Machine Learning for Diabetes Decision Support

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Matthew T. Wiley

August 2011

This thesis titled

Machine Learning for Diabetes Decision Support

by

MATTHEW T. WILEY

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Cynthia R. Marling

Associate Professor of Electrical Engineering and Computer Science

Dennis Irwin

Dean, Russ College of Engineering and Technology

Abstract

WILEY, MATTHEW T., M.S., August 2011, Computer Science

Machine Learning for Diabetes Decision Support (158 pp.)

Director of Thesis: Cynthia R. Marling

This thesis presents work in machine learning that enhances and expands the scope of the 4 Diabetes Support System™ (4DSS). The 4DSS is a decision support system designed to assist patients and physicians with the challenge of managing Type 1 diabetes (T1DM). The objective of the 4DSS is to detect problems in diabetes management and to recommend therapeutic changes to correct these detected problems. This thesis contributes three advances: (1) preprocessing noisy data, preparatory to applying machine learning algorithms; (2) enhancing the automated detection of excessive glycemic variability, a serious problem for patients with diabetes; and (3) predicting patient blood glucose levels, in order to preemptively detect and avoid potential health problems.

In this work, the Continuous Glucose Monitoring (CGM) data is smoothed using cubic spline smoothing with extra weight on fingersticks and local optima. This data preprocessing improves the accuracy of problem detection and blood glucose prediction. Previous work in classifying glycemic variability using a naïve Bayes classifier obtained an accuracy of only 87.1%. Using smoothed CGM data and a rich set of domain independent pattern recognition features to train multilayer perceptrons and support vector machines, a best accuracy of 93.8% has now been obtained. This machine learning classifier improves the ability to detect excessive glycemic variability, an important indicator of risk for diabetic complications.

Accurately predicting blood glucose levels could enhance patient safety by giving patients time to intervene before problems occur. Support Vector Regression (SVR) and AutoRegressive Integrated Moving Average (ARIMA) models were built and tested on data from ten T1DM patients. This resulted in a best Root Mean Square Error (RMSE) of

4.5 mg/dl for 30 minute predictions and 17.4 mg/dl for 60 minute predictions. Clarke Error Grid Analysis (CEGA) showed that 99% of 30 minute predictions and 90% of 60 minute predictions fell within 20% of target, CEGA's most accurate range.

Approved: ____

Cynthia R. Marling

Associate Professor of Electrical Engineering and Computer Science

Acknowledgements

Completion of this thesis would not have been possible without the efforts of my advisor – Dr. Cynthia Marling, and my machine learning mentor – Dr. Razvan Bunescu; words cannot describe my gratitude for their guidance and support with this project. I am thankful for everyone who has helped with the research behind this project, including the patients, the domain experts Dr. Frank Schwartz and Dr. Jay Shubrook, committee member Dr. Jundong Liu, fellow students Stan Vernier and Tessa Cooper, and everyone else who has been or still is involved. I would especially like to thank my mother and her parents for their unconditional love and support through my education. The most important thanks goes to my wife Shu. Her passion for research convinced me to enroll in graduate studies, and I am forever grateful that I did.

Table of Contents

List of Tables

Table Page

List of Figures

1 Introduction

This thesis presents research in machine learning for diabetes management. There are three major contributions:

1. preprocessing noisy data, preparatory to applying machine learning algorithms;

2. enhancing the automated detection of excessive glycemic variability, a serious problem for patients with diabetes; and

3. predicting patient blood glucose levels, in order to preemptively detect and avoid potential health problems.

This work enhances the 4 Diabetes Support System™ (4DSS) and expands its scope. The 4DSS is a decision support system designed to assist physicians and patients with management of Type 1 diabetes (T1DM). T1DM is a disease in which the pancreas is unable to produce insulin. This requires patients with T1DM to constantly monitor their blood glucose values and administer dosages of insulin when appropriate. The 4DSS automatically identifies problems in blood glucose control that require changes to insulin pump therapy. Once these problems are detected, case-based reasoning is used to find similar problems, and therapeutic advice is generated.

The goal for patients with T1DM is to control their blood glucose levels such that they resemble the glucose levels of a person without diabetes. Poor blood glucose control can lead to serious diabetic complications, such as blindness, kidney failure and death (American Diabetes Association, 2011). A recent focus of the 4DSS has been to detect excessive glycemic variability using artificial intelligence (Vernier, 2009). Currently there is no screen for excessive glycemic variability; however, such a tool would be useful for diabetes clinicians. Another approach to detecting problems in blood glucose control is to preemptively detect them using a blood glucose prediction model. As with classification, prediction is accomplished using artificial intelligence. A detailed description of diabetes, 4DSS, artificial intelligence techniques, and evaluation techniques is given in Chapter 2.

The first contribution of this thesis is preprocessing data collected for the 4DSS. This includes inferring missing life event data and smoothing Continuous Glucose Monitoring (CGM) data. Missing life event data was inferred by exploiting other information collected by the 4DSS. Several techniques were investigated for smoothing CGM data. These techniques were evaluated by an endocrinologist, and cubic spline smoothing was determined to give the best results. A detailed explanation of these data preprocessing techniques is given in Chapter 3.

The second contribution of this thesis is enhancements to detection of excessive glycemic variability. These enhancements include smoothing CGM data, using a rich set of pattern recognition features, and evaluating classification algorithms. Feature selection was carried out to find a subset of features which gave optimal results on a validation dataset. This subset of features was then used with k-fold cross validation to evaluate and compare the previous work with the new enhancements. A detailed explanation with results is given in Chapter 4.

The last contribution of this thesis is a new module for the 4DSS that predicts blood glucose values. This module uses data previously collected for the 4DSS to train and evaluate models that predict blood glucose values. This work motivated the need to preprocess data as described in Chapter 3. Using the preprocessed data, Support Vector Regression (SVR) is used to build a prediction model. The SVR model is then evaluated using walk-forward testing. A detailed description along with results is given in Chapter 5. Comprehensive results for 10 different patients are given in Appendix A.

Chapter 6 describes research related to this work. This includes work on other time series prediction problems that use SVR models, other work in predicting blood glucose values, and work towards an artificial pancreas. Chapter 7 presents opportunities for future work. Chapter 8 gives a summary and conclusions.

2 Background

This chapter presents background information relevant to this work. First, diabetes is defined, and the management challenge for diabetes is described in detail; this section defines many of the domain specific terms used throughout this thesis. Next is a description of the 4 Diabetes Support System™, the project within which this work was conducted. Then, a description of the artificial intelligence techniques employed in this work is given. Finally, techniques used for evaluating the artificial intelligence algorithms are described.

2.1 Diabetes

According to the American Diabetes Association (ADA), "Diabetes mellitus (MEL-ih-tus), or simply, diabetes, is a group of diseases characterized by high blood glucose levels that result from defects in the body's ability to produce and/or use insulin" (American Diabetes Association, 2010a). As of 2011, 10.9 million people age 65 years or older (26.9%), and 25.6 million people age 20 or older (11.3%) in the United States have diabetes mellitus (American Diabetes Association, 2011). The total cost of diabetes in the United States for 2007 was $174 billion dollars (American Diabetes Association, 2011).

There are two prevalent forms of diabetes, Type 1 (T1DM) and Type 2 (T2DM). T1DM occurs when the body is no longer able to produce insulin. Onset of T1DM is common in childhood; this disease used to be known as juvenile diabetes. This form of diabetes is less common; only about 5-10% of people with diabetes have T1DM (American Diabetes Association, 2010b). T2DM occurs when the body is unable to utilize the insulin produced or not enough insulin is produced. T2DM is commonly associated with obesity; however, obesity is not the only high risk factor. Certain ethnicities are considered to be high risk groups, as large percentages of those ethnicities have diabetes (American Diabetes Association, 2010c).

2.1.1 Managing Diabetes

Patients with diabetes need to control their blood glucose levels. Insulin or other medication may be used to control blood glucose levels. If blood glucose levels are not adequately controlled, the long term complications can be quite costly in terms of both health and finance. Such complications include increased risk for heart disease and stroke, blindness, kidney failure, and even death (American Diabetes Association, 2011). Typically, patients with T2DM can control their blood glucose levels through medication, exercise, and proper diet. Patients with T1DM require insulin to survive, either from injections or from a pump (Centers for Disease Control and Prevention, 2007). Many patients with T1DM use an insulin pump in conjunction with Continuous Glucose Monitoring (CGM).

The insulin pump allows the patient to administer any amount of insulin. For patients using Metronic pumps, this amount is chosen with the help of the Bolus Wizard®. There are many factors which influence the efficacy of insulin for the patient that the Bolus Wizard®considers. Insulin sensitivity, which varies from patient to patient, is a measure of the patient's responsiveness to insulin. The carb ratio, which is also patient specific, describes the amount of insulin required to cover carbohydrates for a meal. When calculating a recommended bolus amount, the Bolus Wizard® uses the insulin sensitivity and carb ratio parameters, along with a recent bolus history and the current blood glucose reading.

There are two major problems in blood glucose control: hyperglycemia and hypoglycemia. Hyperglycemia, or high blood glucose levels, occurs during diabetes in the absence of treatment. In T1DM, it is pronounced when the insulin pump fails or when the patient does not administer enough insulin. Hypoglycemia, or low blood glucose levels, occurs when the patient administers too much insulin. Recent research indicates that glycemic variability, or fluctuation between highs and lows, is a third problem contributing to increased risk of complications (Ceriello and Ihnat, 2010; Hirsch and Brownlee, 2005; Kilpatrick et al., 2010; Kilpatrick et al., 2006; Monnier and Colette, 2008; Monnier et al., 2006).

Patients with diabetes must regularly monitor their blood glucose levels using fingersticks. A fingerstick is obtained by drawing a small amount of blood for analysis by a personal glucose meter. Fingersticks are used by the Bolus Wizard® for recommending insulin dosages. For patients using CGM, fingersticks are used to calibrate the sensor. Patients are advised to calibrate their sensor three times a day. This calibration may cause discontinuities in the CGM values. However, fingerstick data is more accurate than CGM data and is relied upon when readings disagree. The CGM sensor records sample blood glucose values every five minutes, which allows the patient to closely monitor their blood glucose levels. The CGM sensor lags the real blood glucose values by 10 to 15 minutes, providing values within ±20% of the actual values. (Mastrototaro et al., 2008).

The system of using a CGM sensor and an insulin pump to control blood glucose values is open loop; the patient must intervene with the system for everything to be in flux. Closing the loop with an artificial pancreas is an idea proposed by Dr. Arnold Kadish that dates back to 1964 (Juvenile Diabetes Research Foundation, 2010). If an artificial pancreas could supply the patient with insulin such that the system would not cause hypoglycemia or hyperglycemia, then it would be possible to build a closed-loop system. However, the major challenge to building a closed-loop system is the dynamics of the efficacy of insulin. Every patient reacts differently to insulin. Even the same patient may react differently to insulin at different times. Factors known to influence the efficacy of insulin include exercise, diet, stress and other life events. These factors present many challenges to open loop, as well as closed-loop, control.

2.2 The 4 Diabetes Support System Project

The work described in this thesis was conducted within the context of the 4 Diabetes Support System™ (4DSS) project. The 4DSS is a case-based decision support system aimed to facilitate both physician and patient management of T1DM. The 4DSS completes this task in 3 steps: detecting problems in blood glucose control, generating solutions to these detected problems, and remembering which solutions worked for future reference. 4DSS research and development has been conducted over the course of three clinical research studies, the third of which is still ongoing. These studies are described next.

2.2.1 The Preliminary Study

The purpose of the first 4DSS study was to determine if a decision support system could be developed to help manage patients with T1DM. Before the study was conducted, it was approved by the Institutional Review Board (IRB) at Ohio University. Twenty human subjects with T1DM enrolled for a period of 6 weeks per subject, and 12 subject completed the entire protocol. A variety of patient information was collected, including: background information, insulin pump data, CGM data, and daily life event data. Daily life event data included meal information, sleep information, work information, stress, illness, and other miscellaneous information. In addition, each patient filled out an exit survey at the end of their participation in the study. Using the collected data, a 4DSS prototype was built by knowledge engineers and evaluated by both diabetes experts and knowledge engineers. This study showed that a decision support system for T1DM would be feasible. It identified the needs to address additional problems in blood glucose control and to reduce data entry time for patients (Marling et al., 2008; Schwartz et al., 2008; Marling et al., 2009a).

At the end of the preliminary study, the 4DSS prototype consisted of four different modules and a case base with 49 cases. These modules included a website for data entry, a database for recording patient information, situation assessment for detecting problems, and a case retrieval module for identifying cases with similar problems to those detected by situation assessment. The website for data entry was developed by Anthony Maimone (Maimone, 2006). The database was developed by Anthony Maimone with the help of Kathleen Evans-Romaine and Wesley Miller (Maimone, 2006). The situation assessment module was developed by Wesley Miller (Miller, 2009). The case retrieval module was developed by Donald Walker (Walker, 2007). The case-base was created by the diabetes experts and knowledge engineers, using the collected data.

2.2.2 The Second Study

The purpose of the second 4DSS study was to evaluate the capabilities of the situation assessment and case retrieval modules developed during the preliminary study. The second study received approval from the IRB at Ohio University prior to its beginning. Twenty-six adult human subjects with T1DM enrolled for a period of 5 weeks per subject. Twenty-three subjects completed the entire protocol. Since the case base was built using data from the first study, patients who participated in the first study did not also participate in the second study. This was done to prevent any bias in the evaluation. As in the first study, patient data was recorded in the database (Schwartz et al., 2010).

Evaluation of the problem detection and the case retrieval module were presented in (Schwartz et al., 2010; Marling et al., 2009b; Marling et al., 2009c; Vernier, 2009). For detecting problems, this evaluation showed that the problems detected were correct and useful a majority of the time. The patients' own physicians evaluated these detections and found that 97.9% of the detected problems were correct, and 96.1% were useful. Four diabetes experts evaluated the case retrieval module. The experts found that 79% of the cases retrieved had problems that were similar to the problem that was detected, and 82%

of the associated solutions were helpful to the patients experiencing the problems (Schwartz et al., 2010).

2.2.3 The Third Study

The evaluation from the second study showed that there was room for improvement in the case retrieval module. Collecting more data and adding cases to the case base is one way to improve this performance. This motivated the need for a third study. As with the first two studies, the third study was approved by the IRB at Ohio University. So far, seventeen human subjects with T1DM enrolled for a period of 3 months per subject, and twelve have completed the entire protocol. This study led to several improvements and extensions for the 4DSS project.

New cases were created from the collected data and added to the case-base. This gave the case retrieval module more cases to select from. However, to make retrieved solutions specific to individual patients, solutions needed to be adapted. This resulted in the fifth module of the 4DSS project, which is adaptation. This module was developed by Tessa Cooper (Cooper, 2010). The purpose of this module is to tailor the solution found by the case retrieval module to the specific needs of the patient. An example of a solution requiring adaptation is one that suggests the patient should increase their basal rate before bedtime from 0.9 to 1.0 units. However, if the patient's current basal rate before bedtime is 0.6, adjusting it to 1.0 would not be ideal. The adaptation module can tailor the advice such that an appropriate basal rate is suggested.

Stan Vernier added new functionality to the situation assessment module to automatically detect days containing excessive glycemic variability (Marling et al., 2011; Vernier, 2009). As indicated in Section 2.1.1, excessive glycemic variability may lead to an increased risk of diabetic complications. This work is discussed in greater detail in Chapter 4.

The current work not only enhances the 4DSS, but also expands its scope. Detection of excessive glycemic variability (Vernier, 2009) has been improved as described in Chapter 4. Prediction of future blood glucose values, described in Chapter 5, is new functionality that is useful for the 4DSS in multiple ways. If the predictor were used in real-time, problems could be detected preemptively and prevented. A real-time predictor would also allow the patient to posit hypothetical situations. This would allow the patient to see what could possibly happen in the future if they were to eat a meal or administer a certain amount of insulin. The predictor could act as an educational tool, for example, demonstrating to a patient what could be done to avoid a hypoglycemic episode. The adaptation module could consult the prediction module when generating solutions. This would allow the adaptation module to test solutions before suggesting them. Figure 2.1 gives an overview of all the modules included in the 4DSS project and how these modules are connected.

2.3 Artificial Intelligence

This section describes artificial intelligence techniques and formulations that were used for this work. These techniques include the machine learning algorithms Multilayer Perceptrons (MP)s, Support Vector Machines (SVM) for classification, and Support Vector Regression (SVR). The formulation of a time series prediction problem is important for predicting blood glucose values.

2.3.1 Multilayer Perceptrons

A multilayer perceptron is a type of feedforward neural network. This algorithm is a variant of the perceptron algorithm proposed by (Rosenblatt, 1958). Perceptrons are described in detail in (Rumelhart et al., 1986; Bishop, 1995; Theodoridis and Koutroumbas, 2009; Bishop, 2006; Mitchell, 1997; Rosenblatt, 1958). In this work, MPs

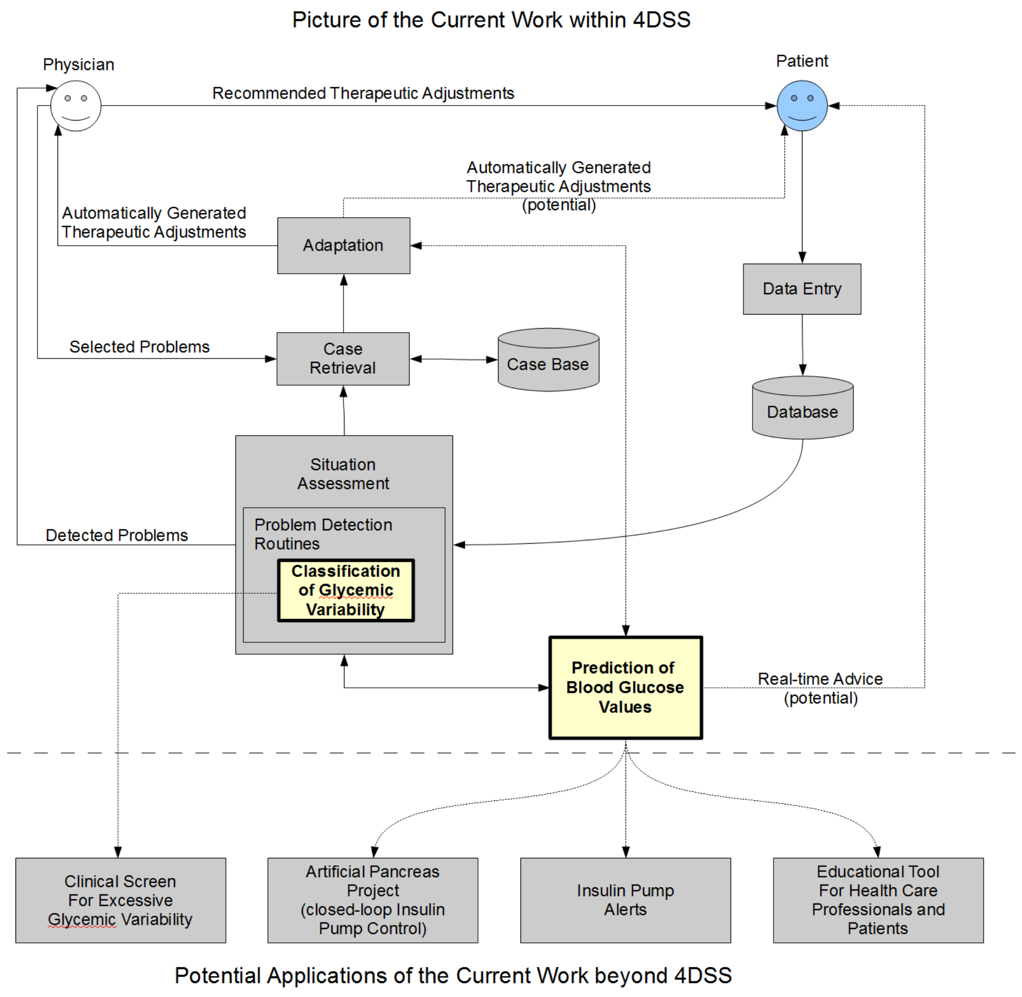

Picture of the Current Work within 4DSS
Physician
Patient
Recommended Therapeutic Adjustments
Automatically Generated Therapeutic Adjustments (potential)
Automatically Generated Therapeutic Adjustments
Adaptation
Data Entry
Selected Problems
Case Retrieval
Case Base
Database
Situation Assessment
Problem Detection Routines
Classification of Glycemic Variability
Detected Problems
Prediction of Blood Glucose Values
Real-time Advice (potential)
Clinical Screen For Excessive Glycemic Variability
Artificial Pancreas Project (closed-loop Insulin Pump Control)
Insulin Pump Alerts
Educational Tool For Health Care Professionals and Patients
Potential Applications of the Current Work beyond 4DSS

Figure 2.1: A flowchart specifying the modules of the 4DSS project.

are used for classification of glycemic variability. A brief introduction to MPs is given in the remainder of this section.

The perceptron algorithm is guaranteed to converge if the data **x** is linearly separable. Linear separability implies that there exists some hyperplane **w** such that:

$$\mathbf{w}^{\top}\mathbf{x} + b > 0 \quad \forall t \in \omega_1 \tag{2.1}$$

$$\mathbf{w}^{\top}\mathbf{x} + b < 0 \quad \forall t \in \omega_2 \tag{2.2}$$

where $\mathbf{w} \equiv (w_1, w_2, \ldots, w_N)^\top$ is a weight vector, $\mathbf{x} \equiv (x_1, x_2, \ldots, x_N)^\top$ is an input vector, b is an offset, t is the example label, and ω represents the class label. The perceptron is said to converge when Equations 2.1 and 2.2 have been satisfied, meaning that the data has been linearly separated. In the case of the data not being linearly separable, the multilayer perceptron achieves convergence with the use of hidden layers and backpropagation. Separation of the data is learned by updating the weight vector $\mathbf{w}$ using a learning rule. When there is an example which does not satisfy Equations 2.1 and 2.2, the learning rule is activated for the hidden layer(s), which updates the weight vector in the following manner:

$$\Delta\mathbf{w}(\tau) = -\eta \;\nabla E\Big|_{\mathbf{w}(\tau)} + \alpha\, \Delta\mathbf{w}(\tau - 1) \tag{2.3}$$

where τ is the iteration step of the perceptron, $\Delta\mathbf{w}(\tau)$ is the weight vector increment at step τ, $\nabla E\Big|_{\mathbf{w}(\tau)}$ is the gradient of the error function in the weight space at step τ, η is the learning rate, and α is the momentum. The learning rate η controls the step size of the weight adjustments. The momentum parameter α – which acts as an exponential decay with values between zero and one – helps the perceptron achieve convergence quicker. This parameter controls the influence of past weight changes on the current weight change, which adds inertia to the weight space. A full derivation of this formula is given in (Bishop, 1995; Rumelhart et al., 1986). An advantage of MPs is that a backpropagation network with enough hidden nodes can approximate any decision surface (Hornik and White, 1989).

2.3.2 Support Vector Machines

SVMs were first described in 1979 in (Vapnik, 1979). The books (Vapnik, 1998; Vapnik, 1995) present an introduction and overview of SVMs. An excellent tutorial on SVMs for pattern recognition is given in (Burges, 1998) and the books (Theodoridis and Koutroumbas, 2009; Bishop, 2006) have chronicled many of the recent developments with

SVMs. In this work, SVMs are used for classifying glycemic variability and predicting blood glucose values. A brief introduction to SVMs for classification is given in the rest of this section.

Like perceptrons, SVMs attempt to find a hyperplane that separates the data; however, an SVM tries to maximize the margin separating the classes. Figure 2.2 illustrates this maximum margin approach. Consider a decision boundary which satisfies Equations 2.1 and 2.2 for a binary classification problem. It can be shown that the distance between a given point $\mathbf{x_n}$ and the decision boundary is given as:

$$\frac{t_n y(\mathbf{x}_n)}{||\mathbf{w}||}$$

where $t_n \in \{-1, 1\}$ and corresponds to the label for the n^th example, $\mathbf{x}_n$ is the n^th feature vector, $||\mathbf{w}|| \equiv \sqrt{\mathbf{w}^\top\mathbf{w}} \equiv \sqrt{w_1^2 + \ldots + w_{|\mathbf{x}_n|}^2}$ and $y(\mathbf{x}_n) = \mathbf{w}^\top\phi(\mathbf{x}_n) + b$ where $\phi(\mathbf{x}_n)$ is a feature space transformation and b is an offset (Bishop, 2006). We want to find the point $\mathbf{x}_n$ with the closest perpendicular distance to the decision boundary while optimizing the parameters $\mathbf{w}$ and b to maximize the distance of the margin. This is formulated in (Bishop, 2006), as a quadratic programming optimization problem, $J(\mathbf{w}, b)$, where we wish to minimize the norm of the weight vector $\mathbf{w}$, as shown in Equation 2.4:

$$J(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 \tag{2.4}$$

subject to:

$$t_n\left(\mathbf{w}^\top\phi(\mathbf{x}_n) + b\right) \geq 1, \quad \text{for all } n = 1, \ldots, N \tag{2.5}$$

This problem can be solved by introducing Lagrange multipliers, $a_n \geq 0$, for each constraint defined in Equation 2.5. The primal form becomes Equation 2.6:

$$L_p(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{n=1}^{N} a_n\left\{t_n\left(\mathbf{w}^\top\phi(\mathbf{x}_n) + b\right) - 1\right\} \tag{2.6}$$

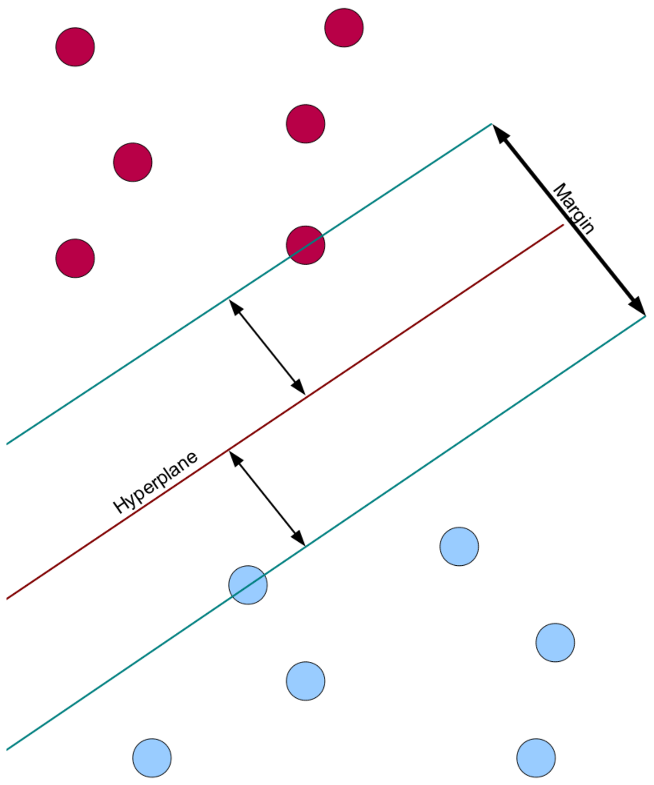


Figure 2.2: Illustration of the maximum margin.

where $\mathbf{a} \equiv (a_1, \ldots, a_N)^\top$. Minimizing $\mathbf{w}$ and b is achieved by setting the derivatives of $L_p(\mathbf{w}, b, \mathbf{a})$ with respect to $\mathbf{w}$ and b to zero. This gives the equality constraints of the dual formulation problem, as defined in Equations 2.7 and 2.8:

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n) \tag{2.7}$$

$$0 = \sum_{n=1}^{N} a_n t_n \tag{2.8}$$

Substituting these constraints into Equation 2.6 gives the dual formulation, shown in Equation 2.9:

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k\left(\mathbf{x}_n, \mathbf{x}_m\right) \tag{2.9}$$

subject to:

$$a_n \geq 0, \quad n = 1, \ldots, N \tag{2.10}$$

$$0 = \sum_{n=1}^{N} a_n t_n \tag{2.11}$$

where $k(\mathbf{x}_n, \mathbf{x}_m) = \phi^\top(\mathbf{x}_n)\,\phi(\mathbf{x}_m)$ is the kernel function. The unique aspect of this dual formulation is that the input is represented as a dot product of feature vectors. Introducing slack variables allows SVMs to converge without linearly separable data. The new constraints with the slack variables, denoted as ξ_n, are defined in Equation 2.12:

$$t_n y_n\left(\mathbf{x}_n\right) \geq 1 - \xi_n, \quad n = 1, \ldots, N \tag{2.12}$$

The purpose of these slack variables is to allow for some of the points to be outside of the decision boundary, while maximizing the margin; this is soft margin maximization. The optimization problem can then be rewritten per Equation 2.13:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n \tag{2.13}$$

The constant C is known as the regularization constant. A larger C will put more emphasis on training errors, whereas a smaller C will put more emphasis on minimizing the norm of the weight vector. A full derivation with theorems is contained within the tutorial in (Burges, 1998).

It has been shown that the only points needed for classification are points either on or inside the margin, illustrated in Figure 2.2 (Bishop, 2006). These examples are known as support vectors. Only support vectors will have non-zero Lagrange multipliers, resulting in sparse solutions. Kernel substitution can be applied to the kernel for SVMs, which allows the kernel to be transformed into a polynomial, or radial basis function (RBF). The kernel function $k(\mathbf{x}_n, \mathbf{x}_m)$ is valid if the result is positive semidefinite for all $\mathbf{x}_n$ (Bishop, 2006). This type of kernel transformation is the biggest advantage of SVMs because it allows the original feature space to be projected into a higher dimensional space. This transformation is key to separating data that is not linearly separable, i.e., when there is no hyperplane $\mathbf{w}$ that satisfies Equations 2.1 and 2.2. An example of a RBF kernel is given in Equation 2.14:

$$k(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||^2}{\sigma^2}\right) \tag{2.14}$$

where $\mathbf{u}$ and $\mathbf{v}$ are two input examples, and σ is the width of the hypersphere in the infinite dimensional feature space corresponding to $k(\mathbf{u}, \mathbf{v}) = \phi^\top(\mathbf{u})\,\phi(\mathbf{v})$.

2.3.3 Support Vector Regression

SVMs were originally used for solving classification problems. They have since been extended to solve regression and ranking problems. Regression analysis with SVMs is known as Support Vector Regression (SVR). Smola and Scholkopf have published a comprehensive tutorial explaining SVR (Smola, A.J. and Scholkopf, B., 2004). The use of SVR to solve a time series prediction problem has became a topic of interest over the past decade (Sapankevych and Sankar, 2009). One appealing aspect of using SVR for time series prediction is that it does not require a predefined mathematical model. The prediction of future values is driven by the training data. In this work, SVR is used to predict future blood glucose values.

Before delving into SVMs for regression, we start by looking at ridge regression. In ridge regression, we seek to minimize a regularized error function given as:

$$\frac{1}{2}\sum_{n=1}^{N}\{y(x_n \mid \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}||\mathbf{w}||^2 \tag{2.15}$$

where the constant $\frac{1}{2}$ is included for convenience, $y(x_n \mid \mathbf{w}) = \sum_{i=0}^{|\mathbf{w}|} w_i\, x^k$ giving the predicted value, t_n is the actual value, and λ is the ridge parameter. The quadratic error function results in solutions which are not sparse; therefore the quadratic error function is replaced with an ϵ-insensitive error function in terms of the prediction $y(\mathbf{x})$ (Vapnik, 1995). This function allows for examples to have zero error if the absolute difference between the predicted value and actual value is less than some constant ϵ. A linear cost function is shown in Equation 2.16:

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon \\ |y(\mathbf{x}) - t|, & \text{otherwise} \end{cases} \tag{2.16}$$

The E_ϵ function allows points to have an error of zero that are inside of an ϵ insensitive region – also known as the ϵ tube – illustrated as the shaded region in Figure 2.3. This results in sparse solutions. The objective is to find an optimal weight vector $\mathbf{w}$ such that

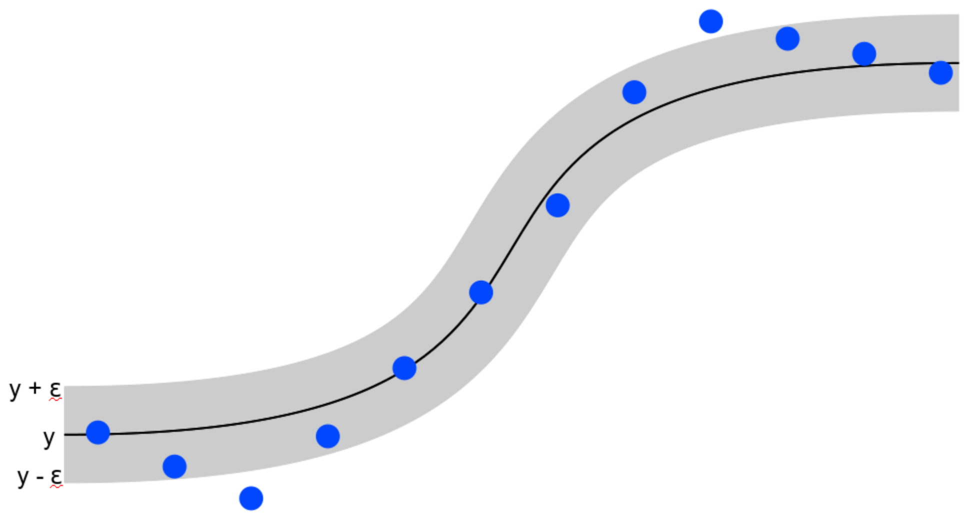


Figure 2.3: Illustration of the ϵ tube.

the Euclidean norm ($||\mathbf{w}^2||$) and the error generated by the estimation process are minimized, as shown in Equation 2.17.

$$J(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N} E_\epsilon\left(y(\mathbf{x}_n) - t_n\right)^2 \tag{2.17}$$

This objective is known as the regularized risk. C is the regularization constant, which is used to give significance to the regularized risk (the error generated by E_ϵ). The quadratic error function (the second term of Equation 2.17), is known as the empirical risk. Slack variables ξ and $\hat{\xi}$ are introduced for points outside of the ϵ tube with the following conditions:

$$t_n \le y(\mathbf{x}_n) + \epsilon + \xi_n \tag{2.18}$$

$$t_n \ge y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n \tag{2.19}$$

Then, Equation 2.16 can be rewritten as:

$$J(\mathbf{w}, b) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) \tag{2.20}$$

As with SVMs for classification, this minimization can be solved using Lagrange multipliers and optimizing the Lagrangian; resulting in a similar dual formulation (Bishop, 2006).

The objective is to minimize both the norm of the weight vector and the empirical risk. The regularization constant C governs the amount of trust the model gives the training data. A large value for C will result in a model that will put emphasis on minimizing the slack errors. This would result in a model that would attempt to perfectly fit the training data, but which would not generalize well on unseen data.

2.3.4 Time Series Prediction

A time series is an ordered sequence of observations from the present and past. Time series datasets differ from regular datasets because there is a natural ordering to the observations. Another unique feature of these datasets is that adjacent observations are dependent (Box et al., 2008). Examples of these types of datasets are common, such as the daily closing price of the Dow Jones index, the monthly sales for a car dealership, and the daily electricity use of New York City, to name a few. Observations are typically measured at uniformly spaced intervals, but this is not always the case, due to missing data points or the general nature of the time series.

Exploiting information from observations up to time t to estimate a future value at time $t + l$ is known as time series prediction. A time series prediction problem can be generalized abstractly as shown in equation 2.21.

$$\hat{y}_{t+l} = f\left(y_t,\ y_{t-1},\ y_{t-2}, \ldots,\ y_{t-n}\right) \tag{2.21}$$

For an observed time series y with n points, where t refers to the most recent observation and $t - n$ is the most distant observation, a future value at $t + l$ can be estimated with a function f. Function f is known as the model, which is used to obtain an estimated value $\hat{y}_{t+l}$.

A time series is stationary if the joint probability distribution does not depend on time. In regards to SVM, any information that influences the output and is not encoded into the model is considered noise; it is impossible to exploit all information that influences real-world time series. Prediction of nonlinear time series that are both noisy and non-stationary is considered to be difficult. In many real-world examples of time series, the datasets are noisy and non-stationary. Blood glucose prediction, financial market prediction, and electricity utility load forecasting are examples of noisy and non-stationary time series.

Real-world time series that exhibit these properties require advanced algorithms to model predictions. This motivates the use of techniques such as SVMs and neural networks. SVMs outperform neural networks and other advanced prediction techniques on time series that are non-stationary, noisy, and not defined a-priori; as shown by many publications listed in (Sapankevych and Sankar, 2009). Due to sparse representation of the solution, SVMs tend to have good generalization performance and testing is done much quicker with SVMs than their counterparts. Therefore, SVMs are favored over other advanced techniques.

CGM data can be formulated as a time series, where the current blood glucose reading is y_t, and the future blood glucose value is y_{t+l}. The goal is to learn a function, f, from CGM data such that the predicted value from the model $\hat{y}_{t+l}$ is as close as possible to the actual value y_{t+l}. The function f can be obtained using many different learning models. SVR is explored in this work due to its resilience to noise, good generalization performance, and sparse solution properties. This allows the SVR to learn from a large number of training examples and features within a practical training time. There are also some theoretical twists that are easily applied to the SVR to boost performance for blood glucose prediction, as explained in Chapter 5.

2.4 Evaluation Techniques

This section describes evaluation techniques used to evaluate classification and regression models. Two different types of evaluation were performed. The first is k-fold cross validation, which was used for evaluating models for classification of glycemic variability. The second is walk-forward testing, which was used for evaluating blood glucose prediction models.

2.4.1 K-fold Cross Validation

To identify a superior machine learning method, k-fold cross validation is performed. This is a standard technique for comparing the performances of different methods using small datasets. A formal explanation of k-fold cross validation is given in (Mitchell, 1997). Using the results from k-fold cross validation, a p-value indicating statistical significance can be obtained. Given that this p-value is small enough, we can accept one method as superior to the other. K-fold cross validation partitions the dataset into k folds. Then, all but one fold is used for training, and the fold that is held out is used as a test set. This is done k times, once for each fold. An illustration of this process is given in Figure 2.4. In this case, a development dataset is used for tuning parameters. This dataset is kept

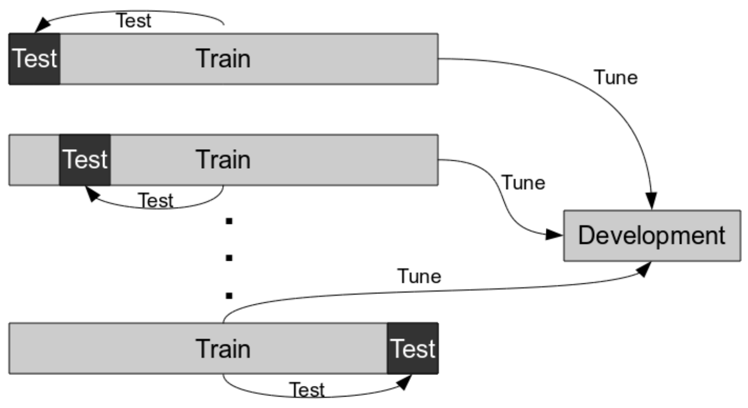


Figure 2.4: Illustration of k-fold cross validation.

separate from the k folds used for training and testing, such that the parameters are not tuned on any test data. For SVM, the regularization constant C and the kernel parameters are tuned. For the MP, the learning rate η and momentum parameter α are tuned.

2.4.2 Walk-forward testing

Walk-forward testing is a commonly used method for assessing the generalization performance of a time series prediction model. For example, this type of testing has been employed for financial time series forecasting in (Refenes et al., 1997; Cao and Tay, 2003; Tay and Cao, 2002a). This method works by using an out-of-sample tuning and testing dataset. The idea is to test the regression model in real time. This is done by partitioning the test dataset, and walking forward through the test set in a certain increment of time, updating each dataset (train, validation, and test) appropriately. Figure 2.5 illustrates this type of testing.

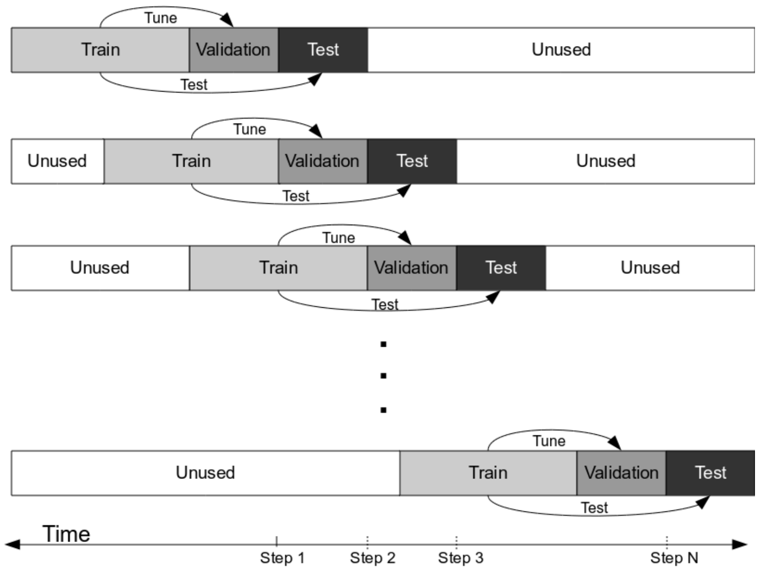


Figure 2.5: Illustration of walk-forward testing.

At each step, the validation set is used to tune the parameters of the SVR. Then, the performance of the test set is calculated and the error measures are recorded. Once testing is complete for that step, the starting and ending times of all datasets are updated by a predefined step size. The data that was previously used for testing becomes part of the validation set and, perhaps, the training set if the validation set is smaller than the test set. The data that was previously used as validation data becomes part of the training dataset. This type of walk-forward testing continues until the test dataset space has been exhausted. The results at each step can be averaged together to get an overall result for the system. Upon completion, the results of the walk-forward testing can be used for statistical significance testing. Statistical significance is used to compare two different learning models for blood glucose prediction.

3 Data Preprocessing

This chapter describes data preprocessing techniques that were carried out on patient data prior to classification and prediction. Features engineered to classify glycemic variability are derived from the values recorded by the CGM sensor. Therefore, noise recorded by the CGM sensor is also encoded in these features. The intuition is that filtering noise before encoding features will improve the discriminative power of these features, resulting in better performance of the classifiers.

Features used to predict blood glucose values depend on CGM data as well as on other data recorded by the patient. As in classification, features derived from CGM data will retain noise. Furthermore, noisy CGM readings can confuse the regression model with unrealistic examples. This motivates preprocessing the CGM data before generating features and examples for prediction. Data recorded by the patient may be missing or invalid, causing incomplete or corrupt features. However, correctly recorded patient data should correspond to changes in blood glucose levels, improving the accuracy of blood glucose prediction models. This motivates the inference of missing data values that were recorded by the patient.

Two forms of data were collected for the 4DSS. These forms are automated data and user-entry data. Automated data consists of CGM data, bolus information, basal information, and fingerstick information. All of this data is recorded automatically by the insulin pump worn by the patient. Automated data is always present and as accurate as the equipment recording the data. User-entry data represents life event information such as exercise, meals, work, and sleep information. This data is collected via a web interface that is filled out by the user. Unlike automated data, user-entry data may be inaccurate or missing. Both forms of data need to undergo noise and/or anomaly detection to obtain accurate examples. For user-entry data, this involves identifying corrupt or missing data, and then removing corrupt records or inferring the missing data. For automated data, this involves smoothing noisy CGM data. The quality of the CGM data is further improved with the use of fingerstick information. This information allows anomalous CGM data points to be detected and corrected.

Anomaly detection is the process of identifying unexpected patterns in data (Chandola et al., 2009). Noise is different from anomalies in that noise comes from inaccuracy in the sensor recording the data. The survey presented in (Chandola et al., 2009) defines 3 different types of anomalies: point anomalies, contextual anomalies, and collective anomalies. Point and collective anomalies are not applicable to CGM data. However, contextual anomalies are quite common for CGM data. These anomalies occur when the CGM sensor becomes uncalibrated. An uncalibrated CGM sensor is corrected when the patient performs a fingerstick. The data points sampled while the CGM sensor is uncalibrated are contextual anomalies. The fingerstick information is used to correct these anomalous readings. Ridge regression and cubic spline smoothing described in Sections 3.2.4 and 3.2.5 attempt to correct these contextual anomalies.

Section 3.1 describes inference techniques that were used for user-entry data. This includes inferring information from two sources: a typical daily schedule specified by the patient and extra information recorded by the insulin pump. Section 3.2 describes techniques used to filter noise from CGM data. Filtering of CGM data was performed for two different tasks: classification of glycemic variability; and prediction of blood glucose values.

3.1 Inferring missing data

At the beginning of each patient's participation in the study, routine information such as wake/sleep times and to/from work times are collected for each day of the week. This information can be used to infer missing data that the patient might have forgotten to record. For example, if the patient forgot to enter the time they went to work on Monday, their daily schedule can be consulted to infer the time the patient was working. However, not all information can be inferred from a daily schedule. For example, patients do not typically eat meals at the same time every day. It is important that the meal information is accurate, as the regression model will learn how carbohydrates affect the patient's blood glucose values.

Meal information is available from two different sources. When the patient does not forget, meal information is recorded by the patient via the web interface. Meal information is also recorded when the patient administers a meal bolus using the Bolus Wizard®. The Bolus Wizard® takes the estimated number of carbohydrates for the meal as input, and recommends a bolus amount to the patient. The estimated number of carbohydrates recorded by the patient is captured in the insulin pump data. This data may be more accurate and consistent than the user-entry meal data. Every patient is advised to use the Bolus Wizard® at the time of the meal; the Bolus Wizard® records the estimated carbohydrates with a time stamp. Considering that the patient uses the Bolus Wizard®at the time of the meal, this time stamp will also represent the time of the meal. On the other hand, the time of the user-entry meals are estimated by the user. These times may be estimated inconsistently, causing confusing examples for the regression model. Therefore, the information recorded by the Bolus Wizard® is used whenever possible.

Inferring missing data from the daily schedule helps deal with missing patient data in respect to sleep and work information. In the case of meal information, daily schedules are not as accurate, and the Bolus Wizard® is used. The patient is expected to use the Bolus Wizard® whenever carbohydrates are consumed, with the exception of correcting for a hypoglycemic episode. Hypoglycemic episodes are recorded by the patient, and while the times are estimated, having this information is better than not having any information at all. Using the estimated carbohydrates from the Bolus Wizard® in combination with the estimated carbohydrates consumed during hypoglycemic episodes gives the most information possible for meal data.

3.2 Filtering Techniques

Smoothing and filtering CGM data has gained interest in the diabetes technology community recently (Bequette, 2010; Sparacino et al., 2008). This is due to the fact that the CGM sensors do not record data with 100% accuracy. The sensors used in this study are known to record values at ±20% of the actual blood glucose level (Mastrototaro et al., 2008). A physician implicitly smooths the values recorded by the CGM sensor when considering the data. An explicit example of how a physician smooths a CGM data plot is shown in Figure 3.1.

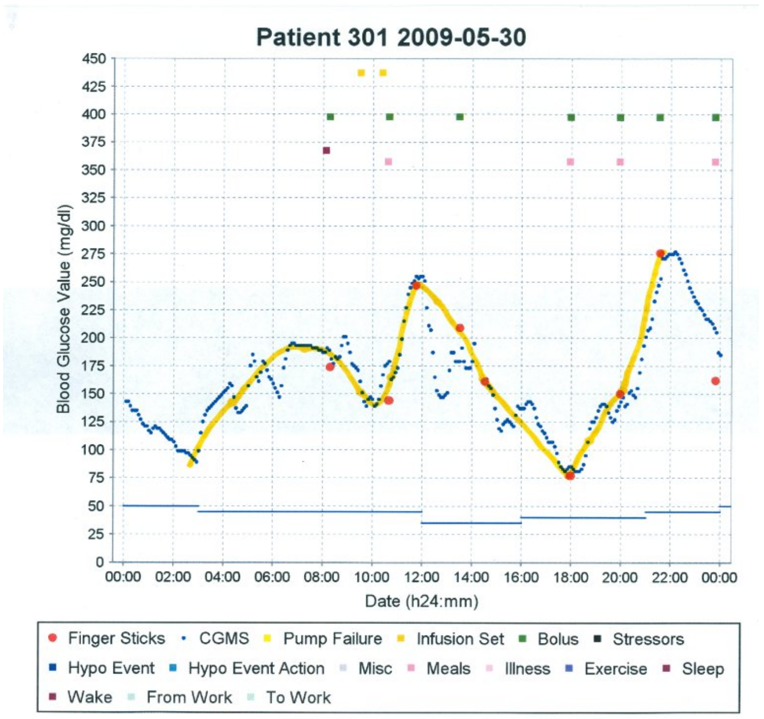


Figure 3.1: CGM data annotated by a physician with a smoothed curve.

An endocrinologist was asked to smooth 10 CGM plots like the one shown in Figure 3.1. In smoothing, the physician gave extra weight to local optima and fingerstick data; fingersticks are known to be more accurate than CGM data. This motivated the use of fingerstick data, along with CGM data, when smoothing. Different techniques were investigated to smooth the CGM data. These techniques included simple moving averages, exponential moving averages, ridge regression, low pass discrete Fourier transform filters, and cubic spline smoothing. The smoothing technique identified by the physicians as matching the implicit smoothing process the closest was cubic spline smoothing.

The original CGM plots annotated by physicians to classify glycemic variability did not contain any fingerstick information. Therefore it was not possible to incorporate fingerstick data while smoothing and the cubic spline smoothing defined in Equation 3.15 was used weighting local optima without fingerstick information. However, fingerstick data was available for blood glucose prediction. Therefore, it was possible to use the cubic spline smoothing defined in Equation 3.15 with local optima and fingerstick information as preferred by physicians.

3.2.1 Simple Moving Average

A moving average filter for CGM data was investigated in depth by (Sparacino et al., 2008), and was found to be inadequate for smoothing CGM data. To verify these findings, this work investigates moving average filters as well. A simple moving average, $\hat{y}_t$, of order k is defined as follows:

$$\hat{y}_t = \frac{1}{k+1} \sum_{i=0}^{k} y_{t-i} \tag{3.1}$$

where t represents an index in the time series, y_t is the data point for which the moving average is calculated, and k is the number of data points before y_t which are included in the calculation.

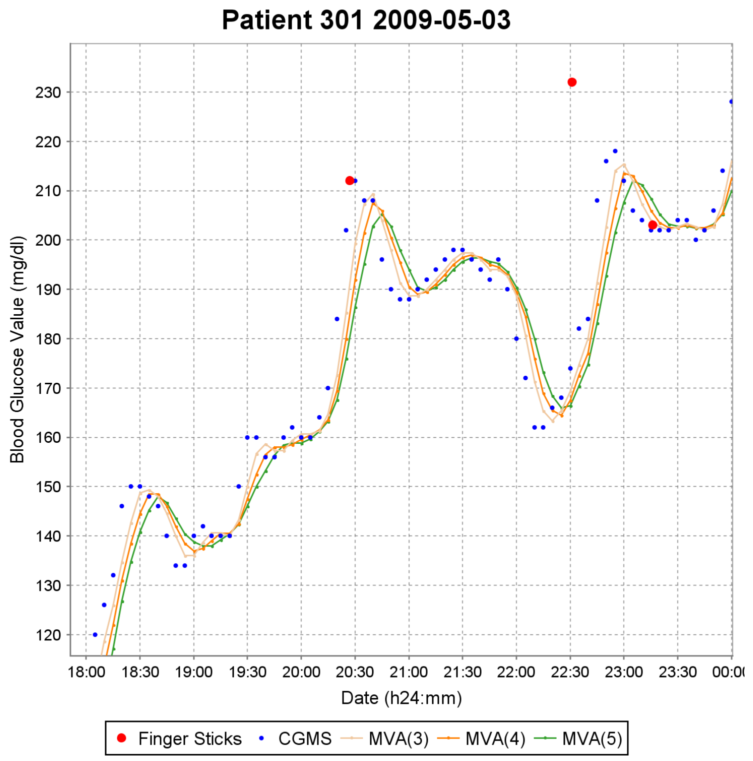


Figure 3.2: Examples of smoothing with moving averages of different orders.

One problem with this type of smoothing is that the moving average tends to lag behind the real blood glucose values. Figure 3.2 shows this type of lag with moving averages of orders 3, 4 and 5. To combat this problem, we came up with a different formulation for the moving average, which could be called the middle moving average. A middle moving average, also denoted by $\hat{y}_t$, of order k is defined as:

$$\hat{y}_t = \frac{1}{2k+1}\left(y_t + \sum_{i=1}^{k} y_{t-i} + \sum_{i=1}^{k} y_{t+i}\right) \tag{3.2}$$

where y_t is the data point for which the moving average is calculated, and k is the number of neighbors on each side of point y_t included in the calculation.

This type of moving average is resistant to the lag noticed with the first moving average. Figure 3.3 shows examples of smoothing with middle moving averages of orders

2, 3 and 4. As shown in this figure, using more points (higher orders) yields less noise, but it also shortens the peaks and nadirs of the original CGM data.

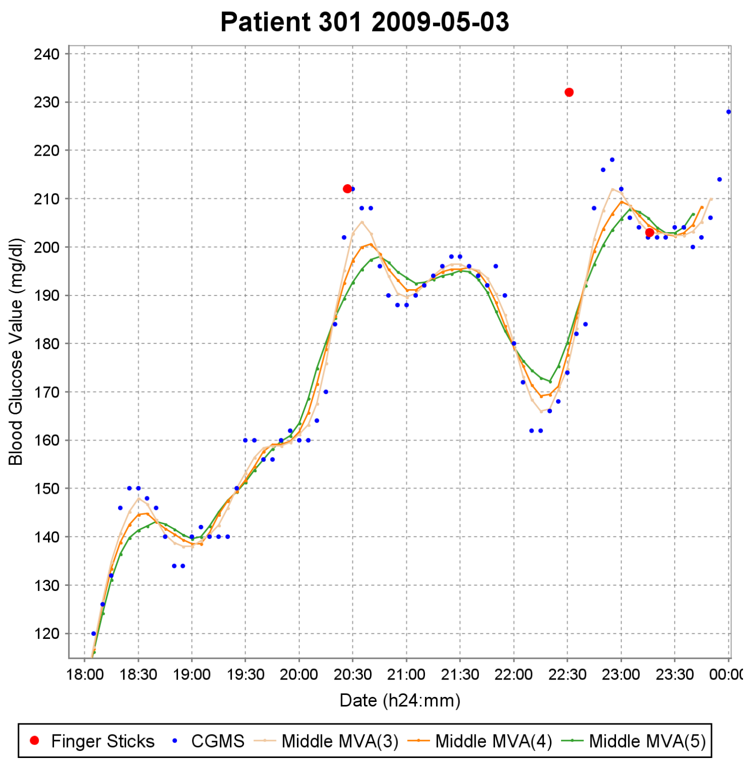


Figure 3.3: Examples of smoothing with middle moving averages of different orders.

3.2.2 Exponential Moving Average

A slight modification to the simple moving average is the exponential moving average. The exponential moving average takes a parameter λ, with values between 0 and 1, that acts as an exponential decay. This type of exponential decay was also investigated in (Sparacino et al., 2008). If λ is 1, then this becomes a simple moving average. An exponential moving average, $\hat{y}_t$, of order k and decay λ is defined as:

$$\hat{y}_t = \frac{\sum_{i=0}^{k} y_{t-i}\lambda^i}{\sum_{i=0}^{k} \lambda^i} \tag{3.3}$$

where y_t is the data point for which the exponential moving average is calculated, k is the number of data points before y_t which are included in the calculation, and λ is the decay parameter.

The exponential moving average also lagged like the simple moving average, motivating the need for a middle exponential moving average. The middle exponential moving average is defined as:

$$\hat{y}_t = \frac{y_t + \sum_{i=1}^{k} y_{t-i}\lambda^i + \sum_{i=0}^{k} y_{t+i}\lambda^i}{1 + \sum_{i=1}^{k} 2\lambda^i} \tag{3.4}$$

where y_t is the data point for which the moving average is calculated, k is the number of neighbors on each side of point y_t included in the calculation, and λ is the decay parameter.

Smoothing with a middle exponential moving average can be used to reduce the influence of distant points. Smoothing with a λ of 0.5 is compared to smoothing with a simple middle moving average in Figure 3.4. A smaller value for λ results in a smoothed graph closer to that of the original CGM data.

Moving averages produce smooth curves. However, moving averages can over-smooth important aspects of data such as true physiological peaks and nadirs. Moving averages can also preserve noise in the CGM data. To reduce noise, it is possible to include the more accurate fingerstick data into the moving average formulations. However, if this information was included, it would be over-smoothed. It would be ideal if the smoothed curve passed through each fingerstick, as is done by the endocrinologist. This is not possible using moving averages, which motivates the use of other smoothing techniques.

3.2.3 Low Pass Discrete Fourier Transform

A Discrete Fourier Transform (DFT) of N CGM measurements yields N complex sinusoidal components (Stanley et al., 1983). This transformation (Equation 3.5) from the

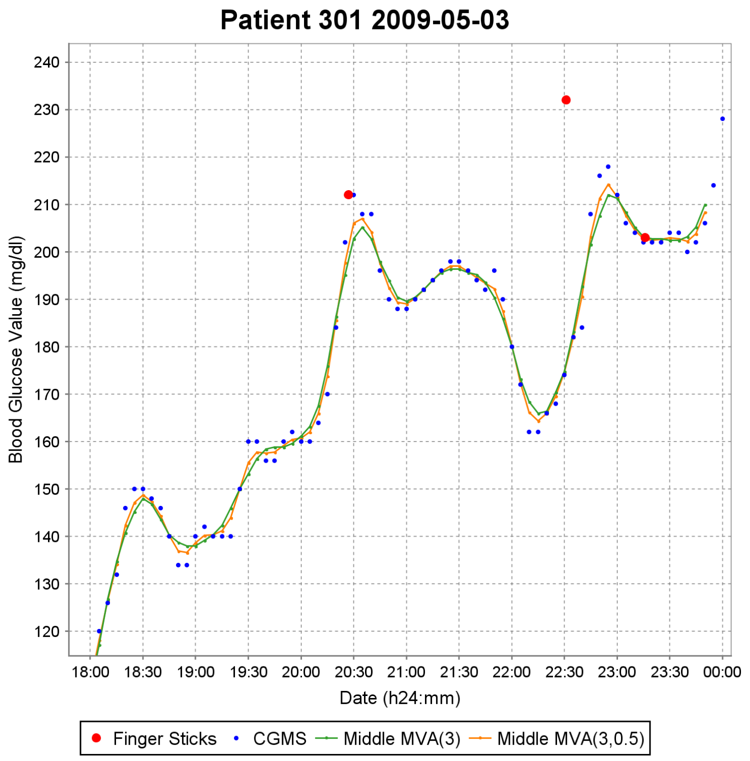


Figure 3.4: The middle exponential moving average compared to a simple middle moving average.

time domain to the frequency domain is helpful because high frequencies in the frequency domain can be discarded as noise. The sequence of complex sinusoidal components in the frequency domain can be transformed to the time domain using the inverse DFT (Equation 3.6). The original CGM measurements can be reconstructed using all of the frequency information when performing the inverse DFT. Equation 3.5 shows how the DFT is computed:

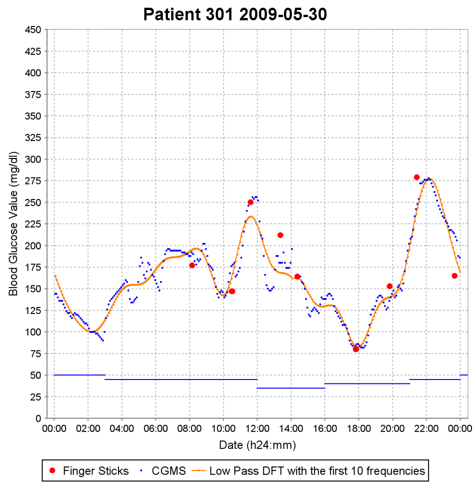
$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(-\frac{2\pi i}{N} kn\right), \quad k = 0, \ldots, N-1 \tag{3.5}$$

where N is the number of CGM measurements, x_n is an individual CGM data point, and i is $\sqrt{-1}$. Equation 3.6 shows how the inverse DFT is computed:

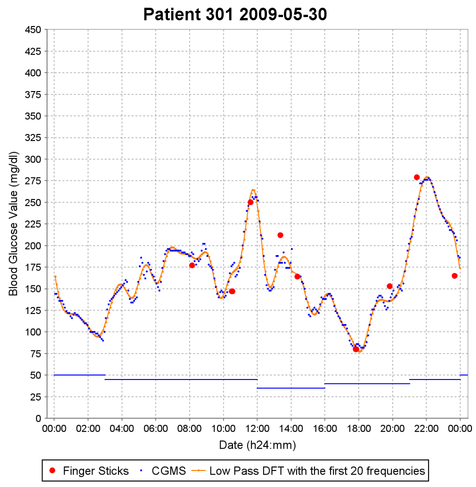
$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k \exp\left(\frac{2\pi i}{N}kn\right), \quad n = 0, \ldots, N-1 \tag{3.6}$$

where N is the number of sinusoidal components, X_k is the frequency obtained from Equation 3.5, and i is $\sqrt{-1}$.

A low pass DFT zeros out all frequencies above a certain threshold. This filter is useful as the lowest frequencies contain a majority of the information about the original signal. This is because each frequency, $X(k)$, represents a sinusoidal component of k/N cycles per day. High frequencies correspond to sinusoidal components with short periods. Noise from the CGM data appears in frequencies with short periods. By ignoring high frequencies, this noise is ignored while a majority of the original CGM data is preserved. Once these frequencies have been zeroed out, the inverse transform is computed, resulting in a smoothed curve. Figures 3.5(a) and 3.5(b) show examples of this type of smoothing.



(a) Low pass DFT with all but the first 10 frequencies discarded.



(b) Low pass DFT with all but the first 20 frequencies discarded.

Figure 3.5: Examples of the low pass DFT filter.

The low pass DFT filter ignores noise from the original CGM data when performing the inverse transform. On the other hand the moving averages include noise in the averaged data values. This aspect of the low pass DFT filter makes it superior to the moving averages for smoothing CGM data. However, the disadvantage of the DFT filter is the lack of an obvious way to integrate fingerstick information.

3.2.4 Ridge Regression

Ridge regression is a statistical approach for solving non-linear least squares problems, first appearing formally in (Hoerl and Kennard, 1970). The basic idea is that any signal containing K points can be fit exactly with a polynomial, $y\,(x \mid \mathbf{w})$, of degree K. Equation 3.7 shows the formula for this polynomial:

$$y\,(x \mid \mathbf{w}) = \sum_{i=0}^{K} w_i\, x^k \tag{3.7}$$

where $\mathbf{w} \equiv (w_0, w_1, \ldots, w_K)^\top$, and x^k is x raised to the power of k. The values of $\mathbf{w}$ can be realized by minimizing the mean square error distance between $y\,(x \mid \mathbf{w})$ and the original points. Forcing a polynomial line to fit every single point from the original signal results in large coefficients, and a volatile curve. Ridge regression addresses this problem by minimizing a trade-off between the mean square error and the norm of the weight vector squared. Equation 3.8 shows how the objective $L\,(\mathbf{w})$ is minimized with this trade-off:

$$L\,(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} (y\,(x_n \mid \mathbf{w}) - t_n)^2 + \frac{\lambda}{2}\, ||\mathbf{w}||^2 \tag{3.8}$$

where $\frac{1}{2}$ is included for mathematical convenience, N is the number of CGM points, t_n is the original value of the nth CGM point, λ is the ridge parameter, and $||\mathbf{w}||^2 \equiv \mathbf{w}^\top\mathbf{w} \equiv w_0^2 + w_1^2 + \ldots + w_K^2$. The ridge parameter controls the aforementioned trade-off. An appealing aspect of ridge regression is that Equation 3.8 can be modified to weigh fingerstick examples more heavily, as shown in Equation 3.9.

$$L(\mathbf{w}) = \sum_{n=1}^{N} \frac{C_n}{2} (y(x_n \mid \mathbf{w}) - t_n)^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{3.9}$$

where C_n is defined as:

$$C_n = \begin{cases} P, & \text{if n is a fingerstick example} \\ 1, & \text{otherwise} \end{cases} \tag{3.10}$$

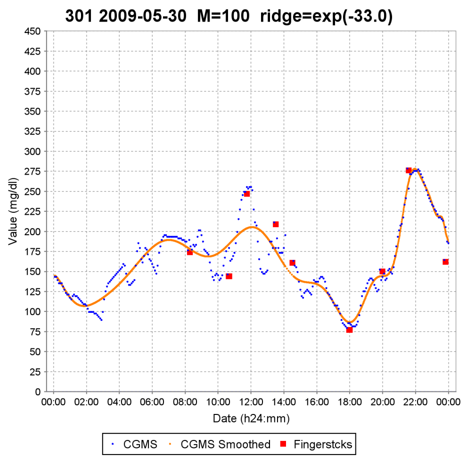
where P is any positive integer greater than one. The solution is found by setting the gradient of the objective function (Equation 3.9) with respect to **w** equal to zero, as shown in equation 3.11.

$$\mathbf{w} = X^\top X (X^\top Y + \lambda I)^{-1} \tag{3.11}$$

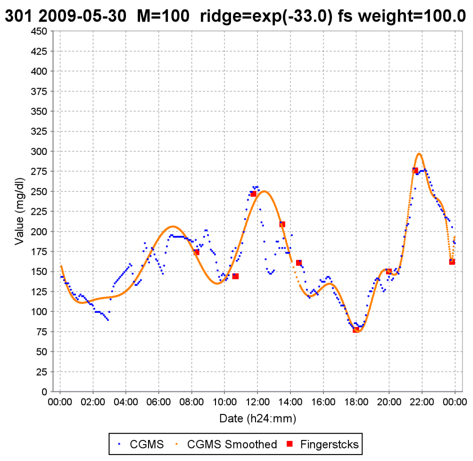
where X is the Vandermone matrix of the form $X \equiv (\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_N})^\top$ where $\mathbf{x_i} \equiv \left(1, x_i, x_i^2, \ldots, x_i^K\right)^\top$, $Y \equiv (t_1, t_2, \ldots t_N)^\top$, and I is the identity matrix. For a weighted solution, each $\mathbf{x_i}$ and t_i are scaled by $\sqrt{C_i}$. Thus, solving for **w** can be written as a system of linear equations with or without weighting. A full derivation can be found in (Bishop, 2006).

Figures 3.6(a) to 3.6(d) show examples of this type of ridge regression. Figure 3.6(a) illustrates ridge regression without fingerstick data. Figure 3.6(b), demonstrates the advantages of weighting fingersticks for this type of smoothing. Figure 3.6(c), shows how the ridge regression reacts to a different weighting of fingersticks. When the CGM curve becomes more complex, a higher degree polynomial can be used to obtain a more complex line, as shown in Figure 3.6(d).

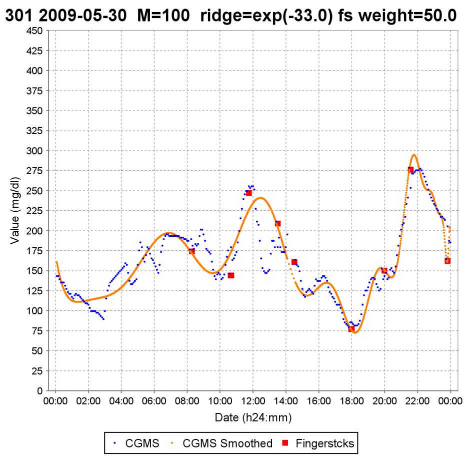
The biggest drawback to ridge regression is that, with a large order, ridge regression becomes numerically unstable with a monomial basis function. Unfortunately this yields inconsistent results, making it inappropriate for smoothing CGM data where a larger order is needed due to the complexity of the CGM plot.



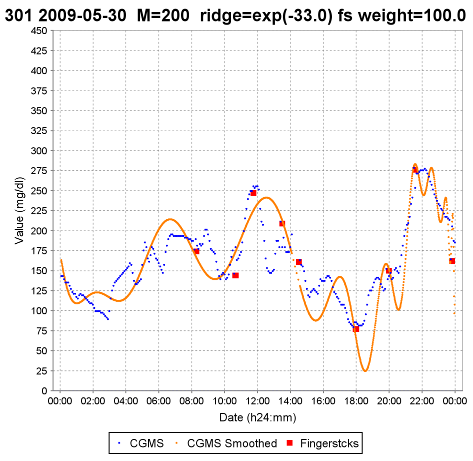
(a) Ridge regression with order = 100, ridge = $\exp^{-33}$, and fingerstick weight = 0.



(b) Ridge regression with order = 100, ridge = $\exp^{-33}$, and fingerstick weight = 100.



(c) Ridge Regression with order = 100, ridge = $\exp^{-33}$, and fingerstick weight = 50.



(d) Ridge regression with order = 200, ridge = $\exp^{-33}$, and fingerstick weight = 100.

Figure 3.6: Examples of ridge regression.

3.2.5 Cubic Spline Smoothing

Cubic spline smoothing is a regularized cubic spline interpolation, described in great detail in (Pollock, 1993). Given a set of n points (x_i, y_i), the objective of the spline interpolation is to connect adjacent points using cubic functions S_i. Each spline function

$S_i(x)$ is computed as per Equation 3.12, where x takes values in the range x_i to x_{i+1}, with the condition that adjacent functions S_i and S_{i-1} give the same value for point (x_i, y_i).

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad \forall x \in [x_i, x_{i+1}] \tag{3.12}$$

where a_i, b_i, c_i, and d_i are the coefficients for the cubic spline function S_i.

While this type of interpolation produces smooth curves, noise from the CGM data is preserved. This noise can be smoothed away by allowing the spline functions S_i to deviate from the original data points, while requiring that the first and second derivatives of the spline function are continuous at all points (x_i, y_i). This is achieved by minimizing a trade-off between the error of the spline functions and the second derivative of the spline functions. The second derivative of Equation 3.12 is given in Equation 3.13.

$$S_i''(x) = 6a_i(x - x_i) + 2b_i \tag{3.13}$$

For spline interpolation, the coefficients a_i and b_i grow large. Large coefficients result in volatile convex/concave curves between data points. By forcing the spline to minimize the area under the second derivative, the smoothed curve is less volatile, as it does not attempt to pass through all of the original data points. The estimated plot L is found with this minimization as shown in Equation 3.14:

$$L = \sum_{i=0}^{n} (Y_i - S_i)^2 + \frac{\lambda}{x_n - x_0} \int_{x_0}^{x_n} S''(x)^2 dx \tag{3.14}$$

where S_i is the cubic spline function, Y_i is the original CGM data point, λ is the ridge parameter, $S''(x)$ is the second derivative of the spline function S_i, and x_0, and x_n are the first and last CGM points which are smoothed.

The first term corresponds to the total square error of the original points and the points produced by the spline functions S_i. The second term corresponds to the average curvature of the splines. By minimizing the curvature, noise is smoothed away from the CGM graph. When $\lambda = 0$, Equation 3.14 results in an interpolating spline that passes through every data point. At the other extreme, a large value for λ results in a straight line. To avoid issues with missing data points and different starting times of the first CGM reading, the area under the second derivative is divided by the range of the X values. The implementation used to compute the smoothed cubic spline follows pseudo-code found in (Pollock, 1993).

As with ridge regression, an advantage to cubic spline smoothing is that weights can be given to specific examples. This allows for extra weighting of fingersticks. Furthermore, it is a concern of the physicians that true peaks and nadirs in the CGM data are not smoothed away, as reflected in Figure 3.1. Therefore, CGM data points corresponding to local minima or maxima should be given extra weight, so that the smoothed curve will pass through these points as well as fingersticks. This is done by modifying Equation 3.14, as shown in Equation 3.15.

$$L = \frac{\sum_{i=0}^{n} w_i \left(Y_i - S_i\right)^2}{\sum_{i=0}^{n} w_i} + \frac{\lambda}{x_n - x_0} \int_{x_0}^{x_n} S''(x)^2 dx \tag{3.15}$$

where S_i is the cubic spline function, Y_i is the original CGM data point, λ is the ridge parameter, $S''(x)$ is the second derivative of the spline function S_i, x_0, and x_n are the first and last CGM points which are smoothed, and w_i is the weight associated with the ith point. Each w_i is defined as:

$$w_i = \begin{cases} P, & \text{if } (x_i, y_i) \text{ is a fingerstick or local optima} \\ 1, & \text{otherwise} \end{cases} \tag{3.16}$$

where P is any positive integer greater than one. The square error should be divided by the sum of the weight vector to achieve consistent results with the spline smoothing. This is because the patients do not take the same number of fingersticks every day nor is the number of local optima constant. Local optima are determined automatically by examining a window of time around each CGM point. If the point is the maximum or minimum value out of all points in the window, then that point is considered to be a local optimum. For this work, all CGM points 90 minutes before and after the CGM point in question are included in this check.

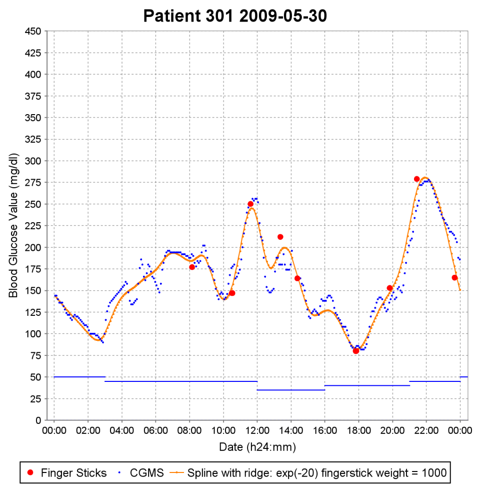
Weighting of local optima and fingersticks becomes problematic when these two events occur close to each other. We propose a technique for dealing with these cases. The weight of a local optimum depends on the distance of the closest fingerstick. If there is not a fingerstick within one hour before or after the local optimum, a weight of k is used. Otherwise, the weight of the local optimum depends on two distances, x_{FC} and y_{FC}.

The distance x_{FC} is the absolute difference in minutes between the fingerstick time and the local optimum time. The distance y_{FC} is the absolute difference in mg/dl between the fingerstick reading and the closest CGM reading. The objective is to formulate these two distances such that the local optimum weight shrinks as the value of x_{FC} shrinks or the value of y_{FC} grows. A large value for y_{FC} implies that the CGM sensor is uncalibrated with respect to the fingerstick, and more emphasis is given to the fingerstick value. A small value for x_{FC} implies that the fingerstick is close to the local optimum, and more emphasis is given to the fingerstick value. This type of formulation is defined in Equation 3.17:

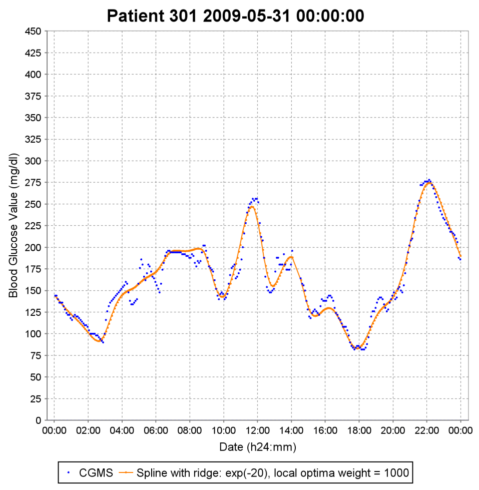
$$w_i = P - \left(a_p\, y_{FC}^{b_p}\,(60 - x_{FC})^{c_p}\right) \tag{3.17}$$

where a_p, b_p, and c_p are positive real numbers greater than 0, P is the weight value used in equation 3.16, x_{FC} is the distance between the closest fingerstick and local optimum in minutes, and y_{FC} is the distance between the fingerstick and the closest CGM reading. A constant of 60 is used as it is the maximum value for x_{FC}; when this maximum is reached, the resulting equation only depends on y_{FC}.

An example of cubic spline smoothing with ridge $\exp^{-20}$, $P = 1000$, and $a_p = b_p = c_p = 1$ is shown in Figure 3.7(a). An example of cubic spline smoothing without fingerstick information, with $\lambda = \exp^{-20}$, $P = 1000$ for local optima, and $a_p = b_p = c_p = 1$ is shown in Figure 3.7(b). Due to the ability to use fingerstick and local optimum weighting, along with consistent smoothed curves across different patients, the physicians identified cubic spline smoothing as the best CGM smoothing technique.



(a) Cubic spline smoothing with ridge $\exp^{-20}$, $P = 1000$, and $a_p = b_p = c_p = 1$.



(b) Example of cubic spline smoothing ignoring fingersticks with ridge $\exp^{-20}$ and $P = 1000$.

Figure 3.7: Examples of cubic spline smoothing.

4 Classification of Glycemic Variability

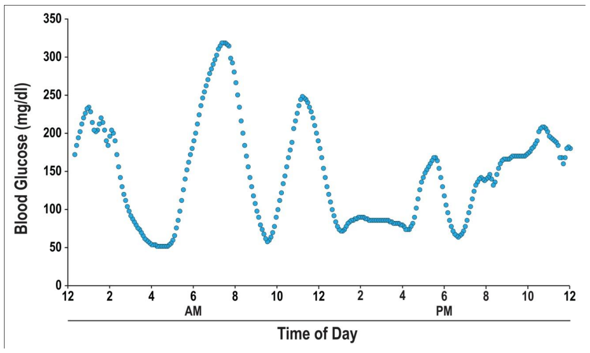
This chapter describes work in classifying daily CGM plots as having excessive or non-excessive glycemic variability. Much of this chapter has been submitted, verbatim, to the International Conference on Machine Learning and Applications (ICMLA) 2011 (Wiley et al., 2011).

A primary concern of diabetes management is blood glucose control. This is because there is no known cure for diabetes; however, diabetes can be treated and managed with blood glucose control. Good blood glucose control can delay or prevent serious diabetic complications, including heart disease, kidney failure, blindness, and strokes (The Diabetes Control and Complications Trial Research Group, 1993). Recent work has shown that excessive glycemic variability is a significant aspect of blood glucose control that contributes to diabetic complications (Ceriello and Ihnat, 2010; Hirsch and Brownlee, 2005; Kilpatrick et al., 2010; Kilpatrick et al., 2006; Monnier and Colette, 2008; Monnier et al., 2006).

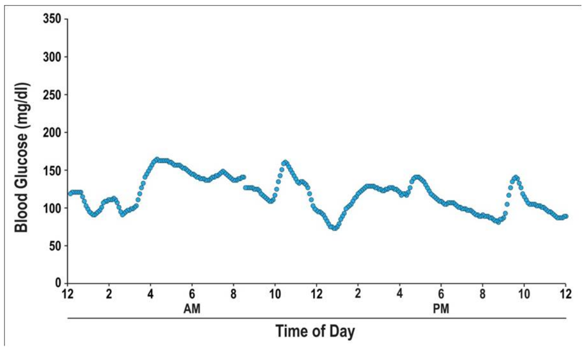
In spite of this, patients are not currently screened for excessive glycemic variability in clinical practice. This is due, in part, to there being no definitive measure for glycemic variability. Even without a definitive measure, diabetes specialists are able to recognize excessive glycemic variability when they see it in daily CGM plots. One way to capture this clinician perception is to build a machine learning classification system, which could act as an automated screen for excessive glycemic variability. Such a system was first explored in (Vernier, 2009). The research described in this chapter extends Vernier's preliminary work, significantly improving system performance.

4.1 Background

The 4DSS was previously extended to automatically detect excessive glycemic variability on daily CGM plots (Vernier, 2009). Blood glucose values were collected using a CGM sensor that recorded sample blood glucose values at 5-minute intervals. Figure 4.1(a) represents a day of excessive variability, while Figure 4.1(b) represents a day of acceptable variability.



(a) Excessive glycemic variability



(b) Acceptable glycemic variability

Figure 4.1: Blood glucose plots obtained through continuous glucose monitoring sensors. Figure to appear in the Journal of Diabetes Science and Technology (Marling et al., 2011).

In the original experiment, two physicians individually classified over 300 days of blood glucose plots as excessively variable or not. The physicians were in agreement on 218 days of the data, which was used to create a training dataset. To evaluate the system, 100 days were chosen at random and presented to the physicians twice. Examples were saved when the physicians were in agreement and consistent with themselves, resulting in 61 examples. Several machine learning classifiers were investigated using the aforementioned datasets with three domain dependent features, described in Section 4.1.1. This was done using the Weka machine learning toolkit (Hall et al., 2009). A naïve Bayes classifier had the best performance, with 85% accuracy.

4.1.1 Domain Dependent Variability Measures

Three different domain dependent metrics were used in the original experiment for classifying glycemic variability (Vernier, 2009). These measures included: Mean Amplitude of Glycemic Excursion (MAGE), Distance Traveled (DT), and Excursion Frequency (EF). MAGE was chosen as it is the most prominent measure for measuring glycemic variability (Service et al., 1970). The physicians developed two additional measures for classifying glycemic variability along with MAGE (Marling et al., 2011). These measures are DT and EF. The purpose of these two measures is to supplement the information provided by MAGE, as MAGE by itself is insufficient for classifying glycemic variability.

4.1.1.1 MAGE

The first measurement for glycemic variability was MAGE (Service et al., 1970), which is still considered to be the "gold standard" for measuring glycemic variability (Robard, 2009; Bolli, 2006; Monnier and Colette, 2008). MAGE is computed as follows. First, the standard deviation for the daily plot of CGM data is computed. Next, each blood glucose excursion which exceeds the standard deviation is detected. The heights of these detected excursions are then averaged together. Depending on which type of excursion occurs first, only peak-to-nadir or nadir-to-peak excursions are included in the calculation.

Figure 4.2 shows a twenty-four hour blood glucose plot with the MAGE calculation. The standard deviation in this case is 63. The first blood glucose excursion which exceeds the standard deviation is a peak-to-nadir excursion starting at 1 am. Therefore, only peak-to-nadir excursions are included. The final value for the MAGE calculation is $\frac{72+126+236}{3} = 144.6$.

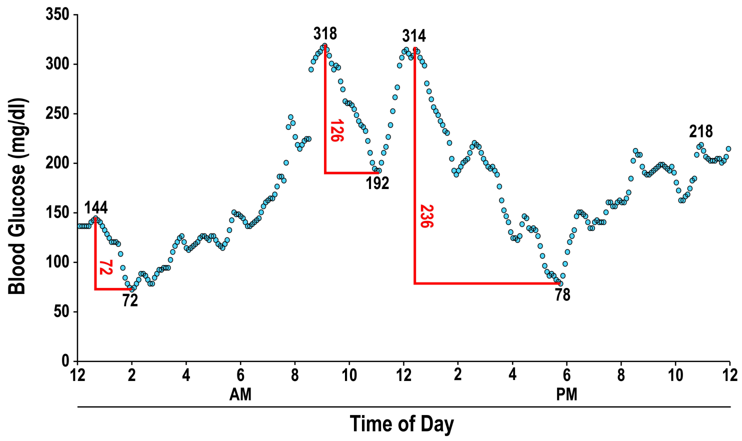


Figure 4.2: Calculation of MAGE for an actual patient's daily blood glucose plot. Figure to appear in the Journal of Diabetes Science and Technology (Marling et al., 2011).

4.1.1.2 Distance Traveled

The DT metric quantifies the total change in blood glucose levels for a given day. This is done by summing the absolute values of the differences between consecutive blood glucose samples. The intuition for this measure is that an excessively variable day will have a greater Distance Traveled than a day exhibiting acceptable glycemic variability. Figure 4.3 shows how DT is calculated.

4.1.1.3 Excursion Frequency

This measure computes the total number of blood glucose excursions exceeding 75 mg/dl that leave the normal range. The 75 mg/dl Excursion Frequency measurement was created to supplement the information from MAGE and DT. Unlike MAGE, it considers

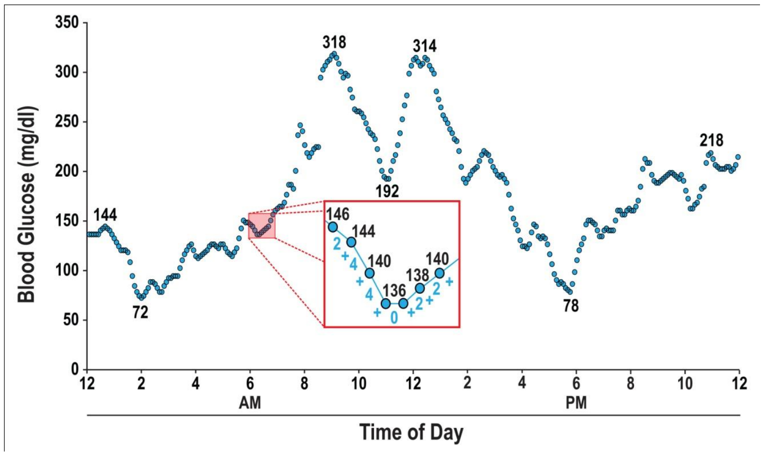


Figure 4.3: Calculation of DT for an actual patient's daily blood glucose plot. Figure to appear in the Journal of Diabetes Science and Technology (Marling et al., 2011).

both nadir-to-peak and peak-to-nadir excursions. Unlike DT, it does not consider fluctuations that occur within the normal range. Figure 4.4 shows how EF is calculated.

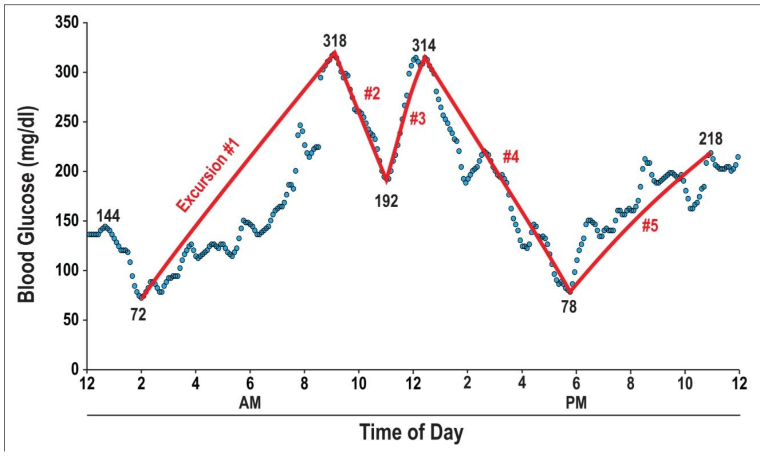


Figure 4.4: Calculation of EF for an actual patient's daily blood glucose plot. Figure to appear in the Journal of Diabetes Science and Technology (Marling et al., 2011).

4.2 Enhancements

Three different enhancements were investigated to improve the classification of glycemic variability. The first was to smooth the CGM data to eliminate noise, as explained in Chapter 3. CGM data was smoothed via cubic spline smoothing with extra weight for local optima. The second enhancement was to develop a rich set of pattern recognition features that capture aspects of glycemic variability not captured by MAGE, DT, and EF. Optimal subsets of these pattern recognition features in conjunction with the three domain dependent features were obtained using feature selection algorithms on a development dataset. These subsets were obtained both independent of, and with the guidance of, the machine learning algorithm. The last approach was to train and evaluate Support Vector Machines (SVM) and Multilayer Perceptrons (MP), both of which are machine learning algorithms known to obtain state-of-the-art generalization performance in several domains. These algorithms were then compared to the original naïve Bayes (NB) approach.

4.3 Feature Engineering

Numerous pattern recognition features were added to the original set of domain dependent features to improve the discriminative performance of the machine learning algorithms. A summary of the features investigated in this chapter is found in Table 4.1. Features $MAGE$, EF, DT are defined in Section 4.1.1. The following subsections describe each of these pattern recognition features in further detail. A description of the feature selection methods is found in Section 4.4.

Table 4.1: Features that were investigated for classification of glycemic variability.

Feature	Description
$MAGE$	Mean Amplitude of Glycemic Excursion
EF	Excursion Frequency
DT	Distance Traveled
σ	Standard Deviation
AUC	Area Under the Curve
μ_{pq}	2-Dimensional central moments of order $p + q \leq 3$
ϵ	Eccentricity
FF_i	Amplitudes of low DFT frequencies for $1 \leq i \leq 24$
RR	Roundness Ratio
BE	Bending Energy
DC_i	Direction Codes, for $1 \leq i \leq 3$

4.3.1 Standard Deviation

This feature is computed as the sample standard deviation over the set of blood glucose measurements. The intuition is that an excessively variable day will have a higher standard deviation than an acceptable day.

4.3.2 Area Under the Curve

This feature is computed as the total area between the CGM graph and a horizontal line corresponding to the minimum blood glucose level measured for that day. Figure 4.5 shows a blood glucose plot in which the shaded region is used to compute the Area Under the Curve (AUC). The intuition is that a larger area correlates with increased glycemic variability.

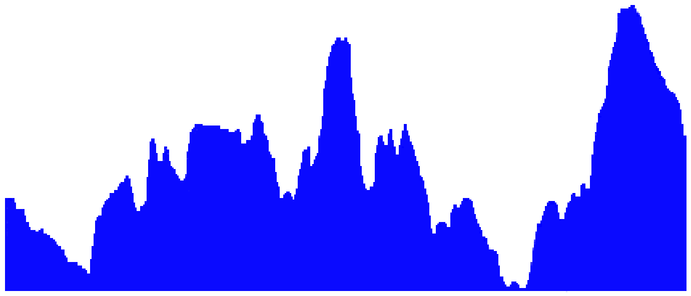


Figure 4.5: Area Under the Curve of an actual patient's CGM plot.

4.3.3 Central Image Moments

Image moments are computed on the pixel intensities of a given image and can be used to derive useful properties of the image such as the total intensity, centroid, orientation, and moment of inertia. To compute the image moments of a CGM graph, we use the 2-dimensional region between the CGM graph and the horizontal line corresponding to the minimum blood glucose level, as shown previously in Figure 4.5. If we use C to denote this region, then the binary intensity $f(x, y)$ at any pixel position can be defined as in Equation 4.1.

$$f(x,y) = \begin{cases} 1, & (x,y) \in C \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

Using these intensity values, image moments of order $p + q$ are calculated as shown in Equation 4.2.

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y) \tag{4.2}$$

For m_{00}, this equation computes the total number of points in the object. If the entire image contains $N \times M$ pixels, then the $N \times M$ moments (m_{pq}) of order $p + q$ uniquely determine the image, where $0 \leq p \leq N$ and $0 \leq q \leq M$. The lower order moments can therefore be used to summarize the image. Since glycemic variability does not change when the CGM region is translated, we will be using instead the lower order *central moments*, modified versions of image moments that are translation invariant.

Using m_{00} along with m_{10} and m_{01}, the center of mass of each axis can be computed, which then gives the centroid of the shape $(\bar{x}, \bar{y})$:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}$$

The central image moments are then computed as follows:

$$\mu_{pq} = \sum_{x} \sum_{y} (x - \bar{x})^p (y - \bar{y})^q f(x, y) \tag{4.3}$$

As features for variability detection we use central moments of order up to 3, i.e. μ_{11}, μ_{20}, μ_{02}, μ_{21}, μ_{12}, μ_{30}, and μ_{03}. Moment μ_{00} is excluded, since it is equivalent to the already considered AUC feature.

4.3.4 Eccentricity

Eccentricity is the ratio between the maximum and minimum distance from the boundary of the object to its centroid (Theodoridis and Koutroumbas, 2009). Eccentricity conveys how much the shape of an object deviates from being circular, or equivalently, how elongated the object is. Eccentricity can be computed as shown in Equation 4.4, using central image moments:

$$\epsilon = \frac{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}}{\mu_{00}} \tag{4.4}$$

The intuition behind using this feature is that an acceptable day is expected to be more elongated than an excessively variable day, and consequently will have a higher eccentricity.

4.3.5 Discrete Fourier Transform Amplitudes

Using the Discrete Fourier Transform (DFT), the sequence $\{y_i\}$ of n CGM measurements in the time domain is transformed into a sequence $\{Y_k\}$ of n complex sinusoidal components in the frequency domain, as shown in Equation 4.5 below (Stanley et al., 1983), where $\exp^{\left(-\frac{2\pi j}{n}\right)}$ is the n-th complex root of unity:

$$Y_k = \sum_{i=0}^{n-1} y_i \exp^{\left(-\frac{2\pi j}{n}ki\right)}, \quad \text{where } k = 0, 1, \ldots, n-1 \tag{4.5}$$

The corresponding time points $\{x_i\}$ are sampled every 5 minutes for an entire day, resulting in $n = 288$ samples $\{y_i\}$. The complex numbers $\{Y_k\}$ represent the magnitude and the phase of the sinusoidal components of the sampled input function $\{y_i\}$. A particular Y_k corresponds to a sinusoidal component with frequency k/n cycles per day. Like the image moments, the DFT uniquely determines the original signal; therefore, we can use the lower frequencies to summarize the CGM graph. Since very rapid fluctuations are indicative of noise, we use as features the magnitudes of the first 24 components, i.e. $\{||Y_k||, 1 \leq k \leq 24\}$. By ignoring the very first component Y_0 (a real number), we make the DFT feature set translation invariant.

4.3.6 Roundness Ratio

This feature is a ratio between the perimeter of the CGM graph squared and its area. If $\{p_i = (x_i, y_i)\}$ is a sequence of n CGM points, then the perimeter and the roundness ratio are defined as in Equation 4.6 below:

$$RR = \frac{P^2}{4\pi\mu_{00}}, \quad \text{where } P = \sum_{i=1}^{n-1} ||p_{i+1} - p_i|| \tag{4.6}$$

In the general case of 2D objects, this feature will take the value of 1 for a perfect circle, and larger values as the objects deviate more from a circular shape. If an acceptable day resembles a rectangle and an excessively variable day resembles a similar rectangle that is much more jagged (a similar area with a larger perimeter), then the roundness ratio of the excessively variable day will be larger than that of the acceptable day. Due to its dependence on the perimeter, smoothing CGM data is expected to enhance the discriminative power of this feature.

4.3.7 Bending Energy

Bending energy computes the average curvature of the CGM graph $\{(x_i, y_i)\}$, as shown in Equation 4.7 below, in which P refers to the perimeter.

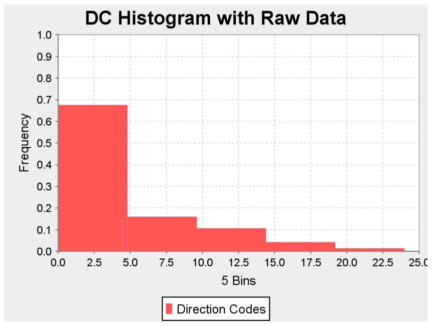
$$BE = \frac{1}{P}\sum_{i=1}^{n-2}(\theta_{i+1} - \theta_i)^2, \quad \text{where } \theta_i = \arctan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) \tag{4.7}$$

Due to larger and more frequent fluctuations in blood glucose levels, an excessively variable day should have a higher bending energy than a day with acceptable variability. Smoothing CGM data is expected to improve the discriminative power of this feature because the angles between consecutive points become less sensitive to random noise.

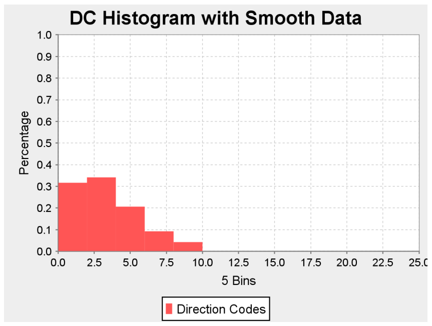
4.3.8 Direction Codes

A direction code (DC) is the absolute difference between the values of two consecutive blood glucose readings. Consequently, a CGM plot with n blood glucose measurements has $n - 1$ direction codes. Direction codes for the entire day are placed into bins, depending on their value. A bin b_i is defined by a minimum DC value low_i and a maximum DC value $high_i$, i.e. $b_i = [low_i, high_i)$. If c_i is the total number of direction codes falling into bin b_i, then the corresponding direction code feature is defined as $DC_i = c_i/(n - 1)$. The bins used to define the DC features for this application are $b_1 = [0, 3)$, $b_2 = [3, 6)$, and $b_3 = [6, 9)$. We arrived at this particular set of bins by analyzing histograms of direction codes on the CGM data.

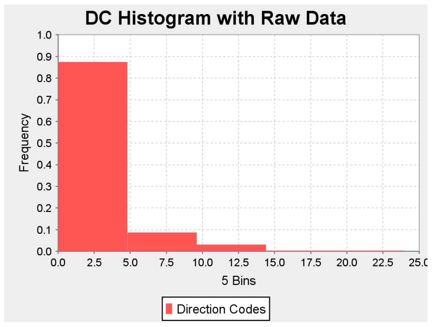
Figures 4.6(a) to 4.6(d) present different situations used to determine the appropriate width and number of bins. Figure 4.6(a) shows the histogram of direction codes on raw CGM data for the excessively variable day presented in Figure 3.1. Figure 4.6(c) shows a



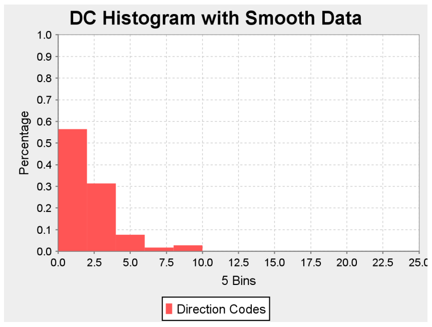
(a) Excessively variable day with raw data.



(b) Excessively variable day with smoothed data.



(c) Acceptable day with raw data.



(d) Acceptable day with smoothed data.

Figure 4.6: Histograms of direction codes.

histogram for a different day with acceptable variability. The width of each bin in these two histograms is 5 mg/dl. When comparing Figure 4.6(a) with Figure 4.6(c), there is an obvious shift of distribution into the first bin, indicating fewer 5-minute blood glucose spikes on the acceptable day. This behavior was consistent across different days of data.

Figures 4.6(b) and 4.6(d) present histograms of direction codes on smoothed data, for the same days used in Figures 4.6(a) and 4.6(c), respectively. The width of each bin in these two histograms is smaller, at 2 mg/dl. The shift in distributions to the first bin is even more obvious in the case of smoothed data. Based on the analysis of these histograms, a bin width of 3 mg/dl was chosen. With a limit of 9 mg/dl these bins would only introduce three DC features, adding finer granularity to the raw data histograms and capturing most of the distribution variation in the smoothed data histograms. Using the three bins $b_1 = [0, 3)$, $b_2 = [3, 6)$, and $b_3 = [6, 9)$ means that direction codes with value greater than 9 are ignored (these direction codes are still counted in the total number $n - 1$).

4.4 Feature Selection

Automatic feature selection of the features defined in Table 4.1 was performed on the development dataset for both raw and smooth CGM data. Two different filter methods plus one wrapper method were investigated. The filter methods were based on the Pearson's Correlation Coefficient (PCC) and Welch's *t*-test. A wrapper approach was investigated using greedy backward elimination with SVM and MP as evaluators.

4.4.1 Filter Methods

Features from Table 4.1 were ranked using the PCC and Welch's *t*-test. Features below a threshold were filtered out based on these rankings. Table 4.2 shows features ranked by PCC and Table 4.3 shows features ranked by Welch's *t*-test.

4.4.1.1 Ranking with PCC

The PCC was used to compute the correlation between each feature and the output label. For feature X and label Y, the linear dependence between the two, $\rho(X, Y)$ is calculated as shown in Equation 4.8:

$$\rho\left(X, Y\right) = \frac{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2}} \tag{4.8}$$

where X is the feature, $\bar{X}$ is the sample mean of feature X, Y is the output label, and $\bar{Y}$ is the sample mean of the output label. The PCC is the covariance between the feature and the label divided by the product of their standard deviations. Using the development dataset to find a threshold, features were filtered from the superset defined in Table 4.1.

No set of features selected by the PCC performed better than the original set of features defined in (Vernier, 2009) for the development dataset. Therefore, the PCC results were not used in for the rest of the experiments.

Table 4.2: Feature Ranking using PCC.

PCC Filter (correlation)	
Raw	Smooth
DC_1 (68%)	DC_1 (70%)
DC_3 (66%)	DT (66%)
AUC (65%)	AUC (65%)
DT (62%)	DC_3 (64%)
EF (60%)	EF (61%)
σ (60%)	σ (61%)
$MAGE$ (56%)	DC_2 (52%)
μ_{pq} (52%)	$MAGE$ (51%)
FF_i (50%)	FF_i (50%)
DC_2 (42%)	RR (49%)
ϵ (33%)	BE (38%)
RR (23%)	μ_{pq} (37%)
BE (11%)	ϵ (35%)

4.4.1.2 Ranking with *t*-test

An unpaired, two sampled, unequal variance *t*-test (Welch's *t*-test) was carried out on the features and the output label. The formula for Welch's *t*-test is given in Equation 4.9:

$$t = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{s_x^2}{N_x} \frac{s_y^2}{N_y}}} \tag{4.9}$$

where $\bar{X}$, s_x^2, and N_x are the sample mean, sample variance and sample size for positive examples, and $\bar{Y}$, s_y^2, and N_y are the sample mean, sample variance and sample size for negative examples. Using this *t*-value with the degrees of freedom approximated from the Welch-Satterthwaite equation, p-values can be computed (Welch, 1947). The p-values obtained from this test were used to filter features using the development dataset. Table 4.3 shows the ranking of the features based on these p-values. Features shown in bold represent the subset which obtained the best performance on the development dataset.

4.4.2 Wrapper Methods

Greedy backward elimination was performed 10 times on the development dataset, using all but one of the 10-folds as training data. This was done starting with the feature set defined in Table 4.1. The features which were most common from the results of wrapper selections across the 10 folds were chosen. This was done with both raw and smooth data. The features selected using these wrapper methods are shown in bold in Table 4.3.

4.4.3 Discussion

The results of the *t*-test filter and the backward elimination wrapper are shown in bold in Table 4.3. The four sets of selected features shown in Table 4.3 are quite different from each other. No feature appears in all four sets. Direction Codes, Excursion Frequency, Standard Deviation, and Distance Traveled were the only features that

Table 4.3: Feature Selection using *t*-test and Backward Elimination.

T-test Filter (p-value)		Backward Elimination	
Raw	Smooth	Raw	Smooth
AUC (2×10^{-11})	$\mathbf{DC_1}$ (2×10^{-9})	$\mathbf{DC_1}$	**DT**
EF (3×10^{-10})	**DT** (4×10^{-8})	$\mathbf{FF_1}$ $\mathbf{FF_{12}}$	**RR**
DT(3×10^{-9})	**AUC** (5×10^{-8})	**EF**	$\mathbf{DC_3}$
$\boldsymbol{\sigma}$ ($\mathbf{5\times10^{-9}}$)	$\mathbf{DC_3}$ (1×10^{-7})	$\boldsymbol{\mu_{20}}$	**MAGE**
DC_1 (1×10^{-8})	**EF** (6×10^{-7})	ϵ	$\boldsymbol{\sigma}$
DC_3 (4×10^{-8})	$\boldsymbol{\sigma}$ ($\mathbf{7\times10^{-7}}$)	σ	AUC
$MAGE$ (1×10^{-7})	μ_{pq} (2×10^{-5})	DT	EF
μ_{pq} (2×10^{-7})	DC_2 (3×10^{-5})	$MAGE$	ϵ
DC_2 (1×10^{-3})	$MAGE$ (4×10^{-5})	RR	DC_1
ϵ (2×10^{-3})	FF_i (8×10^{-5})	DC_3	DC_2
FF_i (3×10^{-2})	RR (9×10^{-5})	BE	BE
RR (0.12)	BE (2×10^{-3})	AUC	FF_i
BE (0.30)	ϵ (5×10^{-3})	DC_2	μ_{pq}

appeared in three of the four feature sets. The only features that were not selected in any of the four sets are: Eccentricity and Bending Energy. Although the four feature sets are very different, there is no substantial difference in their performance (Section 4.5.1). This may indicate that the features overlap in terms of the CGM plot information they encode. In the backward elimination setting, the Fourier features were selected only when using raw data. This behavior is consistent with our expectation that smoothing eliminates some of the random noise from CGM data.

4.5 Experimental Evaluation

From the original work in (Vernier, 2009), the best performing algorithm for glycemic variability detection was a naïve Bayes learning algorithm. We now believe that this algorithm is not the most appropriate for solving this problem. The naïve Bayes algorithm assumes features are independent of each other given the class label, which is not the case for our set of features. For example, EF and DT are not independent in the presence of excessive variability, since a larger DT is likely to increase the number of excursions greater than 75 mg/dl. The DC bins are clearly not independent – as the number of data points distributed among the bins is always constant. This means that if the value of one bin grows, the values of the other bins must shrink. This motivated us to explore Multilayer Perceptrons (MP) (Bishop, 1995) and Support Vector Machines (SVM) (Scholkopf, B. and Smola, A.J., 2002; Vapnik, 1995), two learning algorithms that can seamlessly accommodate overlapping features. MPs that are implemented as a backpropagation network with enough hidden nodes can approximate any decision surface (Hornik and White, 1989). Similarly, an SVM with a Gaussian kernel is a flexible learning model, as it can approximate non-linear decision boundaries. SVMs are known to be resilient to overfitting and to have good generalization performance, due to the max-margin criterion used during optimization. Furthermore, while the MP solution may be only a local optimum, the SVM is guaranteed to converge to a global optimum due to the corresponding convex optimization.

The MP and SVM parameters are tuned using a grid search on a separate development dataset – the same data that is also used for feature selection. There are 262 unique examples in the entire glycemic variability dataset, 187 positive and 75 negative. The development dataset is created from 52 randomly chosen examples, 37 positive and 15 negative. The remaining data is used for training and evaluating the models, using 10-fold cross validation, as described in Section 2.4.1. The distribution of the output label in the development set is similar to the label distribution in each of the 10 folds. The overall evaluation process is illustrated in Figure 2.4. The same setting (same folds, and same development data) is used for evaluating all of the systems.

As fully described in Chapter 2, the MP uses the development set to find optimal values for the learning rate and momentum, parameters that control the speed at which the MP corrects itself. The SVM uses the development set to find the best kernel, and then it re-uses the development set to find optimal parameters for the regularization parameter and the kernel parameters. In all tuning experiments, the Gaussian kernel obtained the best performance. The width of the Gaussian kernel was then optimized on the same development data. CGM data was smoothed using Equation 3.14 ignoring fingerstick information with ridge parameter $\lambda = \exp^{-20}$, and local optima weight $P = 1000$. We used the Weka implementation (Hall et al., 2009) for the naïve Bayes model and the Multilayer Perceptron. We used LIBSVM (Chang and Lin, 2011) for the SVM implementation.

4.5.1 Results and Discussion

Classified examples are categorized as true positive (TP), true negative (TN), false positive (FP), or false negative (FN). We report performance using accuracy, sensitivity, and specificity, three error metrics that are commonly used for analyzing the performance of classifiers in the medical domain.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \text{ Sensitivity} = \frac{TP}{TP + FN}, \text{ Specificity} = \frac{TN}{TN + FP}$$

We evaluated the model developed in the preliminary study by training a naïve Bayes classifier on raw data using only the three original features: MAGE, Excursion Frequency, and Distance Traveled. The 10-fold cross validation results are shown in Table 4.4 as NB Raw.

Table 4.5 shows the results of the 10-fold cross validation evaluation for each of the three learning models, using the features automatically selected by the *t*-test filter. This is shown for both raw and smoothed CGM data. Similarly, Table 4.6 shows the performance of the three learning models when the features are automatically selected using greedy backward elimination.

Table 4.4: Preliminary results using 10-fold cross validation.

Model	Accuracy	Sensitivity	Specificity
NB Raw	87.1%	78.3%	90.6%

Table 4.5: Results of 10-fold evaluation using *t*-test filtering of features

Model	Accuracy	Sensitivity	Specificity
NB Raw	87.1%	81.6%	89.3%
NB Smooth	91.9%	91.6%	92.0%
MP Raw	90.0%	83.3%	92.6%
MP Smooth	91.4%	85.0%	94.0%
SVM Raw	89.5%	78.3%	93.3%
SVM Smooth	**92.8%**	88.3%	94.6%

When using automatic feature selection, the best accuracy of 93.8% is obtained by the MP model trained on smooth data, with a feature set selected through greedy backward elimination. Smoothing the data increases the performance consistently when using feature sets selected through the *t*-test, however it degrades the performance for the SVM and NB models in the greedy backward elimination setting. A one sided, paired

Table 4.6: Results of 10-fold evaluation using greedy backward elimination of features

Model	Accuracy	Sensitivity	Specificity
NB Raw	91.9%	88.3%	93.3%
NB Smooth	89.5%	85.0%	91.3%
MP Raw	91.4%	85.0%	94.0%
MP Smooth	**93.8%**	86.6%	96.6%
SVM Raw	92.8%	85.0%	96.0%
SVM Smooth	91.4%	80.0%	96.0%

t-test was performed to investigate the significance in improvement between the two best systems shown in bold in Tables 4.5, 4.6, and the previous NB system shown in Table 4.4. The improvements are significant at levels $p < 0.01$. Figure 4.5.1 shows the receiver operating characteristic (ROC) curves computed for: the previous NB system (Table 4.4), the current best accuracy system (Table 4.6), and the best accuracy system using a *t*-test filter (Table 4.5).

One interesting result is that, in terms of area under the ROC curve, the best system is the SVM trained on smooth data with filtered features, and not the MP with greedy backward elimination that obtained the best accuracy.

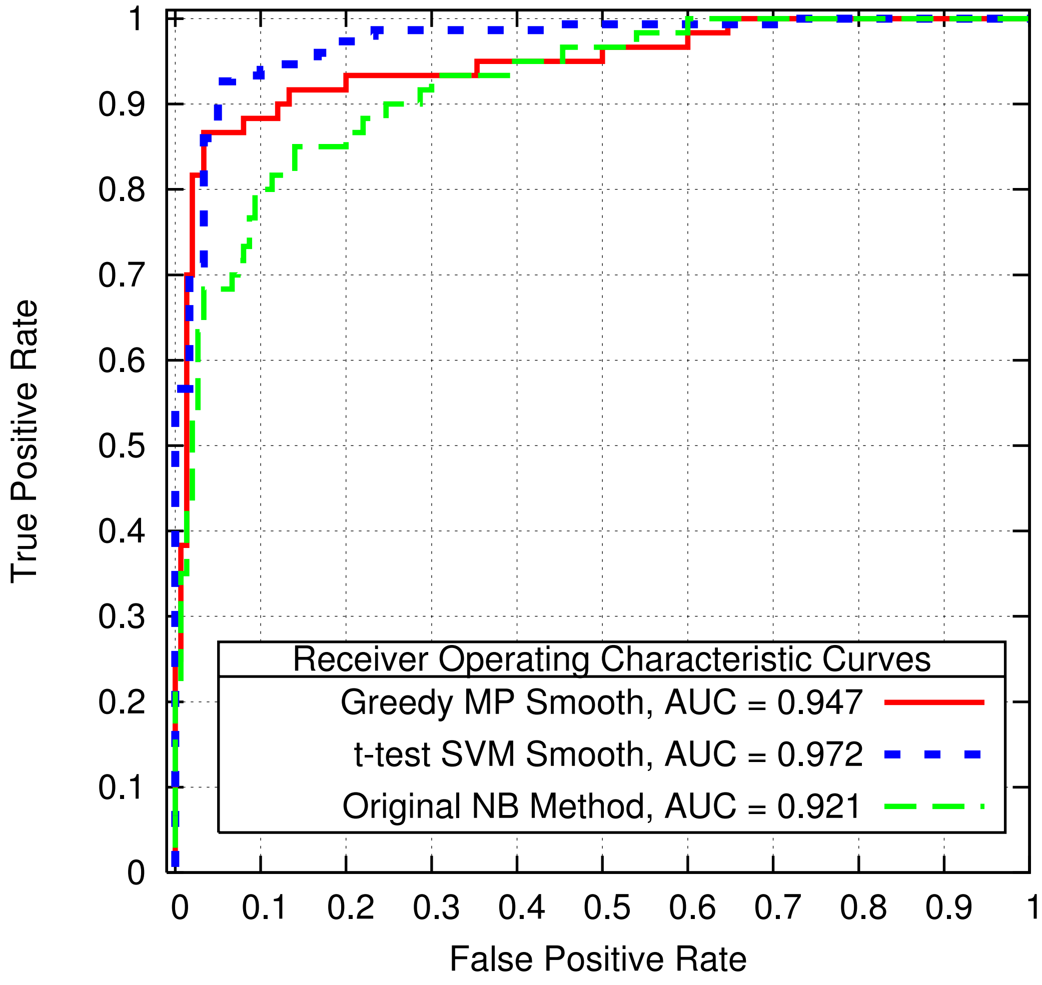


Figure 4.7: Comparison of ROC curves.

5 Blood Glucose Prediction

This chapter presents the results for blood glucose prediction. It describes the features and models used to predict blood glucose values along with the error metrics used to measure the performance of the prediction models.

5.1 Background

As described in Section 2.3.4, using information up to a certain point in time, t, to predict a future value at time $t + l$, is known as time series prediction. In this work, Support Vector Regression (SVR) was used to train a model to predict future blood glucose values at time $t + l$ (Section 2.3.3). A development dataset was used to evaluate sets of features generated by the feature templates. For each feature template, the feature which scored the best Root Mean Square Error (RMSE) on the development data was incorporated into the feature vector utilized by the SVR model. The SVR model was compared to two baselines, a naïve baseline which assumes the blood glucose value is constant, and an AutoRegressive Integrated Moving Average (ARIMA) model. The three models were tested using walk-forward testing as described in Section 2.4.2. The results from each step were then used to generate three domain independent error measures: Mean Absolute Error (MAE), RMSE, and coefficient of determination (R^2). There is also one domain dependent error metric, the Clarke Error Grid Analysis (CEGA). Blood glucose values were smoothed using cubic spline smoothing with extra weight for fingerstick and local optima examples, as described in Section 3.2.5. Furthermore, the SVR model was improved by adapting the C and ϵ parameters for each training example.

5.2 Baselines

Two baselines were used for comparing performance to the SVR model: the t_0 baseline, and the ARIMA model.

5.2.1 t_0 Baseline

The t_0 baseline simply uses the current blood glucose value as the predicted future value, as shown in Equation 5.1:

$$\hat{y}_{t+l} = y_t \tag{5.1}$$

where y_t represents the blood glucose value at time t, and $\hat{y}_{t+l}$ represents the estimated blood glucose value at the future time $t + l$.

5.2.2 Autoregressive Integrated Moving Average Model

ARIMA models are described in detail in (Box et al., 2008). An ARIMA model of order (p, d, q) is useful for describing both stationary and nonstationary time series, which makes these models appropriate for analysis of any time series. The orders p, d, q are integers greater than or equal to zero, which refer to the order of the autoregressive, difference, and moving average components respectively.

A time series $\mathbf{y} = (y_t, y_{t-1}, \ldots, y_1)$ is described by an ARIMA model via Equation 5.2.

$$y_t = \phi_1 y_{t-1} + \ldots + \phi_{d+p} y_{t-d-p} - \beta_1 e_{t-1} - \ldots - \beta_q e_{t-q} + e_t \tag{5.2}$$

where t denotes an index in the time series $\mathbf{y}$, y_t is the value of the time series at index t, $\left(\phi_1, \ldots, \phi_{d+p}\right)$ are the parameters of the autoregressive component, $\left(\beta_1, \ldots, \beta_q\right)$ are the parameters of the moving average component, and e_t denotes a random shock at index t. A random shock is a white noise term that represents the forecast error at index t. If we define the lag operator B, such that $By_t = y_{t-1}$ and $B^i y_t = y_{t-i}$, then, the ARIMA definition can be written in a concise form, as shown in Equation 5.3.

$$\left(1 - \sum_{i=1}^{p} \Phi_i B^i\right) (1 - B)^d\, y_t = \left(1 - \sum_{i=1}^{q} \beta_i B^i\right) e_t \tag{5.3}$$

where $\left(\Phi_1, \ldots, \Phi_p\right)$ are the parameters of the autoregressive component. This equation illustrates the effect of the order d with $(1 - B)^d$. If d is greater than zero, the time series is stationarized (Box et al., 2008). Consider the time series $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. Using a difference of 1, this time series is transformed to $(x_1, x_2, \ldots, x_{n-1})$ where $x_i = y_{i+1} - y_i$.

The goal in forecasting is to predict a value at time $t + l$ using the parameters solved by fitting the ARIMA model to the time series $\mathbf{y}$. It can be shown that the minimum mean square error forecast is the conditional expected value of that same step (Box et al., 2008), as illustrated in equation 5.4.

$$\hat{y}_{t+1} = E\left[y_{t+1}|\mathbf{y}\right] \tag{5.4}$$

where E denotes the expected value, $\mathbf{y} = (y_t, y_{t-1}, \ldots, y_1)$. Once the value for $\hat{y}_{t+1}$ is obtained, it is appended to $\mathbf{y}$, and a forecast $\hat{y}_{t+2}$ for the future time y_{t+2} can be made. To calculate the conditional expectations for any horizon $t + l$, the rules defined in Equation 5.5 are used.

$$\begin{aligned}
[y_{t-j}] &= y_{t-j} & j &= 0, 1, 2, \ldots \\
[y_{t+j}] &= \hat{y}_{t+j} & j &= 1, 2, \ldots \\
[e_{t-j}] &= e_{t-j} & j &= 0, 1, 2 \ldots \\
[e_{t+j}] &= 0 & j &= 1, 2, \ldots
\end{aligned} \tag{5.5}$$

where j is a nonnegative integer and brackets denote the expected value at time t; e.g. $E_t\left[y_{t+i}\right] = [y_{t+i}]$ and $E_t\left[e_{t+i}\right] = [e_{t+i}]$. These rules allow ARIMA to forecast up to any horizon, $t + l$ using information from previous forecasts $\hat{y}_{t+i}$ where $0 < i < l$. Expanding the right hand side of Equation 5.4 gives Equation 5.6.

$$\hat{y}_{t+l} = \left[y_{t+l}\right] = \phi_1\left[y_{t+l-1}\right] + \ldots + \phi_{d+p}\left[y_{t+l-p-d}\right] - \beta_1\left[e_{t+l-1}\right] - \ldots - \beta_q\left[e_{t+l-q}\right] + \left[e_{t+l}\right] \tag{5.6}$$

where the rules defined in Equation 5.5 are used for each y_i and e_i. The ARIMA model and its forecasts are based solely on previous values of the time series $\mathbf{y}$. Using ARIMA to predict blood glucose values implies that only the actual blood glucose values are used for prediction, other factors such as insulin and life event data are not considered.

Consider a time series **y** represented by an ARIMA model of order $(1, 1, 0)$ in the form of Equation 5.3, with $\Phi_1 = 0.8$:

$$(1 - 0.8B)(1 - B)\, y_{t+1} = e_{t+1}$$

which can be written as:

$$\left(1 - 1.8B + 0.8B^2\right) y_{t+1} = e_{t+1}$$

Carrying the y_{t+1} term through gives:

$$y_{t+1} - 1.8y_t + 0.8y_{t-1} = e_{t+1}$$

Generalizing this equation for any horizon $t + l$, gives Equation 5.7.

$$y_{t+l} = 1.8y_{t+l-1} - 0.8y_{t+l-2} + e_{t+l} \tag{5.7}$$

Using the rules from Equation 5.5, the forecasts at the origin t are given as:

$$\begin{aligned} \hat{y}_{t+1} &= 1.8y_t - 0.8y_{t-1} \\ \hat{y}_{t+2} &= 1.8\hat{y}_{t+1} - 0.8y_t \\ \hat{y}_{t+l} &= 1.8\hat{y}_{t+l-1} - 0.8\hat{y}_{t+l-2} \quad l = 3, 4, \ldots \end{aligned} \tag{5.8}$$

This example shows how the forecasts generated by the ARIMA model are computed in the recursive order, $\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots$

The ARIMA model used in this work is implemented in the R statistical package (R Development Core Team, 2011). Identifying the model was completed with the R statistical function *auto.arima*, which chooses (p, d, q) based on the examples provided. The Bayes information criterion is used to determine the orders p and q. The Phillips-Perron unit root test is used to determine order d.

5.3 Feature Templates

This section defines the templates used to create features for predicting blood glucose values. Each template defines a set of possible features which can be employed by the SVR feature vector.

5.3.1 Baseline Features

The two baselines described in Section 5.2 are also used as features for the SVR model. The t_0 feature is defined in Equation 5.9 and the ARIMA feature is defined in equation 5.10.

$$\varphi_{t_0}(x,t) = y_t \tag{5.9}$$

where x is the patient, t is the index of the time series, and y_t is the blood glucose value at time t.

$$\varphi_{ARIMA}(x,t,l) = \hat{y}_{t+l} \tag{5.10}$$

where $\hat{y}_{t+l}$ is the expected value from the ARIMA model for time $t + l$.

5.3.2 Moving Average

This template defines features that encode a moving average of past observations starting from a time t for patient x. The moving average feature template is defined in Equation 5.11.

$$\varphi_{MVA}(x,t,n,\lambda) = \frac{\sum_{i=0}^{n} \lambda^i y_{t-i}}{\sum_{i=0}^{n} \lambda^i} \tag{5.11}$$

where n is the number of past observations, and λ is a decay factor between zero and one. If $\lambda = 1$ then there is no decay, and Equation 5.11 computes a simple moving average.

5.3.3 Rate of Change

This template defines features that encode the rate of change in blood glucose over a specified period of time. These features are similar to taking the average of a differenced time series with $d = 1$. For example, this feature template computes the average of the series $(x_1, x_2, \ldots, x_{n-1})$ where $x_i = y_{i+1} - y_i$. The rate of change feature template is defined in Equation 5.12.

$$\varphi_{RoC}(x, t, n, \lambda) = \frac{\sum_{i=0}^{n-1} \lambda^i \left[y_{t-i} - y_{t-i+1}\right]}{\sum_{i=0}^{n-1} \lambda^i} \tag{5.12}$$

where n is the number of past observations, and λ is a decay factor between zero and one.

5.3.4 Bolus

This template defines features that compute the total amount of insulin administered with a bolus over a given period of time. The type of bolus needs to be considered, as a square wave bolus releases a specified amount of insulin over time, whereas a regular bolus releases insulin instantaneously. The bolus feature template is defined in equation 5.13.

$$\varphi_{Bolus}(x, t, \Delta) = \varphi_{Bolus}(x, t - \Delta, t) \tag{5.13}$$

where Δ is the period of time to use before time t, and $\varphi_{Bolus}(x, t - \Delta, t)$ is the total amount of insulin from all boluses between time $t - \Delta$ and t, including any insulin from a square wave bolus which overlaps the period $(t - \Delta, t)$.

5.3.5 Basal Rate

This template defines features that compute the total amount of insulin from the patient's basal rate. While this template is similar to the bolus template in the sense that it measures insulin, when tuning the bolus and basal rate templates separately, different values were obtained for the size of the bin and the number of bins for the two templates. This indicates that the SVR model exploits the basal and bolus features in different ways, and these features should be more useful if they are represented as separate features. The basal rate feature template is defined in Equation 5.14.

$$\varphi_{Basal}(x, t, \Delta) = \varphi_{Basal}(x, t - \Delta, t) \tag{5.14}$$

where Δ is the period of time to use before time t, and $\varphi_{Basal}(x, t - \Delta, t)$ is the total amount of insulin from the basal rate over the period $(t - \Delta, t)$.

5.3.6 Basal Rate Area from Mean

The purpose of this template is to encode drastic changes in the patient's basal rate, such as a temporary basal or pump suspension event. For each bin that this feature encodes, the mean basal rate is calculated. Then, the absolute area between the basal rates in the bin and the mean is computed. The basal area from mean feature template is defined in Equation 5.15.

$$\varphi_{BasalArea}(x, t, \Delta) = \int_{t-\Delta}^{t} |\varphi_{Basal}(x, t) - \bar{\varphi}_{Basal}(x, t - \Delta, t)| \, dt \tag{5.15}$$

where Δ is the period of time to use before time t, $\varphi_{Basal}(x, t)$ is the basal rate at time t, and $\bar{\varphi}_{Basal}(x, t - \Delta, t)$ is the average basal rate over the period $(t - \Delta, t)$.

5.3.7 Carbohydrates

This template defines features that encode the estimated number of carbohydrates from the Bolus Wizard® and hypoglycemic episodes over a given period of time. The carbohydrate feature template is defined in Equation 5.16.

$$\varphi_{Meal}\left(x, t, \Delta\right) = \varphi_{BWZ}\left(x, t - \Delta, t\right) + \varphi_{HE}\left(x, t - \Delta, t\right) \tag{5.16}$$

where $\varphi_{BWZ}\left(x, t - \Delta, t\right)$ is the number of estimated carbohydrates entered into the Bolus Wizard® between time $t - \Delta$ and t, and $\varphi_{HE}\left(x, t - \Delta, t\right)$ is the number of estimated carbohydrates from hypoglycemic episodes between time $t - \Delta$ and t.

5.3.8 Exercise

This template defines features that describe the amount of time the patient is exercising over a given period of time. The exercise feature template is defined in Equation 5.17.

$$\varphi_{Exercise}\left(x, t, \Delta\right) = \frac{1}{\Delta} \int_{t-\Delta}^{t} Exercise\left(x, t\right) dt \tag{5.17}$$

where $Exercise\left(x, t\right)$ is 1 if the patient is exercising at time t, and 0 otherwise.

5.3.9 Work

This template defines features that describe the amount of time the patient is at work over a given period of time. The work feature template is defined in Equation 5.18.

$$\varphi_{Work}(x, t, \Delta) = \frac{1}{\Delta} \int_{t-\Delta}^{t} Work\left(x, t\right) dt \tag{5.18}$$

where $Work\left(x, t\right)$ is 1 if the patient is at work at time t, and 0 otherwise.

5.3.10 Sleep

This template defines features that describe the amount of time the patient is sleeping over a given period of time. The sleep feature template is defined in Equation 5.19.

$$\varphi_{Sleep}(x,t,\Delta) = \frac{1}{\Delta}\int_{t-\Delta}^{t} Sleep\,(x,t)\,dt \tag{5.19}$$

where $Sleep\,(x,t)$ is 1 if the patient is sleeping at time t, and 0 otherwise.

5.4 Adapting Parameters

Two different parameters can be adapted for the SVR such that more recent training examples are given more emphasis than distant training examples. The intuition is that examples from the recent past will likely resemble examples in the near future.

5.4.1 Ascending C

The C parameter for the SVR controls the emphasis given to errors within the training data (Section 2.3.2). By increasing the value of C for more recent examples, the SVR model will put more emphasis on errors from recent examples and less emphasis on errors of distant examples. This idea first appeared as a back propagation rule for neural networks in (Refenes et al., 1997). Tay and Cao later adapted this idea for SVM (Cao and Tay, 2003; Tay and Cao, 2002a).

The index of each example in the time series is exploited to compute its C value. An index of 1 denotes the most distant training example, and an index of n denotes the most recent training example. Using this formulation, the value of C for each example is computed as shown in Equation 5.20.

$$C_i = C\frac{2}{1+\exp\left(a - 2a\frac{i}{n}\right)} \tag{5.20}$$

where i represents the index of the training example, n is the total number of training examples, and a is the parameter that controls the rate at which C is increased. This function resembles a logistic function that is centered about the midpoint of the data. When $i = \frac{n}{2}$, the corresponding C_i will have a weight of C. As i approaches zero, the value of C_i will reach its minimum. As i approaches n, the value of C_i approaches $2C$. This type of function is illustrated in Figure 5.1, for $n = 4000$.

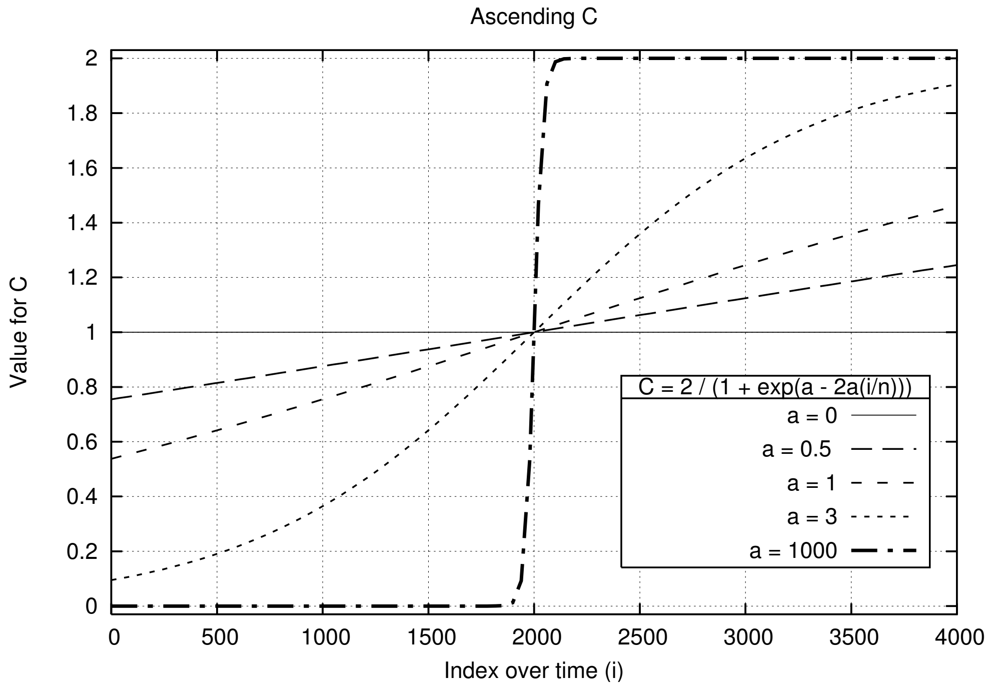


Figure 5.1: Illustration of ascending the C parameter.

5.4.2 Descending Epsilon

The ϵ parameter controls the sparsity of solutions by ignoring training errors within a certain distance of the target value. As described in Section 2.3.3, this represents a tube around the output label, illustrated in Figure 2.3. By decreasing the value of ϵ for more recent examples, the SVR model will be more precise when learning from recent training examples. Tay and Cao proposed descending the ϵ parameter in (Tay and Cao, 2002b; Cao and Tay, 2003).

As with adapting the C parameter, the index of each example is exploited to compute its ϵ value. An index of 1 denotes the most distant training example, and an index of n denotes the most recent training example. The value of ϵ for each example is computed as shown in Equation 5.21.

$$\epsilon_i = \epsilon \frac{1 + exp\left(b - 2b\frac{i}{n}\right)}{2} \tag{5.21}$$

where i represents the index of the training example, n is the total number of training examples, and b is the parameter which controls the rate at which ϵ is descended. This function resembles an exponential decay which passes through 1 at the midpoint of the training data. When $i = \frac{n}{2}$, the corresponding ϵ_i will have a weight of ϵ. This type of function with $n = 4000$ is illustrated in Figure 5.2.

5.5 Error Metrics

This section describes the error metrics used to measure and compare the performance of the two baselines and the SVR model. These metrics are Mean Absolute Error (MAE), Root Mean Square Error (RMSE), coefficient of determination (R^2), and the Clarke Error Grid Analysis (CEGA).

5.5.1 Mean Absolute Error

The mean absolute error represents the average overall error between the actual and predicted values. The definition is given in Equation 5.22.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{5.22}$$

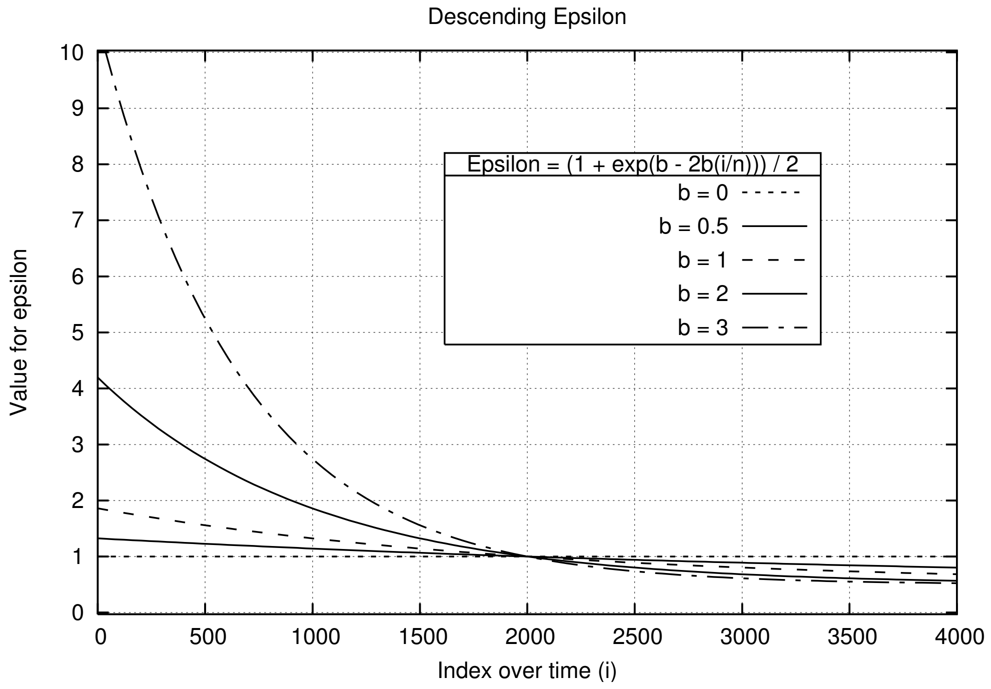


Figure 5.2: Illustration of descending the ϵ parameter.

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of test examples.

5.5.2 Root Mean Square Error

The Root Mean Square Error is similar to the standard deviation of the error. It is defined in Equation 5.23.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{5.23}$$

where y_i is the actual value and $\hat{y}_i$ is the predicted value.

5.5.3 Coefficient of Determination

R^2, or the coefficient of determination, provides information about the goodness of fit of the model. An R^2 score of 1 means a perfect fit.

$$R^2 = 1 - \frac{ESS}{TSS} \tag{5.24}$$

where ESS is the residual sum of squares error and TSS is the total sum of squares, defined as follows:

$$ESS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{5.25}$$

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{5.26}$$

where $\bar{y}$ is the mean of the actual values.

5.5.4 Clarke Error Grid Analysis

The Clarke Error Grid Analysis (CEGA) is used to analyze clinical accuracy of blood glucose values. It was first proposed in (Clarke et al., 1987), and it has become a clinical standard for assessing the quality of blood glucose sensors as well as predictions (Kovatchev et al., 2004). The grid breaks down a scatter plot into 5 regions as illustrated in Figure 5.3.

These regions are defined, from best to worst, as:

A Predicted values within 20% of the actual values

B Points outside of 20% that would not lead to inappropriate treatment

C Points leading to unnecessary treatment

D Points indicating a potentially dangerous failure to detect hypoglycemia or hyperglycemia

E Points that would confuse treatment of hypoglycemia for hyperglycemia or vice-versa

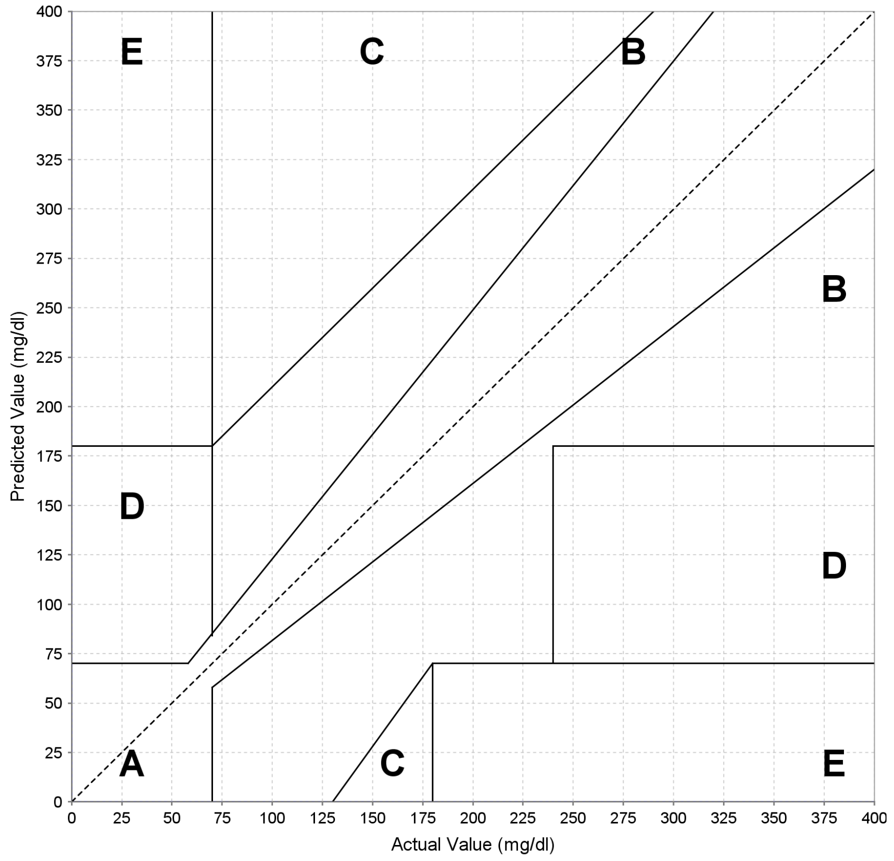


Figure 5.3: The Clarke Error Grid.

5.6 Experimental Evaluation

Data from ten different patients who completed the third study of the 4DSS™ was used to predict blood glucose values. For each patient, a pivot date was chosen such that it was always a Sunday about 1 month into the patient's study. The feature templates were tuned using a grid search with 2 weeks of training data and 1 week of development data prior to the pivot point. This gave ten unique feature vectors tailored for each individual at each prediction horizon. The next 14 days after this pivot date were used for testing, while the 14 days prior to the pivot date were used for training, and the pivot date itself was used for tuning. Walk-forward testing, as defined in Section 2.4.2, was performed with a step size of 1 on all 14 days of test data. For each test day, the RMSE, MAE, R^2, and CEGA were computed for the t_0 baseline, ARIMA baseline, and the SVR model. All results are on smoothed data, using cubic spline smoothing with extra weight for local optima and fingersticks. A ridge parameter of $\lambda = \exp^{-20}$, a local optima and fingerstick weight of $P = 1000$, and $a_p = b_p = c_p = 1$ were used. $\left(a_p, b_p, c_p\right)$ are the constants from Equation 3.17 which decay a local optimum's weight due to a fingerstick within one hour of the optimum's time.

The ARIMA model was built using 4 days of training data. An exploratory data analysis showed that 4 days gave the lowest RMSE for the ARIMA model. While the ARIMA model uses 10 fewer days of data than the SVR model, the ARIMA model is slower to train and predict than the SVR model. The tuning dataset was used to tune the parameters for the SVR. A grid search is carried out with a linear, polynomial, and Radial Basis Function (RBF) kernels. Along with the type of kernel, the regularization parameter C, the degree of the polynomial kernel, the width of the hypersphere γ, and the tube width ϵ are tuned. Then, the tuning dataset was used again to tune the a and b parameters; a controls the rate at which C is increased, and b controls the rate at which ϵ is decreased. This experiment is carried out for prediction horizons at 30 and 60 minutes, or $l = \{6, 12\}$. For each horizon, two different feature vectors are used for the SVR model, one including both insulin and life event data, and one without this data.

Appendix A contains the details for each patient's feature vector that was tuned using the feature templates in Tables 5.1 and 5.2. Table 5.1 defines templates for the SVR model that include insulin and life event data, defined as SVR_1. Table 5.2 defines templates for the SVR model that exclude insulin and life event data, defined as SVR_2. For the moving average and rate of change features, the decay parameter λ was tuned. For the life event features, the size of the bin, k, and the number of bins of that size are tuned.

Table 5.1: Feature vector with insulin and life event data for the SVR model, SVR_1.

Feature	Description
$\varphi_{t_0}(x,t)$	Glucose value at time t
$\varphi_{ARIMA}(x,t,k)$	ARIMA forecast at time t, $k = l$
$\varphi_{MVA}(x,t,k,\lambda)$	Moving Average, $k = \{3,6,9,12\}$
$\varphi_{RoC}(x,t,k,\lambda)$	Rate of Change, $k = \{3,6,9,12\}$
$\varphi_{Bolus}(x,t-k,k)$	Bolus
$\varphi_{Basal}(x,t-k,k)$	Basal Rate
$\varphi_{BasalArea}(x,t-k,k)$	Basal Rate Area from Mean
$\varphi_{Meal}(x,t-k,k)$	Carbohydrates
$\varphi_{Exercise}(x,t-k,k)$	Exercise
$\varphi_{Work}(x,t-k,k)$	Work
$\varphi_{Sleep}(x,t-k,k)$	Sleep

Table 5.2: Feature vector without insulin and life event data for the SVR model, SVR_2.

Feature	Description
$\varphi_{t_0}(x,t)$	Glucose value at time t
$\varphi_{ARIMA}(x,t,k)$	ARIMA forecasts at time t, $k = l$
$\varphi_{MVA}(x,t,k,\lambda)$	Moving Average, $k = \{3,6,9,12\}$
$\varphi_{RoC}(x,t,k,\lambda)$	Rate of Change, $k = \{3,6,9,12\}$

5.6.1 Results and Discussion

This section reports the results for the 30 minute and 60 minute prediction horizons. At each prediction horizon, the average RMSE for each patient and the error metrics averaged over all patients are reported. Also reported is the RMSE results of walk-forward testing for two patients, along with two examples of one day's predictions and CEGA. Comprehensive results for each patient can be found in Appendix A.

5.6.1.1 30 Minutes

This section presents results for predictions made at 30 minutes. Table 5.3 presents the four error metrics averaged over all 10 patients. An interesting result is that for a 30 minute prediction horizon, the ARIMA and SVR_2 models, which do not use insulin or life event data, scored a better RMSE and MAE than SVR_1, which included them. These results suggest that a model based only on blood glucose values may be appropriate for making 30 minute predictions.

Table 5.4 shows the average RMSE for each patient's test data. For 30 minute predictions, insulin and life event data either had no effect or hindered the performance of

Table 5.3: Averages over all patients for each error metric with a 30 minute prediction horizon.

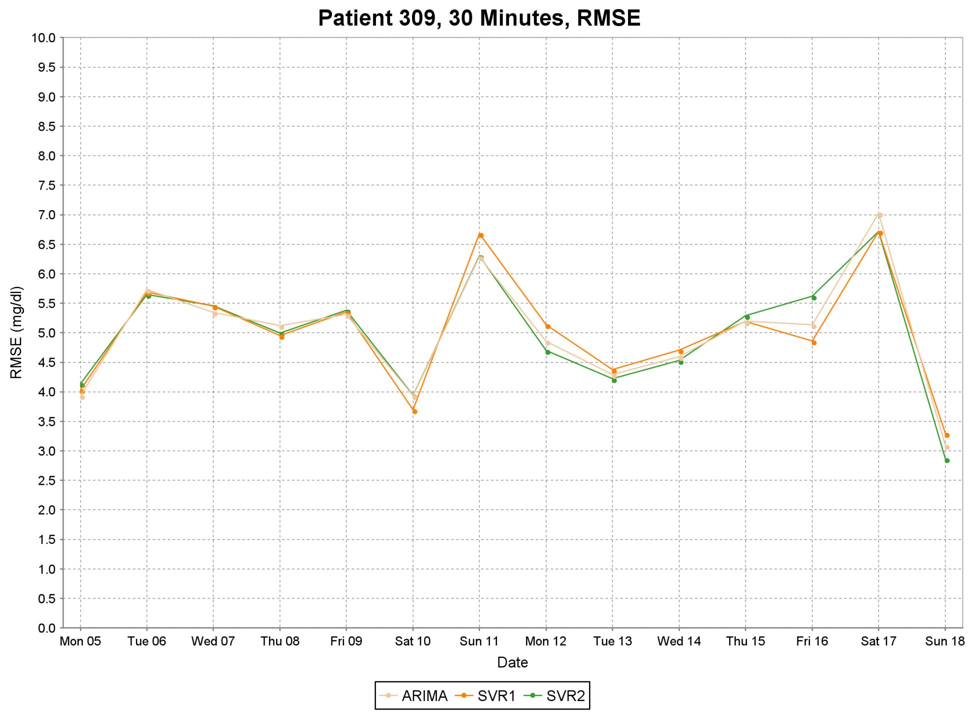
Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	19.5	4.5	4.7	4.5
MAE	15.2	3.2	3.5	3.4
R^2	0.80	0.98	0.98	0.98
CEGA (A%)	87%	99%	99%	99%

Table 5.4: RMSE for each patient with a 30 minute prediction horizon.

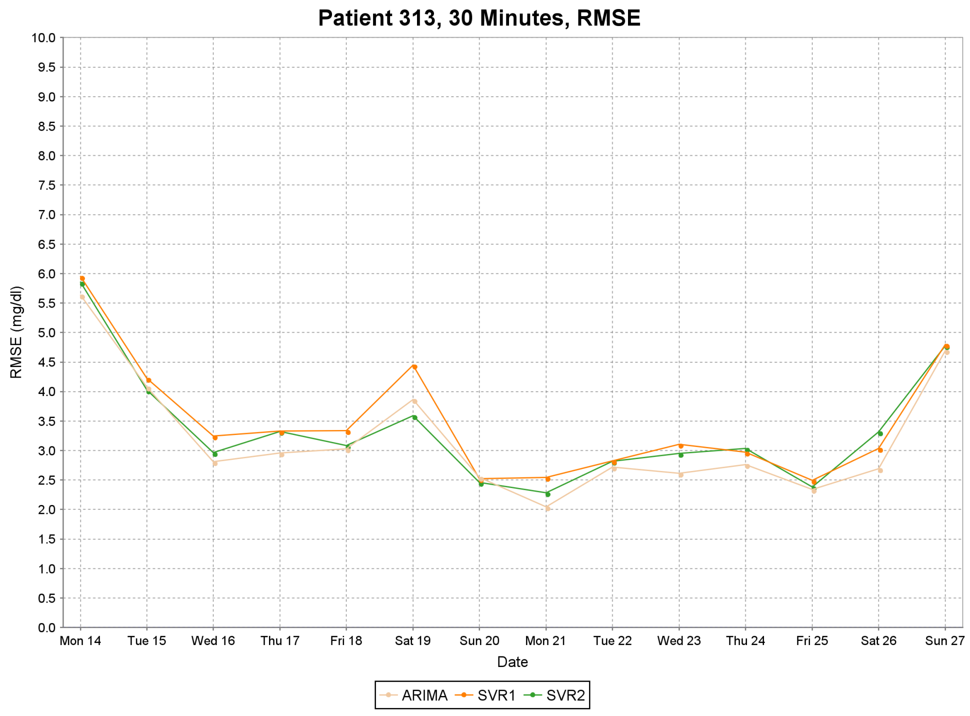
Patient	t_0	ARIMA	SVR_1	SVR_2
301	15.6	2.8	3.2	3.1
302	17.0	5.3	5.5	5.3
304	20.7	4.0	4.4	4.2
306	26.5	6.1	6.1	6.0
307	17.6	4.2	4.6	4.3
308	19.2	4.8	5.2	4.9
309	22.2	4.9	5.0	4.9
310	22.0	5.8	6.1	5.8
312	15.6	3.3	3.6	3.6
313	18.2	3.1	3.4	3.3

the SVR_1 model when compared with SVR_2. Figure 5.4 shows the average daily RMSE for patients 309 and 313 over the entire 2 weeks of testing. For patient 309, the curves for SVR_1 and SVR_2 intersect each other, indicating that, on different days, different models gave the best performance. On Friday, the 16th, SVR_1 greatly outperformed SVR_2 for patient 309. Figure 5.5 shows the predictions for this day in detail along with the CEGA results. As this figure illustrates, SVR_1 was more accurate at the peaks and nadirs of the actual blood glucose plot. Furthermore, the CEGA shows the predictions for SVR_1 are closer to the target values, although all predicted values for both models fall within the ideal A range of the Clarke error grid. For patient 313, SVR_2 consistently outperformed SVR_1. However, ARIMA did better than SVR_2 on many of the test days as well, such as Wednesday the 24th. Figure 5.6 shows the predictions and CEGA for ARIMA and SVR_2

on this day. As this figure illustrates, even though SVR_2 has a higher RMSE than the ARIMA model, its predictions are contained within the A range of the Clarke error grid.

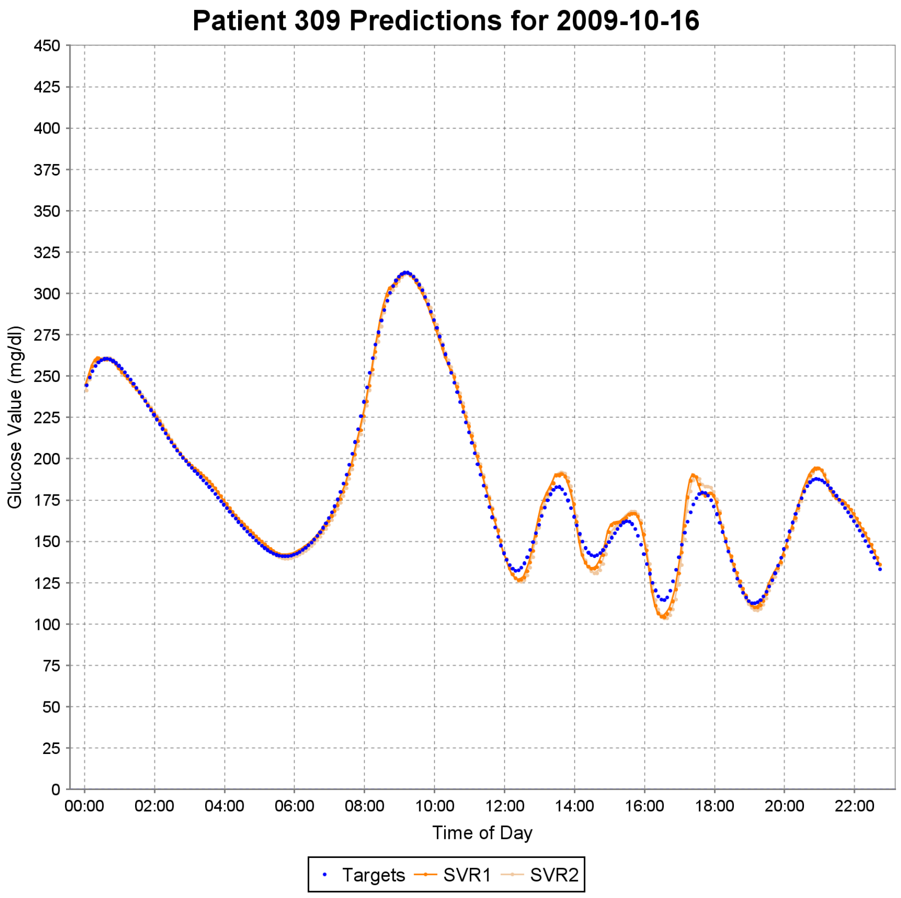


(a) RMSE for patient 309 with a 30 minute horizon.

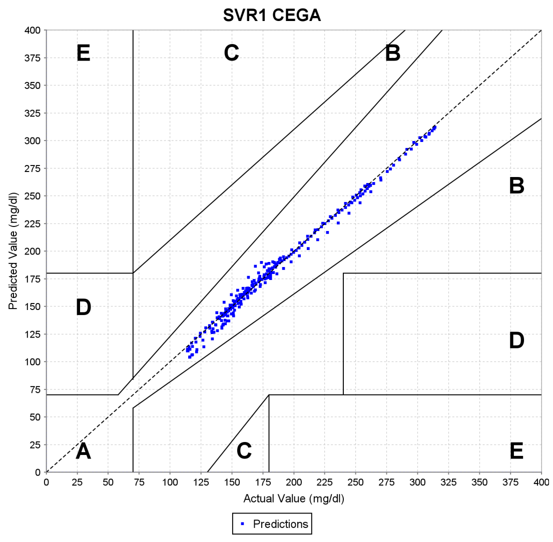


(b) RMSE for patient 313 with a 30 minute horizon.

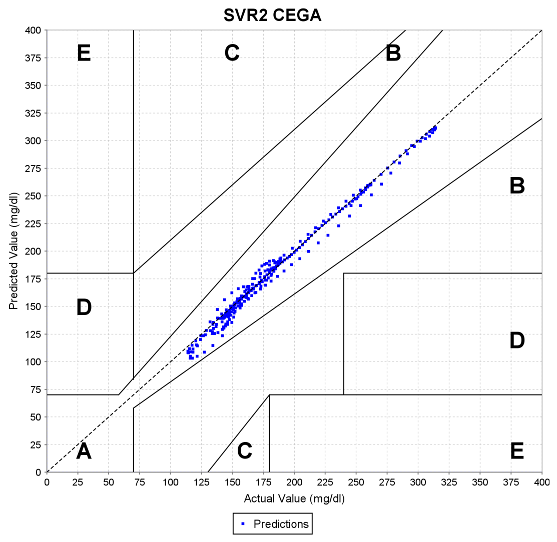
Figure 5.4: Walk-forward testing results over 2 weeks for patients 309 and 313 with a 30 minute prediction horizon.



(a) 30 minute predictions for SVR_1 and SVR_2, along with the actual target values.

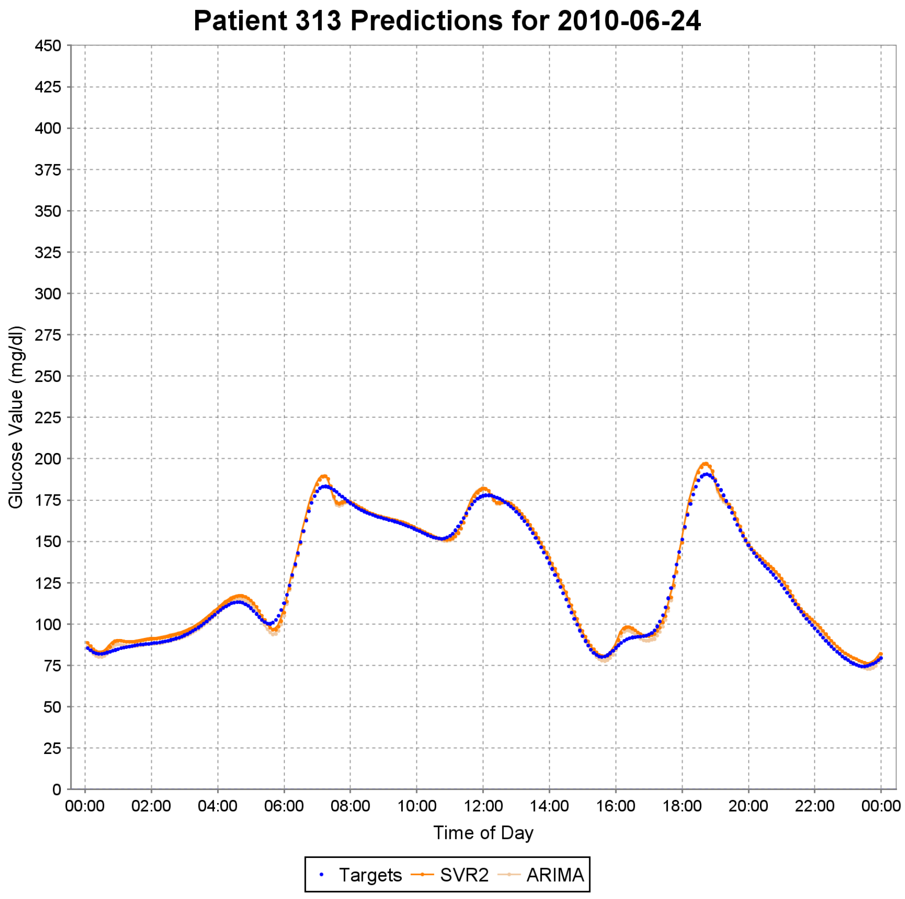


(b) SVR_1 CEGA

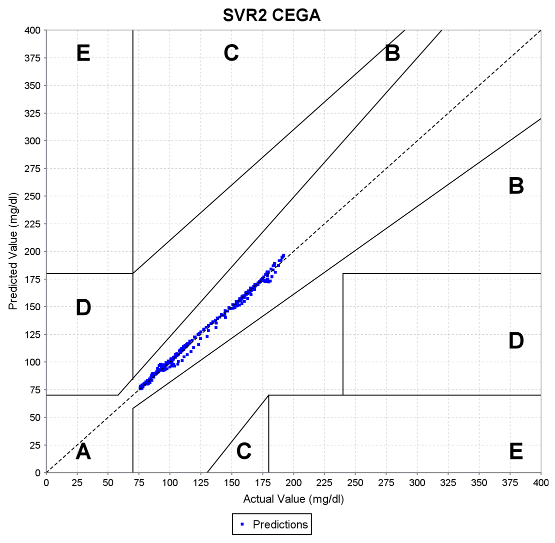


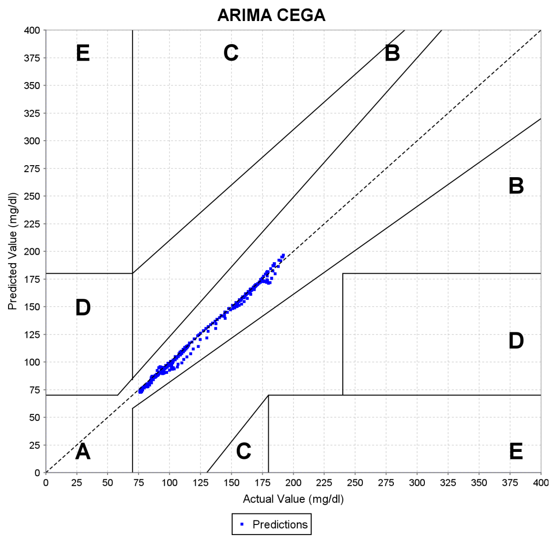
(c) SVR_2 CEGA

Figure 5.5: 30 minute predictions and CEGA results for patient 309.



(a) 30 minute predictions for SVR_2 and ARIMA, along with the actual target values.





(b) SVR_2 CEGA

(c) ARIMA CEGA

Figure 5.6: 30 minute predictions and CEGA results for patient 313.

5.6.1.2 60 Minutes

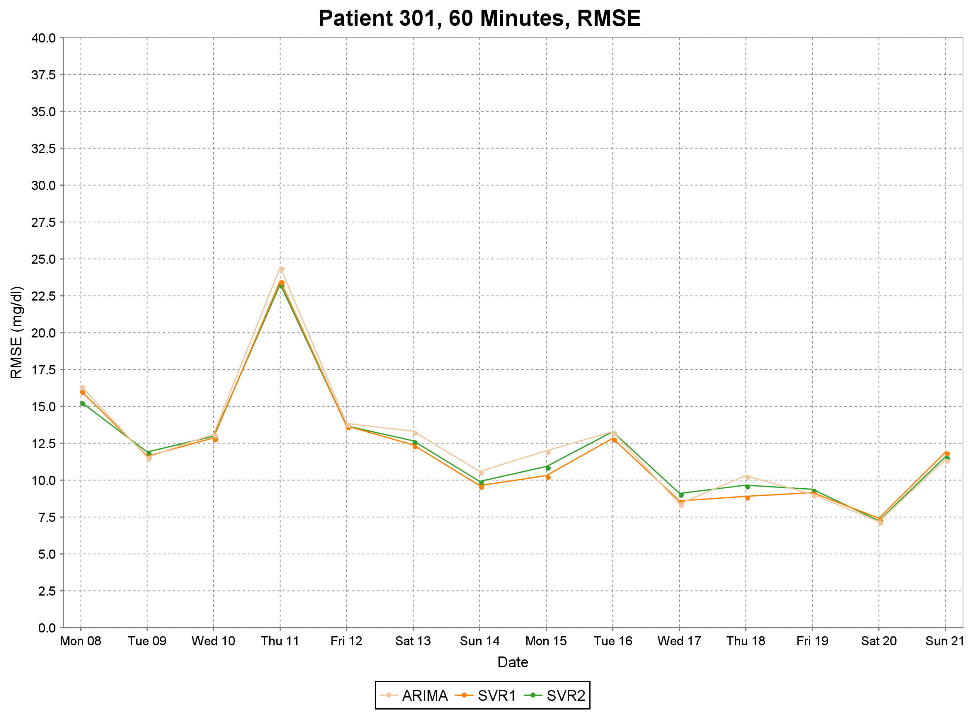
This section presents results for predictions made at 60 minutes. Table 5.5 presents the four error metrics averaged over all 10 patients. For 60 minute predictions, SVR_2 was the most accurate, followed by SVR_1. These results are consistent with the conclusions made in Duke's thesis (Duke, 2009), that autoregressive models tend to be less accurate for prediction horizons longer than 45 minutes. While the SVR_2 model outperformed SVR_1 across all patients, some patients benefited from the insulin and life event data, as shown in Table 5.6. Patients 301, 309, and 312 had lower RMSEs when using the SVR_1 model. Figure 5.7 shows the average daily RMSE for patients 301 and 312 over 2 weeks of testing. As can be seen from this figure, the improvement from including insulin and life event data in SVR_1 was consistent for patient 301. The best improvement observed from using SVR_1 is on Thursday, the 18th. The predictions and CEGA for this day for SVR_1 and SVR_2 are shown in detail in Figure5.8. For patient 312, there were different days on which each SVR model outperformed the other. For example, on Sunday the 23rd, SVR_2 outperformed SVR_1, while on Saturday the 22nd, SVR_1 outperformed SVR_2. The predictions and CEGA for the 23rd for SVR_1 and SVR_2 are shown in Figure 5.9.

Table 5.5: Averages over all patients for each error metric with a 60 minute prediction horizon.

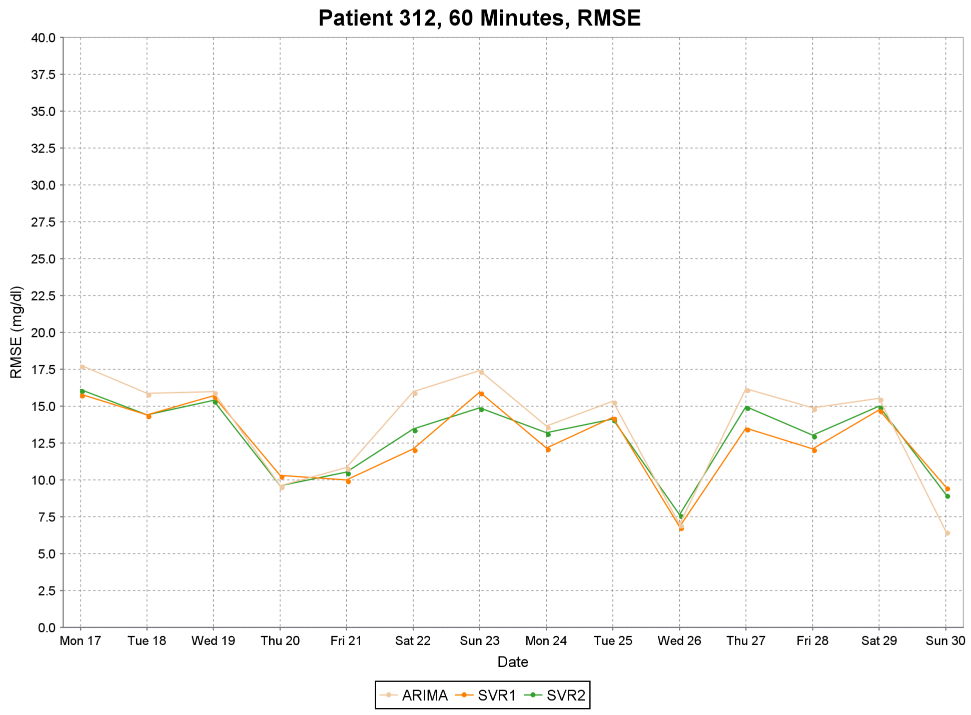
Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	35.5	17.9	17.7	17.4
MAE	27.9	13.5	13.4	13.2
R^2	0.36	0.82	0.82	0.83
CEGA (A%)	63%	89%	89%	90%

Table 5.6: RMSE for each patient with a 60 minute prediction horizon.

Patient	t_0	ARIMA	SVR_1	SVR_2
301	28.8	12.5	12.0	12.2
302	29.7	18.2	18.4	17.5
304	37.7	17.2	17.3	17.2
306	47.7	23.9	24.7	23.7
307	26.0	16.8	16.7	16.5
308	34.3	18.4	18.2	18.2
309	40.7	20.7	19.6	19.9
310	40.0	23.7	23.2	22.4
312	29.2	13.7	12.6	12.9
313	33.8	14.3	14.0	13.7

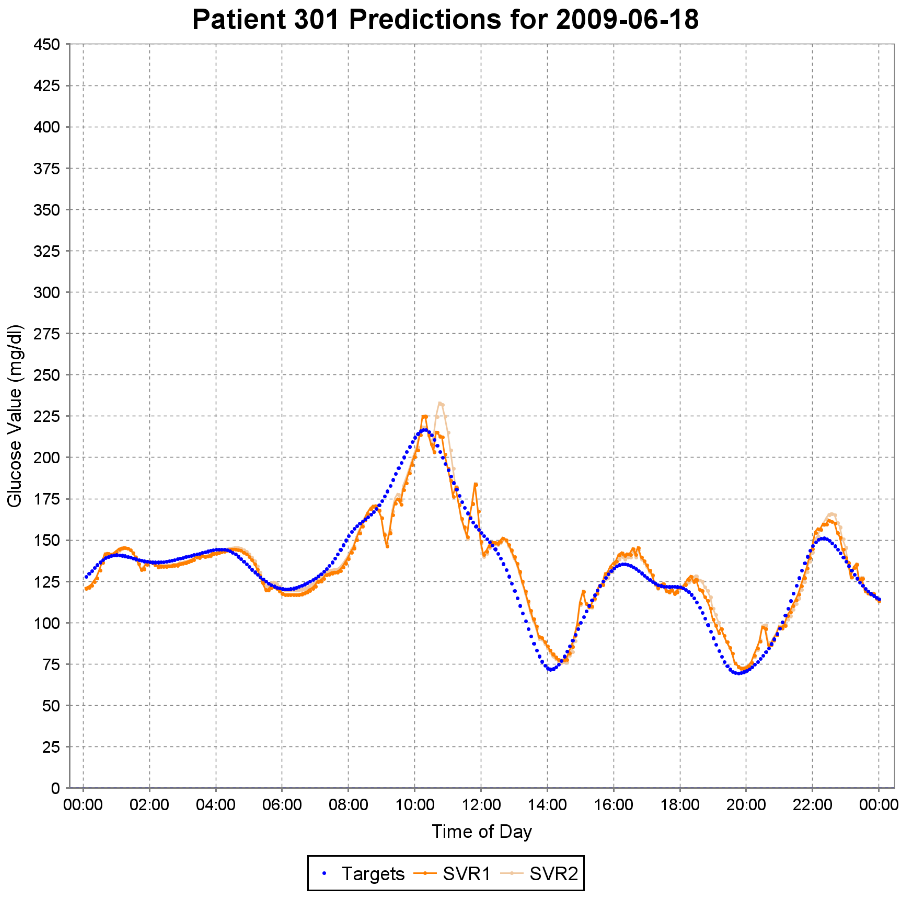


(a) RMSE for patient 301 with a 60 minute horizon.

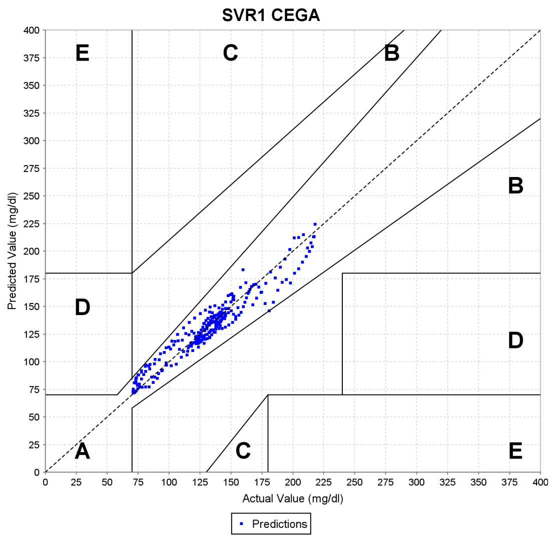


(b) RMSE for patient 312 with a 60 minute horizon.

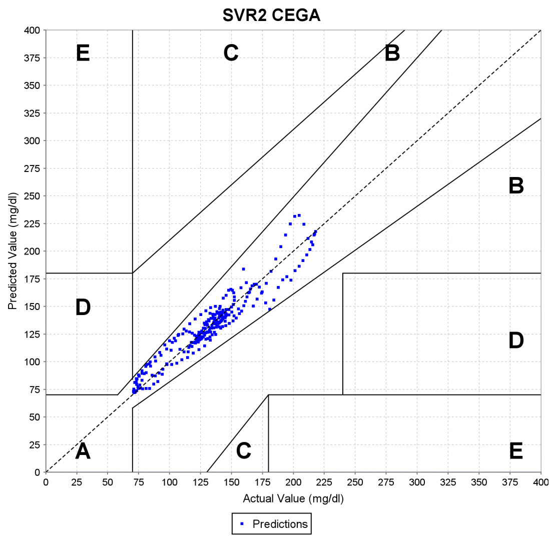
Figure 5.7: Walk-forward testing results for patients 301 and 312 over 2 weeks with a 60 minute prediction horizon.



(a) 60 minute predictions for SVR_1 and SVR_2, along with the actual target values.

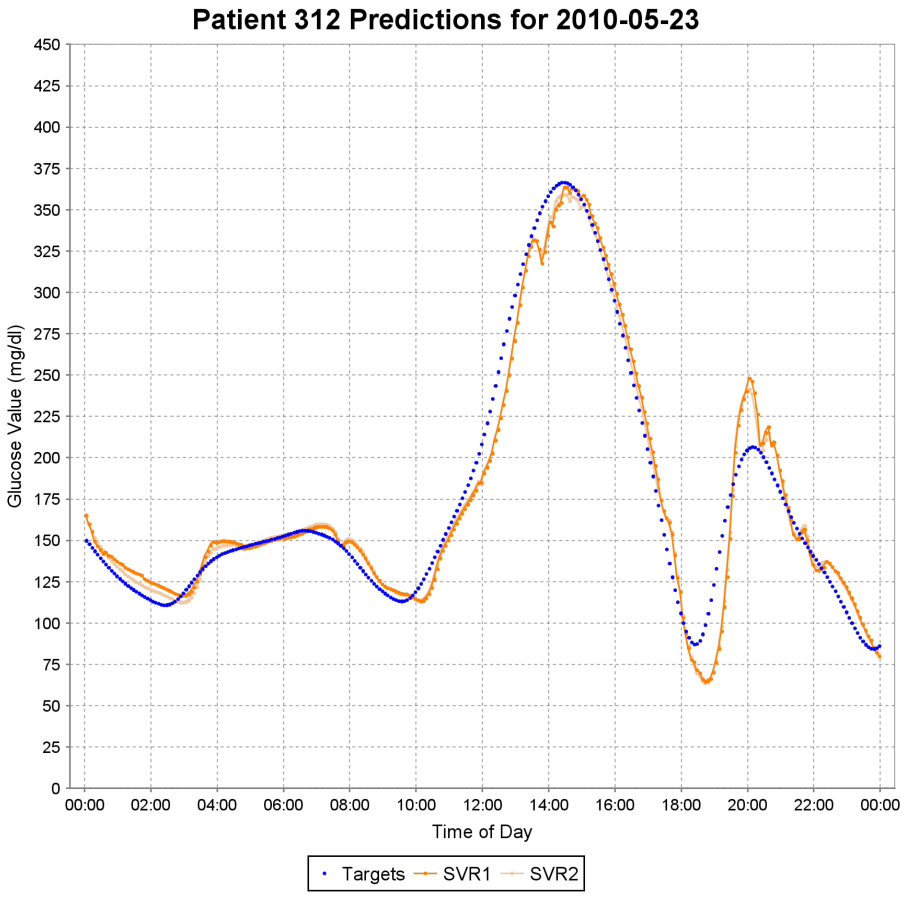


(b) SVR_1 CEGA

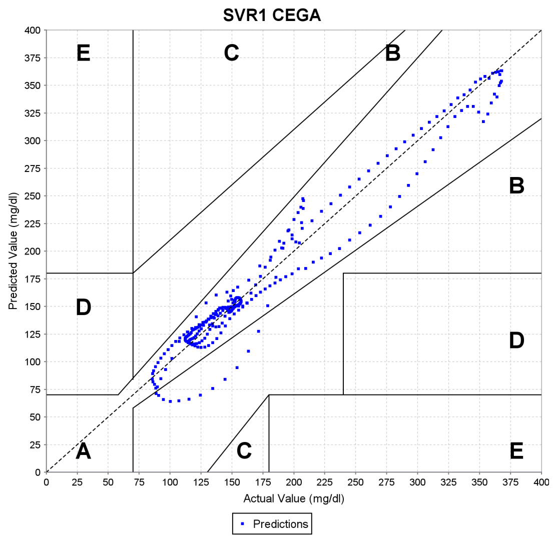


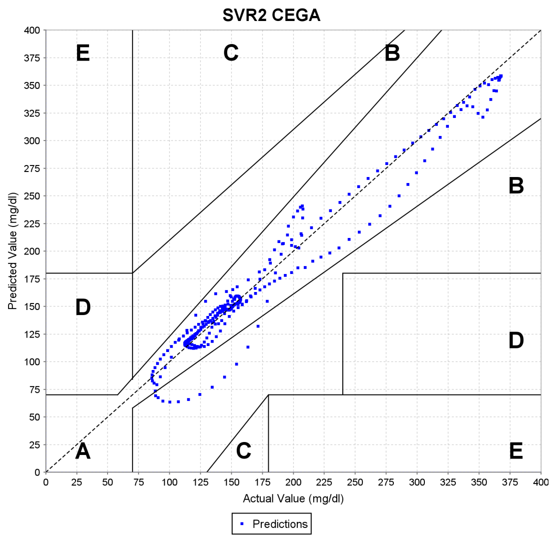
(c) SVR_2 CEGA

Figure 5.8: 60 minute predictions and CEGA results for patient 301.



(a) 60 minute predictions for SVR_1 and SVR_2, along with the actual target values.





(b) SVR_1 CEGA

(c) SVR_2 CEGA

Figure 5.9: 60 minute predictions and CEGA results for patient 312.

6 Related Research

This chapter reviews research related to SVR for time series prediction as well as other approaches to blood glucose prediction.

6.1 Support Vector Regression for Time Series Prediction

SVR for time series prediction, while new to diabetes management, has been applied in other application domains, including financial market prediction and electric utility load forecasting.

6.1.1 Financial Market Prediction

Financial market prediction has been the most focused application of SVR for time series prediction. Sapankevych and Sankar discuss 21 research publications for financial market prediction using SVR (Sapankevych and Sankar, 2009). Between 2001 and 2003, Tay and Cao published a series of four articles on the application of SVR for financial market prediction (Tay and Cao, 2001; Tay and Cao, 2002a; Tay and Cao, 2002b; Cao and Tay, 2003). These articles show various ways the SVR algorithm can be optimized for this domain.

The first publication of the series gives a comprehensive description of the datasets, input and output variables, and error metrics (Tay and Cao, 2001). The datasets consist of the daily closing price for five real futures contracts. A transformation of the five day relative difference in price is applied to the closing prices. The output variable for observation t_i is replaced with $t_i - t_{i-5}$. Tay and Cao (2001a) show two important properties of this transformation by comparing histograms of the datasets. This transformation modifies the datasets such that the distribution is more symmetrical and normal. Once the output variable has been transformed to relative differences, a three day exponential moving average is applied to smooth the data. Taking a relative difference and smoothing the data with an exponential moving average generally increases predictive accuracy in neural networks and SVR (Tay and Cao, 2001).

The authors also give a comparison between SVR and multi-layered back-propagation (BP) neural networks (Tay and Cao, 2001). Four metrics were used for comparing the performance of the two algorithms on test data: Normalized Mean Square Error (NMSE), Mean Absolute Error (MAE), Directional Symmetry (DS), and Weighted Directional Symmetry (WDS). The SVR outperformed the BP network on all four metrics. The authors conclude that SVR is much more promising than BP networks due to the following reasons: BP networks tries to minimize the training error, even when using a validation dataset; over-fitting can degrade accuracy. In contrast, SVR try to minimize the generalization error. This is known as the structural risk minimization principal, and generally results in better performance on test data. A BP network has many free parameters, whereas a SVR model only has three. This makes obtaining an optimal combination of parameters much easier for the SVR. Finally, the BP network uses a gradient descent algorithm, which is not guaranteed to find a global optimum, whereas the solution for a SVR model is a global optimum due to its solving a constraint optimization problem (Tay and Cao, 2001).

The other three articles had a common theme for improving performance in financial market prediction using SVR (Cao and Tay, 2003; Tay and Cao, 2002a; Tay and Cao, 2002b). All three articles address properties of financial time series that make accurate forecasting hard for the SVR. These properties are the non-stationarity of the financial time series and the natural dependence of time between observations. The dependence comes from the notion that recent observations have more influence than distant observations. To account for this property, the authors suggest making modifications to both parameters C and ϵ such that recent training points are given more weight than distant training points. The C parameter controls the emphasis given to errors in the training data (Section 2.3.2); The ϵ parameter is used to give zero error to training examples within the distance ϵ from the target value (Section 2.3.3). By ascending the value of the C parameter, ϵ-insensitive errors of recent observations have a heavier penalty (Tay and Cao, 2002a). This is done with a logistic function, centered about the midpoint of the training dataset. The logistic function used takes one parameter, which controls the rate at which C is increased. Descending the ϵ parameter with respect to time also has a similar effect on the penalty of recent training points (Tay and Cao, 2002b). An exponential function was used to increase the value of ϵ for distant points. These modifications resulted in a solution with fewer support vectors, and improvement in generalization performance was observed over standard SVR (Cao and Tay, 2003).

Another interesting application in financial market forecasting is using SVM to predict the direction of the daily price change in the Korea composite stock price index (KOSPI) (Kim, 2003). This is different from many of the other publications presented in (Sapankevych and Sankar, 2009), as (Kim, 2003) used SVM rather than SVR for prediction. The authors extracted 3000 examples from 10 years of data for training and tuning different machine learning algorithms. Twelve features were created from the datasets. These features included a relative change in price, moving averages of the relative change, rate of change, distance of the current price and the moving averages, and price oscillators between two moving averages. Using these features, SVM, three-layer back-propagation network, and case-based reasoning (nearest neighbor) models were compared. Experimental results from the study showed that the SVM outperformed the back-propagation network and case-based reasoning systems (Kim, 2003). The authors attribute this increase in performance to the structural risk minimization principle of SVMs.

Blood glucose prediction and financial market prediction data have similar characteristics. In both cases the data is nonlinear, non-stationary, and noisy. The data is nonlinear in the sense that there is not a linear equation that can easily solve the forecasting problem. Dynamic trends in the time series data implies non-stationarity. These trends change over time. In the case of stock markets, the trends can be influenced by the time of the year. For blood glucose prediction, trends change according to the state of the patient; even an individual's trends may change over time due to small factors such as stress, illness, or lack of sleep. The noise aspect is demonstrated by the fact that there are many features that influence the output of both prediction problems. Not all features can be exploited for prediction, and these missing factors are considered to be noise (Tay and Cao, 2001).

The non-stationarity of blood glucose time series can be addressed in the same way that Tay and Cao address non-stationarity of financial market data. Emphasis can be put on more recent training examples by modifying the C and ϵ parameters to obtain a more accurate prediction. This technique trains the SVR to ignore examples that are distant from the present, and puts a greater emphasis on training errors close to the horizon (Cao and Tay, 2003; Tay and Cao, 2002a; Tay and Cao, 2002b). This same kind of transformation of parameters would be useful for blood glucose prediction. To put more emphasis on minimizing the error of recent training examples, the value of C is increased and the value of ϵ is decreased.

6.1.2 Electric Utility Load Forecasting

Another area of application for time series prediction is the forecasting of electrical power consumption demands. The ability to forecast these demands on a short term basis is essential for system stability, as the voltage generated by the utility must match the demand. Mid-term and long term forecasts allow for better maintenance scheduling, more efficient use of generated electricity, and lower costs for both consumers and suppliers.

The SVR survey presents 17 different research publications related to this problem (Sapankevych and Sankar, 2009).

In 2001, the EUNITE network held a competition for predicting the daily maximum electricity load of the next 31 days. The winning solution to the problem used SVR (Chang et al., 2001). The competitors were given 2 years of electricity load demand data where the demand was recorded every half hour, the average daily temperature for three years, and the dates of holidays for the training and testing datasets.

The authors point out some interesting patterns in the data that is given. The value for the maximum daily load has a seasonal pattern: higher demand during the winter and lower demand during the summer. The data values are also periodic throughout the week, as there is lower demand during the weekend and higher demand during the weekdays. There is a negative correlation between the average daily temperature and the daily maximum electricity load. The demand for electricity is lower on holidays, and certain holidays, such as Christmas, have a bigger effect on demand.

Several steps were taken to prepare the data for the SVR model. Binary calendar attributes, such as the day of the week, weekday vs. weekend, and holiday information is encoded. Temperature was not used as a feature, because it was not given for the test set and would be difficult to estimate with the limited data. Information about previous daily maximums was included as well; however, for the test dataset, these values were estimated. The last step for preparing the data was segmenting the data based on the season. The authors argued that training a model for each season separately would improve the overall performance, because of the non-stationary aspect of the time series. The authors concluded that choosing the appropriate data segments was the key to enhancing performance.

Pai and Hong present an interesting twist to SVR for forecasting Taiwanese electricity load (Pai and Hong, 2005). Their SVR uses simulated annealing (SA) for choosing the free parameters σ, C, and ϵ. The SA algorithm initializes random values for these parameters, and randomly adjusts them until a stopping criterion is reached.

The dataset consisted of 59 years of load data, which was used for training, tuning, and testing. The objective was to predict the total electricity load of the last 9 years using the first 50 years for training and tuning. The input for the SVR was the electricity load for each year over a predefined time span. The authors found that encoding the total electric load from the past 30 years as features gave the best performance using the training and tuning set. An ARIMA model and a general regression neural network model are compared to the SVR with SA. The SVR greatly outperformed the other two models. The authors attribute this success to using the SA algorithm to choose the free parameters and to the structural risk minimization principle of SVM.

Electric utility load forecasting has some similarities to blood glucose prediction. Electric utility load data is cyclic over time, as was shown with the periodicity of the data in (Chang et al., 2001). The same is true for blood glucose values. Blood glucose patterns differ if the patient is sleeping or awake. There are also differences for exercise, work, and the day of the week. As with electric utility load data, holidays have profound effects on blood glucose values. This is primarily due to extra carbohydrates typically ingested on holidays. There are also cyclic aspects of blood glucose levels with respect to changing the insulin reservoir. The FDA recommends that the reservoir be changed every 72 hours. The efficacy of insulin may depend on the time since the insulin reservoir was changed.

6.2 Blood Glucose Prediction

This section presents research in the area of blood glucose prediction. Three projects, the Artificial Pancreas project, the Intelligent Diabetes Assistant, and the AIDA[1] diabetes simulator are described.

[1] AIDA originally stood for Automatic Insulin Dosage Advisor but is now used solely as an acronym.

6.2.1 Artificial Pancreas Project

The objective of the Artificial Pancreas Project is to close the loop, removing the need for a patient to control the insulin pump. As mentioned in section 2.1.1, closing the loop has been a goal of diabetes researchers for quite some time (Juvenile Diabetes Research Foundation, 2010). The Juvenile Diabetes Research Foundation (JDRF) has been advancing research for a closed-loop system since 2005; this project is known as the JDRF Artificial Pancreas Project. The JDRF, founded in 1970, is the world leader in charitable donations for T1DM research (JDRF, 2011a). In 2005, the Artificial Pancreas Project was approved, and funding for an artificial pancreas started the following year. JDRF defines an artificial pancreas as "an automated system to disperse insulin based on real time changes in blood sugar levels" (JDRF, 2011a). With the present open loop system, patients are required to manually manage their blood glucose levels. Human errors contribute to poor blood glucose control, increasing the risk of long term complications. Therefore, a closed-loop system would be ideal.

In 2009, Aaron Kowalski presented a road map to the artificial pancreas and published some results using a partially automated insulin pump for patients with T1DM (Kowalski, 2009). In his publication, Kowalski raises the question, "Why hasn't an artificial pancreas been realized?" The author posits that patient safety is why such a system has not been realized. This is because inaccurate CGM sensors may report artificially high blood glucose levels, leading to overdosing of insulin, causing hypoglycemia. Until the CGM sensor technology becomes more reliable, a fully automated closed-loop system cannot be realized. However, Kowalski believes that a partial closed-loop system is possible using today's technology.

The road map to an artificial pancreas is described in stages. The first stage deals with hypoglycemic episodes. An alarm would sound when hypoglycemia is detected. If the patient is not responsive to the hypoglycemia alarm, the insulin pump should automatically shut off. While this does not avoid hypoglycemia completely, prolonged episodes of hypoglycemia can be avoided. The second stage is to predict hypoglycemia to reduce the patient's exposure to hypoglycemic episodes. An alarm would sound when hypoglycemia is predicted. If the patient is unresponsive to the alarms, insulin delivery would be reduced. The third stage is to reduce the patient's exposure to both hypoglycemia and hyperglycemia by predicting these episodes. The fourth step is to keep the patient within the target range overnight. The system in this stage would also contain a meal announcement so the system could handle bed time snacks. This announcement would contain information that the system would use to reduce hyperglycemia exposure. The last stage is to move to a fully closed-loop system, such that user interactions are nearly eliminated and the patient's blood glucose levels mimic those of a person without diabetes.

The author concludes that there are a number of challenges that need to be addressed before a closed-loop system will be available to patients. The biggest challenge is inaccuracy of available CGM devices. These devices base readings on interstitial fluid, which lags behind actual blood glucose levels. The CGM devices also become unreliable due to miscalibration. Therefore a closed-loop system must be able to calibrate itself. The dynamics of the efficacy of insulin from patient to patient are also a concern for closed-loop systems. Stress, exercise, and hormone changes may impact the efficacy of insulin. This requires an algorithm that models the physiological aspects of insulin and blood glucose. In summary, Kowalski states that closed-loop systems in stage 3 are possible with today's technology, and notes that such a system is available in Europe. He predicts that a fully closed-loop system is still more than five years away (Kowalski, 2009).

Currently, the JDRF Artificial Pancreas Project has developed a partially automated insulin pump. This pump requires the user to administer meal boluses and to adjust basal rates depending on the patient's activity. However, the system aims to keep the patient between a set range, such as 80mg/dl and 180mg/dl. This is done by adjusting the insulin delivery based on the current blood glucose reading (JDRF, 2011b). This mimics the actions of the human pancreas to a certain degree. JDRF defines this system as a hypoglycemia-hyperglycemia minimizer, the third stage of Kowalski's road map. This type of insulin pump is currently awaiting approval from the Food and Drug Administration (FDA) (JDRF, 2011b).

The approach to blood glucose prediction developed in this work could be applied towards the control of an artificial pancreas. Once reliable CGM sensors become available, the prediction model could be used to complete the final stage of the artificial pancreas road map (Kowalski, 2009). Potential problems in blood glucose control could be preemptively identified and corrected over time by learning a model of the patient's reaction to insulin and life events. This model would be tailored for an individual by learning from that individual's data. As of May, 2011, JDRF is only using the current blood glucose level for adjusting the delivery of insulin (JDRF, 2011b).

6.2.2 Intelligent Diabetes Assistant

In his PhD thesis, David Duke presents an outline for an Intelligent Diabetes Assistant (IDA) (Duke, 2009). IDA is a system that aims to collect information about a patient via telemedicine, share this information with a health care team, and analyze this information for three purposes. These purposes are: (1) predicting two hour postprandial blood glucose levels; (2) generating therapy advice; and (3) continuous glucose modeling. The data collected consists of meal, insulin, other medications, and exercise information. Information was collected through a cell phone interface, with meal information collected via a cell phone picture of the meal.

When a patient with diabetes eats a meal, the number of carbohydrates consumed must be estimated so that the correct amount of insulin can be delivered. Errors in this estimation may result in postprandial hypoglycemia or hyperglycemia. Duke posited that essentially, patients are trying to predict their own postprandial blood glucose levels, which is a difficult task. He proposed Gaussian process regression to predict two hour postprandial glucose values. A Gaussian process with a Gaussian kernel is advantageous because nonlinearities in the input data can be learned (Duke, 2009). Another interesting point Duke makes about Gaussian process regression, is that the physiology of each patient is unique, and Gaussian process regression allows for a unique model to be trained for each patient.

The input for the regression is a combination of past and anticipated data measurements with respect to a meal time *t*. The input consists of the current blood glucose measurement, average exercise before and after the meal, time of the day, estimated carbohydrates, insulin, and medication information. The output of the regression is a two hour postprandial blood glucose prediction. Gaussian process regression with linear and Gaussian kernels and different types of training datasets were investigated. The types of training sets investigated were individual, joint datasets among patients, and weighted datasets between all patients where data for similar patients is weighted more heavily. Duke showed that modeling the effects of patient behavior on blood glucose levels at meal time and improving communication between patients and physicians resulted in improved patient care. Duke notes that this automated method of predicting postprandial blood glucose levels performs better than humans.

Continuous dynamic modeling of blood glucose levels was also explored with the data that was collected. This was done so that therapy advice could be generated at times outside of meals. Two models were investigated, an autoregressive model with exogenous inputs and a physiological model with and without exercise. The inputs to both models consisted of the current CGM reading, ingested carbohydrates, ingested fat, ingested protein, exercise rate, and insulin information. Predictions for 15 minutes, 45 minutes, and 120 minutes were compared for the two models. Interestingly, the autoregressive models outperformed the physiological model for short term predictions up to 45 minutes. For predictions beyond 45 minutes, the physiological model with exercise performed the best (Duke, 2009).

Duke proposes two different approaches for generating therapy advice: real-time and retrospective. Real-time advice would preemptively mitigate hypoglycemia and hyperglycemia, similar to the artificial pancreas. Real-time advice would also be used to generate bolus advice. Retroactive advice would be used to educate the patient. This could be done by adjusting insulin and carbohydrate parameters to show different outcomes to the patient. This would allow the patient to learn optimal blood glucose control behavior.

Meal information was generated using machine learning algorithms on images of meals. A training set of images was labeled with the number of carbohydrates by two dietitians. A regression model was then trained to predict the number of carbohydrates given an image of a meal. Along with estimating the portion size of the meals, the images were also used to automatically recognize food. An interesting point Duke makes is that people eat consistent meals. The problem is not recognizing food from all the possible meals in the world, but rather to recognize food that is similar to meals in the patient's past. This makes the overall task much simpler.

The prediction described in this thesis is different from the models proposed in the IDA system (Duke, 2009). Prediction of blood glucose values can subsume both the physiological model and the autoregressive model proposed in IDA. SVR is more dynamic than autoregressive models, and should be able to achieve better generalization performance on nonlinear, non-stationary, and noisy data. SVR is also less prone to over-fitting than autoregressive models. The most important aspect of SVR is learning a weight vector. Exploiting features correctly allows the SVR model to individualize the physiological dynamics of each patient by using this weight vector. To facilitate this learning, aspects of the physiological model can be exploited as features for the SVR.

6.2.3 AIDA

The AIDA diabetes simulator is a research project that has been ongoing for over 20 years now. According to the website, "AIDA is a freeware educational simulator program of glucose-insulin interaction and insulin dosage & dietary adjustment in diabetes mellitus" (AIDA Diabetes Simulator, 2011). The authors stress that this software is for educational purposes only, as it lacks the ability to accurately predict glucose levels for individual patients. However, the model does attempt to accurately reflect blood glucose profiles of patients with T1DM. The model's performance is validated against real patient data. Along with the software that is available on the website, there are also tutorials and guides for both the technical aspects of the AIDA simulator, and general information on glucose-insulin interaction.

The first publications presenting the AIDA simulator debuted in 1991 (Lehmann and Deutsch, 1991a; Lehmann and Deutsch, 1991b). These publications present an overview of the model that is used to do 24 hour simulations of blood glucose profiles. The models that are described are then illustrated using case studies with patients who are treated with insulin. The model built by the authors consisted of a series of differential equations which attempt to describe the glucose-insulin interaction for a specific patient. These equations depend on parameters, and these parameters must be optimized for each individual patient. The authors concluded that their model does work in a strictly defined domain for some patients, but acknowledge that it is not possible to model a patient's blood glucose profile with a series of differential equations. This is due to specific clinical scenarios which change the dynamics of glucose-insulin interaction. The authors note that this limitation is common to all physiological models. Despite these limitations, the model still has valuable use as an educational tool.

Since these first publications, the AIDA simulator has become available free of charge on the Internet. There is a demonstration of the simulator available for download as well. The most recent version of the AIDA simulator (version 4.3b) has been described in (Lehmann et al., 2011). This publication presents the growing interest in the AIDA simulator by showing the number of visits to the AIDA website and the number of downloads of the AIDA simulator by month. This newer version of the AIDA simulator has been extended to work seamlessly on all Windows and Macintosh platforms. An explanation of how to install the software is also given. The simulator has been modified further to individualize parameters for each patient. The simulator has also been modified for plasma insulin simulations for different types of insulin: rapid-acting, short-acting, intermediate-acting, and very long-acting insulin for insulin doses up to 60 units. These type of simulations use insulin absorption models, which the authors conclude are in need of validation with future work. The authors plan to extend the AIDA simulator such that it is possible to model patients with T2DM (non-insulin-dependent).

The authors are aware that not all patient scenarios are accounted for using the type of model employed by AIDA (Lehmann et al., 2011). The blood glucose prediction approach presented in this thesis attempts to model multiple scenarios. The model described in this work is more general, allowing it to complement the AIDA simulator. Similar to the AIDA simulator, the model in this work could also be used for educational purposes.

7 Future Work

This chapter identifies future work to extend the contributions presented in this thesis. Future work could further improve the results of both glycemic variability classification and blood glucose prediction. Another area of future work is to make the contributions of this thesis available to both patients and health care professionals.

7.1 Data Preprocessing

Future work in data preprocessing involves both life event data and smoothing CGM data, as described in Chapter 3. Collection of future meal and exercise information could be enhanced to make it more accurate, which would make it more useful to the blood glucose prediction models. Current plans are to collect life event data with a mobile interface, which will allow for more accurate timestamps with life events. It would be even better if, as done with the IDA project, meal information could be captured using a camera phone. Then, pattern recognition could be used to automatically determine the number of carbohydrates associated with the meal. Furthermore, the picture could be saved with a time stamp such that the time of carbohydrate ingestion is more precise. Also done with the IDA project, exercise information could be collected using a sensor. In the IDA project the sensor measured skin temperature, movement via accelerometers, and estimates on the number of calories burned per minute. This information would be valuable to any blood glucose prediction model.

While smoothing CGM data produced results closer to the actual blood glucose values, this accuracy could be further improved. Even though fingerstick data is more accurate than CGM data, there are cases of noisy fingerstick readings. The meters that record fingerstick values are only accurate to ±7% of the actual values. Identifying noisy fingerstick readings as anomalies and ignoring them while smoothing would increase the physiological accuracy of the smoothed curve. Furthermore, using a simple function to estimate the weight of a local optimum, while useful, may be suboptimal. Improvements might be made by identifying specific cases and modifying the local optima weight based on these cases. Each local optima's case would be determined with a classification scheme. This scheme could be a set of rules or a machine learning classifier.

7.2 Classification of Glycemic Variability

The model for detecting excessive glycemic variability described in Chapter 4 could be further improved. A constraint that we faced during development of the models was the relatively small size of the glycemic variability dataset. The current dataset contains 262 unique examples; 187 positive and 75 negative examples. In order to obtain statistically significant results from the 10-fold cross validation, we had to limit the size of the development set, which may have led to suboptimal parameters and imperfect feature selection. Collecting more annotated CGM plots from diabetes experts is therefore a high priority for future work.

We also plan to change the binary annotation scheme to a "5-star" ordinal scheme wherein physicians would annotate CGM plots using 5 ordered labels, from least variable (1 star) to excessively variable (5 stars). This new annotation scheme would alleviate the problem of disagreement between annotators, with minimal additional effort for the annotators. The new annotations have the potential to further improve accuracy when used with ordinal regression (Chu and Keerthi, 2005). Furthermore, we plan to bootstrap the new annotation process by using the currently developed systems via active learning (Tong and Koller, 2000), thus maximizing the utility of the newly annotated examples.

The ultimate aim of this classification is to develop a practical clinical tool to automatically screen for excessive glycemic variability in patients with diabetes. A routine clinical screen for excessive glycemic variability would allow early identification of patients at risk for preventable diabetic complications. Preventing diabetic complications improves quality of life for patients while reducing the financial burden of health care costs. Therefore, future plans include development of a commercial software package for clinical use.

7.3 Blood Glucose Prediction

The model used to predict blood glucose values described in Chapter 5 can be improved in various ways. The current SVR model exploits insulin and carbohydrate information separately. However, the dynamics of insulin and carbohydrates are not independent. Similar to the forecast of the ARIMA model exploited as a feature for the SVR model, the SVR model could be further improved by using features that represent the pharmacokinetics of insulin and carbohydrates. These features would account for the dependence between insulin and carbohydrates, allowing the SVR model to exploit this information directly. The features generated by the pharmacokinetic model may be used with the ARIMA model, resulting in more accurate ARIMA forecasts which would result in a more accurate SVR model. It is an empirical question whether the pharmacokinetic features are more beneficial with the SVR or ARIMA model. The pharmacokinetic features could be generated using an existing algorithm, such as the one described in the IDA project or the AIDA simulator. With features that accurately model the pharmacokinetics of insulin and carbohydrates, the insulin and carbohydrate parameters become more useful to the overall prediction model.

It is possible that the accuracy of the prediction models may vary given the situation. For example, there are fewer factors that influence blood glucose values while the patient is asleep, and these predictions may be more accurate than predictions made while the patient is awake. Evaluation of the accuracy of predictions during meals and exercise should also be captured. There are many factors which influence blood glucose during these situations, and the intuition is that a model with insulin and life events will obtain better accuracy. The accuracy of the prediction models given the range of the blood glucose values is also of interest. Hyperglycemia and hypoglycemia are more important clinically than the normal range, and the accuracy of the prediction models should be captured at these abnormal ranges. Insulin and life events may influence transitions in and out of the normal range. Evaluating predictions during these transitions may show that the model which includes insulin and life events results in better accuracy.

The patient specific part of the SVR model can be further improved. This work used a separate set of data to tune the features only once before measuring the performance of the prediction models. This could be improved upon by optimizing the features using the validation data as a guide. There are two different approaches to this optimization: optimizing the parameters of each feature template, and optimizing combinations of features. Optimizing the parameters for each feature template could be formulated as an optimization problem. For example, we would want to minimize the validation error for the moving average feature with parameters $\{n, \lambda\}$, where n is the number of points to include, and λ is the exponential decay applied to the average. On the other hand, combinations of features could be optimized using feature selection methods. This could be done with a filter or a greedy wrapper, as was done for classification of glycemic variability (Chapter 4). Optimization of features should be done on a patient by patient basis, with the intuition that the features that optimize a patient model are different from patient to patient.

Another way to improve the accuracy of prediction would be to build models based on specific scenarios. For instance, training two models, one while the patient is sleeping and one while the patient is awake, may increase the overall accuracy of the predictions. Another example would be to use one model for weekdays and one model for weekends. This could also be done to handle insulin efficacy cycles, using the time of the last insulin reservoir change to partition the models.

Fully integrating the blood glucose prediction model with the 4DSS™ is also left for future work. One approach would be to use the predictions to add real-time situation assessment to the 4DSS™. The predictions would enable the situation assessment module to preemptively detect problems. When a problem is detected preemptively, the patient would be alerted and a solution for the problem would be generated. The system could then monitor the patient to check if the generated advice was followed, and if it was, record the outcome of the real-time solution. 4DSS™ would become a real-time case-based reasoning system, with automated extensions to the case-base. This extends the current 4DSS™ functionality, in that information about patient outcomes must now be manually entered.

Another way to integrate the predictions with the 4DSS™ is to create a module to critique advice generated by the adaptation module. The adaptation module makes several different suggestions for correcting detected problems. Some of these suggestions include administering a bolus or consuming carbohydrates. These suggestions can be sent as parameters to the blood glucose prediction module. Then, predictions can be made to determine the effect of the following the suggestions before they are recommended to a patient or health care provider.

There are three potential real-world applications for blood glucose prediction, beyond its role in 4DSS™. First, it could be incorporated in an educational tool, which could be used to educate health care professionals and patients. Such a tool would be able to show patients the effects of insulin, carbohydrates, and exercise on their blood glucose values. Second, the prediction model could be incorporated in the insulin pump, so that insulin pump alarms could be generated when hypoglycemia or hyperglycemia is predicted. This would contribute to patient safety. It would be especially useful for waking sleeping patients before impending nocturnal hypoglycemia. Third, the prediction model could be integrated into an artificial pancreas, helping to create a closed-loop system. This could benefit the JDRF's Artificial Pancreas Project. The model proposed in this work may prove to outperform the models currently envisioned by the JDRF. It offers the advantages of: (1) incorporating life event data known to impact blood glucose values; and (2) enabling patient specific models to be developed for each individual patient.

8 Summary and Conclusion

This thesis has presented research in machine learning for diabetes decision support. The contributions of this thesis are:

1. preprocessing noisy data, preparatory to applying machine learning algorithms;
2. enhancing the automated detection of excessive glycemic variability, a serious problem for patients with diabetes; and
3. predicting patient blood glucose levels, in order to preemptively detect and avoid potential health problems.

These contributions expand the scope of the 4 Diabetes Support System™ (4DSS) and potentially enable new clinical applications for diabetes management. At the time of this writing, the machine learning approaches for classifying glycemic variability and predicting blood glucose values offer new solutions to these problems.

As described in Chapter 3, preprocessing data included inferring life event data and smoothing CGM data. Sleep and work data was inferred from the patient's daily schedule, and carbohydrate data was inferred from information stored in the bolus wizard. Several techniques were investigated for smoothing CGM data, including moving averages, exponential moving averages, low pass DFT filter, ridge regression, and cubic spline smoothing. Cubic spline smoothing most closely matched the smoothing of an endocrinologist, due to its ability to give extra weight to fingersticks and local optima. Smoothing CGM data improved the performance of both classification of glycemic variability and prediction of blood glucose values.

As described in Chapter 4, the automated detection of excessive glycemic variability was improved via three different enhancements. These enhancements were smoothing CGM data, adding a rich set of pattern recognition features, and evaluating different classification algorithms. Feature selection was carried out using a *t*-test filter and a backward greedy elimination wrapper. The selected features were then used with k-fold cross validation to compare the results of the previous work with the new results. The enhancements resulted in better performance over the previous work, which had an accuracy of 87.1%. The backward greedy elimination obtained a best accuracy of 93.8% with a multilayer perceptron on smoothed data. The *t*-test filter obtained a best accuracy of 92.8% with a Support Vector Machine on smoothed data.

As described in Chapter 5, a new module was created for the 4DSS™ that predicts blood glucose values. Blood glucose prediction was formulated as a time series prediction problem, and the synthesis of two regression models was used to solve it. The two models were an AutoRegressive Integrated Moving Average (ARIMA) and Support Vector Regression (SVR). The ARIMA model was used as a baseline for the SVR model, and it was also used to supply features for the SVR model. A naïve baseline, which used the current blood glucose value as its prediction was used for comparison.

Three models were tested across 10 patients with two weeks of test data using walk-forward testing with a step size of one day, after tuning the feature templates for each patient. Two different SVR models were tested, one with insulin and life event data, and one without. This was done for 30 and 60 minute prediction horizons. Four different error metrics were recorded for each day: (1) Root Mean Square Error (RMSE), (2) Absolute Mean Error (MAE), (3) Coefficient of Determination (R^2), and (4) Clarke Error Grid Analysis (CEGA). The validation dataset was used to tune the parameters of the SVR. Not only were the parameters for the SVR tuned, but they were also adapted such that more recent training examples were given more emphasis during training. The rate at which the parameters were adapted was also tuned on the validation dataset.

For 30 minute predictions, the best RMSE of 4.5 mg/dl was obtained across all patients using either ARIMA or a SVR model without insulin and life event data. For 60 minute predictions, a best RMSE of 17.4 mg/dl was obtained across all patients using a SVR model without insulin and life event data. A SVR model with insulin and life event data did better than ARIMA at 60 minutes, and it outperformed a SVR model without insulin and life event data for three patients.

Future work is planned to build and improve upon the three contributions presented in this thesis. Future plans include: collecting life event data via a cellphone interface for better data accuracy; expanding the size of the glycemic variability dataset for improved classification; and investigating additional feature templates and modeling advances for better blood glucose prediction. There are also potential clinical applications of this work, including routine screening for excessive glycemic variability and safety alarms to alert patients to predicted hypoglycemia.

Bibliography

AIDA Diabetes Simulator (2011). AIDA free diabetic software simulator program of glucose-insulin action. http://www.2aida.net/aida/index.shtml, accessed May, 2011.

American Diabetes Association (2010a). Diabetes basics. http://www.diabetes.org/diabetes-basics/, accessed November, 2010.

American Diabetes Association (2010b). Type 1. http://www.diabetes.org/diabetes-basics/type-1, accessed Novemeber, 2010.

American Diabetes Association (2010c). Type 2. http://www.diabetes.org/diabetes-basics/type-2/, accessed November, 2010.

American Diabetes Association (2011). Diabetes statistics. http://www.diabetes.org/diabetes-basics/diabetes-statistics/, accessed May, 2011.

Bequette, B. W. (2010). Continuous glucose monitoring: Real-time algorithms for calibration, fitlering, and alarms. *Journal of Diabetes Science and Technology*, 4(2):404–418.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*, volume 4. Springer New York.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, New York.

Bolli, G. (2006). Glucose variability and complications. *Diabetes Care*, 29(7):1707–1709.

Box, G., Jenkins, G., and Reinsel, G. (2008). *Time Series Analysis: Forecasting and Control*. John Wiley, Hoboken, New Jersey, fourth edition.

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

Cao, L. and Tay, F. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518.

Centers for Disease Control and Prevention (2007). National diabetes fact sheet. http://www.cdc.gov/diabetes/pubs/pdf/ndfs_2007.pdf, accessed November, 2010.

Ceriello, A. and Ihnat, M. (2010). 'Glycaemic variability': A new therapeutic challenge in diabetes and the critical care setting. *Diabetic Medicine*, 27(8):862–867.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3). Article 15.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chang, M., Chen, B., and Lin, C. (2001). EUNITE network competition: Electricity load forecasting. Technical report, National Taiwan University.

Chu, W. and Keerthi, S. (2005). New approaches to support vector ordinal regression. In *Proceedings of the 22nd International Conference on Machine learning*, ICML'05, pages 145–152, New York, NY, USA. ACM.

Clarke, W., Cox, D., Gonder-Frederick, L., Carter, W., and Pohl, S. (1987). Evaluating clinical accuracy of systems for self-monitoring of blood glucose. *Diabetes care*, 10(5):622–628.

Cooper, T. (2010). Case adaptation for an intelligent decision support system for diabetes management. Master's thesis, Ohio University, Athens, Ohio.

Duke, D. L. (2009). *Intelligent Diabetes Assistant: A Telemedicine System for Modeling and Managing Blood Glucose*. PhD thesis, Carnegie Mellon University.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations Newsletter*, 11:10–18.

Hirsch, I. B. and Brownlee, M. (2005). Should minimal blood glucose variability become the gold standard of glycemic control? *Journal of Diabetes and its Complications*, 19(3):178–181.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthognoal problems. *Technometrics*, 12(1):55–67.

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

JDRF (2011a). *JDRF Artifcial Pancreas Project*. http://artificialpancreasproject.com, accessed May, 2011.

JDRF (2011b). *JDRF Artificial Pancreas Project FAQ*. http://www.artificialpancreasproject.com/faq/default.html, accessed May, 2011.

Juvenile Diabetes Research Foundation (2010). Countdown, An Artificial Pancreas, How Close Are We To Closing The Loop? http://www.jdrf.org/files/APP/Countdown 06.pdf, accessed November, 2010.

Kilpatrick, E. S., Rigby, A. S., and Atkin, S. L. (2006). The effect of glucose variability on the risk of microvascular complications in type 1 diabetes. *Diabetes Care*, 29(7):1486–1490.

Kilpatrick, E. S., Rigby, A. S., and Atkin, S. L. (2010). For debate. Glucose variability and diabetes complication risk: We need to know the answer. *Diabetic Medicine*, 27(8):868–871.

Kim, K. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307–319.

Kovatchev, B., Gonder-Frederick, L., Cox, D., and Clarke, W. (2004). Evaluating the accuracy of continuous glucose-monitoring sensors. *Diabetes Care*, 27(8):1922–1928.

Kowalski, A. (2009). Can we really close the loop and how soon? Accelerating the availability of an artificial pancreas: A roadmap to better diabetes outcomes. *Diabetes Technology & Therapeutics*, 11(S1):113–119.

Lehmann, E. and Deutsch, T. (1991a). A clinical model of glucose-insulin interaction. In *Proceedings of the MIE9́1 Satellite Conference on Computer Modelling*, pages 101–111.

Lehmann, E. and Deutsch, T. (1991b). A physiological model of glucose-insulin interaction. In *Proceedings of the 13th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 13, pages 235–242. IEEE.

Lehmann, E., Tarin, C., Bondia, J., Teufel, E., and Deutsch, T. (2011). Development of AIDA v4. 3b Diabetes Simulator: Technical Upgrade to Support Incorporation of Lispro, Aspart, and Glargine Insulin Analogues. *Journal of Electrical and Computer Engineering*, 2011:1–17.

Maimone, A. (2006). Data and knowledge acquisition in case-based reasoning for diabetes management. Master's thesis, Ohio University, Athens, Ohio.

Marling, C., Shubrook, J., and Schwartz, F. (2008). Case-based decision support for patients with type 1 diabetes on insulin pump therapy. In Althof, K.-D., Bergmann, R., Minor, M., and Hanft, A., editors, *Advances in Case-Based Reasoning: 9th European Conference, ECCBR*, pages 325–339, Berlin. Springer.

Marling, C., Shubrook, J., and Schwartz, F. (2009a). Toward case-based reasoning for diabetes management: A preliminary clinical study and decision support system prototype. *Computational Intelligence*, 25(3):165–179.

Marling, C., Vernier, S., Cooper, T., Shubrook, J., and Schwartz, F. (2009b). Evaluating the automated detection of blood glucose control problems. In *Ninth Annual Diabetes Technology Meeting, Diabetes Technology Society*, San Fransisco, CA, USA. Abstract A103.

Marling, C., Vernier, S., Cooper, T., Shubrook, J., and Schwartz, F. (2009c). Towards evaluation of a medical case-based decision support system. In *Workshop Proceedings of the 8th International Conference on Case-Based Reasoning*, Seattle, Washington.

Marling, C. R., Shubrook, J. H., Vernier, S. J., Wiley, M. T., and Schwartz, F. L. (2011). Characterizing blood glucose variability using new metrics with continuous glucose monitoring data. *Journal of Diabetes Science and Technology, in press*.

Mastrototaro, J., Shin, J., Marcus, A., and Sulur, G. (2008). The accuracy and efficacy of real-time continuous glucose monitoring sensor in patients with type 1 diabetes. *Diabetes Technology & Therapeutics*, 10(5):385–390.

Miller, W. A. (2009). Problem Detection for Situation Assessment in Case-Based Reasoning for Diabetes Management. Master's thesis, Ohio University, Athens, Ohio.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Monnier, L. and Colette, C. (2008). Glycemic variability. *Diabetes Care*, 31(s2):S–150–S–154.

Monnier, L., Mas, E., Ginet, C., Michel, F., Villon, L., Cristol, J., and Colette, C. (2006). Activation of oxidative stress by acute glucose fluctuations compared with sustained chronic hyperglycemia in patients with type 2 diabetes. *Journal of the American Medical Association*, 295(14):1681–1687.

Pai, P. and Hong, W. (2005). Support vector machines with simulated annealing algorithms in electricity load forecasting. *Energy Conversion and Management*, 46(17):2669–2688.

Pollock, D. (1993). Smoothing with cubic splines. Technical report, Queen Mary and Westfield College, The University of London.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Refenes, A., Bentz, Y., Bunn, D., Burgess, A., and Zapranis, A. (1997). Financial time series modeling with discounted least squares back-propagation. *Neurocomputing*, 14:123–138.

Robard, D. (2009). Interpretation of continuous glucose monitoring data: glycemic variability and quality of glycemic control. *Diabetes Technology & Therapeutics*, 11(s1):S–55–S–67.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65:386–408.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). *Learning internal representations by error propagation*, pages 318–362. MIT Press, Cambridge, MA, USA.

Sapankevych, N. I. and Sankar, R. (2009). Time series prediction using support vector machines: A survey. *Computational Intelligence Magazine, IEEE*, 4(2):24–38.

Scholkopf, B. and Smola, A.J. (2002). *Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond.* MIT Press, Cambridge, MA.

Schwartz, F., Shubrook, J., and Marling, C. (2008). Use of case-based reasoning to enhance intensive management of patients on insulin pump therapy. *Journal of Diabetes Science and Technology*, 2(4):603–611.

Schwartz, F., Vernier, S., Shubrook, J., and Marling, C. (2010). Evaluating the automated blood glucose pattern detection and case-retrieval modules of the 4 Diabetes Support System (TM). *Journal of Diabetes Science and Technology*, 4(6):1563–1569.

Service, F. J., Molnar, G. D., Rosevear, J. W., Ackerman, E., Gatewood, L. C., and Taylor, W. (1970). Mean amplitude of glycemic excursions, a measure of diabetic instability. *Diabetes*, 19(9):644–655.

Smola, A.J. and Scholkopf, B. (2004). A tutorial on support vector regression. In *Statistics and Computing*, volume 14, pages 199–222. Springer, Netherlands.

Sparacino, G., Facchinetti, A., Maran, A., and Cobelli, C. (2008). Continuous glucose monitoring time series and hypo/hyperglycemia prevention: Requirements, methods, open problems. *Current Diabetes Reviews*, 4(3):181–192.

Stanley, W., Dougherty, G., and Dougherty, R. (1983). *Digital Signal Processing*. Reston Publishing Company, Reston, VA, second edition.

Tay, F. and Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega-International Journal of Management Science*, 29(4):309–317.

Tay, F. and Cao, L. (2002a). Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48:847–861.

Tay, F. and Cao, L. (2002b). ε-Descending support vector machines for financial time series forecasting. *Neural Processing Letters*, 15(2):179–195.

The Diabetes Control and Complications Trial Research Group (1993). The effect of intensive treatment of diabetes on the development and progression of long-term complications in insulin-dependent diabetes mellitus. *New England Journal of Medicine*, 329(14):977–986.

Theodoridis, S. and Koutroumbas, K. (2009). *Pattern Recognition*. Elsevier, fourth edition.

Tong, S. and Koller, D. (2000). Support vector machine active learning with applications to text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML'00, pages 999–1006, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Vapnik, V. (1979). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag New York Inc.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley-Interscience, New York.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin.

Vernier, S. J. (2009). Clinical evaluation and enhancement of a medical case-based decision support system. Master's thesis, Ohio University.

Walker, D. (2007). Similarity determination and case retrieval in an intelligent decision support system for diabetes management. Master's thesis, Ohio University, Athens, Ohio.

Welch, B. (1947). The generalization of student's problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35.

Wiley, M. T., Bunescu, R., Marling, C., Shubrook, J., and Schwartz, F. (2011). Automatic detection of excessive glycemic variability for diabetes management. Submitted to ICMLA 2011.

Appendix: Prediction Results

This appendix contains detailed results for the blood glucose prediction models at all horizons for all patients. This appendix is broken down into 10 sections, one section for each patient. Each section is broken down into two subsections, for 30 and 60 minute horizons. Each patient section begins with the features found by tuning the feature templates. The baseline features, $\varphi_{t_0}(x, t)$ and $\varphi_{ARIMA}(x, t, l)$, are omitted from the tables containing the feature vectors as these features were not tuned. Each prediction horizon subsection presents a plot of the RMSE of the walk-forward testing for all models and baselines, along with a table showing the averages for all 4 error metrics from this test data.

A.1 Patient 301

Table A.1 shows the feature vectors found from tuning the feature templates for patient 301 at 30 and 60 minute prediction horizons.

Table A.1: Feature vectors for patient 301.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 0.5$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 0.5$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 0.5$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 0.5$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 1$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 1$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 1$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 1$	$\lambda = 0.25$
$\varphi_{Bolus}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 10, i = \{0, \ldots, 5\}$
$\varphi_{Basal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{BasalArea}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Meal}(x, t - i\,k, k)$	$k = 10, i = \{0, \ldots, 5\}$	$k = 10, i = \{0, \ldots, 5\}$
$\varphi_{Sleep}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$

A.1.1 30 Minutes

Table A.2: Patient 301 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	15.6	2.8	3.2	3.1
MAE	12.4	2.0	2.5	2.4
R^2	0.81	0.99	0.99	0.99
CEGA (A%)	91%	100%	100%	100%

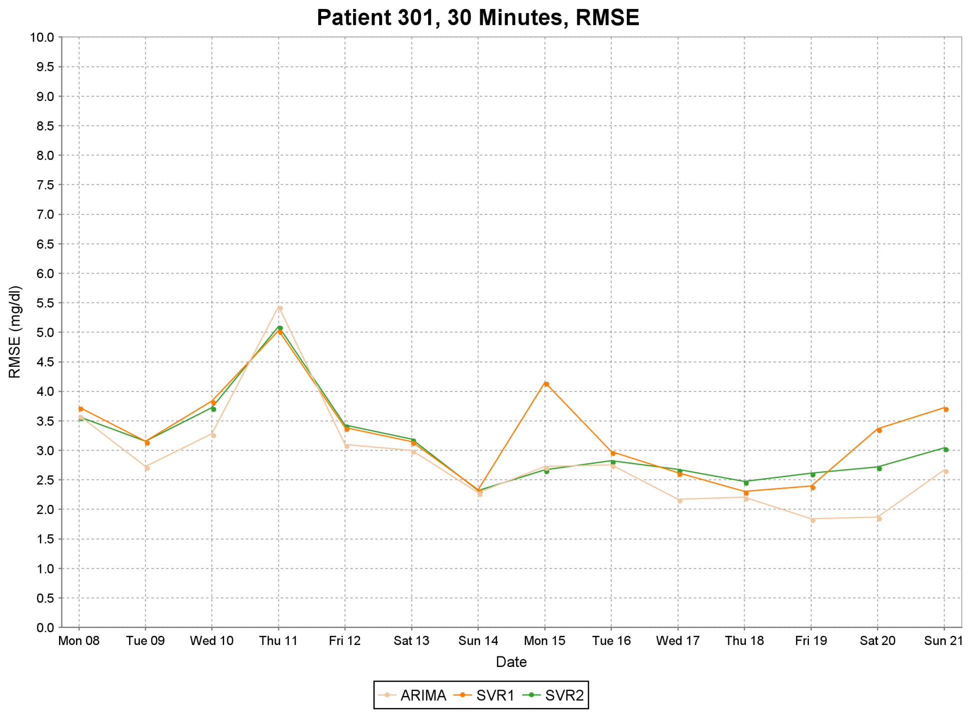


Figure A.1: RMSE plot of test data for patient 301 with 30 minute predictions.

A.1.2 60 Minutes

Table A.3: Patient 301 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	28.8	12.5	12.0	12.2
MAE	22.8	9.5	9.4	9.4
R^2	0.36	0.88	0.89	0.88
CEGA (A%)	68%	96%	96%	96%

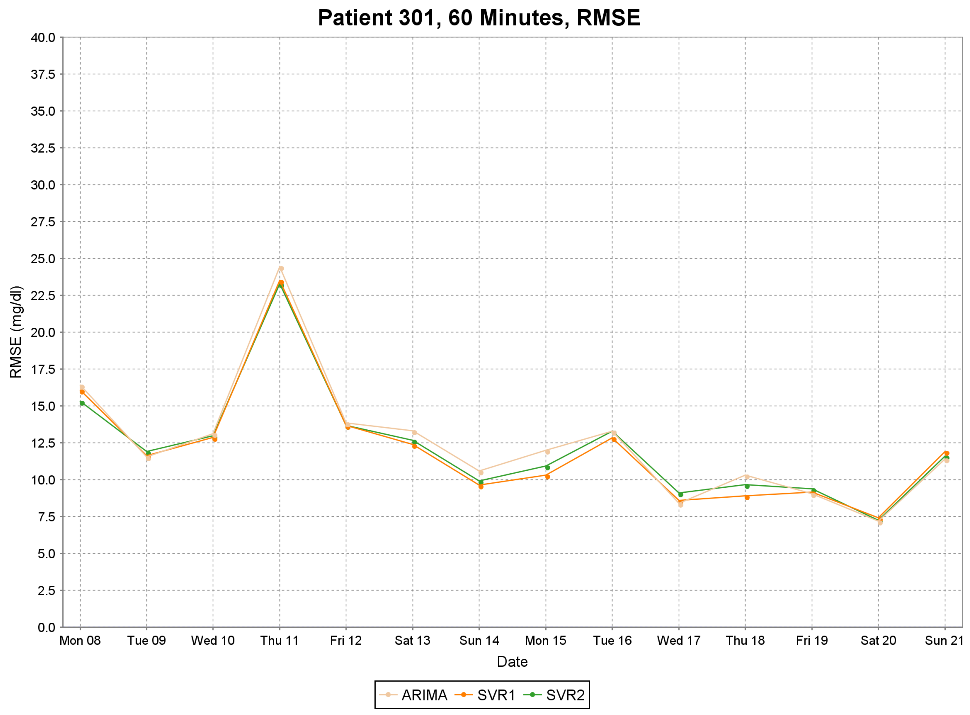


Figure A.2: RMSE plot of test data for patient 301 with 60 minute predictions.

A.2 Patient 302

Table A.4 shows the feature vectors found from tuning the feature templates for patient 302 at 30 and 60 minute prediction horizons.

Table A.4: Feature vectors for patient 302.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 0.25$	$\lambda = 1$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 0.25$	$\lambda = 1$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 0.25$	$\lambda = 1$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 0.25$	$\lambda = 1$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 0.25$	$\lambda = 1$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 0.25$	$\lambda = 0.75$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 0.25$	$\lambda = 0.75$
$\varphi_{Bolus}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Basal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 10, i = \{0, \ldots, 5\}$
$\varphi_{BasalArea}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Meal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Exercise}(x, t - i\,k, k)$	$k = 30, i = \{0, \ldots, 5\}$	$k = 10, i = \{0, \ldots, 17\}$
$\varphi_{Sleep}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$

A.2.1 30 Minutes

Table A.5: Patient 302 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	17.0	5.3	5.5	5.3
MAE	13.2	4.0	4.1	3.9
R^2	0.77	0.97	0.97	0.97
CEGA (A%)	89%	99%	99%	99%

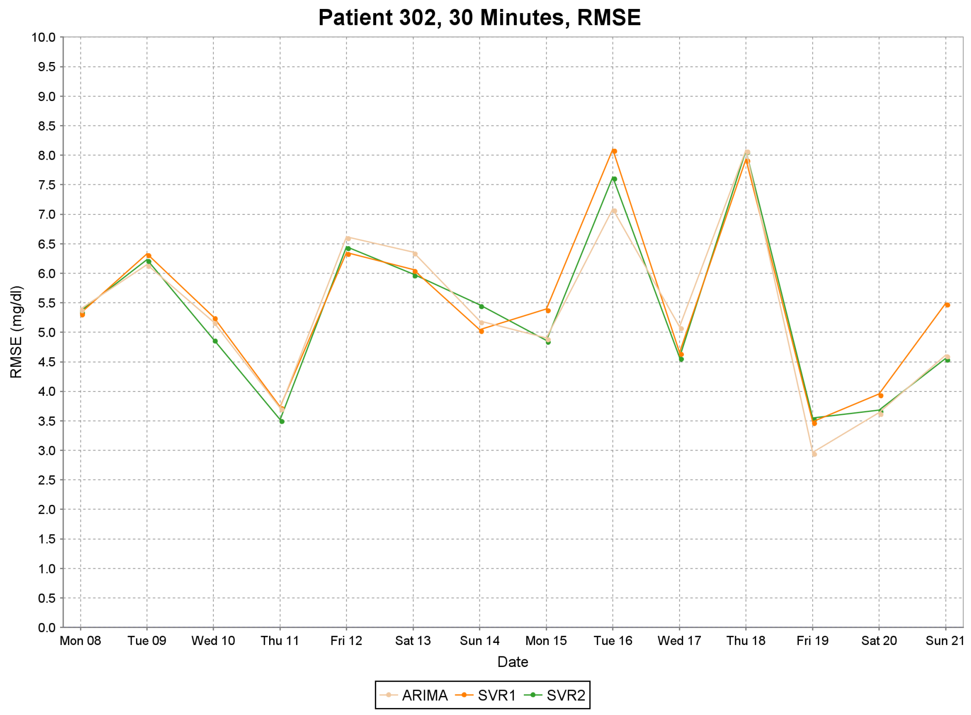


Figure A.3: RMSE plot of test data for patient 302 with 30 minute predictions.

A.2.2 60 Minutes

Table A.6: Patient 302 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	29.7	18.2	18.4	17.5
MAE	30.6	15.2	13.8	13.3
R^2	0.30	0.72	0.72	0.75
CEGA (A%)	66%	88%	87%	88%

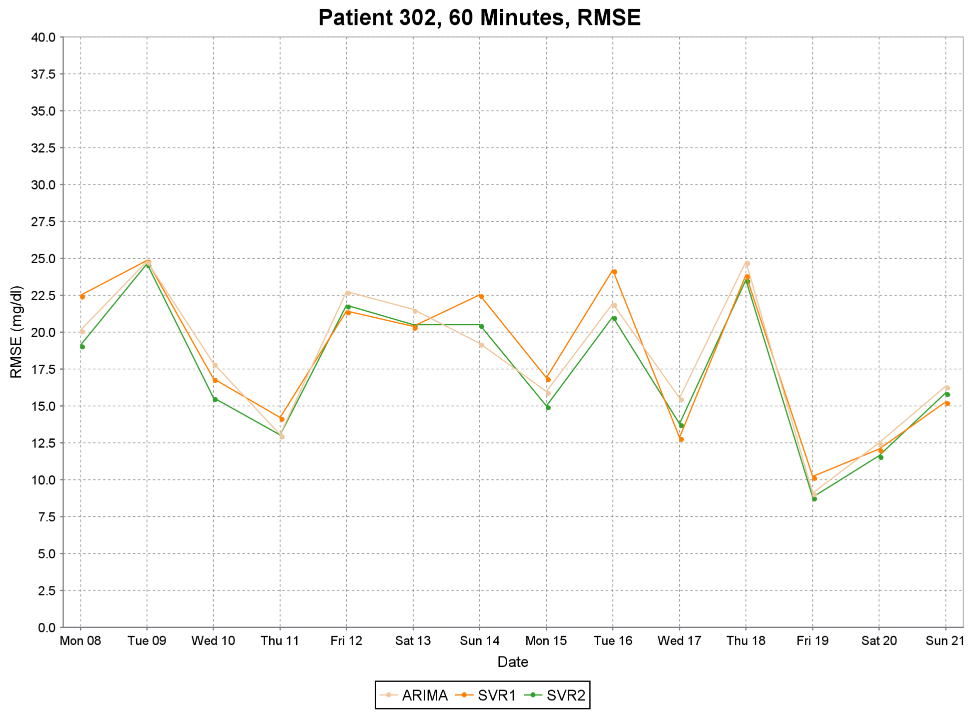


Figure A.4: RMSE plot of test data for patient 302 with 60 minute predictions.

A.3 Patient 304

Table A.7 shows the feature vectors found from tuning the feature templates for patient 304 at 30 and 60 minute prediction horizons.

Table A.7: Feature vectors for patient 304.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 0.25$	$\lambda = 1$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 0.25$	$\lambda = 1$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 0.25$	$\lambda = 0.9$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 0.25$	$\lambda = 0.75$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 0.25$	$\lambda = 0.75$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 0.25$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 0.25$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 0.25$	$\lambda = 0.5$
$\varphi_{Bolus}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 10, i = 0$
$\varphi_{Basal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{BasalArea}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 30, i = \{0, 1\}$
$\varphi_{Meal}(x, t - i\,k, k)$	$k = 10, i = \{0, \ldots, 5\}$	$k = 10, i = \{0, \ldots, 5\}$
$\varphi_{Exercise}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Work}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Sleep}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$

A.3.1 30 Minutes

Table A.8: Patient 304 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	20.7	4.0	4.4	4.2
MAE	16.3	3.0	3.3	3.2
R^2	0.76	0.99	0.99	0.98
CEGA (A%)	82%	99%	99%	99%

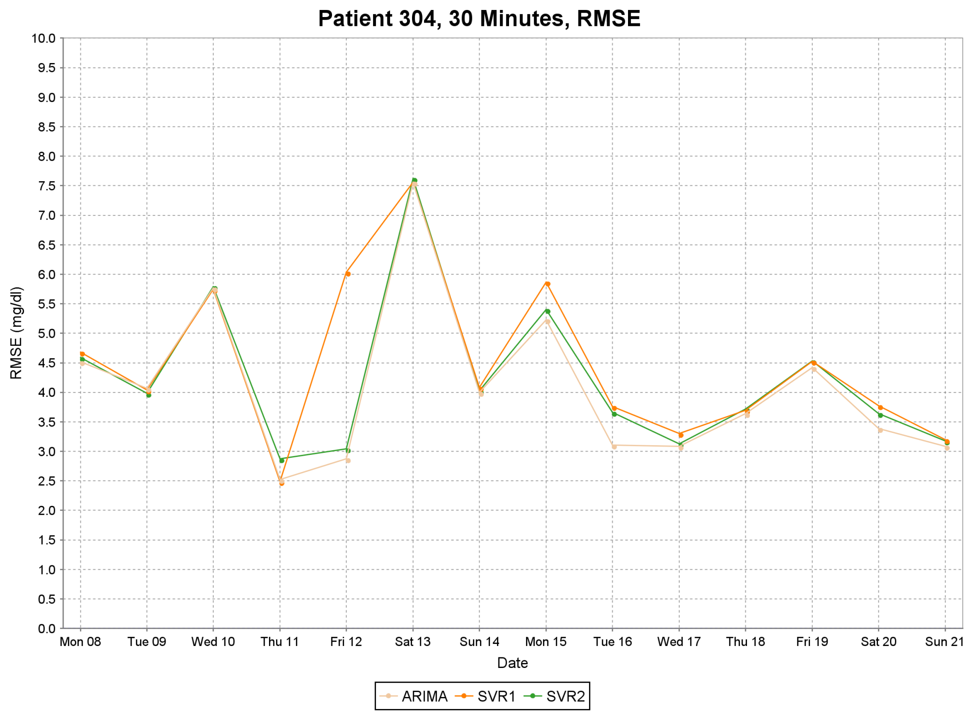


Figure A.5: RMSE plot of test data for patient 304 with 30 minute predictions.

A.3.2 60 Minutes

Table A.9: Patient 304 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	37.7	17.2	17.3	17.2
MAE	29.7	13.3	13.5	13.3
R^2	0.24	0.82	0.82	0.82
CEGA (A%)	58%	87%	85%	86%

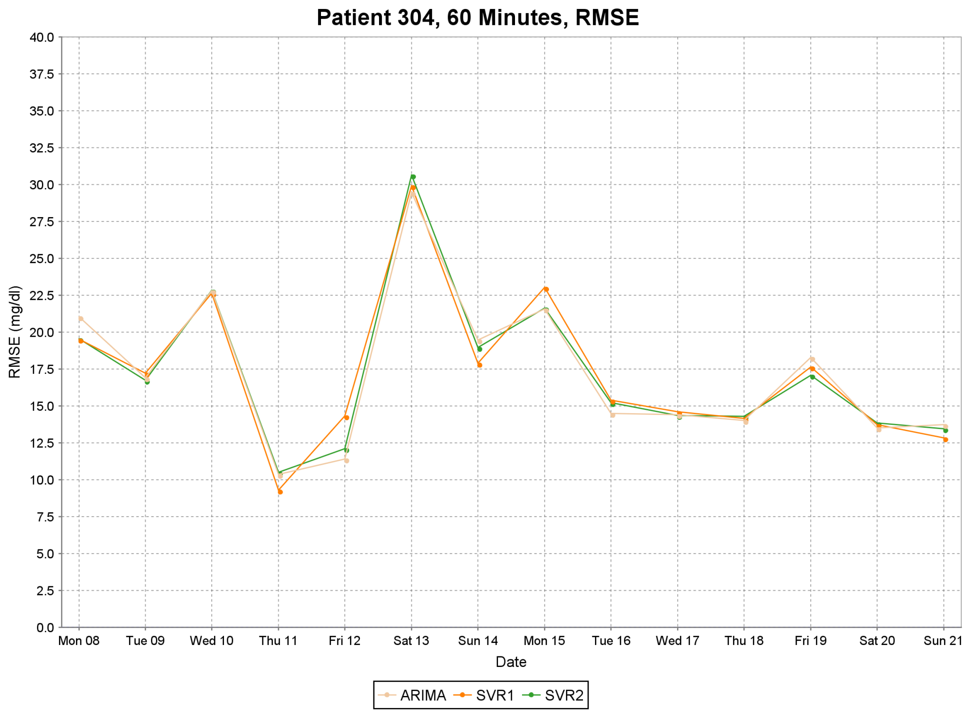


Figure A.6: RMSE plot of test data for patient 304 with 60 minute predictions.

A.4 Patient 306

Table A.10 shows the feature vectors found from tuning the feature templates for patient 306 at 30 and 60 minute prediction horizons.

Table A.10: Feature vectors for patient 306.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 0.25$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 0.25$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{Bolus}(x, t - i\,k, k)$	$k = 60,\, i = 0$	$k = 20,\, i = \{0, 1, 2\}$
$\varphi_{Basal}(x, t - i\,k, k)$	$k = 60,\, i = 0$	$k = 60,\, i = 0$
$\varphi_{BasalArea}(x, t - i\,k, k)$	$k = 60,\, i = 0$	$k = 60,\, i = 0$
$\varphi_{Meal}(x, t - i\,k, k)$	$k = 60,\, i = 0$	$k = 60,\, i = 0$
$\varphi_{Exercise}(x, t - i\,k, k)$	$k = 60,\, i = \{0, 1\}$	$k = 30,\, i = \{0, 1\}$
$\varphi_{Work}(x, t - i\,k, k)$	$k = 60,\, i = 0$	$k = 60,\, i = 0$
$\varphi_{Sleep}(x, t - i\,k, k)$	$k = 60,\, i = 0$	$k = 60,\, i = \{0, 1\}$

A.4.1 30 Minutes

Table A.11: Patient 306 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	26.5	6.1	6.1	6.0
MAE	20.4	4.5	4.5	4.4
R^2	0.81	0.98	0.98	0.98
CEGA (A%)	76%	99%	99%	99%

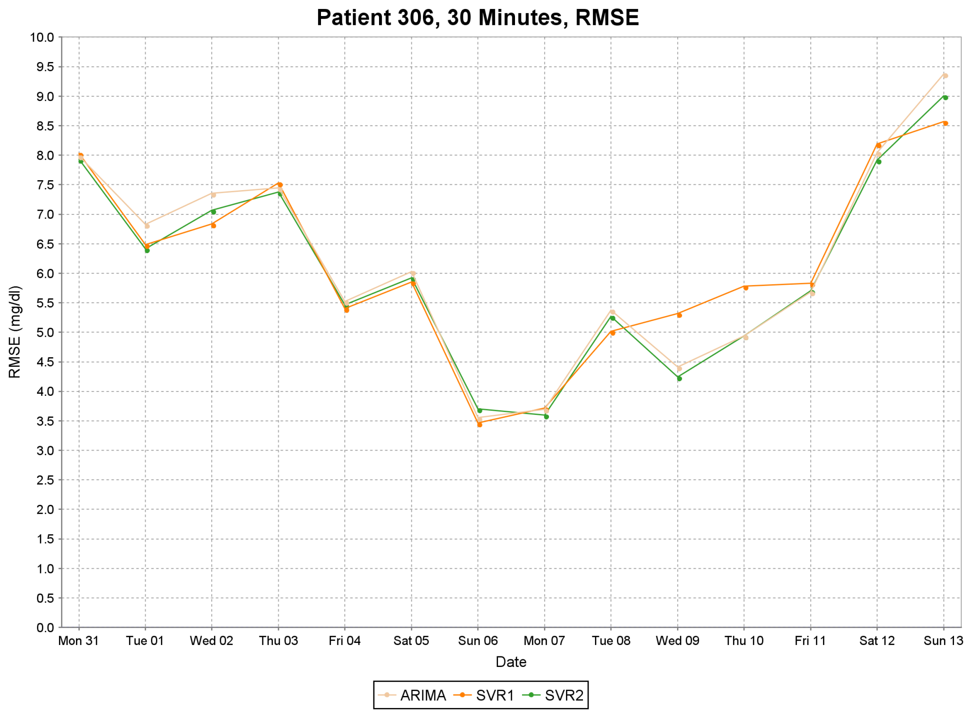


Figure A.7: RMSE plot of test data for patient 306 with 30 minute predictions.

A.4.2 60 Minutes

Table A.12: Patient 306 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	47.7	23.9	24.7	23.7
MAE	37.3	18.4	18.8	17.8
R^2	0.37	0.83	0.82	0.83
CEGA (A%)	51%	83%	82%	83%

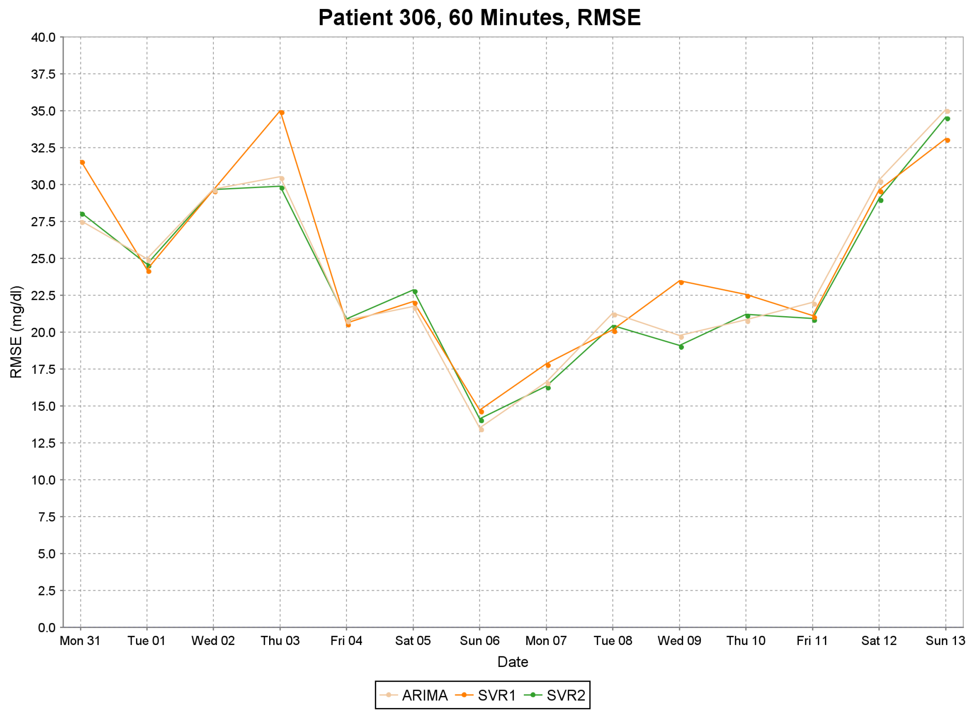


Figure A.8: RMSE plot of test data for patient 306 with 60 minute predictions.

A.5 Patient 307

Table A.13 shows the feature vectors found from tuning the feature templates for patient 307 at 30 and 60 minute prediction horizons.

Table A.13: Feature vectors for patient 307.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 1$	$\lambda = 0.5$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 1$	$\lambda = 1$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 1$	$\lambda = 0.9$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 1$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 0.75$	$\lambda = 1$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 1$	$\lambda = 1$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 1$	$\lambda = 0.75$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 1$	$\lambda = 0.5$
$\varphi_{Bolus}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 10, i = \{0, \ldots, 11\}$
$\varphi_{Basal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{BasalArea}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Meal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = \{0, 1, 2\}$
$\varphi_{Work}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Sleep}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$

A.5.1 30 Minutes

Table A.14: Patient 307 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	17.6	4.2	4.6	4.3
MAE	13.9	3.1	3.5	3.3
R^2	0.86	0.99	0.99	0.96
CEGA (A%)	92%	100%	100%	100%

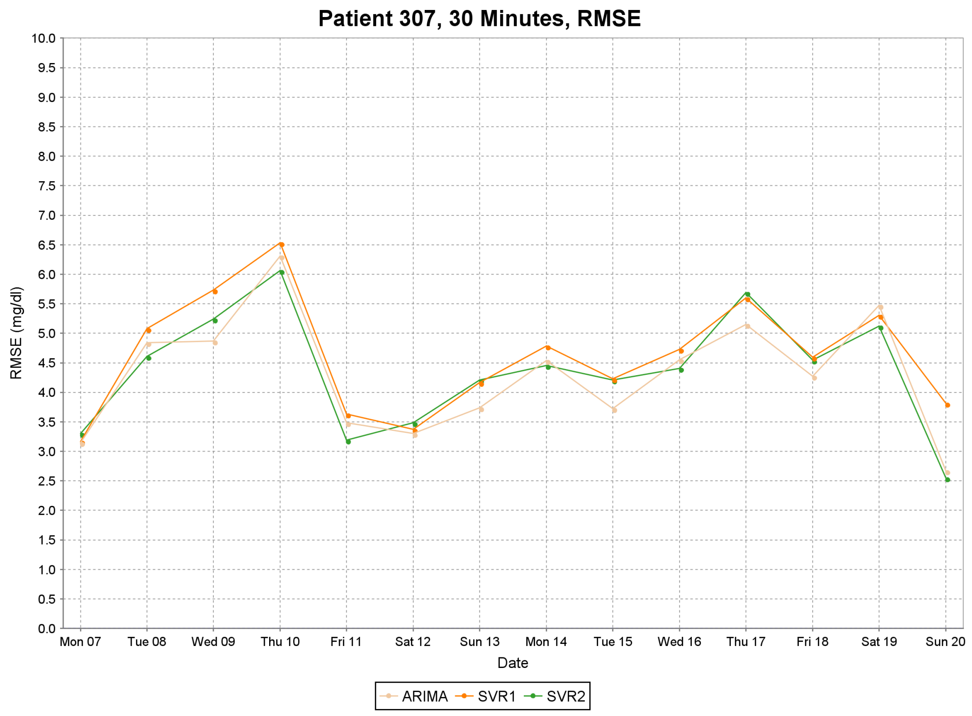


Figure A.9: RMSE plot of test data for patient 307 with 30 minute predictions.

A.5.2 60 Minutes

Table A.15: Patient 307 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	32.6	16.8	16.7	16.5
MAE	26.0	12.7	12.7	12.8
R^2	0.54	0.84	0.84	0.83
CEGA (A%)	67%	91%	90%	90%

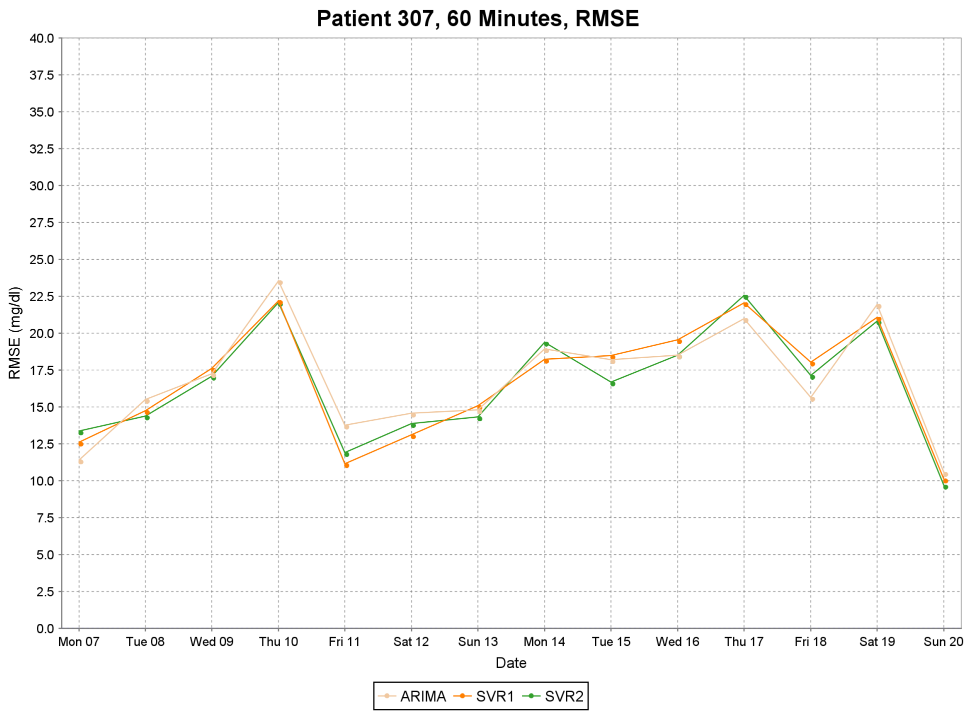


Figure A.10: RMSE plot of test data for patient 307 with 60 minute predictions.

A.6 Patient 308

Table A.16 shows the feature vectors found from tuning the feature templates for patient 308 at 30 and 60 minute prediction horizons.

Table A.16: Feature vectors for patient 308.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 1$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 0.75$	$\lambda = 1$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 0.75$	$\lambda = 1$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 0.75$	$\lambda = 0.9$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 0.9$	$\lambda = 1$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 1$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 1$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 0.9$	$\lambda = 0.5$
$\varphi_{Bolus}(x, t - i\,k, k)$	$k = 30, i = \{0, 1\}$	$k = 20, i = \{0, 1, 2\}$
$\varphi_{Basal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{BasalArea}(x, t - i\,k, k)$	$k = 10, i = \{0, \ldots, 5\}$	$k = 10, i = \{0, \ldots, 5\}$
$\varphi_{Meal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 30, i = \{0, 1\}$
$\varphi_{Exercise}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = \{0, 1\}$
$\varphi_{Work}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Sleep}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$

A.6.1 30 Minutes

Table A.17: Patient 308 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	19.2	4.8	5.2	4.9
MAE	14.0	3.3	3.8	3.5
R^2	0.76	0.99	0.99	0.99
CEGA (A%)	89%	99%	99%	99%

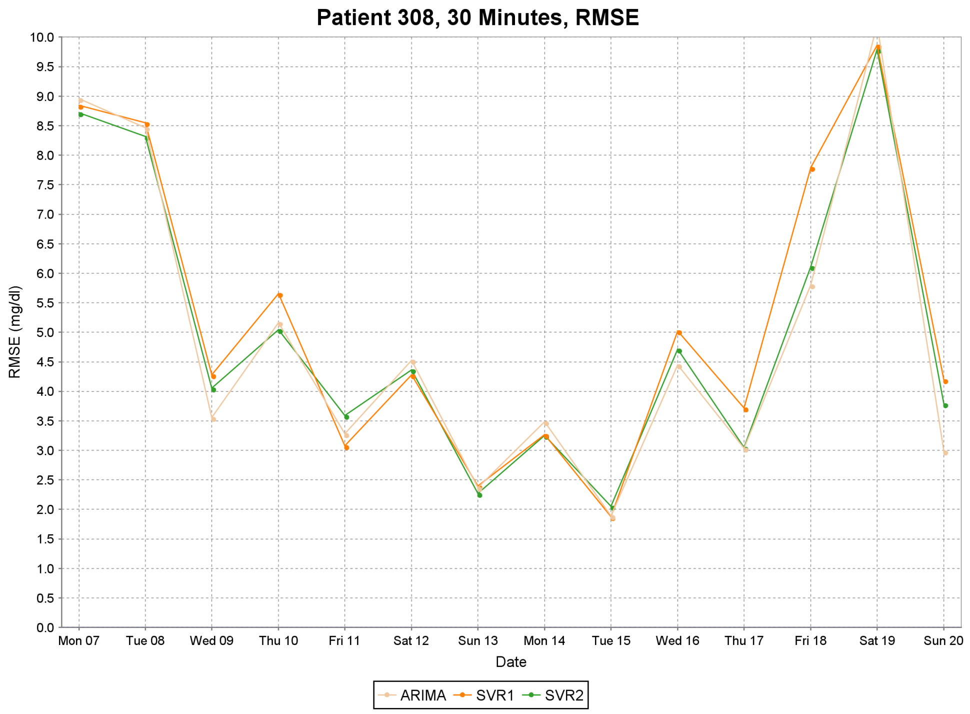


Figure A.11: RMSE plot of test data for patient 308 with 30 minute predictions.

A.6.2 60 Minutes

Table A.18: Patient 308 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	3.4.3	18.4	18.2	18.2
MAE	25.1	13.3	13.3	13.2
R^2	0.23	0.77	0.77	0.77
CEGA (A%)	69%	90%	90%	90%

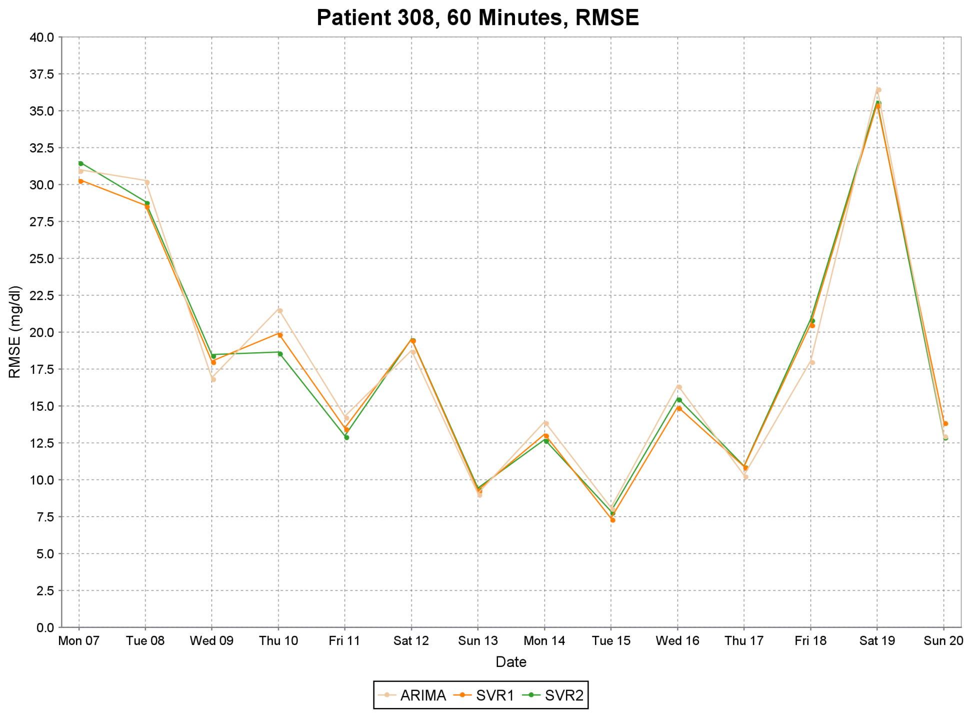


Figure A.12: RMSE plot of test data for patient 308 with 60 minute predictions.

A.7 Patient 309

Table A.19 shows the feature vectors found from tuning the feature templates for patient 309 at 30 and 60 minute prediction horizons.

Table A.19: Feature vectors for patient 309.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 0.5$	$\lambda = 1$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 0.9$	$\lambda = 1$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 1$	$\lambda = 0.9$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 1$	$\lambda = 0.75$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 0.25$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{Bolus}(x, t - i\ k, k)$	$k = 60, i = 0$	$k = 60, i = \{0, \ldots, 3\}$
$\varphi_{Basal}(x, t - i\ k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{BasalArea}(x, t - i\ k, k)$	$k = 60, i = 0$	$k = 10, i = \{0, \ldots, 5\}$
$\varphi_{Meal}(x, t - i\ k, k)$	$k = 60, i = 0$	$k = 20, i = \{0, 1, 2\}$
$\varphi_{Work}(x, t - i\ k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Sleep}(x, t - i\ k, k)$	$k = 60, i = 0$	$k = 15, i = \{0, \ldots, 3\}$

A.7.1 30 Minutes

Table A.20: Patient 309 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	22.2	4.9	5.0	4.9
MAE	17.5	3.6	3.8	3.7
R^2	0.83	0.99	0.99	0.99
CEGA (A%)	87%	100%	100%	100%

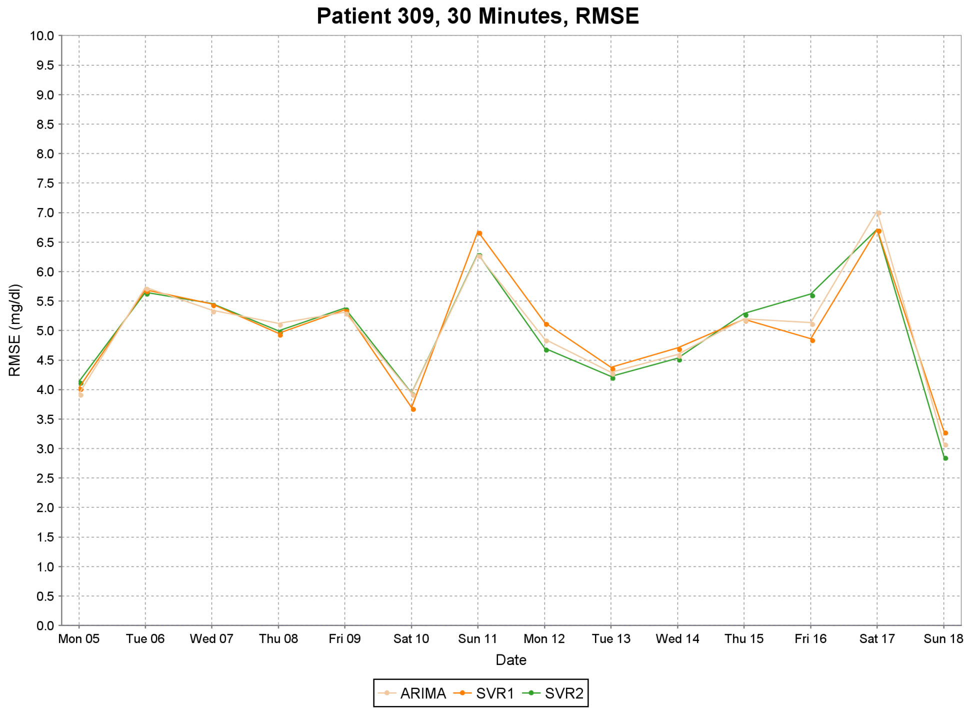


Figure A.13: RMSE plot of test data for patient 309 with 30 minute predictions.

A.7.2 60 Minutes

Table A.21: Patient 309 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	40.7	20.7	19.6	19.9
MAE	32.3	15.8	13.1	15.0
R^2	0.43	0.85	0.86	0.86
CEGA (A%)	65%	89%	90%	89%

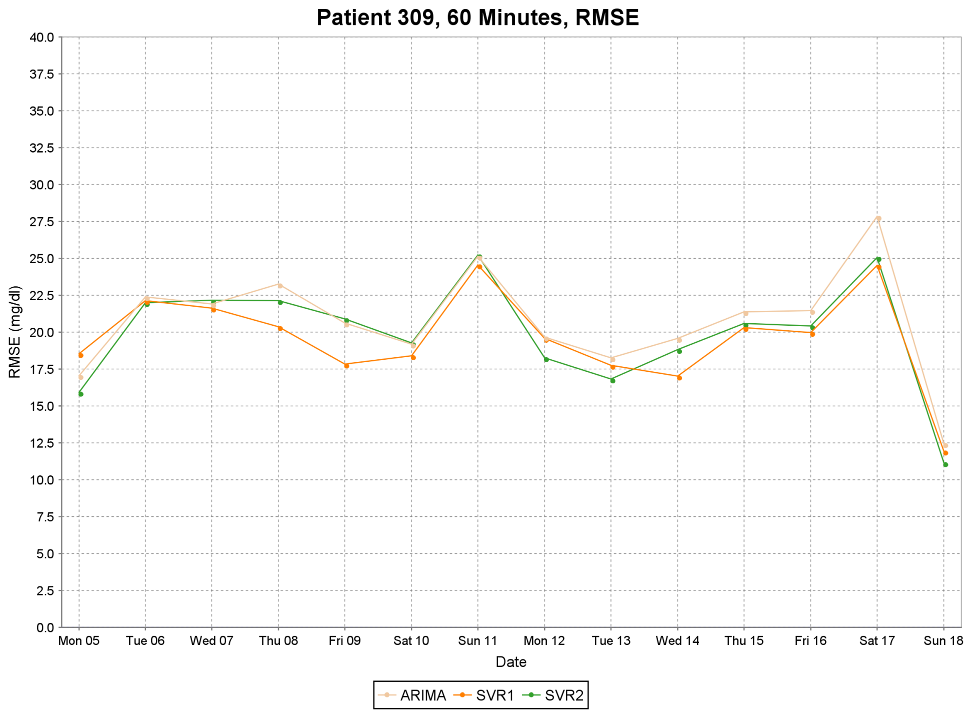


Figure A.14: RMSE plot of test data for patient 309 with 60 minute predictions.

A.8 Patient 310

Table A.22 shows the feature vectors found from tuning the feature templates for patient 310 at 30 and 60 minute prediction horizons.

Table A.22: Feature vectors for patient 310.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 0.75$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 0.75$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 0.75$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 0.5$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 1$	$\lambda = 0.9$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 1$	$\lambda = 1$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 0.75$	$\lambda = 0.75$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 0.5$	$\lambda = 0.5$
$\varphi_{Bolus}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 15, i = \{0, \ldots, 3\}$
$\varphi_{Basal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{BasalArea}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Meal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Exercise}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Sleep}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = \{0, 1\}$

A.8.1 30 Minutes

Table A.23: Patient 310 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	22.0	5.8	6.1	5.8
MAE	17.1	4.0	4.4	4.2
R^2	0.78	0.98	0.98	0.98
CEGA (A%)	85%	99%	99%	99%

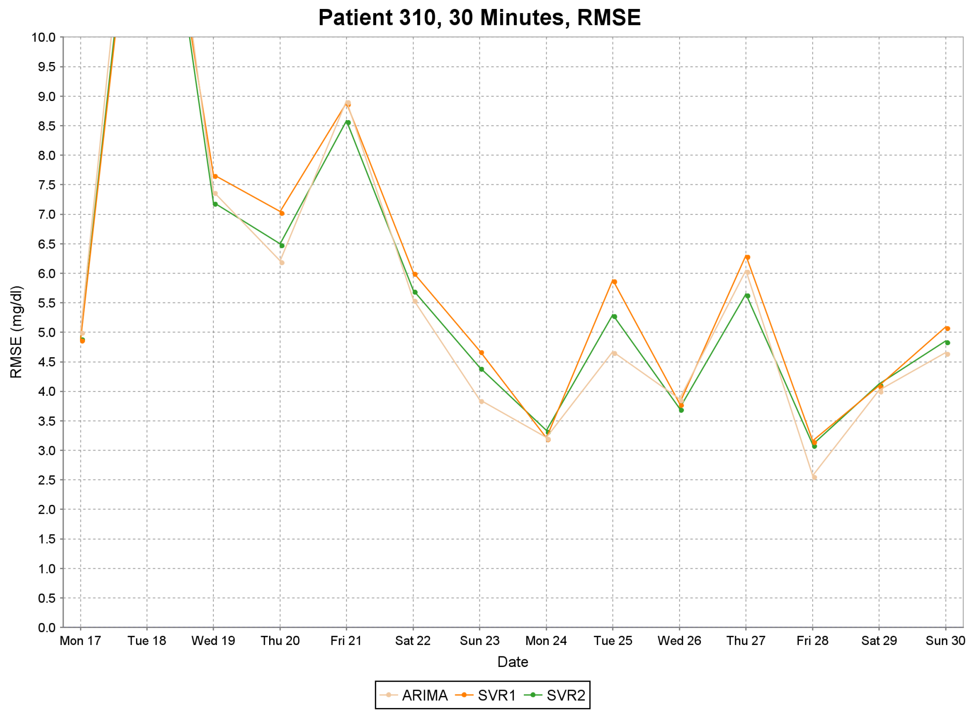


Figure A.15: RMSE plot of test data for patient 310 with 30 minute predictions.

A.8.2 60 Minutes

Table A.24: Patient 310 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	40.0	23.7	23.2	22.4
MAE	31.5	17.3	17.1	16.7
R^2	0.27	0.74	0.75	0.76
CEGA (A%)	58%	84%	85%	86%

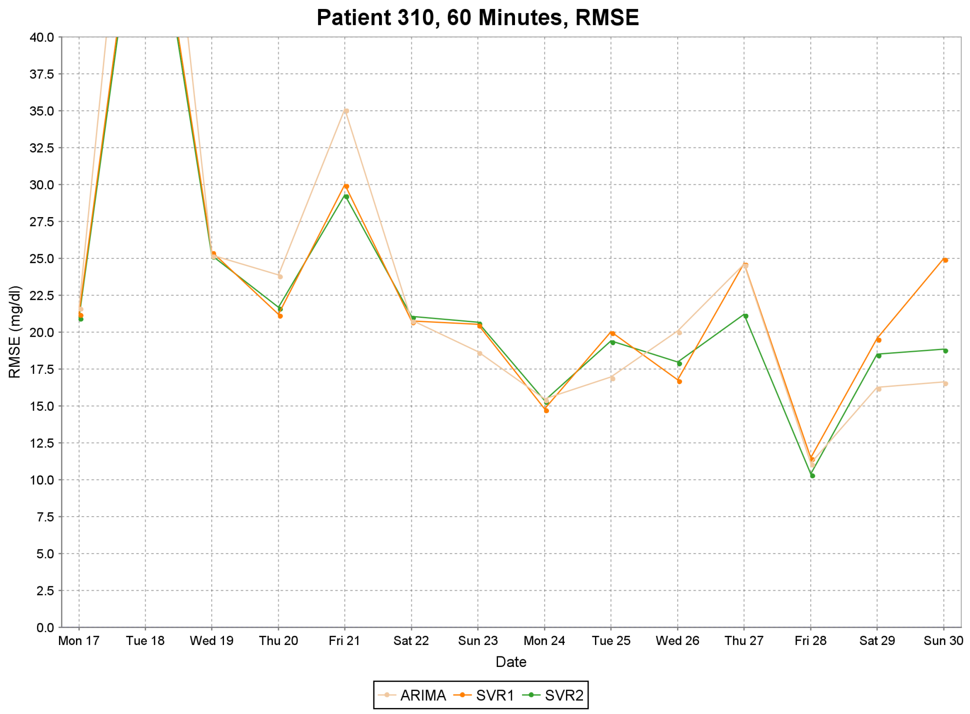


Figure A.16: RMSE plot of test data for patient 310 with 60 minute predictions.

A.9 Patient 312

Table A.25 shows the feature vectors found from tuning the feature templates for patient 312 at 30 and 60 minute prediction horizons.

Table A.25: Feature vectors for patient 312.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 0.25$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 0.75$	$\lambda = 1$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 0.9$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 0.5$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 0.5$	$\lambda = 0.5$
$\varphi_{Bolus}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 30, i = \{0, 1\}$
$\varphi_{Basal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{BasalArea}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Meal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 20, i = \{0, 1, 2\}$
$\varphi_{Work}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 10, i = \{0, \dots, 5\}$
$\varphi_{Sleep}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$

A.9.1 30 Minutes

Table A.26: Patient 312 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	15.6	3.3	3.6	3.6
MAE	12.7	2.4	2.7	2.8
R^2	0.84	0.99	0.99	0.99
CEGA (A%)	92%	100%	100%	100%

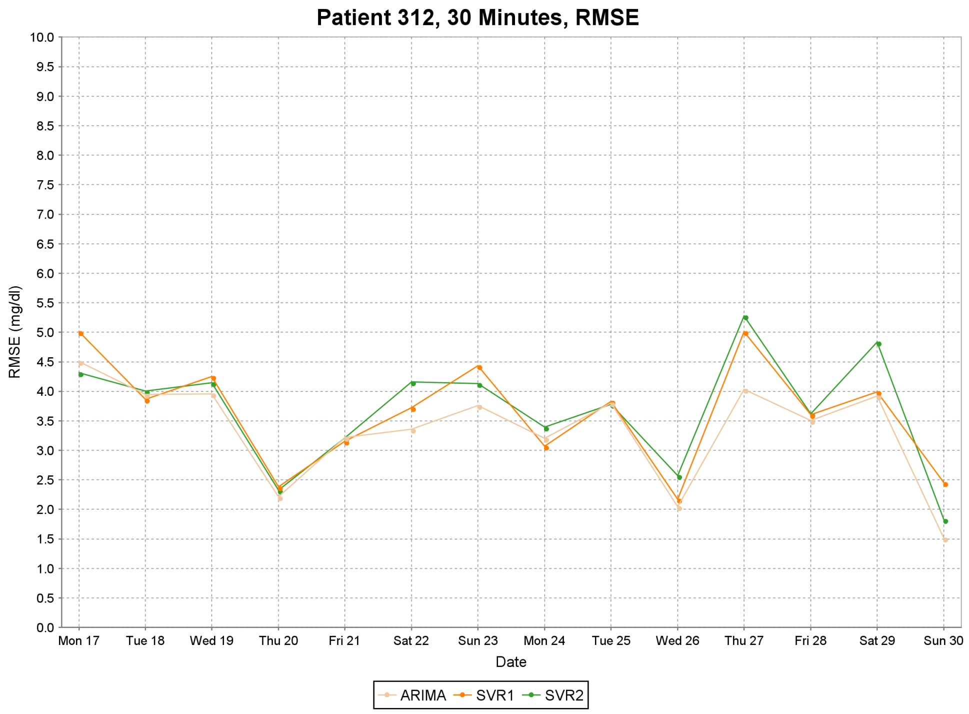


Figure A.17: RMSE plot of test data for patient 312 with 30 minute predictions.

Table A.27: Patient 312 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	29.2	13.7	12.6	12.9
MAE	23.9	10.5	9.6	9.9
R^2	0.44	0.87	0.89	0.88
CEGA (A%)	69%	93%	94%	94%

A.9.2 60 Minutes

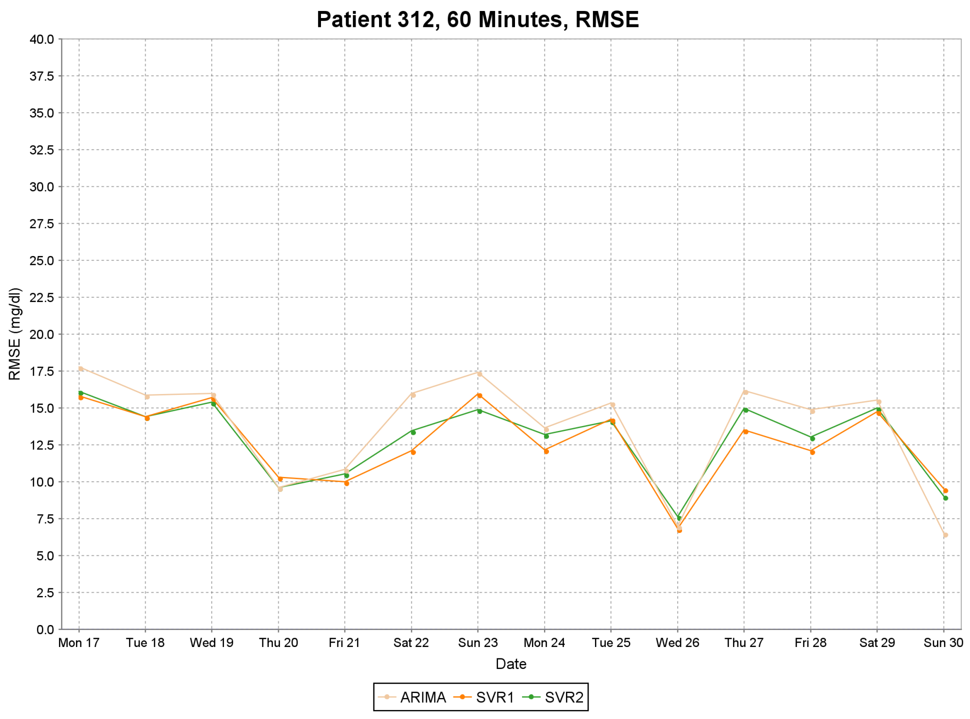


Figure A.18: RMSE plot of test data for patient 312 with 60 minute predictions.

A.10 Patient 313

Table A.28 shows the feature vectors found from tuning the feature templates for patient 313 at 30 and 60 minute prediction horizons.

Table A.28: Feature vectors for patient 313.

Feature	30 Minutes	60 Minutes
$\varphi_{MVA}(x, t, 3, \lambda)$	$\lambda = 0.5$	$\lambda = 1$
$\varphi_{MVA}(x, t, 6, \lambda)$	$\lambda = 0.5$	$\lambda = 0.5$
$\varphi_{MVA}(x, t, 9, \lambda)$	$\lambda = 0.5$	$\lambda = 0.5$
$\varphi_{MVA}(x, t, 12, \lambda)$	$\lambda = 0.5$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 3, \lambda)$	$\lambda = 0.9$	$\lambda = 0.5$
$\varphi_{RoC}(x, t, 6, \lambda)$	$\lambda = 0.5$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 9, \lambda)$	$\lambda = 1$	$\lambda = 0.25$
$\varphi_{RoC}(x, t, 12, \lambda)$	$\lambda = 1$	$\lambda = 0.25$
$\varphi_{Bolus}(x, t - i\,k, k)$	$k = 10, i = \{0, \ldots, 5\}$	$k = 60, i = 0$
$\varphi_{Basal}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{BasalArea}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Meal}(x, t - i\,k, k)$	$k = 10, i = \{0, \ldots, 5\}$	$k = 10, i = \{0, \ldots, 5\}$
$\varphi_{Exercise}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$
$\varphi_{Sleep}(x, t - i\,k, k)$	$k = 60, i = 0$	$k = 60, i = 0$

A.10.1 30 Minutes

Table A.29: Patient 313 averages for each error metric with a 30 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	18.2	3.1	3.4	3.3
MAE	14.2	2.3	2.6	2.5
R^2	0.82	0.99	0.99	0.99
CEGA (A%)	88%	99%	99%	99%

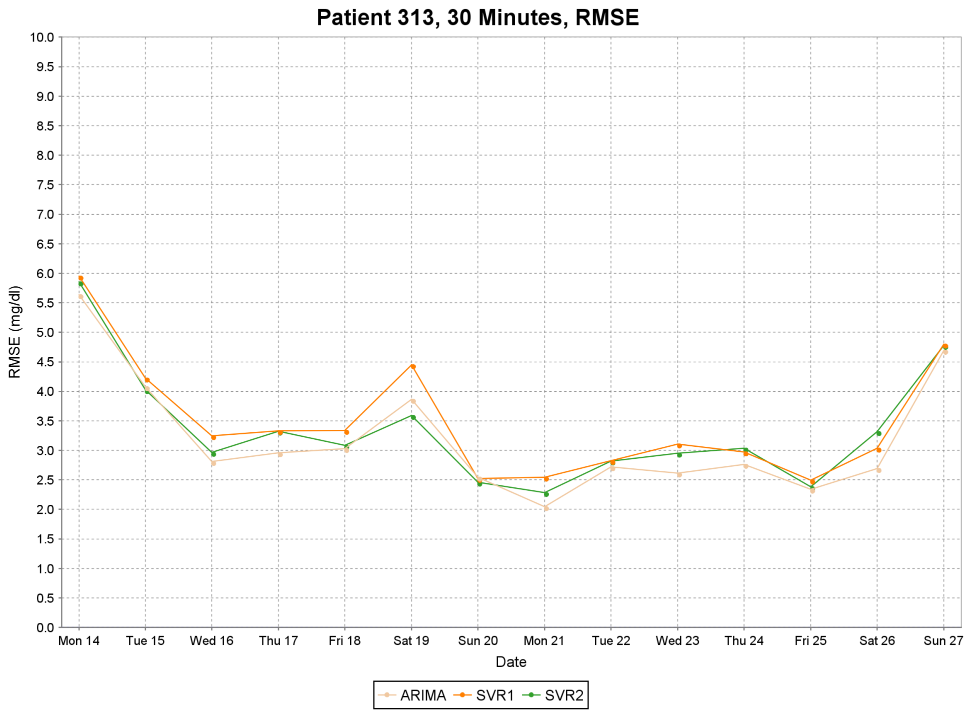


Figure A.19: RMSE plot of test data for patient 313 with 30 minute predictions.

Table A.30: Patient 313 averages for each error metric with a 60 minute prediction horizon.

Error Metric	t_0	ARIMA	SVR_1	SVR_2
RMSE	33.8	14.3	14.0	13.7
MAE	26.5	10.7	10.6	10.4
R^2	0.38	0.87	0.88	0.88
CEGA (A%)	64%	94%	94%	94%

A.10.2 60 Minutes

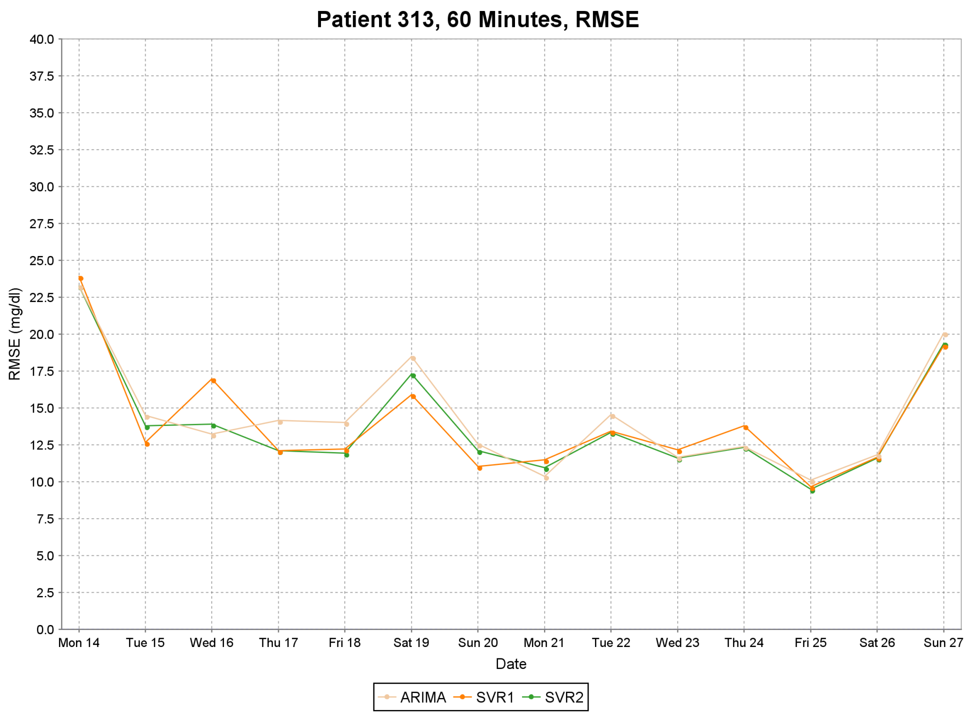


Figure A.20: RMSE plot of test data for patient 313 with 60 minute predictions.