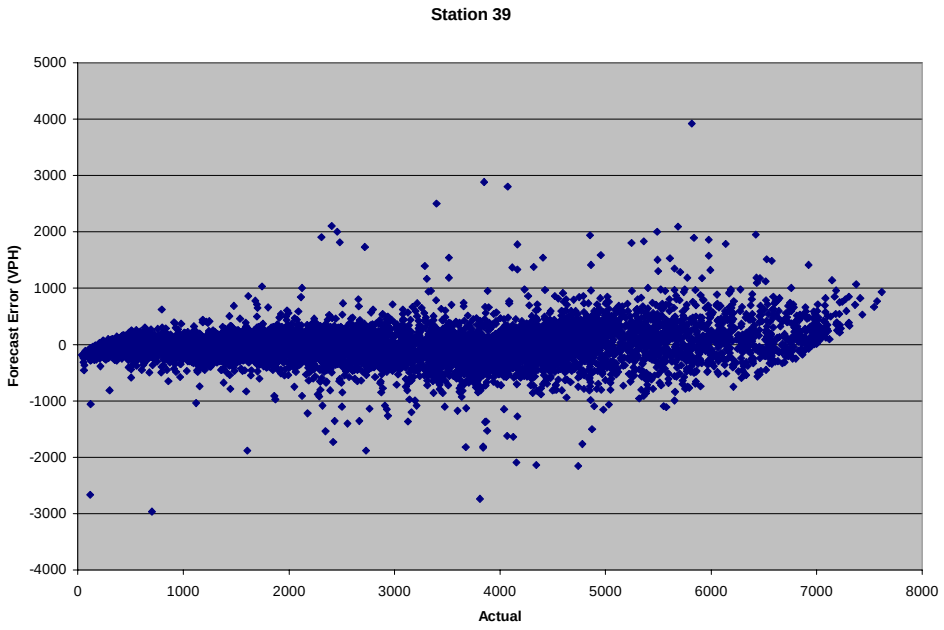


**Figure 43 - Residuals for Station 39**

However, for small values of the actual flow observed, say less than 250 vehicles per hour, the residuals are all negative, which indicates that the forecast algorithm over-estimates the actual flow rate during very light travel conditions. Such low flow rates usually only occur in the early morning when demand for the roadway is minimal and poor forecasts during these conditions may not be as important as during more heavily traveled times. For observed flow rates above 7,000 VPH, all residuals are positive, indicating that the forecasting algorithm under-estimates the actual flow rate when demand is very high. Proactive management of a surface transportation system relies on accurate forecasts, especially during times of heavy demand and the fact that ANN-NPR under-estimates flow rates during these periods for this location may be problematic for a traffic management system. Because so few of the test cases (less than 1%) have an observed flow rate above 7,000 VPH, one potential solution is to develop an ANN-NPR model specifically for periods of heavy demand, with such a model likely using very few neighbors (i.e. small *k*) to calculate the estimate.

On the whole, the plot of the residuals for Station 39 indicates that the ANN-NPR model can be very useful in estimating future traffic conditions as measured by flow rates. However, during very low or very high demand the model may over-estimate or under-estimate the conditions, respectively.

### *5.6.2 Model Inadequacy Arising from Changes in Tuning Option Values*

Another type of model inadequacy may arise when the underlying process changes over time. The form of the model as defined by the state vector may still represent the underlying process but the model is more accurate using other values for the remaining tuning options. In surface transportation systems, such a change may result from long-term construction or, perhaps, deterioration of the roadway over time. Weather systems may be similarly effected by natural and manmade changes to the environment.

To adjust for changes in the underlying process, the tuning option values that were optimized in section 5.5 may need to be optimized again. The test set should consist of cases observed only after the change is thought to have taken place, thus ensuring that the model is optimized for the current conditions.

## *5.7 Summary*

This chapter proposed an iterative methodology for applying ANN-NPR in real-time system, as shown in Figure 39. The methodology consists of five phases.

- Identify System Constraints
- Postulate General Class of Models
- Identify a Tentative Model
- Optimization
- Diagnostic Checking

The limitations imposed by the real-time system must be fully understood and identified for use in the optimization step of the methodology. When enumeration is used to determine optimal values for the tuning options, the modeler uses the limitations of the real-time system during the analysis of the forecasts. For genetic algorithms, the limitations are incorporated into the constraints of the objective function.

Deciding to use kernel or nearest neighbor nonparametric regression occurs in the second step, Postulate General Class of Models. Nearest neighbor NPR has the advantage of always generating an estimate whereas kernel nonparametric regression will not calculate a forecast if no historical observations are close enough to the current conditions. Advanced data structures can be used with both kernel and nearest neighbor nonparametric regression to significantly reduce execution time but approximate nearest neighbors are not easily applied to kernel NPR.

A model to be tentatively entertained consists of specific values or ranges of values and increments to consider for every tuning option. Theoretical knowledge and practical experience with the system being modeled are employed in the selection process.

Values for the remaining tuning options that were not assigned in the previous step are determined through optimization. Enumeration examines every combination of tuning option values to determine an optimal set whereas genetic algorithms may be used to heuristically search the space of decision variable values. Enumerating all possible combinations of tuning options may necessitate a large number of iterations and requires

the modeler to analyze the resulting forecasts to select optimal values for the tuning options. Genetic algorithms, on the other hand, often require fewer iterations but the optimal set of tuning option values must be re-determined if the constraints of the real-time system change.

The final step, Diagnostic Checking, validates the results found from the previous step. If the performance of the model is acceptable and meets the constraints of the real-time system, it may be used in a production system. Otherwise, repeat the steps of Model Identification, Optimization, and Diagnostic Checking to find a more appropriate nonparametric regression model.

# 6 Conclusions

The purpose of this research was to enhance nonparametric regression for use in real-time systems by first reducing execution time using advanced data structures and imprecise computations and then developing a methodology for applying NPR. Because each application of NPR will be specific for each system, this research provides general guidelines for deploying NPR, similar to how Box and Jenkins (1970) provided a methodology for conducting time series analysis. It was assumed that the decision to use nonparametric regression to model the system was made a priori.

The introductory chapter provided an overview of nonparametric regression in relation to case-based reasoning. Chapter 1 identified slow execution time and the lack of a systematic methodology as two major obstacles hindering the widespread use of NPR.

Chapter 2 provided a theoretical description of nonparametric regression and decomposed the model into three fundamental components. The two classes of nonparametric regression, kernel and nearest neighbor, were also discussed. Several previous applications of nonparametric regression were described to highlight several of the key advantages and disadvantages.

One specific disadvantage of nonparametric regression is its slow execution time relative to parametric techniques. Chapter 3 addressed the concerns of using nonparametric regression in real-time systems where the time required to execute an algorithm is as important as the accuracy of the calculations. Chapter 3 also described the theoretical foundation of advanced data structures and approximate nearest neighbors as methods of reducing the execution time of NPR. Other methods of speeding nonparametric regression were briefly discussed. However, the use of advanced data structures and approximate nearest neighbors increases the complexity of nonparametric regression, furthering the motivation for a systematic methodology for deploying NPR, especially in real-time systems.

Six case studies were examined in Chapter 4, five involving the short-term prediction of highway volume and one estimating maximum daily temperature. The case studies demonstrated the effectiveness of advanced data structures and approximate nearest neighbors to reduce the execution time of nonparametric regression. Because

approximate nearest neighbors are a type of imprecise computation, the effect of ANN on forecast accuracy was also examined.

Finally, Chapter 5 detailed a methodology for applying nonparametric regression based on the author's experiences with the case studies. Special attention was given to deploying NPR in real-time systems but the methodology is general enough to apply nonparametric regression in any operating environment. The process, which is iterative like many other model-building methods, offers varying amounts of involvement by the modeler.

## *6.1 Contributions*

This research effort resulted in significant contributions to nonparametric regression users and engineers using forecasting techniques running on real-time systems. The enhancements to nonparametric regression that reduce execution time benefit users of the forecasting technique while the methodology improves the ability to effectively and successfully implement NPR. The improvements to NPR also provide engineers with another potential forecasting technique that may be run on real-time systems.

### *6.1.1 Nonparametric Regression Enhancements*

This research effort applied and extended existing technologies (i.e. advanced data structures and approximate nearest neighbors) to enhance the performance of nonparametric regression for use in real-time systems by significantly reducing execution time. While there are tradeoffs associated with using advanced data structures and approximate nearest neighbors, the execution time of NPR can be reduced to the point where nonparametric regression is a viable forecasting technique for use in real-time systems.

As mentioned previously, numerous research efforts cited slow execution time as a significant disadvantage of nonparametric regression. Section 4.6.2 demonstrated that nonparametric regression using advanced data structures such as KD trees is roughly 1,000 times faster per forecast than nonparametric regression with sequential searching.

Advanced data structures add complexity to the forecasting algorithm in terms of code and tuning options requiring values but do not sacrifice forecast accuracy because they return the same neighbors that would have been found using a sequential search.

Approximate nearest neighbors were shown to further reduce the execution time of nonparametric regression beyond the time-savings already achieved using advanced data structures. Figure 27 through Figure 32 (pages 57-60) demonstrate the savings in execution time resulting from approximate nearest neighbors with different values of $\varepsilon$. As $\varepsilon$ increases, average query time decreases nonlinearly and appears to reach an asymptote near $\varepsilon = 10$. In other words, the most significant decrease in average execution time is achieved by using approximate nearest neighbors with $\varepsilon = 1$ compared to exact nearest neighbors ($\varepsilon = 0$). Furthermore, improvements in average query time appear negligible as $\varepsilon$ approaches 10.

Since first introduced, there have been very few refinements to nonparametric regression. This research provides significant enhancements to NPR that overcome one of its most considerable shortcomings – slow execution time. These enhancements allow nonparametric regression to be used in real-time systems where it was previously infeasible due to its slow execution time.

### *6.1.2 Methodology*

This research developed a methodology for applying nonparametric regression in real-time systems. The motivation for developing a general methodology for applying NPR stemmed from the author's personal experiences with nonparametric regression and specifically addressed the complexity introduced by the modifications required to successfully use nonparametric regression in real-time systems.

Figure 39 (page 73) summarizes the iterative approach to NN-NPR model building employed in this research effort. The methodology consists of five steps.

- Identify System Constraints
- Postulate General Class of Models
- Identify a Tentative Model
- Optimization

- Diagnostic Checking

The first step forces the modeler to identify and fully understand the limitations imposed by the real-time system. The second phase involves the selection of kernel or nearest neighbor nonparametric regression as the fundamental class of models used to represent the system. Step three, Identifying a Tentative Model, allows the modeler to add practical knowledge and personal experience with the system to intelligently restrict the range of values for the tuning options considered. The optimization portion of the methodology focuses on the tradeoff between tuning options and the resulting effects on forecast accuracy and average query time to select optimal values. Finally, diagnostic checks are applied to the model to determine its adequacy.

The methodology, combined with the enhancements to nonparametric regression, makes nonparametric regression better suited for use in real-time systems. The methodology provides guidelines for modelers to answer questions that frequently arise when applying nonparametric regression. It is hoped that, when appropriate, the methodology will allow nonparametric regression to be used in place of less-suitable forecasting techniques.

## *6.2 Future Research*

While this research effort reduced the execution time of nonparametric regression and provided a methodology for applying NPR, there are still several topics worthy of further investigation.

### *6.2.1 Nonparametric Regression*

#### 6.2.1.1 Assumptions

Many of the assumptions of nonparametric regression remain untested. For example, NPR assumes that neighbors closer to the current conditions more accurately represent those conditions. To a degree, this assumption was verified in section 4.6.1 when forecasts calculated with exact nearest neighbors were consistently more accurate than

estimates computed using approximate nearest neighbors. However, it is unknown whether the assumption is independent of the state space definition, distance metric, or forecast function.

Similarly, an implicit assumption used in this research is that $k$ and $\varepsilon$ (and all tuning options) effect forecast accuracy independently of the forecast function employed. The results presented in Smith et. al. (2000) appear to indicate that this is true, at least for different values of $k$. If in fact the tuning options effect forecast accuracy independently of the forecast function employed, the NPR methodology may be simplified accordingly if more than one forecast function is being considered.

#### 6.2.1.2 Approximate Nearest Neighbors with Kernel Nonparametric Regression

The performance of kernel nonparametric regression may also be improved through the use of advanced data structures similar to how NN-NPR was enhanced. Using approximate nearest neighbors with kernel nonparametric regression, on the other hand, is not as easily accomplished. Approximate nearest neighbors, as implemented in this research effort, find the $k$ neighbors that are within relative error $\varepsilon$ of the true nearest neighbors. Kernel nonparametric regression, on the other hand, uses all historical points within a predetermined distance from the current conditions to generate a forecast. Herein lies the incompatibility – ANN searches for $k$ neighbors whereas kernel nonparametric regression selects an unspecified number of neighbors based on distance.

#### 6.2.1.3 Approximate Nearest Neighbor Tuning Options

A thorough study of how the tuning options associated with approximate nearest neighbors effect nonparametric regression would reinforce section 5.4 of the methodology by providing more explicit guidelines for specifying values for the tuning options based on theoretical knowledge and practical experience with the system. For example, Mount (1998) suggests using the standard search for small values of $\varepsilon$, but does not stipulate a cutoff value below which standard search is preferred. Furthermore, the effects of bucket size on query times are not well understood. Arya et. al. (1998) point out that “adjusting bucket size affects the search time” but provide no insight about the

relationship between bucket size and query time. A more thorough examination would consider the effect of bucket size on search performance in conjunction with splitting rules and search methods.

Arya et. al. (1998) tested a new multidimensional search tree using seven data sets from different distributions include uniform, Gaussian, Laplace, and various other distributions containing increasingly more clustered data. Similar data sets could be used to determine the effects of the ANN tuning options on nonparametric regression.

### *6.2.2 Methodology*

The methodology for applying nonparametric regression presented in Chapter 5 is heuristic in nature. Many of the decisions are subjective without statistical procedures to recommend a course of action. The first iteration of Box and Jenkins' (1970) methodology for time series analysis was similar. In fact, the authors recommend that the modeler "know the facts as clearly as they can be known – then use judgment" (Box and Jenkins, 1970).

Recommended improvements to the Box-Jenkins methodology suggest using information criterion to aid in model selection. "With information criterion that are well developed and tested and with improved software and computing power, … optimizing a chosen information criterion is the best way to select ARIMA models" (Williams, 2000), thus eliminating most of the subjectivity present in time series analysis. The methodology for nonparametric regression would similarly benefit from the use of information criterion to reduce some of the subjective decisions required by the modeler. Most information criterion, including Akaike's and Bayesian information criterion, penalize models based on the number of parameters in an effort to reach a balance between accurate yet parsimonious models. Nonparametric models may be penalized by the number of elements in the state space.

## 6.3 Summary

This research effort achieved the stated objectives of reducing the execution time of nonparametric regression and providing a systematic methodology for applying NPR. Advanced data structures such as KD trees were used to reduce execution time by as much as 1,000 times without sacrificing forecast accuracy. Imprecise computations as implemented via approximate nearest neighbors further reduced execution time but at the expense of forecast accuracy. A methodology for applying nonparametric regression was given, motivated by the author's personal experiences with nonparametric regression, especially with the added complexity introduced by the modifications required to successfully use nonparametric regression in real-time systems.

# Bibliography

Altman, N. S. “An Introduction to Kernel and Nearest Neighbor Nonparametric Regression.” The American Statistician. Vol. 46, No. 3. August 1992: 175-185.

Arya, Sunil, and David M. Mount. “Approximate Nearest Neighbor Searching.” 4th Annual ACM-SIAM Symposium on Discrete Algorithms. 1993: 271-280.

Arya, Sunil, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. “An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions.” Journal of the ACM. Vol. 45. 1998: 891-923.

Bentley, Jon L. “Multidimensional Binary Search Trees in Database Applications.” IEEE Transactions on Software Engineering. Vol. SE-5, No. 4. July 1979: 333-340.

Bentley, Jon L. “Multidimensional Binary Search Trees Used for Associative Searching.” Journal of the ACM. Vol. 18, No. 9. September 1975: 509-517.

Box, George E. P. and Gwilym M. Jenkins. Time Series Analysis: Forecasting and Control. California: Holden-Day. 1970.

Charytoniuk, W., M. S. Chen, and P. Van Olinda. “Nonparametric Regression Based Short-Term Load Forecasting.” Submitted for publication in IEEE Transactions on Power Systems. 1997.

Clark, S. D. “Traffic Prediction Using Multivariate Nonparametric Regression.” Report for Institute for Transport Studies University of Leeds. September 30, 1999.

Clarkson, Kenneth L. “Nearest Neighbor Queries in Metric Spaces.” Proceedings of the 29th Annual ACM Symposium on Theory of Computing. 1997: 609-617

Davis, Gary A. and Nancy L. Nihan. “Nonparametric Regression and Short-Term Freeway Traffic Forecasting.” Journal of Transportation Engineering. Vol. 117, No. 2. March/April 1991: 178-188.

Eastman, Caroline M. “Nearest Neighbor Searching Using *k*-d Trees: Some Preliminary Experimental Results on the Impact of Query and Record Distribution.” Technical Report Southern Methodist University. 1984.

Friedman, Jerome H., Jon L. Bentley, and Raphael A. Finkel, and. “An Algorithm for Finding Best Matches in Logarithmic Expected Time.” ACM Transactions on Mathematical Software. Vol. 3, No.3. September 1977: 209-236.

Ignizio, James P. and Tom M. cavalier. Linear Programming.New Jersey: Prentice Hall, Inc. 1994.

Karlsson, M. and S. Yakowitz. “Rainfall-Runoff Forecasting Methods, Old and New.” Stochastic Hydrology and Hydraulics. 1987: 303-318.

Kennedy, Ruby L., Yuchun Lee, Benjamin Van Roy, Christopher D. Reed, and Richard P. Lippmann. Solving Data Mining Problems Through Pattern Recognition. New Jersey: Prentice Hall, Inc. 1998.

Liu, Jane W. S., Wei-Kuan Shih, Kwei-Jay Lin, Riccardo Bettati, and Jen-Yao Chung. “Imprecise Computations.” Proceedings of the IEEE. Vol. 82, No. 1. January 1994: 55-67.

Maneewongvatana, Songrit and David M. Mount. “Analysis of Approximate Nearest Neighbor Searching with Clustered Point Sets.” ALENEX 99. 1999.

Mendenhall, William and Terry Sincich. A Second Course in Statistics: Regression Analysis. New Jersey: Prentice Hall, Inc. 1996.

Michalewicz, Zbigniew. Genetic Algorithms + Data Structures = Evolution Programs. New York: Springer. 1996.

Mount, David M. "ANN Programming Manual." University of Maryland. 1998.

Mulhern, Francis J. and Robert J. Caprara. "A Nearest Neighbor Model for Forecasting Market Response." International Journal of Forecasting. Vol. 10, No. 2. 1994: 191-207.

Neufeldt, Victoria, ed. Webster's New World Dictionary. 3rd ed. New York: Time Warner. 1990.

Ramamrithan, Krithi and John A. Stankovic. "Scheduling Algorithms and Operating Systems Support for Real-Time Systems." Proceedings of the IEEE. Vol. 82, No. 1. January 1994: 55-67.

Ripley, B. D. Pattern Recognition and Neural Networks. United Kingdom: Cambridge University Press. 1999.

Schaal, Stefan. "Nonparametric Regression for Learning." Proceeding of the Conference on Prerational Intelligence. Germany. 1994.

Smith, Brian L. "Forecasting Freeway Traffic Flow for Intelligent Transportation Systems Application." Diss. University of Virginia. 1995.

Smith, Brian L., Billy M. Williams, and R. Keith Oswald. "Comparison of Parametric and Nonparametric Models for Traffic Condition Forecasting." Accepted for publication in Transportation Research Record. January 2000.

Thearling, Kurt. “Massively Parallel Architectures and Algorithms for Time Series Analysis.” Lectures in Complex Systems. 1993.

Transportation Research Board. Highway Capacity Manual. 3rd ed. Washington DC: Transportation Research Circular. 1997.

Williams, Billy M. Letter to the author. July 7, 2000.

Yianilos, Peter N. “Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces.” Proceedings of the ACM-SIAM Symposium on Discrete Algorithms. 1993: 311-321.