CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Blood Glucose Prediction for Type 1 Diabetes using Machine Learning

Long Short-term Memory based models for blood glucose prediction

Master's thesis in Computer Science

Christian Meijner, Simon Persson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Master's thesis 2017

Blood Glucose Prediction for Type 1 Diabetes using Machine Learning

Long Short-term Memory based models for blood glucose prediction

Christian Meijner, Simon Persson

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2017

Blood Glucose Prediction for Type 1 Diabetes using Machine Learning
Long Short-term Memory based models for blood glucose prediction
Christian Meijner, Simon Persson

© Christian Meijner, Simon Persson, 2017.

Supervisor: Olof Mogren, Department of Computer Science and Engineering
Examiner: Alexander Schliep, Department of Computer Science and Engineering

Master's Thesis 2017
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

Blood Glucose Prediction for Type 1 Diabetes using Machine Learning
Long Short-term Memory based models for blood glucose prediction
Christian Meijner, Simon Persson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In this thesis, walk forward testing is used to evaluate the performance of two long short-term memory (LSTM) models for predicting blood glucose values for patients with type 1 diabetes. The models are compared with a support vector regression (SVR) model as well as with an auto regressive integrated moving average (ARIMA) model, both of which have been used in related research within the area.

The best performing long short-term model produces results equal to those of the SVR model and it outperforms the ARIMA model for all prediction horizons. In contrast to models in related research, this LSTM model also has the ability to assign a level of confidence to each prediction, adding an edge in practical usability.

Keywords: computer science, long short-term memory, LSTM, recurrent neural network, RNN, type 1 diabetes, blood glucose prediction.

Acknowledgements

First and foremost we would like to thank our supervisor Olof Mogren for his interest and dedication throughout the project. Furthermore, we want to express our gratitude towards Björn Eliasson for sharing his domain expertise. Finally, we would like to acknowledge Sofia and Josefin whose thorough peer-review helped us write a better thesis.

Christian Meijner, Simon Persson, Gothenburg, May 2017

Contents

List of Figures

List of Tables

1
Introduction

The number of patients with diabetes mellitus has increased significantly during the last 30 years. In 1980 it was estimated that about 108 million adults had the disease. Today that number is estimated to be around 422 million (WHO, 2016). Type 1 diabetes (T1D) is the more severe variant of the disease where the pancreas has stopped producing insulin, an important protein used to regulate blood glucose levels. Patients with T1D therefore have to manually inject insulin using either syringes or insulin pumps.

Daily treatment of T1D can be cumbersome and involves a lot of decision making regarding how much and how often to inject insulin throughout the day. As a result, it can be hard for diabetics to keep their blood glucose levels stable and within a safe range. To remedy this problem, attempts have been made to use machine learning as a way of optimising insulin dosage and, in extension, blood glucose levels. Many machine learning techniques and algorithms have the capability of predicting future outcomes of a certain data set. Knowing what the blood glucose levels might be in the very near future can be of great importance for a T1D patient when deciding on the amount of insulin to inject in a given situation.

In related research, machine learning methods such as feed-forward neural networks (Zecchin et al., 2012),(Mougiakakou et al., 2006), recurrent neural networks (RNN) (Pappada et al., 2011) and support vector regression (SVR) (Wiley, 2011),(Bunescu et al., 2013) have been used for predicting blood glucose values. However, there seems to be no research made using long-short term memory (LSTM) based models for this task. Blood glucose data can be interpreted as a time series (3.1) and multiple applications of LSTM suggest that it is useful on other kinds of temporal data, (Ma et al., 2015),(Eck and Schmidhuber, 2002), (Malhotra et al., 2015), which is why the overall aim of this work is to investigate the use of LSTM in the blood glucose prediction context.

1.1 Goal

The main goal in this thesis is to expand on current research on predicting blood glucose values for T1D patients using machine learning methods. In particular, the intention is to evaluate models based on LSTM networks and compare the performance of such models with SVR and ARIMA models, both of which have been explored in earlier work (Wiley, 2011),(Bunescu et al., 2013).

1.2 Delimitations

There is no underlying, exact model for how the body behaves and responds to different life events, but there are several variables known to affect the blood glucose levels in a patient with type 1 diabetes (Cryer et al., 2003). Due to limitations in the data available, the models used in this work only utilise blood glucose values, insulin dosages and carbohydrate intake only. These features are important and good predictions have been made using only past blood glucose values as input (Wiley, 2011).

Another limitation is the number of patients of which data the models will be evaluated on. Ideally the models created needs to be verified on a large set of patients. However, in this work, data from only four patients is used.

2
Background

What follows are two sections that give the knowledge required to understand the problem this work tries to solve. The first section gives a brief introduction to type 1 diabetes, what the treatment looks like and risks related to improper treatment. The second section connects diabetes treatment to machine learning and summarise previous research on the topic of blood glucose prediction using machine learning techniques.

2.1 Type 1 Diabetes

When consuming food with carbohydrates, the blood glucose level in the human body increases (Jenkins et al., 1981). To regulate the blood glucose, a hormone called insulin is used (Atkinson et al., 2014). Normally, the hormone is produced by beta cells in the pancreas, but for individuals with type 1 diabetes these beta cells are destroyed by the body's own immune system, eventually resulting in a complete lack of insulin. This deficiency can, if untreated, cause serious implications such as hyperglycemia (abnormally high blood glucose levels) and, by extension, a potentially life threatening state called diabetic ketoacidosis (Daneman, 2006). To compensate for the lack of insulin in the body it has to be manually injected, often using either syringes or an insulin pump.

When using syringes, one typically injects a direct-acting insulin before meals or for smaller adjustments of the blood glucose levels. A more longer-lasting insulin is injected once or a few times per day to sustain a base level of insulin in the body. In contrast with a syringe, an insulin pump is directly connected to the body at all times and is thus able to better simulate the behaviour of a real pancreas. It uses direct-acting insulin not only for meal and adjustment injections (bolus), but also for sustaining the insulin base level in the body. This is done by making evenly spread out injections (basal) through out the day according to a 24 hour program defined by the patient.

Whether using syringes or an insulin pump to treat T1D, the patient still has to decide on how much insulin should be injected throughout the day. When making such decisions, one has to take a number of metrics into account, including but not exclusively, the current and past blood glucose values in the body, insulin dosage, carbohydrate intake (Jenkins et al., 1981), as well as exercise (Association et al., 2004).

Too little insulin in the body might cause hyperglycemia and, in turn, diabetic ketoacidosis which can be a direct life threatening complication (Foster and McGarry, 1983). One can also get more long-term complications such as retinopathy, nephropathy, neuropathy, and cardiovascular disease (Haller et al., 2005). The opposite condition of hyperglycemia is called hypoglycemia and occurs if a patient injects too much insulin, causing a glucose deficiency in the blood stream . If untreated, it can lead to serious and rather direct complications such as cognitive dysfunction, seizures, coma and even death (Cryer et al., 2003). Hypoglycemia and hyperglycemia are both prevented in the same manner, by constantly monitoring blood glucose levels and adjusting the insulin dosage accordingly. Hypoglycemia can often be treated by consuming sugar, while hyperglycemia is treated by injecting more insulin (and possibly also treating the ketoacidosis that might have occurred).

The treatment of T1D can be summarised as a process of constantly balancing food intake with insulin dosage, taking into account other factors such as exercise, sleep cycles and other aspects in life that might affect the blood glucose levels and reception of insulin in the body. In this constant balancing, it is a clear advantage to be able to accurately predict future outcomes based on current actions, such as different levels of insulin dosages.

2.2 Blood glucose prediction using machine learning

In the 1970s and 1980s, the treatment of type 1 diabetes saw great improvements. New strategies were developed with both multiple daily injections and self-monitoring of blood glucose (SMBG) (Boland et al., 2001). Although SMBG is helpful for keeping a good long-term metabolic control (Haller et al., 2004), it usually generates very few blood glucose value data points; the recommended frequency of SMBG for a patient with type 1 diabetes is at least four times a day in conjunction with meals, when fasting and before bedtime (Benjamin, 2002). Making accurate blood glucose predictions with so few data points per day can be challenging. However, with the introduction of continuous glucose monitoring systems (CGMs), the prospects of using machine learning methods as a tool for predicting future blood glucose values are improved. Such systems measure blood glucose continuously and typically record the results every 5-15 minutes instead of around only four times a day.

There is plenty of research on the use of machine learning algorithms for predicting glucose values. For example, an artificial neural network has recently been used in combination with a first-order polynomial extrapolation algorithm to successfully make good 30 minute predictions (Zecchin et al., 2012). A feed-forward neural network has also been evaluated for 75 minute prediction in (Pappada et al., 2011), and the network performed well enough to provide therapeutic guidance. Furthermore, recurrent neural networks and feed forward neural networks together with compartmental models for blood glucose prediction for children with type 1 diabetes has

been explored in (Mougiakakou et al., 2006), and the recurrent network was shown to be superior.

Yet another recent example of how data collected by CGM systems can be used to make good blood glucose predictions with machine learning methods is the thesis "Machine Learning for Diabetes Decision Support" (Wiley, 2011). In it, two different data models are evaluated using SVR. One model consists of attributes solely based on previous blood glucose values, while the other contains information such as insulin dosage, carbohydrate intake, exercise, work and sleep patterns. The prediction performance of the SVR is reported for both 30 and 60 minute blood glucose predictions, and although the more advanced model with more data attributes performed better for some patients participating in the experiment, the simpler model performed better overall. A surprising result, suggesting that the SVR might not be able to fully take advantage of the extra life data attributes in the more complex model.

A related study, "Blood Glucose Level Prediction using Physiological Models and Support Vector Regression", further explores the use of extra life data such as insulin dosage and carbohydrate intake when predicting blood glucose values with SVR (Bunescu et al., 2013). In this study, a physiological model aimed at better capturing the dynamic between blood glucose and daily events is introduced. Using SVR, a new data model with features from the physiological model is compared to a data model similar to the one in Wiley's work. The data model using physiological features turns out to perform slightly better, with an improvement in prediction accuracy over the older model.

There seems to be no previous work on the use of LSTM networks for blood glucose prediction. However, there are multiple examples that demonstrates the usefulness of LSTM networks for learning other temporal data. For example, LSTM was found to outperform multiple algorithms such as SVM, ARIMA and Kalman filtering when predicting traffic speed using data from remote traffic microwave sensors (Ma et al., 2015). LSTM has also been successfully used to compose blues music (Eck and Schmidhuber, 2002), as well as for detecting anomalies in time series data by comparing the error of predicted values with observed values (Malhotra et al., 2015).

The previous approaches mentioned above all do work for predicting future blood glucose values. And while the data sets they are evaluated on vary, many report accuracy numbers comparable to those by (Bunescu et al., 2013). Furthermore, Bunescu used raw CGM data similar to ours with minimal pre-processing. Therefore, the SVR first introduced by (Wiley, 2011) and then further evaluated in (Bunescu et al., 2013) has been the method to which the models presented in this work are compared to.

3
Theory

The intention of this section is to work as a "mini encyclopedia", to which the reader can return whenever a concept or term might need further clarification. It discusses time series, machine learning concepts and the theory for ARIMA, SVR and LSTM.

3.1 Time series

A time series is a chronologically ordered sequence where each element constitutes an observation at that specific moment in time. An important aspect is that each observation might be dependent on the previous observation. Also, observations further back in time might have less impact on future observations. For example, in the context of this work, the blood glucose level from 15 minutes ago tells us more about the immediate future than levels from 5 days ago do.

3.2 Machine Learning - general concepts

Machine learning is a sub area within the field of computer science that studies how to construct and use algorithms that can learn from and analyse data (Ayodele, 2010). Common applications for such algorithms are data categorisation as well as prediction of future data points. For example, machine learning algorithms can be used to tell whether a digital image contains a certain certain subject or not (Shan, 2012), or what the rainfall will be like the next day (Hong, 2008).

There are two main categories of learning: supervised and unsupervised. In supervised learning, the data from which an algorithm should learn is labeled, meaning that each input to an algorithm has a known value that the algorithm ideally should output, given the corresponding input (Kotsiantis et al., 2007). A data set could, for example, consist of multiple pictures, each labeled with a 1 if the image contains a certain subject, and with a 0 otherwise. The goal then is to have an algorithm learn and recognise subjects in the images by outputting the value 1 if an image contains the subject, and 0 if it does not. This is an example of binary classification. However, there are other learning tasks as well. For example, some data might have more than two classes, or be labeled with real values instead of discrete ones.

In contrast to supervised learning, the data in unsupervised learning is unlabeled. Thus, there is no desired or correct value for each input known beforehand. A common use case for unsupervised learning is clustering, where an algorithm is used

to learn what different clusters (or labels) the input data can be split into (Kotsiantis et al., 2007). For example, the input data might be a list of different wine flavors, each described with a set of characteristics such as sweetness, acidity, tannin etcetera. The goal could then be to learn if there is a natural grouping of the different wines, how many such groups there are and what the common characteristics are for each wine group found.

In this work, only supervised learning algorithms have been used, and thus any other learning techniques have been left out of the remaining discussion. For an algorithm to learn what target value an input should be associated with, it must be trained. First, the data set is usually split into three subsets: a training set, a test set and a validation set. During training, the algorithm is applied to all data points in the training set, and its output used to determine how big the error in its classifications or predictions is. The error is usually defined as a loss function E, and the overall goal of the training is to tweak the parameters of the algorithm so that E is minimised.

The learning process can be repeated for multiple iterations, or epochs, on the same training data set to improve performance. However, too many iterations can cause overfitting, a phenomena in which the algorithm becomes overly adjusted to the training data and, as a result, gives poor performance when run on new data points (Caruana et al., 2000). To avoid this, the algorithm's performance is usually evaluated over the validation data set after each iteration. If the performance over the training set is increasing (or staying the same), but the performance over the validation set starts decreasing, the training is stopped.

3.3 Auto-Regressive Integrated Moving Average

When working with time series data an auto-regressive integrated moving average (ARIMA)-model can be used to gain further insights of the data or to predict future values in the time series (Box et al., 2015). More specifically, ARIMA-models are useful on data that is non-stationary; i.e. data whose mean and variance is not constant over time.

Given a time series $\mathbf{x} = \{x_1, x_2, \ldots, x_t\}$ where x_i denotes the data value at point i in time, an ARIMA-model of order (p,q,d) is defined as

$$x_t = \theta_1 x_{t-1} + ... + \theta_{d+p} x_{t-d-p} - \beta_1 \epsilon_{t-1} - ... - \beta_q \epsilon_{t-q} + \epsilon_t \,. \tag{3.1}$$

Parameter p is the number of auto-regressive terms, q the order of the moving average and d is the number of differences needed to make the data stationary. Furthermore, (β_1,...,β_q) are the parameters of the moving average, (θ_1,...,θ_{d+p}) the parameters of the auto-regressive part and ϵ is an error term sampled from a Normal distribution with zero mean.

The goal of an ARIMA is to find parameters that make Eq. 3.1 fit the data as well as possible. Once fitted, the ARIMA model can be used to predict values at time

$t + i$. It has been shown (Box et al., 2015) that this is equivalent to finding the expected value

$$\hat{x}_{t+i} = \mathbf{E}[x_{t+i}|\mathbf{x}]\,. \tag{3.2}$$

The following rules are used when calculating the expected values,

$$\begin{aligned} \mathbf{E_t}[x_{t-j}] &= x_{t-j}\,, \\ \mathbf{E_t}[x_{t+j}] &= \hat{x}_{t+j}\,, \\ \mathbf{E_t}[\epsilon_{t-j}] &= \epsilon_{t-j}\,, \\ \mathbf{E_t}[\epsilon_{t+j}] &= 0\,, \end{aligned} \tag{3.3}$$

where j is a non-negative integer. With these rules, Eq 3.2 can be rewritten as

$$\begin{aligned} \hat{x}_{t+i} = \mathbf{E_t}[x_{t+i}] = \theta_1\mathbf{E_t}[x_{t+i-1}] + ... + \theta_{d+p}\mathbf{E_t}[x_{t+i-d-p}] - \\ \beta_1\mathbf{E_t}[\epsilon_{t+i-1}] - ... - \beta_q\mathbf{E_t}[\epsilon_{t+i-q}] + \mathbf{E_t}[\epsilon_{t+i}]\,. \end{aligned} \tag{3.4}$$

Using this, an ARIMA-model can predict values for any future time step. Once a prediction for time step $t + 1$ is made this value is added to the time series and a prediction for time step $t + 2$ can be made and so forth. In other words, prediction further into the future is based on previous predictions.

3.4 Support Vector Regression

Support Vector Regression (SVR) is a machine learning algorithm that has been used extensively for time series prediction in various applications such as financial market forecasting, weather and environmental parameter estimation and electrical utility load prediction and more (Sapankevych and Sankar, 2009).

SVR is an extension on Support Vector Machines (SVM), a binary classifier that tries to classify data by finding a hyperplane separating two different classes of input data. Such a hyperplane can be defined as

$$\mathbf{w} \cdot \mathbf{x} + b = 0\,. \tag{3.5}$$

In this equation $\mathbf{x}$ is an input vector, $\mathbf{w}$ is a normal to the plane and $\frac{b}{||\mathbf{w}||}$ is the perpendicular distance to the origin. A key feature of an SVM is that it tries to fit the plane so the margin between the classes is maximised, as seen in Fig. 3.1. The vectors with circles around them in Fig. 3.1, i.e. the vectors on the lines H_1 and H_2, are called support vectors. These are of extra importance since they determine the margin.

If some linearly separable input $\mathbf{x}_i$ belonging to either class $y_i = 1$ or class $y_i = -1$ is used as training data, an SVM finds values for $\mathbf{w}$ and b so the data can be described according to the inequalities in Eq. 3.6 and Eq. 3.7.

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq 1, y_i = 1 \tag{3.6}$$

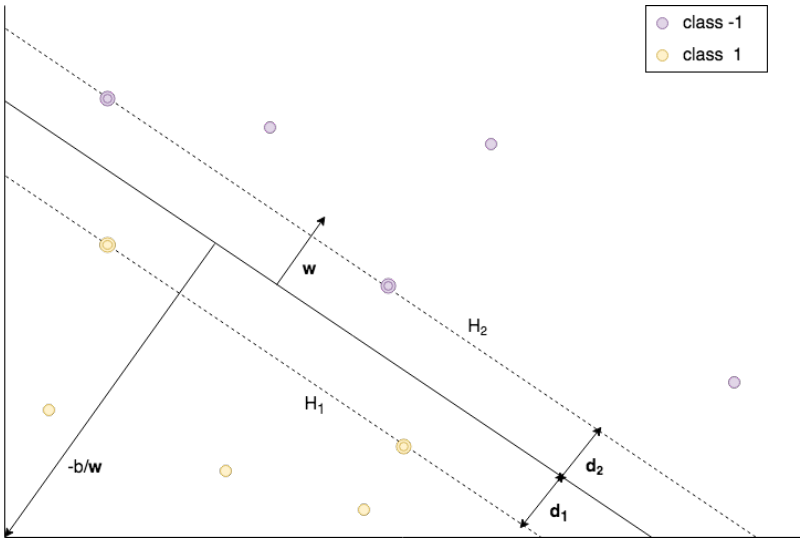

Figure 3.1: A hyperplane separating two classes.

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1, y_i = -1 \tag{3.7}$$

These inequalities can be combined into one as

$$\forall i \; y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0\,. \tag{3.8}$$

The distance from the hyperplane to the lines H_1 and H_2 in Fig. 3.1 needs to be the same which implies $d_1 = d_2$. This distance is the margin and the goal is to maximize it. The margin is equal to $\frac{1}{||\mathbf{w}||}$ and maximising it is equivalent to minimising $||\mathbf{w}||$ subject to condition (3.8).

$$\min ||\mathbf{w}|| \text{ subject to } \forall i \; y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \tag{3.9}$$

This minimisation is the core of an SVM and solving it yields the values for $\mathbf{w}$ and b required to classify new data. Any new data point $\hat{\mathbf{x}}$ is then classified by $\hat{y} = \text{sgn}(\mathbf{w} \cdot \hat{\mathbf{x}} + b)$.

An SVM works well for data that is linearly separable and by introducing slack variables ξ_i for the constraints and assigning penalties to points on the wrong side of the hyperplane the SVM can converge even if the data is not fully separable. This is called a soft margin SVM. With the slack variables, Eq. (3.8) is rewritten as

$$\forall i \; y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i \geq 0 \text{ where } \xi_i \geq 0\,. \tag{3.10}$$

The minimisation then becomes

$$\min ||\mathbf{w}|| + C \sum_i \xi_i \text{ s.t. } \forall i \; y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i \geq 0 \text{ where } \xi_i \geq 0\,. \tag{3.11}$$

Here C is a cost parameter that controls the relation between the penalties of the slack variables ξ_i and the size of the margin.

While an SVM works well for binary classification ($\hat{y} = \pm 1$) of not yet seen data points it lacks the property to classify the input with a real value $\hat{y}$.
SVM works well for binary classification ($\hat{y} = \pm 1$) of not yet seen data points. This is, however, not enough to assign a real value to the input. For this purpose the SVR is more appropriate. Now, the data is of the form $\{\mathbf{x}_i, t_i\}$ ($\mathbf{x}_i \in \mathbb{R}^D$, $t_i \in \mathbb{R}$), where $\mathbf{x}_i$ is the input and t_i its corresponding target value.
The input is no longer classified by $y_i = \pm 1$. Instead it is assigned a real value by:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b\,. \tag{3.12}$$

Expanding on the idea of the soft margin SVM, an SVR uses a slightly more advanced method to calculate the penalties. No penalty is given if the predicted value $f(\mathbf{x})$ is less than some distance ϵ from the actual value t_i, i.e. if $|t_i - f(\mathbf{x}_i)| < \epsilon$. In Fig. 3.2 the grey region bounded by $f(\mathbf{x}) \pm \epsilon$ is called an ϵ-insensitive tube. Target values t_i (the blue dots in 3.2) that lies outside of this tube are given different slack variable penalties depending on the distance to the tube and whether they lie above or below it.

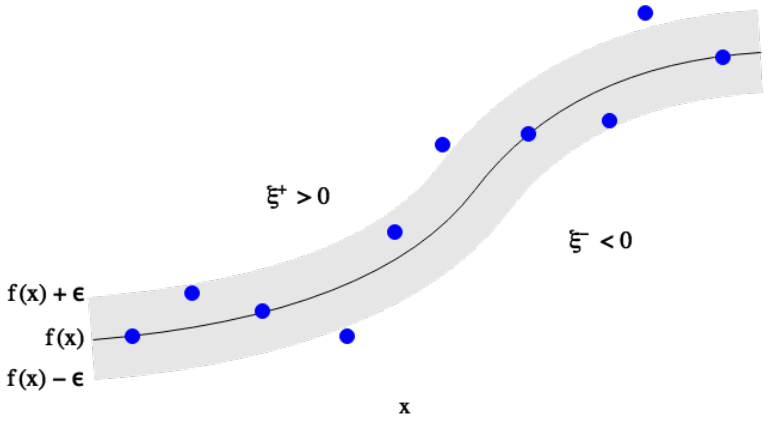

Figure 3.2: Support Vector Regression with an ϵ-insensitive tube.

This yields the following inequalities

$$\begin{aligned} t_i &\leq f(\mathbf{x}_i) + \epsilon + \xi_i^+\,, \\ t_i &\geq f(\mathbf{x}_i) - \epsilon - \xi_i^-\,, \\ \xi_i^+ &\geq 0,\, \xi_i^- \geq 0\ \forall i\,. \end{aligned} \tag{3.13}$$

Thus the minimisation becomes

$$\min\, ||\mathbf{w}|| + C \sum_i (\xi_i^+ + \xi_i^-)\,, \tag{3.14}$$

subject to the constraints given by the inequalities in Eq. 3.13. Solving this minimisation yields the following expression to predict real values $\hat{y}$ for unseen data $\hat{\mathbf{x}}$.

$$\hat{y} = \sum_i (\alpha_i^+ - \alpha_i^-)\mathbf{x}_i \cdot \hat{\mathbf{x}} + b \tag{3.15}$$

In classification and regression problems the data is not always linearly separable in the space of the inputs. However, it might be if the input data is mapped to some higher dimension space. This mapping is done by using kernels (Scholkopf and Smola, 2001) that projects the data into a higher dimension as seen in Fig. (3.3). Common choices of kernels for SVR are the linear kernel, the polynomial-kernel, the sigmoid-kernel and the radial basis function-kernel (Hsu et al., 2003). The SVR-implementation used in this work utilizes the radial basis function-kernel which is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)}, \tag{3.16}$$

where $||\mathbf{x}_i - \mathbf{x}_j||^2$ is the euclidean distance between the input vectors and σ is a free parameter.

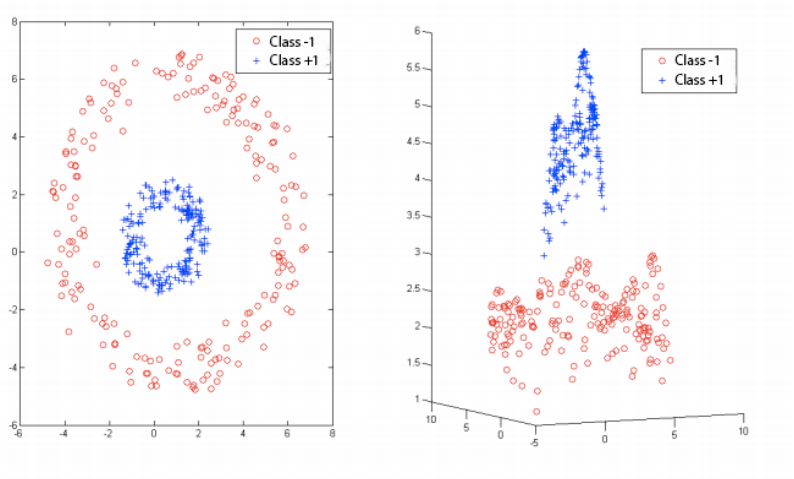

Figure 3.3: The left plot shows an example of data that is not linearly separable. In the right plot, a radial basis function-kernel has been used to project that data into a higher dimension, making it linearly separable.

3.5 Artificial Neural Networks

This section describes some of the important concepts related to artificial neural networks that are needed to fully understand the models evaluated in this work. It details what feed-forward neural networks are, how networks can learn and be trained as well as what a long-short term memory network (LSTM) is.

3.5.1 Single Neuron Network

The human brain is a neural network that consists of billions of interconnected neurons, each of which is capable of receiving, transmitting and processing information. In an ANN, these neurons are represented as a mathematical model that calculates a weighted sum of the signals from other neurons in the network (Abraham, 2005).

More formally, let $\mathbf{x} = [x_1, x_2, \ldots, x_p]$ be a vector of p input signals. A neuron then outputs the scalar value $\hat{y}$ as

$$\hat{y} = f\left(\sum_{i=1}^{p} w_i x_i\right) = f\left(\mathbf{w}^T \mathbf{x}\right) , \tag{3.17}$$

where $\mathbf{w} = \{w_1, w_2, \ldots, w_p\}$ is a vector of weights. Although omitted here to simplify calculations, a bias term b is usually also added to the sum in Eq. 3.17 above. The function f, sometimes referred to as an activation function, can be any non-linear function, and the choice of f depends on the application. If, for example, the neuron is to be used for performing logistic regression, f can be defined as the standard logistic function. This function squashes its input to a real value between $[0, 1]$:

$$f(x) = \frac{1}{1 + e^{-x}} . \tag{3.18}$$

By combining Eq. 3.17 with Eq. 3.18 above, the output of a neuron for a single data point $\mathbf{x}$ becomes

$$\hat{y} = f(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} . \tag{3.19}$$

To demonstrate the dynamics of an ANN, consider a simple network consisting of a single neuron (see Fig. 3.4) with an output $\hat{y}$ as in Eq. 3.19. Furthermore, define

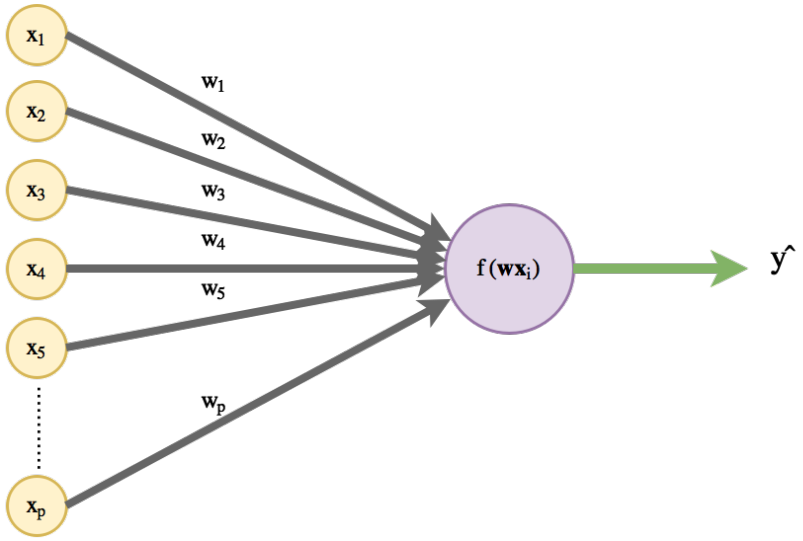

Figure 3.4: A single neuron demonstrating the core concepts of ANNs. The neuron calculates a weighted sum of the p-dimensional input and produces a scalar output $\hat{y}$.

$X = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$ as a set of N data points $\mathbf{x}_i$, each with a corresponding label $\mathbf{y}_i$. Then the goal of the ANN is to learn the weight vector $\mathbf{w}$ such that

$$\hat{y} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} = y , \tag{3.20}$$

for all pairs $(\mathbf{x}, \mathbf{y}) \in X$. This can be formulated as a minimisation problem, for example using the mean squared error (MSE) as objective function over all N data points:

$$E = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{2N}\sum_{i=1}^{N}\left(y_i - f(\mathbf{w}^T\mathbf{x}_i)\right)^2 . \tag{3.21}$$

The objective function can also be referred to as the loss function, or simply the loss, and the goal is to find a weight vector $\mathbf{w}$ that minimises it. A common method for doing this is gradient decent (GD). In GD, the weight vector is adjusted iteratively using a learning rule based on the gradient of the loss function with respect to $\mathbf{w}$

$$\mathbf{w} = \mathbf{w} - \eta\frac{\partial E}{\partial \mathbf{w}} , \tag{3.22}$$

where η denotes the learning rate, i.e how much the weight vector should be adjusted in each iteration.

An alternative method to GD is stochastic gradient descent (SGD). It works in the same way as GD, except that the loss is defined as the the error over a single, randomly chosen data point instead of as the average error over all points:

$$E = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}\left(y - f(\mathbf{w}^T\mathbf{x})\right)^2 . \tag{3.23}$$

Yet another form of gradient descent, and the one used for all ANN related optimisations in this work, is mini-batch GD. In mini-batch GD the loss is defined as the average error over a small set of randomly selected data points, i.e a mini-batch. Denote the size of the mini-batch as k. The loss then becomes

$$E = \frac{1}{2k}\sum_{i=1}^{k}(y_i - \hat{y}_i)^2 = \frac{1}{2k}\sum_{i=1}^{k}\left(y_i - f(\mathbf{w}^T\mathbf{x}_i)\right)^2 . \tag{3.24}$$

Note that $k = 1$ would be the same as running SGD, and $k = N$ would be equivalent to traditional GD.

The gradient of the loss function for mini-batch GD in Eq. 3.24 with respect to an arbitrary element w_j in $\mathbf{w}$ can be calculated using the chain rule:

$$\begin{aligned}\frac{\partial E}{\partial w_j} &= \frac{1}{k}\sum_{i=1}^{k}\frac{\partial}{\partial w_j}\frac{1}{2}\left(y_i - \hat{y}_i\right)^2 \\ &= \frac{1}{k}\sum_{i=1}^{k}\frac{\partial}{\partial \hat{y}_i}\left(\frac{1}{2}\left(y_i - \hat{y}_i\right)^2\right)\cdot\frac{\partial \hat{y}_i}{\partial \mathbf{w}^T\mathbf{x}_i}\cdot\frac{\partial \mathbf{w}^T\mathbf{x}_i}{\partial w_j} \\ &= \frac{1}{k}\sum_{i=1}^{k}(\hat{y}_i - y_i)\hat{y}_i(1 - \hat{y}_i)x_{i,j} .\end{aligned} \tag{3.25}$$

Here, $x_{i,j}$ denotes the j'th element of the i'th input vector.

Using the gradient in Eq. 3.25, the learning rule in Eq. 3.22 for mini-batch GD can be written (in vector notation) as

$$\mathbf{w} = \mathbf{w} - \eta\frac{1}{k}\sum_{i=1}(\hat{y}_i - y_i)\hat{y}_i(1 - \hat{y})\mathbf{x_i} . \tag{3.26}$$

3.5.2 Feed-Forward Neural Networks

An ANN can vary in both size, shape and arrangement of its neurons. The simplest and one of the earliest forms of ANNs explored is feed-forward neural networks (FFNNs). They consist of one or more neurons connected to each other in a way that does not form any cycles. The single neuron ANN described in 3.5.1 is a trivial example of an FFNN; it only has one neuron and therefore cannot contain any cyclic connections. However, more advanced models can consist of multiple neurons, and the neurons can be arranged in one or more layers as in Fig. 3.5.

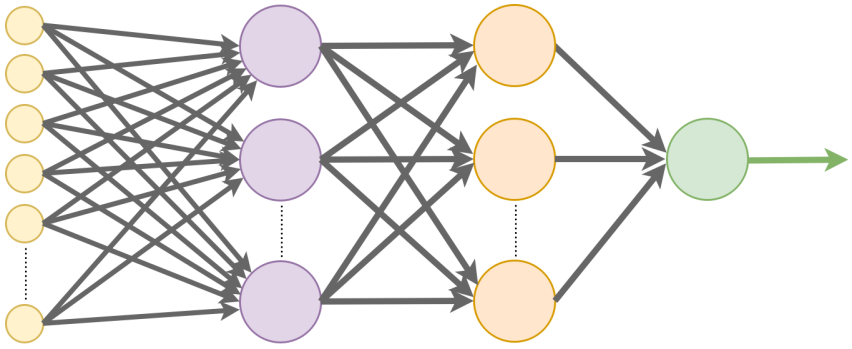

Figure 3.5: A feed-forward neural network with multiple layers. The information flows from left to right in the network, with no cyclic connections in-between the nodes.

The first layer can be referred to as the input layer, and the last as the output layer. All layers in between are called hidden layers. Note that the output layer can have more than one neuron. Every neuron has its own weight vector $\mathbf{w}$ and bias term b, but works according to the same principles as described in 3.5.1. In other words, a neuron in the i'th layer calculates a weighted sum of the output from all neurons in the $i-1$'th layer and so on. The calculation of the network output is made "from left to right", starting with the neurons in the first layer and ending with the ones in the output layer.

The appropriate weight vectors for fitting a multilayer network to its input data can be found by using the different forms of gradient descent described in 3.5.1. First, the gradient descent optimisation is performed on the weights in the output layer. The errors calculated are propagated backwards in the network and used in the gradient descent for the previous layer, and so on for all weights in the network. This method of propagating the errors backwards is called backpropagation.

3.5.3 Recurrent Neural Networks and LSTM

A recurrent neural network (RNN) can be seen as multiple, chained together modules, each containing a simple feed-forward neural networks as in Fig. 3.6. Each module has its own output $\mathbf{h}_i$, and its input $\mathbf{x}_i$ is concatenated with the output $\mathbf{h}_{i-1}$ of the previous module in the chain. This makes the network capable of handling a sequence of inputs of any length and to pass information between states in an input

sequence. In other words, the network can express relations between data points in a sequence of inputs, a characteristic well suited for working with temporal, dynamic data such as a blood glucose value time series. In theory, RNNs are well suited for

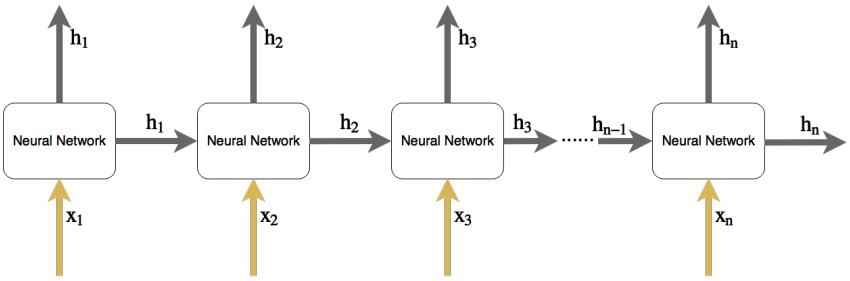

Figure 3.6: An unrolled recurrent neural network, each module with its own input and output pair. The output of the i'th module is the input of the $i+1$'th module in the chain, making RNNs useful for data with a sequential structure.

sequential and temporal data because of the connections in between the modules. However, as pointed out in (Bengio et al., 1994), in practicality traditional RNNs express difficulty to learn dependencies as the duration of those dependencies increases.

Hochreiter and Schmidhuber introduced a new kind of RNN called LSTM in 1997 (Hochreiter and Schmidhuber, 1997) that remedies some of the problems of the more classical RNN models. An LSTM network can remember long-term dependencies, and has proven to work efficiently in numerous applications (Weninger et al., 2015),(Graves et al., 2004), (Breuel et al., 2013).

Just as a classical RNN, an LSTM consist of chained together modules, or cells, where information can pass in-between. The difference lies in the internals of the cells themselves. Each cell sends its cell state and a hidden state to the next cell in the LSTM, and every cell contains four interacting neural network layers instead of a single, simpler feed-forward neural network as in a classical RNN. These four layers together form three "gates" that decide what information from the previous cell should be forgotten, updated and outputted to the next one (see Fig. 3.7).
The first gate in an LSTM cell is the "forget gate". It consists of a simple sigmoid layer that takes the hidden state $\mathbf{h}_{t-1}$ from the previous cell and the input $\mathbf{x}_t$ to create a concatenated vector used in a sigmoid layer. Note that all bold upper-case letters in all equations below denote matrices, as opposed to vectors that are bold and lower-case.

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f\right) \tag{3.27}$$

The output $\mathbf{f}_t$ of the forget gate will be between zero or one. A one can be thought of as "do not forget anything" while a zero means "forget all", since $\mathbf{f}_t$ is multiplied with $\mathbf{c}_{t-1}$, the cell state from the previous cell.

The second gate is the last step in creating the new state $\mathbf{c}_t$ from the old state $\mathbf{c}_{t-1}$. In this gate $\tilde{\mathbf{c}}_t$ is calculated as in Eq. 3.29 below.

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i\right) \tag{3.28}$$

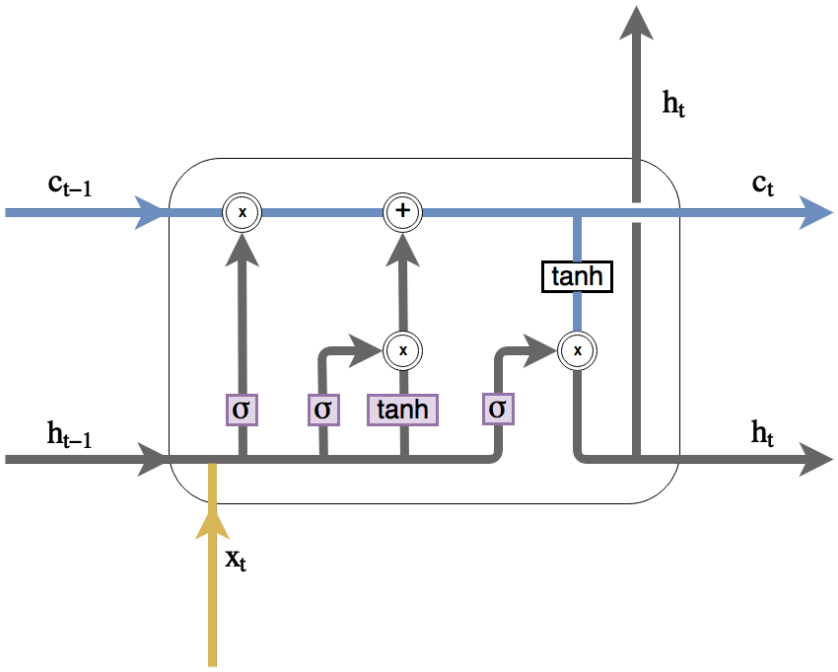

Figure 3.7: The internals of an LSTM cell, with the four interacting neural network layers marked in purple. These four layers together form three "gates" used to decide what information should be forgotten, added and outputted to the next cell.

$$\tilde{\mathbf{c}}_t = tanh\left(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c\right) \tag{3.29}$$

The new state $\mathbf{c}_i$ is finally defined as in Eq. 3.30 below.

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{c}}_t \tag{3.30}$$

The third and final gate is used to decide the output $\mathbf{h}_t$ of the LSTM cell. This output $\mathbf{h}_t$ is defined as in Eq. 3.32, where *tanh* squashes the values of the cell state $\mathbf{c}_t$ to values between -1 and 1, and the sigmoid layer decides what parts of the cell state that should be in the output.

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o\right) \tag{3.31}$$

$$\mathbf{h}_t = \mathbf{o}_t * tanh(\mathbf{c}_t) \tag{3.32}$$

4

Data

A future glucose value depends not only on past observations of glucose values, but also on insulin dosage, carbohydrate intake and other life data. In fact, the blood glucose value depends on more factors than one can realistically include in a model, less collecting the actual data for. Blood glucose values together with insulin dosage and carbohydrate intake, however, is enough for the models in this work to perform well.

4.1 Blood Glucose Values

Blood glucose values are typically measured in mmol/L or mg/dl, the former being the unit used in this report. A person with type 1 diabetes must manage the glucose levels manually with insulin injections, the goal being to keep blood glucose values within a safe range. Both too high and too low blood glucose values may cause serious short and long term complications.

There are two predominant ways to monitor blood glucose levels. Either by fingersticks or with an automated, electronic continuous glucose monitoring (CGM) system. Blood glucose monitoring by fingerstick means using a lancing device, typically on a finger, to get a drop of blood that can then be put on a test strip and be analysed by a traditional blood glucose meter. A CGM system, on the other hand, is constantly attached to the body and measures the blood glucose value continuously, in small and equally spaced time increments. Such systems typically have less accuracy than a traditional blood glucose meter with fingersticks, but collects larger volumes of data. A person using a traditional blood glucose meter might have 4-10 data points per day, while a CGM system might collect several hundreds.

All data used in this work comes from some kind of CGM system, although the actual hardware used differs from patient to patient. Most data sets contain data collected by a meter called FreeStyle Libre. The meter works together with an on-body patch sensor that samples the blood glucose value every 15 minutes. The Libre differs a bit from other CGM systems since it does not need any fingerstick calibration. This simplifies the processing of the gathered data, as one does not need to take any fingerstick data used for calibration into account.

4.2 Insulin Dosage

All patients included in the experiment used an insulin pump to administer their injections of insulin. An insulin pump is typically connected via a tube to the patient at all times and offers two different kinds of injections. Injections made manually by the user to compensate for food eaten or to make direct adjustments to the blood glucose levels are called bolus injections. A bolus injection is usually given instantly, all at once, but some insulin pumps also has functionality for prolonged bolus where the bolus insulin is injected in small bursts over a time interval specified by the user.

The other kind of insulin injection, basal, are smaller evenly distributed injections made automatically throughout the day according to a prespecified program in the insulin pump. The basal program in an insulin pump usually comes in the form of a 24 hour schedule where the user gets to specify exactly how much insulin should be injected each hour. Some insulin pumps also allow for the basal rate to be temporarily adjusted to a higher or lower rate for a certain amount of time.

Insulin injections are measured in units (U), and both basal and bolus injections are stored together with the corresponding timestamp in the device memory. All patients used either an Accu-Chek Spirit Combo or Animas Vibe insulin pump.

While they are both insulin injections, it can be beneficial to treat bolus and basal separately, as pointed out in (Wiley, 2011). For example, when tuning the feature sets for the SVR-models, the number of instances of the bolus and basal attributes might differ (5.6).

4.3 Carbohydrates

One of the most important factors that greatly affects the blood glucose value, apart from insulin dosage, is food consumption. Especially, the amount of carbohydrates consumed. Just as insulin lowers the blood glucose value, carbohydrates raises it. The patients included in the experiments recorded their carbohydrate intake by entering the estimated amount of carbohydrates eaten into a blood glucose meter whenever food was consumed. The carbohydrate entries consist of the amount of carbohydrates in grams, together with a time stamp indicating when the carbohydrates were consumed. And while these entries are estimations, they could serve as an indicator for how much insulin is needed and what future blood glucose values might be.

5
Methods

This section describes the process used when predicting blood glucose values. It explains the error metric used to determine performance, the input vectors utilised by the machine learning models and as well as the evaluation methods. Furthermore it details how the data was pre-processed and why. It also explains how the baseline models and the LSTM models are constructed.

5.1 Time series prediction

The data utilised by the machine learning models is a time series (3.1) consisting of data points evenly spaced by 15 minutes arranged in chronological order. Each $\mathbf{x}_i$ in the series contains a vector of data features at point i in time. With this interpretation in mind, predicting future blood glucose values can be seen as the process of using a time series of size n with values up to point t in time (Eq. 5.1) to predict a value $\mathbf{x}_{t+i}$, i steps into the future.

$$\{\mathbf{x}_{t-(n-1)}, \mathbf{x}_{t-(n-2)}, \mathbf{x}_{t-(n-3)}, ..., \mathbf{x}_t\} \tag{5.1}$$

5.2 Error metric

The performance of all models has been measured using the Root Mean Square Error (RMSE), because it its the main error metric used in several previous studies of blood glucose prediction (Zecchin et al., 2012), (Bunescu et al., 2013). The RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}, \tag{5.2}$$

where $\hat{y}$ is the predicted value and y is the target value.

5.3 Evaluation method

Walk forward testing is an evaluation method used to the determine how well a time series prediction model performs over a period of time. The idea is to take a segment of a given data set and partition it into three different parts. The first part of the segment is used as training data for the model, the second part for validation and tuning of the model parameters and the third is used as test data for measuring performance. The segment is then evaluated by calculating the performance and

error measurements for the test partition.

In order to evaluate performance over time for the entire data set, the segment is moved across the data set by some time step, changing what parts of the data is used for training, validation and testing. Depending on the size of the partitions. Fig. 5.1 illustrates how the segment moves across the data set. For each time step

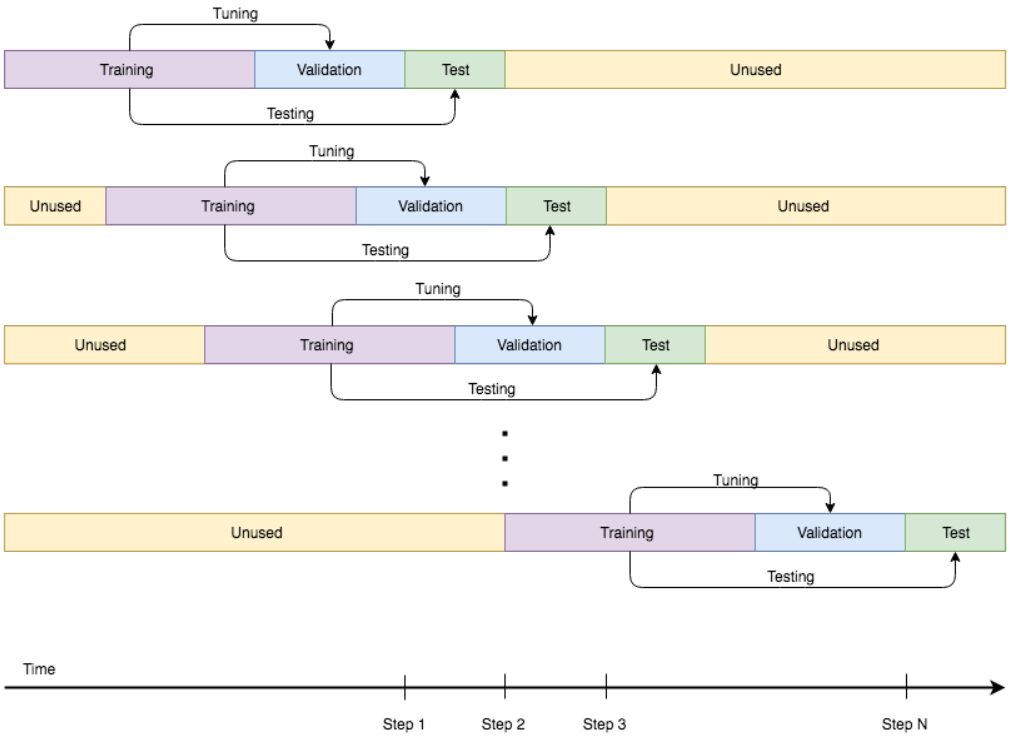

Figure 5.1: An illustration of how a data segment moves across a data set in walk forward testing.

the model is reset and reevaluated with the content of the new segment. Overall performance of the model is then measured by evaluating the performance for each segment.

This method has previously been used for evaluation of time series models for financial forecasting (Żbikowski, 2015) as well as evaluating performance for blood glucose prediction (Wiley, 2011)

5.4 Data pre-processing

External conditions such as temperature as well as limitations in hardware can sometimes cause gaps in CGM data readings. For example, the FreeStyle Libre system mentioned in 4.1 utilises a sensor that can store only the last eight hours of data. If more than eight hours pass between two scans of the sensor, a gap is introduced in the data, as seen in Fig. 5.2. Simple linear interpolation was used to deal with these gaps and to keep the data pre-processing minimal. In other words,

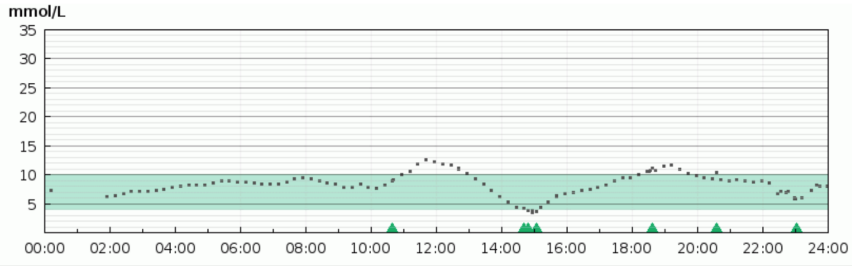

Figure 5.2: A data gap in a CGM plot that occurred during night time between 00:00 and 02:00.

a gap between any two points (x_0, y_0) and (x_1, y_1) in the data was replaced with the straight line

$$f(x) = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}\,. \tag{5.3}$$

All CGM system used in this study records the blood glucose value in intervals, meaning that there is a small "intentional" gap of about 5-15 minutes between every data point as well. To be able to sample a blood glucose value from a patient's data set at any given time t between the start and end time of the patient's entire data set, linear interpolation was used between all data points. Also, when evaluating the SVR, ARIMA and LSTM, the test days for each patient were chosen in a way that minimised the occurrences of such gaps.

5.5 Data features

This section describes the individual data features used in the feature sets detailed in 5.6.

5.5.1 Current blood glucose value

This is simply the current blood glucose value and it is represented as

$$f_{\text{glucose}}(t) = x_t\,, \tag{5.4}$$

where t is an index in the time series and x is the blood glucose value at t.

5.5.2 Blood glucose slope

The slope for glucose values at time t is simply defined as

$$f'_{\text{glucose}}(t) = x_t - x_{t-1}\,. \tag{5.5}$$

5.5.3 ARIMA prediction

Predictions from the ARIMA-model, also used as a baseline, are incorporated as a data feature in some input vectors. This feature is defined as

$$f_{\text{ARIMA}}(t) = y_{t+i}\,, \tag{5.6}$$

where y_{t+i} is the value predicted by the ARIMA-model at time $t + i$.

5.5.4 Bolus

The bolus feature is defined as

$$f_{\text{bolus}}(t, \Delta) = \sum_{x \in I_{\text{bolus},t,\Delta}} x \,, \tag{5.7}$$

where $I_{\text{bolus},t,\Delta}$ is the set of bolus insulin injections made between time t and $t - \Delta$.

5.5.5 Basal

The basal feature is defined as

$$f_{\text{basal}}(t, \Delta) = \sum_{x \in I_{\text{basal},t,\Delta}} x \,, \tag{5.8}$$

where $I_{\text{basal},t,\Delta}$ is the set of basal insulin injections made between time t and $t - \Delta$.

5.5.6 Carbohydrates

The carbohydrate feature is represented as

$$f_{\text{carbs}}(t, \Delta) = \sum_{x \in C_{t,\Delta}} x \,, \tag{5.9}$$

where $C_{t,\Delta}$ is the set of carbohydrate intakes done between time $t - \Delta$ and t.

5.5.7 Exponential Moving Average

A moving average is an average that moves across a time series. Such an average can, in combination with the most recent value, give an indication of a trend in the data. This feature is defined as

$$f_{\text{MVA}}(t, k, \lambda_{mva}) = \frac{\sum_{i=0}^{k} \lambda_{mva}^{i} x_{t-1}}{\sum_{i=0}^{k} \lambda_{mva}^{i}} \,, \tag{5.10}$$

where t is an index in the time series, x is the value at the given index and k is the number of previous values in the series to include in the average. λ_{mva} is a decay parameter valued between zero and one which is used to make older values contributes less to the average thus enabling the more recent values to have a greater impact.

5.5.8 Rate of Change

The rate of change is used to measure the speed at which values in a time series changes. This gives an indication of momentum in the data and in what direction it is heading. The rate of change feature is defined as

$$f_{\text{RoC}}(t, k, \lambda_{roc}) = \frac{\sum_{i=0}^{k-1} \lambda_{roc}^{i} (x_{t-i} - x_{t-i-1})}{\sum_{i=0}^{k-1} \lambda_{roc}^{i}} \,, \tag{5.11}$$

where λ_{roc} is a decay parameter valued between zero and one, t is an index in the time series, x is the value at the given index and k is the number of values in the time series to describe the rate of change metric with.

5.6 Feature sets

This section details the different features sets used. They are all constructed from the data features detailed in section 5.5. Each set utilizes the features as described in its respective table. However, features sets 5.3 and 5.4 can have multiple instances of their bolus, basal and carbohydrate features. For example, set 5.3 may have several different bolus features covering different ranges of past injections. e.g. features for $f_{\text{bolus}}(t, \Delta)$, $f_{\text{bolus}}(t - \Delta, \Delta)$, $f_{\text{bolus}}(t - 2\Delta, \Delta)$.

Feature	**Description**
$f_{\text{glucose}}(t)$	Blood glucose value at time t

Table 5.1: Feature set 1, used for evaluating the performance of the ARIMA, SVR and LSTM models.

Feature	**Description**
$f_{\text{glucose}}(t)$	Blood glucose value at time t
$f_{\text{ARIMA}}(t)$	ARIMA prediction for time $t + \Delta$
$f_{\text{MVA}}(t, k, \lambda)$	Moving average for last k time steps with decay λ
$f_{\text{RoC}}(t, k, \lambda)$	Rate of Change for last k time steps with decay λ

Table 5.2: Feature set 2, used for evaluating the SVR model.

Feature	**Description**
$f_{\text{glucose}}(t)$	Blood glucose value at time t
$f_{\text{ARIMA}}(t)$	ARIMA prediction for time $t + \Delta$
$f_{\text{MVA}}(t, k, \lambda)$	Moving average for last k time steps with decay λ
$f_{\text{RoC}}(t, k, \lambda)$	Rate of Change for last k time steps with decay λ
$f_{\text{bolus}}(t, \Delta)$	Sum of all bolus injections between time step t and $t - \Delta$
$f_{\text{basal}}(t, \Delta)$	Sum of all basal injections between time step t and $t - \Delta$
$f_{\text{carbs}}(t, \Delta)$	Sum of all carbohydrates eaten between time step t and $t - \Delta$

Table 5.3: Feature set 3, used for evaluating the SVR model.

Feature	**Description**
$f_{\text{glucose}}(t)$	Blood glucose value at time t
$f'_{\text{glucose}}(t)$	Blood glucose value slope at time t

Table 5.4: Feature set 4, used for evaluating the SVR model.

Feature	Description
$f_{\text{glucose}}(t)$	Blood glucose value at time t
$f_{\text{bolus}}(t, \Delta)$	Sum of all bolus injections between time step t and $t - \Delta$
$f_{\text{basal}}(t, \Delta)$	Sum of all basal injections between time step t and $t - \Delta$
$f_{\text{carbs}}(t, \Delta)$	Sum of all carbohydrates eaten between time step t and $t - \Delta$

Table 5.5: Feature set 5, used for evaluating the LSTM model.

5.7 Baselines

To measure the performance of the LSTM-models, two baselines were implemented: an ARIMA model and an SVR model. This section details how each model was used.

5.7.1 ARIMA

For each prediction, 4 days of prior blood glucose data was used to create an ARIMA-model (3.3). For example, in the case of a 30 minute prediction (time step $t + 2$), a model was created using data up to time step t. Both the SVR-model and the LSTM-models were trained with 14 days of data, but empirical evaluation showed that the ARIMA-model performed best when trained with only 4 days of data.

The ARIMA-model was implemented in python and the model specific parameters were found using *auto.arima*, a function in the statistical programming language R. It uses the Philips-Perron (Phillips and Perron, 1988) unit root test for finding parameter d, and Bayes information criterion (Bhat and Kumar, 2010) to determine p and q.

5.7.2 SVR

The approached used in (Wiley, 2011) has been the main source of inspiration in the creation of the SVR-model. The model was created and evaluated using the first four feature sets detailed in (5.6). The radial basis function-kernel (RBF-kernel) (3.16) was the kernel of choice, since it only has one parameter and converges fast. Also, experimentation showed that it gave the best results.

The SVR-model was trained and evaluated using walk forward testing as described in section 5.3. A start date, or pivot date, was chosen in a way that minimised the number of gaps (5.4) in the data set for each patient. Data corresponding to 14 days prior to the pivot date was used as training data. Seven days of data prior to the initial pivot date was used as a development set for tuning. This tuning was only done once for each patient, i.e. it is not part of the walk forward procedure. The three days following the pivot date were used for validation, and the day after that

was used for testing and evaluation.

The development data set was used to find optimal parameters for the data features (5.5) and, due to the dependency between them, also for finding the hyper parameters of the SVR-model. As stated above, this tuning was only done once for each patient. However, the hyper parameters were re-tuned on the validation data ahead of each test day.

The parameters tuned for the data features are described in table 5.6 while table 5.7 describes the hyper parameters tuned for the SVR-model.

Name	**Description**
MVA decay	The decay parameter for the moving average feature.
ROC decay	The decay parameter for the rate of change feature.
Bolus delta	How far back in time bolus values will be included.
Bolus max k	The number of bolus features included in the vector.
Basal delta	How far back in time basal values will be included.
Basal max k	The number of basal features included in the vector.
Carbs delta	How far back in time carbohydrate values will be included.
Carbs max k	The number of carbohydrate features included in the vector.

Table 5.6: A list of feature parameters for the SVR model.

Name	**Description**
C	The cost parameter that controls the relation between the penalties of the slack variables and the size of the margin.
ϵ	Sets the margin for how wrong predictions can be before being penalised.
Gamma	Used internally by the RBF-kernel.

Table 5.7: A list of hyper parameters for the SVR model.

A good selection of the parameters is very important since they have a significant impact on overall performance (Bao and Liu, 2006). Tuning of hyper and feature parameters can not be done separately, thus creating a very large search space for the tuning. This makes a standard grid search too time consuming. Instead, tuning has been done with random search over the parameter space. This approach has proved to achieve as good, sometimes even better, results in a fraction of the time it would take a grid search (Bergstra and Bengio, 2012).

The parameters found in tuning, both for the feature and hyper parameters, were used to train the SVR-model with all 14 days of training data. The validation days are then used to once again tune the hyper parameters using random search.
Performance was measured by calculating the RMSE for the test day. After each test day the segment for training, validation and testing data was reset and moved forward 24 hours. This process was repeated for 7 days and the RMSE was calculated for each test day during this period. The overall performance was then calculated by taking the average RMSE over the 7 test days.

The SVR-model was implemented in python using LIBSVM (Chang and Lin, 2011), a library for SVM and SVR.

5.8 LSTM models

As opposed to the input for the SVR model consisting of single vectors, the input to the LSTM network is a sequence $\{\mathbf{x}_{t-l+1}, \mathbf{x}_{t-l+2}, \ldots, \mathbf{x}_t\}$ of l chronologically ordered vectors. Each such sequence has, in turn, a scalar valued target. These vectors are evenly spaced with 15 minutes between each time stamp. The LSTM network has as many cells as there are vectors in the input sequence, and each vector in the sequence is fed to exactly one cell, in chronological order. That is, the i'th vector in the sequence is the input of the i'th cell (Fig. 5.3). Knowledge about the different data attributes can be passed from an earlier point in time to subsequent ones, thus utilising the temporal properties of the LSTM network.

Just as the SVR model, the LSTM networks were evaluated using walk-forward testing with 14 training days, 3 validation days and 7 test days. An LSTM network is trained by feeding training data to the network and adjusting the weights associated with each gate in the cell (3.5) so that a loss function E on the output $\hat{y}$ is minimised. All such minimisations were done using backpropagation with the Adam algorithm (Kingma and Ba, 2014), a variation of SGD mentioned in 3.5.1.

Yet another important difference in the training of the LSTM to that of the SVR is that the training data is split up into multiple mini-batches that are then fed to the model. In other words, the backpropagation is not run after every single input sequence, but rather after a set of multiple ones. This way, the network generalises better and the training is faster to perform.

The hypersurface on which an LSTM network tries to find an optimum value generally can not be assumed to be convex. In other words, there can be many local optimas scattered across the surface, some of which might have values worse than the global one. The initial weights of an LSTM decide where on the hypersurface the optimisation starts. Since an LSTM can get stuck in a local optimum, it is important to start in the right place. In this work, the weight of all LSTM networks as well as their fully connected output layers have been randomly initialised. Although this yields good results most of the time, unfortunate initial values might leave the

optimisation stuck in bad local optimas. To accommodate for this, the training is repeated multiple times.

In contrast to an SVR, an LSTM can benefit from repeatedly being trained on the same data. Each repetition of training over the full training data set (all mini-batches in the case of mini-batch GD) is called an epoch, and the LSTM models evaluated here have been trained using multiple epochs. To determine when to stop the training, and thus avoid overfitting (3.2), a method called early stopping has been used. After each epoch, the performance of the LSTM is evaluated by measuring its prediction accuracy over the validation data set. When the performance has been decreasing for a predetermined amount of consecutive epochs, the training is stopped and the weights for the best performing epoch is used when evaluating the model over the test data.

The LSTM networks have a larger number of hyperparameters than the SVR and takes a longer time to train. Thus, performing a grid search to find the optimal ones would not be feasible. Training and evaluating an LSTM is significantly slower than an SVR, and the amount of hyperparameters makes the search space too vast. Instead, the hyperparameters used for the LSTM networks have been found by manually experimenting with values that seemed reasonable in this context. The most important hyperparameters related to the LSTM models are listed in table 5.8. The hyperparameters used for producing the final results reported in chapter 6 are presented in A.1.

Name	**Description**
Learning Rate	A parameter for the Adam optimiser that determines at what rate it should change the weights.
Batch size	The size of each mini-batch.
Sequence length	The length of the input sequence as well as the length of the unrolled LSTM network.
State size	The number of neurons in the interacting neural network layers within an LSTM cell. This number also decide the size of the cell state and the hidden state vectors.
Forget bias	A ratio determining how much information should be forgotten in the forget gate of an LSTM cell. This parameter has been set to 1 in all experiments based since research suggests this to usually be the optimal value for such LSTM here (Jozefowicz et al., 2015)
Resolution	The distance in time between each step (vector) in the input sequence.

Table 5.8: A summary of the most important hyperparameters of the LSTM models.

5.8.1 LSTM-1: Single layer LSTM with next-step prediction output

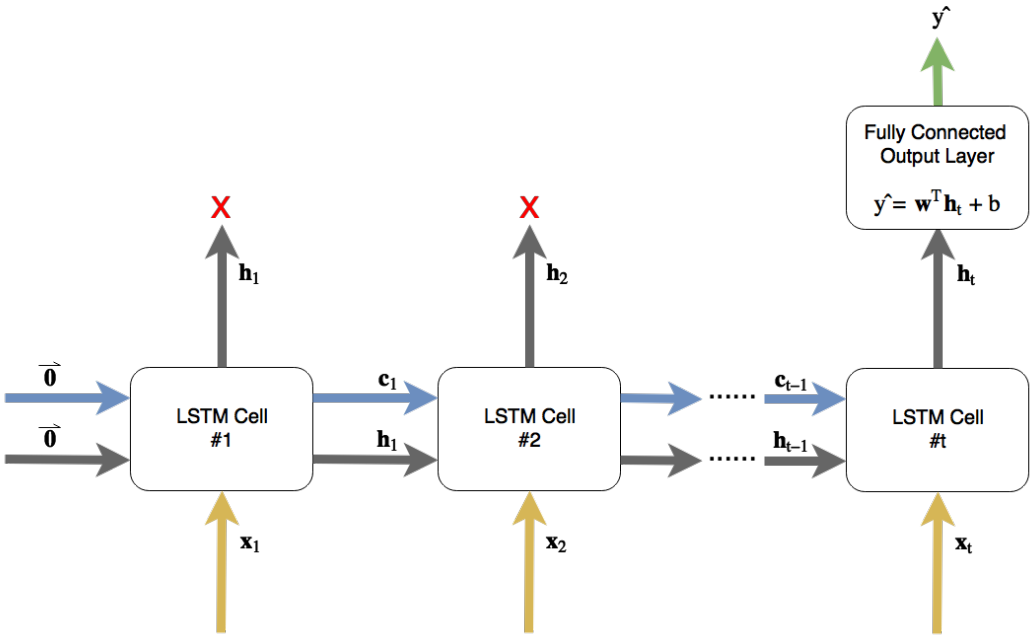

Figure 5.3: An illustration of the LSTM-1 model. The output of the last cell is used as input in the fully connected output layer which in turn produces a scalar output prediction.

In this model, from here on referred to as LSTM-1, only the output $\mathbf{h}_t$ from the last cell is used, and put through a fully connected output layer of size 1 to produce the final scalar output prediction $\hat{y}$ (Fig. 5.3).

During the training process, both the weights inside the LSTM network and in the fully connected output layer are tuned to minimise the output error. The loss is defined as the mean square error (MSE) over all predictions in a mini-batch of size k:

$$E = \frac{\sum_{i=1}^{k} (\hat{y}_i - y_i)^2}{k}, \tag{5.12}$$

where y_i is the target value for the i'th sequence in the batch and $\hat{y}$ the output from the fully connected output layer:

$$\hat{y} = \mathbf{w}^T \mathbf{h}_t + b\,. \tag{5.13}$$

The cell output $\mathbf{h}_t$ is defined in Eq. 3.32.

5.8.2 LSTM-2: Single layer LSTM with next-step distribution parameter output

This model is exactly as LSTM-1, with the exception of the output form and the use of a different loss function. Instead of outputting the predicted blood glucose

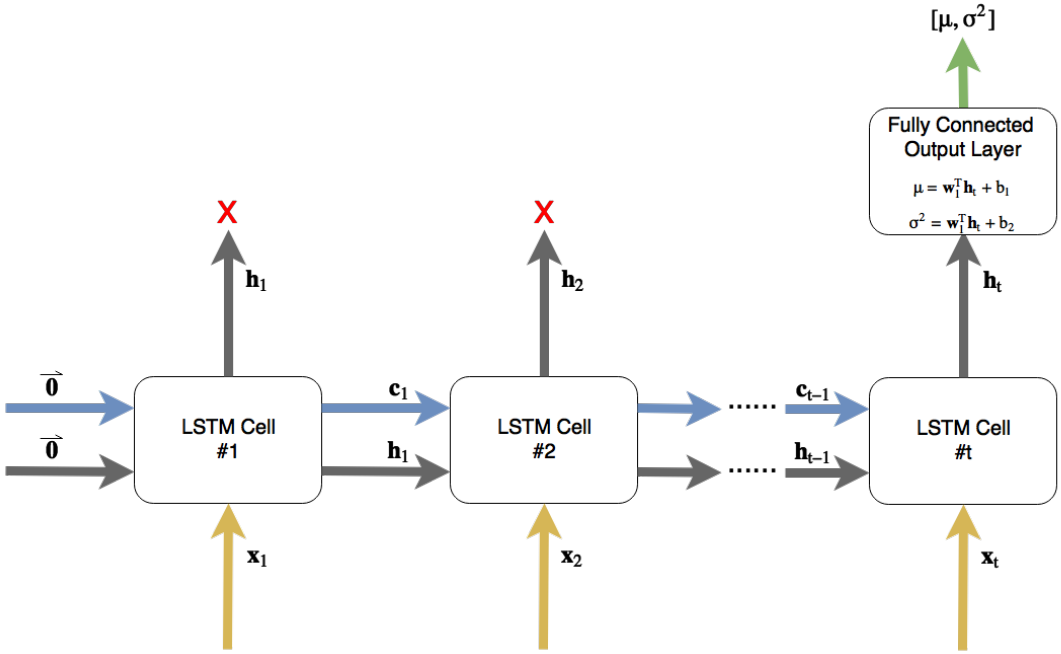

Figure 5.4: An illustration of the LSTM-2 model. The output of the last cell is used as input in the fully connected output layer which in turn produces a the mean μ and variance σ^2 of a Normal distribution.

value straight away, LSTM-2 outputs the mean μ and variance σ^2 of a univariate, Normal distribution (Fig. 5.4). Here, there are two weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$ in the output layer, and the mean and variance are calculated as follows:

$$\begin{aligned} \mu_i &= \mathbf{w_1}^T \mathbf{x_i} + b_1 \,, \\ \sigma_i^2 &= \mathbf{w_2}^T \mathbf{x_i} + b_2 \,. \end{aligned} \tag{5.14}$$

The loss over a mini-batch of size k is defined as

$$E = \frac{\sum_{i=1}^{k} -\log\left(N(y_i|\mu_i, \sigma_i^2)\right)}{k} \,, \tag{5.15}$$

where y_i is the target value for the i'th sequence in the batch and the probability density function $N(y_i|\mu_i, \sigma_i^2)$ is defined as

$$N(y_i|\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\sigma_i^2 \pi}} e^{-\frac{(y_i - \mu_i)^2}{2\sigma_i^2}} \,. \tag{5.16}$$

Minimising loss with the above defined loss function corresponds to maximising the probability density of the distribution around the target blood glucose value. The mean μ in the output vector is also used as the prediction itself.

The main benefit with this model compared to LSTM-1 is that it outputs a measure of "confidence" in its predictions. A lower variance can be interpreted as the network being confident in its prediction, while a higher variance signals the opposite.

Note that although LSTM-2 uses a different loss function than LSTM-1, the performance is evaluated in the same manner using RMSE as error metric over the test data.

6
Results

This chapter presents the performance of the SVR model in 5.7.2, the ARIMA-model in 5.7.1 and the two different LSTM based models in 5.8. The results presented are for 30 and 60 minute prediction horizons. Refer to appendix [A.3] for prediction horizons beyond 60 minutes.

The SVR was evaluated on feature sets 1-4. Sets 2 and 3 are based on the work of (Wiley, 2011). Set 4 was included because of its minimal way of representing change in blood glucose compared to set 2 and 3. Finally, set 1 was included as a reference for how the SVR performs with as little feature engineering as possible. The ARIMA-model used only feature set 1.

Due to the inherent sequential structure of an LSTM network, both LSTM-1 and LSTM-2 were evaluated on feature sets without any manually engineered interpretation of change in blood glucose (such as rate of change, moving average or slope). Intuitively, an LSTM should be able to learn such properties without manually incorporating them into the feature sets themselves.

The ARIMA-model was trained and evaluated using 4 days of training data and 7 days as test data while the SVR and LSTM-models were trained using walk forward testing with 14 days of training data, 3 days of validation data and 7 days of test data.

Note that all RMSEs are given in mmol/L. Also, see A.4 for a visual comparison.

6.1 30 minute predictions

As expected, the worst SVR performance was with feature set 1. It does not include any information about in which direction the blood glucose is heading, and thus the SVR has a harder time making accurate predictions.

Surprisingly, the SVR performs equally well using feature set 2 and 4, even though the former contains the ARIMA prediction as well as a more expressive representation of the blood glucose change.

Embedding information about insulin dosage and carbohydrate intake in the data does not induce any significant performance boost. In fact, on average, it causes

Patient	**SVR** set 1	**SVR** set 2	**SVR** set 3	**SVR** set 4
#1	1.32	0.98	1.12	1.00
#2	1.09	0.94	0.90	0.90
#3	1.48	1.19	1.16	1.19
#4	1.14	0.91	0.92	0.93
Avg.	1.26	**1.00**	1.02	**1.00**

Table 6.1: 30 minute prediction RMSEs for the SVR model.

slightly worse performance. Both LSTM-1 and LSTM-2 perform similarly, with a

Patient	**LSTM-1** set 1	**LSTM-1** set 5	**LSTM-2** set 1	**LSTM-2** set 5
#1	1.09	1.03	1.02	1.05
#2	0.91	0.90	0.91	0.91
#3	1.15	1.16	1.15	1.13
#4	0.92	0.96	0.91	0.98
Avg.	1.02	1.01	**1.00**	1.02

Table 6.2: 30 minute prediction RMSEs for the LSTM models.

slight edge in prediction performance to LSTM-2. The difference is in the order of magnitude that could be discarded as variations due to the stochasticity involved in the training. Note, however, that the difference in performance for larger prediction horizons is greater, thus suggesting that LSTM-2 is indeed the better of the two.

Another observation is that, just as for the SVR model, there seems to be no significant benefit from embedding information about insulin and carbohydrates in the input. LSTM-1 performed slightly better with feature set 5, but the difference is too small to establish feature set 5 as superior to feature set 1.

Patient	**LSTM-1** set 5	**LSTM-2** set 1	**SVR** set 2	**ARIMA**
#1	1.03	1.02	0.98	1.03
#2	0.90	0.91	0.94	0.93
#3	1.16	1.15	1.19	1.19
#4	0.96	0.91	0.91	0.92
Avg.	1.01	**1.00**	1.00	1.02

Table 6.3: Side by side comparison of 30 minute prediction RMSEs for all models.

The side by side comparison of the different models show that the best LSTM model, LSTM-2, has performance equal to the best-performing SVR model. In fact, it performs better than the SVR on patient #2 and #3 and shows equal performance on patient #4. The RMSE for patient #1 is slightly worse than for the SVR, balancing out the gain in performance on the other patients.

Fig. 6.1 shows an example output for the LSTM-2 model on a day with hard to predict excursions in the blood glucose values. Still, the LSTM-2 expresses high confidence in its predictions. The target values are often inside the range $[\mu_i - \sigma_i, \mu_i + \sigma_i]$, and the standard deviation generally does not fluctuate by much. The model shows the lowest level of confidence around 17:00 where the blood glucose is unusually high for this particular patient.

On average, over all test days for all patients, the target value lies inside $[\mu_i - \sigma_i, \mu_i + \sigma_i]$ about 80% of the time.

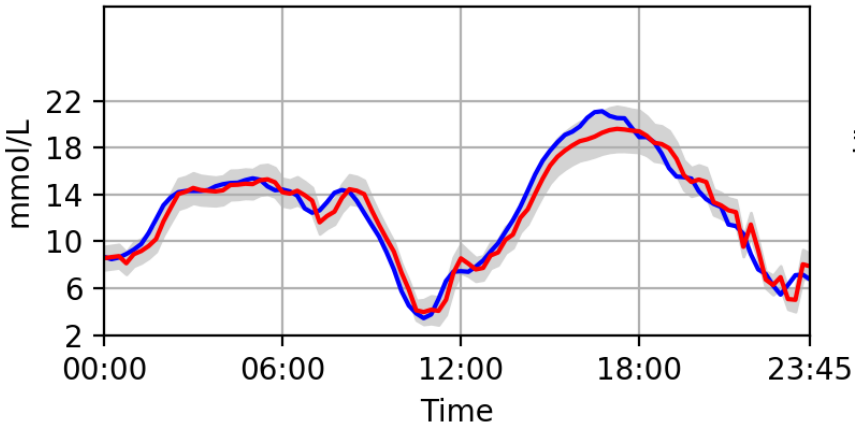

Figure 6.1: A plot of the 30 minute predictions of LSTM-2 (in red) versus the target values (in blue) for one of SP92's test days. The grey area around the prediction line shows the standard deviation of the output distribution.

6.2 60 minute predictions

Patient	**SVR** Set 1	**SVR** Set 2	**SVR** Set 3	**SVR** Set 4
#1	2.24	1.86	2.03	1.86
#2	1.74	1.62	1.55	1.65
#3	2.39	2.14	2.11	2.17
#4	1.97	1.65	1.77	1.67
Avg.	2.09	**1.82**	1.87	1.84

Table 6.4: 60 minute prediction RMSEs for the SVR model.

Patient	**LSTM-1** set 1	**LSTM-1** set 5	**LSTM-2** set 1	**LSTM-2** set 5
#1	1.98	2.04	1.90	1.96
#2	1.63	1.65	1.58	1.58
#3	2.13	2.08	2.08	2.11
#4	1.83	1.83	1.67	1.77
Avg.	1.89	1.90	**1.81**	1.86

Table 6.5: 60 minute prediction RMSEs for the LSTM models.

Patient	**LSTM-1** set 1	**LSTM-2** set 1	**SVR** set 2	**ARIMA**
#1	1.98	1.90	1.86	1.99
#2	1.63	1.58	1.62	1.65
#3	2.13	2.08	2.14	2.22
#4	1.83	1.67	1.65	1.76
Avg.	1.89	**1.81**	1.82	1.91

Table 6.6: Side by side comparison of 60 minute prediction RMSEs for all models.

As expected, a longer prediction horizon produces worse performance for all models. LSTM-2 performs similar to the SVR. In fact, it performs slightly better, even though the difference is very small.

As mentioned in the previous section, there is a greater difference in performance between the LSTM models that can no longer be attributed to the stochasticity in the training process.

Fig. 6.2 shows that the LSTM-2 model outputs a standard deviation that generally is higher than during the same day for a 30 minute prediction horizon 6.1. Also, the target value is outside of the range $[\mu_i - \sigma_i, \mu_i + \sigma_i]$ more often.

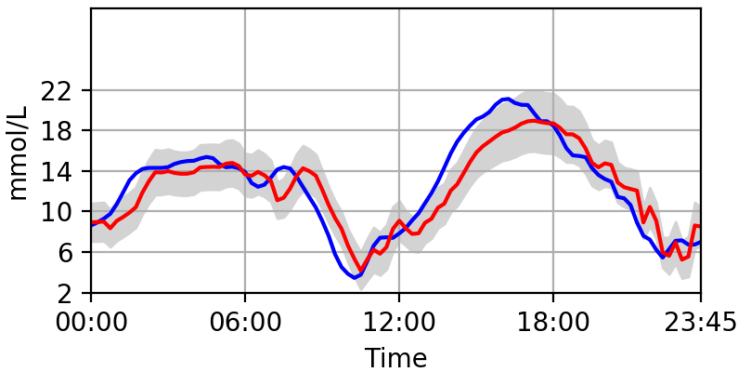

Figure 6.2: A plot of the 60 minute predictions of LSTM-2 (in red) versus the target values (in blue) for one of SP92's test days. The grey area around the prediction line shows the standard deviation of the output distribution.

For prediction horizons larger than 60 minutes, please refer to A.3.

6.3 Statistical significance

To establish the statistical significance of this work, a two-tailed T-test was performed. The hypotheses were defined as

H_0 : LSTM has performance equal to SVR on average. ($\mu_{\text{LSTM}} = \mu_{\text{SVR}}$)

H_1 : LSTM performs better or worse than SVR on average ($\mu_{\text{LSTM}} \neq \mu_{\text{SVR}}$)

where H_0 is the null hypothesis, H_1 the alternative, μ_{LSTM} is the mean LSTM RMSE and μ_{SVR} the mean SVR RMSE. The best performing combination of model and/or feature set was used for both SVR and LSTM.

The mean RMSE for SVR and LSTM (LSTM-2) was both $\mu_{\text{SVR}} = \mu_{\text{LSTM}} = 1.00$. The sample standard deviation for the LSTM was $\sigma_{\text{LSTM}} = 0.16$. Denote $N = 28$ as the total number of days from which the errors were calculated. The t-statistic was then calculated as

$$T = \frac{\mu_{\text{SVR}} - \mu_{\text{LSTM}}}{\frac{\sigma_{\text{LSTM}}}{\sqrt{N}}} = 0\,. \tag{6.1}$$

This yields a p-value of 0.95, suggesting that the alternative hypothesis H_1 can be rejected, and the null hypothesis H_0 accepted. In other words, on average, the LSTM model has performance equal to the baseline SVR model.

7
Discussion

This section details how the results presented in this work compares to related research. It also discusses possible reasons for why the LSTM models did not show an increase in performance compared to the baseline. Furthermore, some ethical issues are explored and suggestions are made for how this work can be improved upon in the future.

7.1 Comparison with related research

Since related research have been made using data from other patients than ours, a direct side by side comparison with our models' prediction errors could not be made. Instead, an SVR-model based on the one created by (Wiley, 2011) was implemented and evaluated on the same data sets as our LSTM models. An ARIMA model identical to the one used as a baseline by Wiley was used to verify the performance of the SVR. In the work by Wiley, the SVR-model performed nearly identical to, or marginally better then the ARIMA-model in all cases. Our SVR-model beats the ARIMA-model in all cases, thus suggesting it performs at least as good as Wiley's implementation.

For 30 minute predictions, our models had performance equal to that of the SVR, and LSTM-2 seems to be the better of the two models presented. Our hope was that LSTM would better capture the dynamics between blood glucose, insulin and carbohydrate intake with its ability to model long-term dependencies. Unfortunately, this does not seem to be the case. The LSTM models do not show any boost in performance when incorporating such data in the input, and the reasons could be many. Bunescu implemented a physiological model utilised by the SVR in (Bunescu et al., 2013). This model gave 12.2% decrease in RMSE, compared to when representing the model in the same way as Wiley. Such a physiological model could not be implemented here, due to the lack of specific patient data such as body mass etcetera, but might have been beneficial for the LSTM models as well.

Another reason for why incorporating extra data about insulin and carbohydrate in the data did not increase prediction accuracy might be that the data is sparse. Most hours during the day, no bolus is injected and no food ingested. This means the related attributes in the input vectors will have a value of zero most of the time. This, in turn, means the number of examples in the training data when carbohydrates intake and/or insulin dosage has played an important role are few in comparison to

the number of input vectors that only contain information about past blood glucose values.

Even though the LSTM models were not able to utilise information related to insulin and carbohydrates, an overall performance improvement independently of the feature set used could have been expected since an LSTM network has a structure suitable for temporal data. It has an internal state in which dependencies between the different values in a given input sequence can be captured, something that, for example, the SVR does not have. Despite this, the LSTM models does not perform better, and the reason could be that there are not that many complex dependencies between time steps in a sequence.

Fig. 7.1 compares the predictions made by the SVR model and the LSTM-1 model during a given day. There is a striking similarity between the two, and one can see that both models seem to struggle in the same areas. For example, just after 06:00, there is a zigzag pattern in the predictions for both models. Also, both models seem to express a slight delay in their predictions in some areas of the plots. These similarities between the two models could indicate that it is hard to make better predictions given this particular data. The sampling rate (every 15 minutes) as well as the accuracy of the underlying CGM system might not be enough. And there might be more factors that affect the blood glucose level not contained in this data. Also, the way we have chosen to represent the data might be a limiting factor.

Finally, one reason for the lack of any significant performance increase with the new models might be that the problem itself is indeed hard to solve, and that more novel approaches are required to further improve prediction accuracy. The raw RMSE numbers we get for both the baseline models and the new LSTM models are comparable to the ones presented in related research. For example, (Bunescu et al., 2013) is able to make 30 minute predictions with an average RMSE of 1.08 mmol/L (19.5 mg/dl) with the physiological model to represent the data mentioned earlier. Another example is (Mougiakakou et al., 2006), in which the average RMSE using a feed-forward neural network on data from four patients is 1.23 mmol/L (22.17 mg/dl). To our knowledge, the best performing prediction model is presented in (Zecchin et al., 2012). It uses a neural network in combination with a first-order polynomial extrapolation algorithm, and an average RMSE of 0.78 mmol (14 mg/dl) is achieved. The model in that research was able to better capture the effects of carbohydrates, hence the lower RMSE.

It is important to note that although number by number comparisons such as the ones made above are interesting, they can also be misleading since overall blood glucose control differs between patients. I.e. the data sets evaluated could significantly affect the results.

Although the LSTM models presented here do not show an increase in prediction accuracy compared to the SVR model, LSTM-2 outputs a measure of confidence together with its predictions. Having a confidence measure at hand can be an

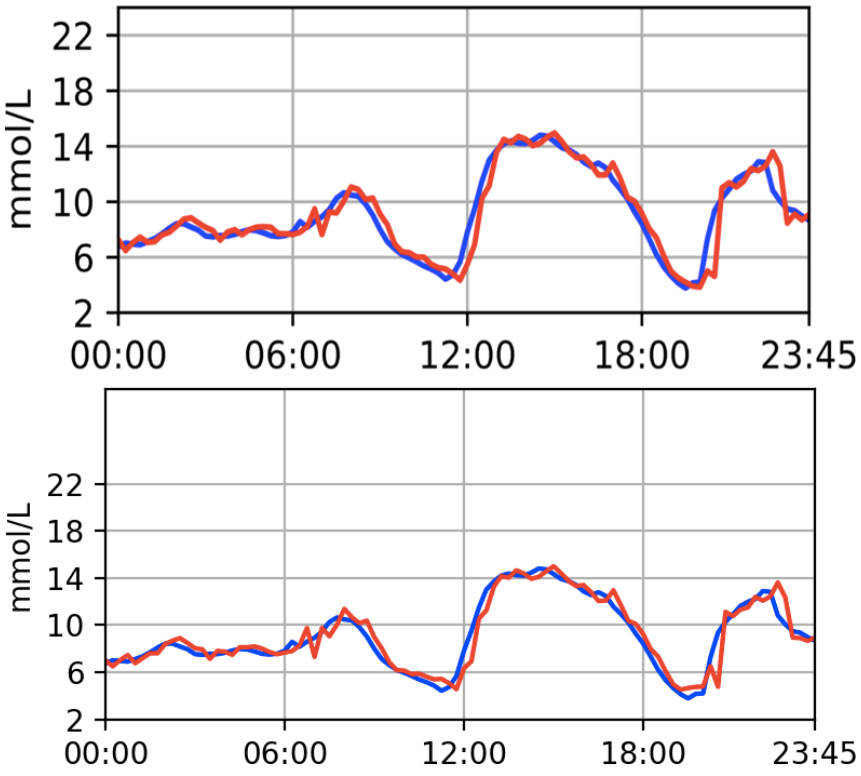

Figure 7.1: A plot comparing the predictions of the SVR model (top) and LSTM-1 model (bottom) during a particular day. Predicted values are shown in red, and target values in blue.

advantage, since a possible end-user could use this information to determine the reliability of the predictions.

7.2 Usability

If the average error over the four patients in this work is a good indication of how the models perform in general, it means that a prediction on average differs by ±1.00 mmol/L. Combining such predictions with other therapeutic knowledge could help in the treatment of type 1 diabetes. Especially the LSTM-2 model, that provides a form of uncertainty together with its prediction; information that could further assist the patient in its treatment.

It is important to note that the results reported are averages. Some predictions might be far worse than the average. In such situations, the LSTM-2 model could prove especially useful since it outputs the standard deviation together with the prediction. Looking at Fig. 7.2, it is evident that the real values mostly are within the standard deviation of the predicted value. There are some instances when the LSTM model is certain of its predictions and still misses the target. However, it accurately captures the trend in how the blood glucose level changes, suggesting the predictions are still useful.

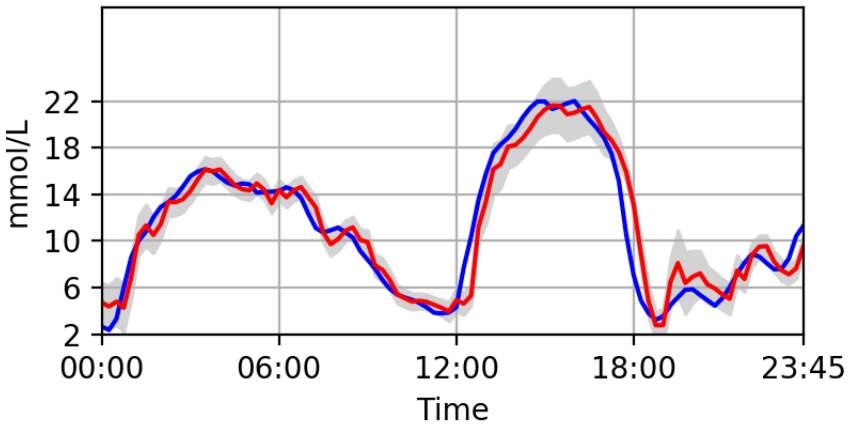

Figure 7.2: A plot of predicted versus target values over a relatively challenging day.

The day in 7.2 is particularly difficult to predict since there are drastic changes in blood glucose throughout the day. Such days are still of relevance though, since the occasions when the blood glucose value changes drastically are the most important to catch. A further exploration on how to best handle these situations could therefore be relevant to conduct in possible future works on blood glucose prediction.

From a purely academical perspective, it is disappointing that the LSTM models failed to utilise insulin and carbohydrates to further increase the accuracy of the predictions. From a practical viewpoint, however, it is easier to implement and maintain a model that uses only blood glucose values for its predictions. Keeping note of and feeding a model with additional data such as carbohydrates can be cumbersome and impracticable for the patient. The process of feeding a model with the current blood glucose (and potentially insulin dosages) could, on the other hand, be automated and consequently more easier to use.

Predictions made further than 30 minutes into the future show drastic decrease in performance. It is possible that the models could prove useful for 60 minute predictions, although mainly as an indicator of where the blood glucose level is headed.

7.3 Other models and evaluation methods

The big difference between walk forward testing and other more common evaluation methods, is that the model is retrained for every new test day. I.e. older data is dropped and are no longer used as the segment walks forward. Since training the SVR-model with more than 14 days of data does not improve performance, this evaluation method is appropriate.

Our initial thought was that the LSTM, on the other hand, would benefit from more data, suggesting that another evaluation method than walk forward testing would be

more suitable. We therefore tried increasing the amount of training data beyond 14 days all the way up to over 60 days, the maximum number of training days available in the biggest data set. Adding more days of data to the training set did not give better performance. Instead, the performance saw a decrease, suggesting that recent blood glucose values have a greater impact on future values. For example, a patient can have a bad month with a lot of variations in the glucose levels. A month later when the patient has stabilized his/her levels, the old data would probably not help when making new predictions. This could obviously differ from patient to patient, and it is possible that a patient with similar trends in blood glucose levels over long periods could benefit from more training data. Of course, both the SVR and the LSTM were also evaluated using fewer days of data, but that gave a decrease in performance.

In an attempt to improve performance we tried splitting the data in order to run it on different LSTM-models and concatenating their results into a fully connected layer. This enabled us to use different time intervals for the different parts of the input data. For example, we tried to look more on specific periods of the day as well as which day of the week it is. I.e., when trying to predict blood glucose values for 12:30 on a Monday, we tried using the data for the same time of day from the days prior. For the patients with enough data we also tried looking at that specific day and time of the week for previous weeks. Running different "time lines", on different LSTM-models and concatenating the results. In an attempt to better utilise the life data, we used a similar approach and tried having only blood glucose in one LSTM and life data in another and again concatenating their results into a fully connected output layer. This, however, did not improve performance.

Adding stochasticity to the training process was also tested by splitting and shuffling the data used for training and validation. This, however, always resulted in worse performance. Having the validation data chronologically just before the test data gives better performance, again suggesting that more recent blood glucose values are more relevant when predicting.

For all methods mentioned above we also tried adding more layers to the LSTM models, i.e. using multiple stacked layer of LSTM networks, in attempts to boost performance. This, unfortunately, did not achieve better results. Instead, it only we only saw an increase in running time.

7.4 Real time prediction system

From a technical point of view, there are some hurdles to overcome in order to create a real time prediction system for blood glucose values. Ideally such a system should warn a patient when to take action in order to avoid possibly harmful future blood glucose levels. To accomplish this, the system need to be continuously updated with the latest blood glucose levels.

This can be done in several ways. The ideal solution would be to run the entire

prediction system on the same hardware that performs the CGM, but training an LSTM-model can be computationally hard on such hardware. However, making predictions is not computationally hard once the prediction model is trained. Thus one possibility is to train the LSTM-model on external hardware and then let the CGM-system do only the actual predictions. Another solution would be if CGM-systems send data to external hardware such as a smartphones. Then training and predictions can then be made by an app on the phone hardware. However, even though these technical challenges are interesting solving them goes beyond the scope of this thesis.

7.5 Ethical considerations

While a system that recommends therapeutic changes to a patient could prove useful, it also introduces some moral dilemmas. Instructing a person on treatment of a disease that in some cases can be deadly, is dangerous. No matter the disclaimers such a system would be advertised with to potential users, there is always a chance that someone would blindly follow the recommendations given by the system. In a worst-case scenario, doing so might result in serious complications for the patient. For example, an inappropriate insulin dosage can lead to hypoglycaemia, a deficiency of glucose in the bloodstream that can cause illness, seizures and even death if not treated properly.

If a recommendation by a decision support system result in a serious complications for the patient, it is not clear who should be held responsible. It could be argued that if the system is advertised with a disclaimer explicitly saying not to trust all its recommendations blindly, the user is the only one to blame if she goes ahead and does so anyway. But what if the patient did not read the disclaimer too carefully, or worse, if there was a misunderstanding leading to the patient fully trusting the system even though one should not?

From an ethical viewpoint there is no simple answer to who should be held responsible in case of an accident caused by such a product. There is, however, a way to minimise the number of such accidents: clear, non-ambiguous warnings and disclaimers that the user is more or less forced to read and understand before using the product. Maybe a decision support system for diabetes should be handed over to patients by their personal diabetes team in order to guarantee that he/she knows the risks.

7.6 Future work

There are multiple ways in which this work could be continued. One very straightforward improvement would be to evaluate the LSTM models over data from a larger set of patients. Doing so would not only further establish (or disestablish) the significance of the LSTM models, but possibly also generate new ideas and insights as

to how they can be improved.

The data used in this work was not only limited in size, but also the amount of data features. It might very well be the case that information other than past blood glucose values, insulin dosage and carbohydrate intake is relevant when predicting blood glucose values. Having a data set with a richer set of features could also help in constructing new data models that render even better performance. Taking metrics such as exercise and sleep patterns, weight or even pulse into account might improve the prediction performance. Furthermore, having more data features available would also make it possible to combine LSTM with the physiological data models explored in (Bunescu et al., 2013), something that could further improve performance further.

An interesting aspect of our results is that the LSTM models do not seem to benefit from having carbohydrate intake or insulin dosage as attributes in the input data. Food ingestion generally raise the blood glucose level, while insulin lowers it. The fact that the LSTM cannot benefit from such information suggests that further research might be needed on how to better utilise the data.

8

Conclusion

The goal of this thesis was to extend current research on predicting blood glucose values for patients with type 1 diabetes. Two LSTM models (5.8.1) were implemented and evaluated on data from four different patients using walk forward testing (5.3) with 14 training days, 3 validation days and 7 test days.

In order to quantify the level of uncertainty in the predictions, one of the LSTM model (5.8.2) was implemented with a Gaussian output. More specifically, an LSTM that outputs the mean and standard deviation of a normal distribution of future values. The mean can be seen as the prediction itself, while the standard deviation can be interpreted as the level of confidence in the prediction. Such information could be useful when determining how much a prediction can be trusted.

The LSTM-models achieves these results using only previous blood glucose values, i.e. not considering any additional data such as insulin and carbohydrates. This can be beneficial for the potential user of a blood glucose prediction system since blood glucose values are easy to record, while carbohydrate intake is not.

It was established with high statistical significance that the LSTM model with Gaussian output had a prediction error similar to the SVR baseline. This, in combination with the added ability to output prediction confidence suggests LSTM networks could be used to provide valuable insights in the treatment of type 1 diabetes.

Bibliography

A. Abraham. Artificial neural networks. *handbook of measuring system design*, 2005.

A. D. Association et al. Physical activity/exercise and diabetes. *Diabetes care*, 27 (suppl 1):s58–s62, 2004.

M. A. Atkinson, G. S. Eisenbarth, and A. W. Michels. Type 1 diabetes. *The Lancet*, 383(9911):69–82, 2014.

T. O. Ayodele. *Machine learning overview*. INTECH Open Access Publisher, 2010.

Y. Bao and Z. Liu. A fast grid search method in support vector regression forecasting time series. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 504–511. Springer, 2006.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

E. M. Benjamin. Self-monitoring of blood glucose: the basics. *Clinical diabetes*, 20 (1):45–47, 2002.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

H. S. Bhat and N. Kumar. On the derivation of the bayesian information criterion. *School of Natural Sciences, University of California*, 2010.

E. Boland, T. Monsod, M. Delucia, C. A. Brandt, S. Fernando, and W. V. Tamborlane. Limitations of conventional methods of self-monitoring of blood glucose. *Diabetes care*, 24(11):1858–1862, 2001.

G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait. High-performance ocr for printed english and fraktur using lstm networks. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 683–687. IEEE, 2013.

R. Bunescu, N. Struble, C. Marling, J. Shubrook, and F. Schwartz. Blood glucose level prediction using physiological models and support vector regression. In *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, volume 1, pages 135–140. IEEE, 2013.

R. Caruana, S. Lawrence, and L. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *NIPS*, pages 402–408, 2000.

C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

P. E. Cryer, S. N. Davis, and H. Shamoon. Hypoglycemia in diabetes. *Diabetes care*, 26(6):1902–1912, 2003.

D. Daneman. Type 1 diabetes. *The Lancet*, 367(9513):847–858, 2006.

D. Eck and J. Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 747–756. IEEE, 2002.

D. W. Foster and J. D. McGarry. The metabolic derangements and treatment of diabetic ketoacidosis. *New England Journal of Medicine*, 309(3):159–169, 1983.

A. Graves, D. Eck, N. Beringer, and J. Schmidhuber. Biologically plausible speech recognition with lstm neural nets. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pages 127–136. Springer, 2004.

M. J. Haller, M. S. Stalvey, and J. H. Silverstein. Predictors of control of diabetes: monitoring may be the key. *The Journal of pediatrics*, 144(5):660–661, 2004.

M. J. Haller, M. A. Atkinson, and D. Schatz. Type 1 diabetes mellitus: etiology, presentation, and management. *Pediatric Clinics of North America*, 52(6):1553–1578, 2005.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

W.-C. Hong. Rainfall forecasting by technological machine learning models. *Applied Mathematics and Computation*, 200(1):41–57, 2008.

C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification. 2003.

D. Jenkins, T. Wolever, R. H. Taylor, H. Barker, H. Fielden, J. M. Baldwin, A. C. Bowling, H. C. Newman, A. L. Jenkins, and D. V. Goff. Glycemic index of foods: a physiological basis for carbohydrate exchange. *The American journal of clinical nutrition*, 34(3):362–366, 1981.

R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350, 2015.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques, 2007.

X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187–197, 2015.

P. Malhotra, L. Vig, G. Shroff, and P. Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

S. G. Mougiakakou, A. Prountzou, D. Iliopoulou, K. S. Nikita, A. Vazeou, and C. S. Bartsocas. Neural network based glucose-insulin metabolism models for children with type 1 diabetes. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 3545–3548. IEEE, 2006.

S. M. Pappada, B. D. Cameron, P. M. Rosman, R. E. Bourey, T. J. Papadimos, W. Olorunto, and M. J. Borst. Neural network-based real-time prediction of glucose in patients with insulin-dependent diabetes. *Diabetes technology & therapeutics*, 13(2):135–141, 2011.

P. C. Phillips and P. Perron. Testing for a unit root in time series regression. *Biometrika*, pages 335–346, 1988.

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

B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

C. Shan. Learning local binary patterns for gender classification on real-world face images. *Pattern Recognition Letters*, 33(4):431–437, 2012.

F. Weninger, H. Erdogan, S. Watanabe, E. Vincent, J. Le Roux, J. R. Hershey, and B. Schuller. Speech enhancement with lstm recurrent neural networks and its application to noise-robust asr. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 91–99. Springer, 2015.

W. H. O. WHO. Global report on diabetes (p. 6). Report, 2016. Geneva.

M. T. Wiley. *Machine learning for diabetes decision support (pp. 55-72)*. PhD thesis, Ohio University, 2011.

K. Żbikowski. Using volume weighted support vector machines with walk forward testing and feature selection for the purpose of creating stock trading strategy. *Expert Systems with Applications*, 42(4):1797–1805, 2015.

C. Zecchin, A. Facchinetti, G. Sparacino, G. De Nicolao, and C. Cobelli. Neural network incorporating meal information improves accuracy of short-time prediction of glucose concentration. *IEEE Transactions on Biomedical Engineering*, 59 (6):1550–1560, 2012.

A

Appendix 1

A.1 LSTM-1 Hyperparameters

What follows is a list of the hyperparameters used for each prediction horizon when evaluating LSTM-1.

30 minutes
- Learning rate: 0.01
- Batch size: 288
- Sequence length: 16
- State size: 4
- Forget bias: 1.0
- Resolution: 15 minutes

60 minutes
- Learning rate: 0.01
- Batch size: 288
- Sequence length: 16
- State size: 4
- Forget bias: 1.0
- Resolution: 15 minutes

90 minutes
- Learning rate: 0.01
- Batch size: 288
- Sequence length: 4
- State size: 4
- Forget bias: 1.0
- Resolution: 15 minutes

120 minutes
- Learning rate: 0.01
- Batch size: 288
- Sequence length: 8
- State size: 4
- Forget bias: 1.0

- Resolution: 15 minutes

A.2 LSTM-2 Hyperparameters

What follows is a list of the hyperparameters used for each prediction horizon when evaluating LSTM-2.

30 minutes
- Learning rate: 0.002
- Batch size: 96
- Sequence length: 16
- State size: 8
- Forget bias: 1.0
- Resolution: 15 minutes

60 minutes
- Learning rate: 0.002
- Batch size: 288
- Sequence length: 4
- State size: 8
- Forget bias: 1.0
- Resolution: 15 minutes

90 minutes
- Learning rate: 0.002
- Batch size: 288
- Sequence length: 4
- State size: 8
- Forget bias: 1.0
- Resolution: 15 minutes

120 minutes
- Learning rate: 0.002
- Batch size: 288
- Sequence length: 4
- State size: 8
- Forget bias: 1.0
- Resolution: 15 minutes

A.3 90 and 120 minute predictions

Patient	**LSTM-1** set 1	**LSTM-2** set 5	**SVR** set 2	**ARIMA**
#1	2.68	2.64	2.51	2.80
#2	2.45	2.07	2.15	2.16
#3	2.90	2.88	2.79	2.84
#4	2.38	2.37	2.23	2.45
Avg.	2.60	2.49	**2.42**	2.56

Table A.1: Side by side comparison of 90 minute prediction RMSEs for all models.

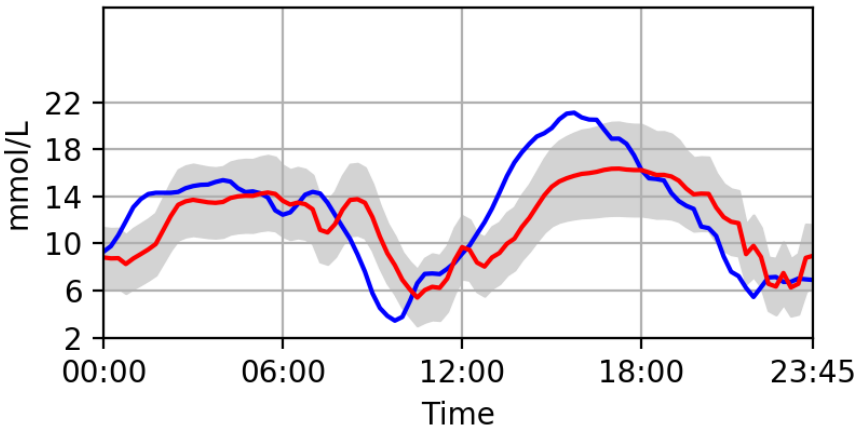

Figure A.1: A plot of the 90 minute predictions of LSTM-2 (in red) versus the target values (in blue) for one of SP92's test days. The grey area around the prediction line shows the standard deviation of the output distribution.

Patient	**LSTM-1** set 1	**LSTM-2** set 1	**SVR** set 4	**ARIMA**
#1	3.25	3.17	3.06	3.45
#2	2.67	2.57	2.51	2.47
#3	3.32	3.13	3.14	3.21
#4	2.94	2.86	2.69	2.82
Avg.	3.04	2.93	**2.85**	2.99

Table A.2: Side by side comparison of 120 minute prediction RMSEs for all models.

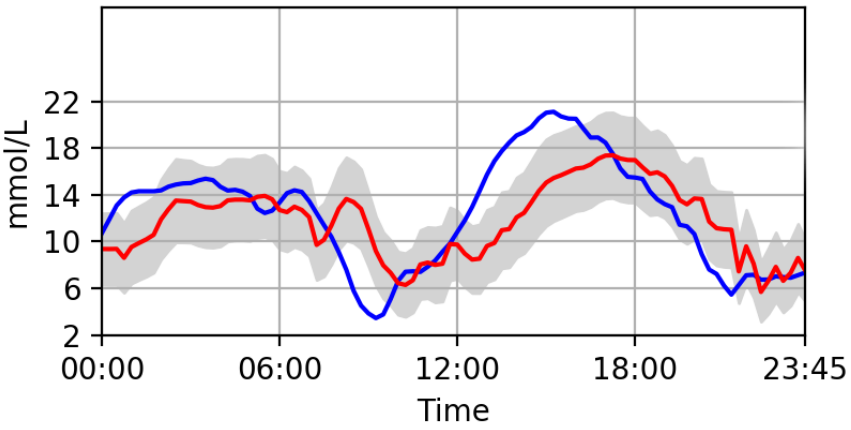

Figure A.2: A plot of the 120 minute predictions of LSTM-2 (in red) versus the target values (in blue) for one of SP92's test days. The grey area around the prediction line shows the standard deviation of the output distribution.

A.4 RMSE comparisons for SVR and LSTM

A.4.1 30 minute prediction horizon

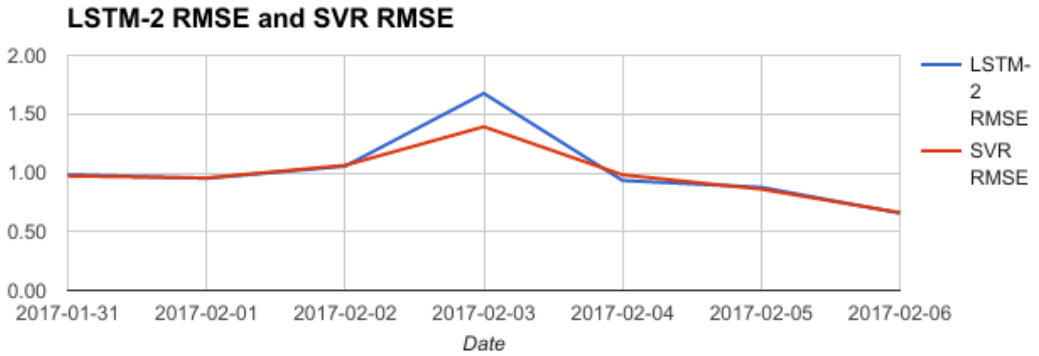

Figure A.3: The average 30 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #1.

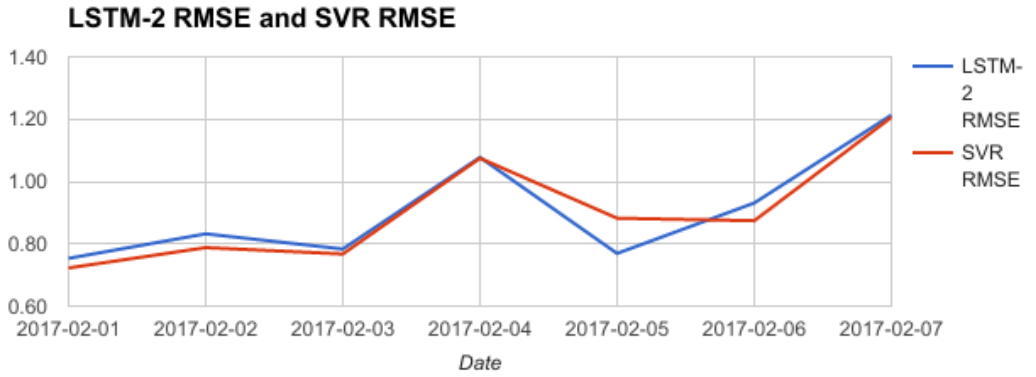

Figure A.4: The average 30 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #2.

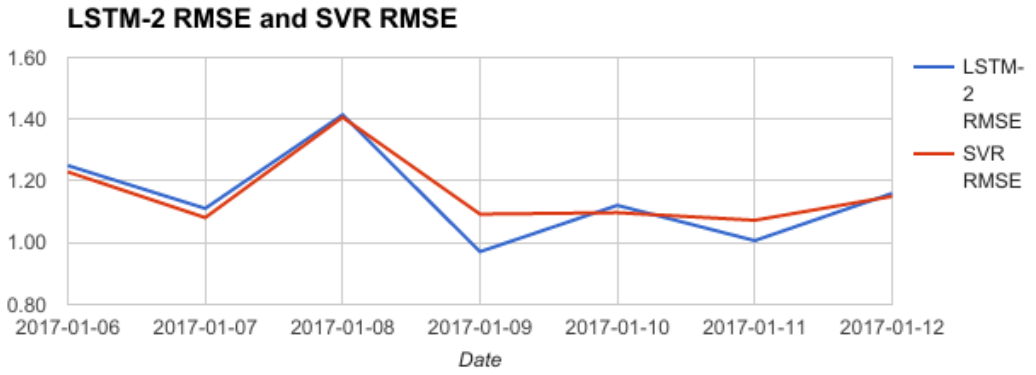

Figure A.5: The average 30 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #3.

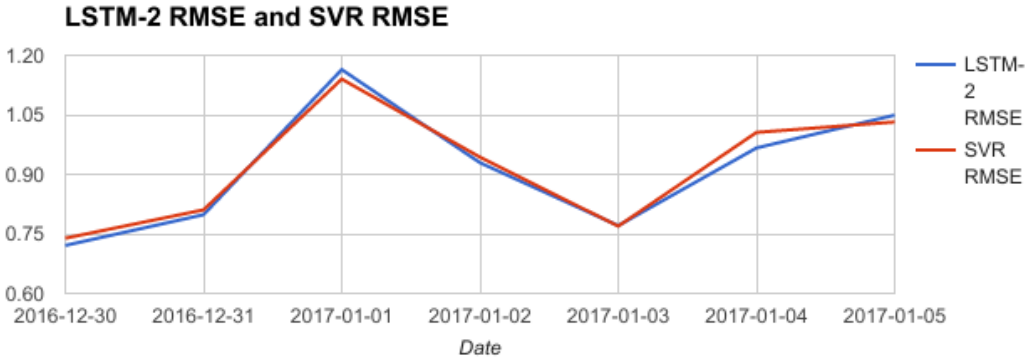

Figure A.6: The average 30 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #4.

A.4.2 60 minute prediction horizon

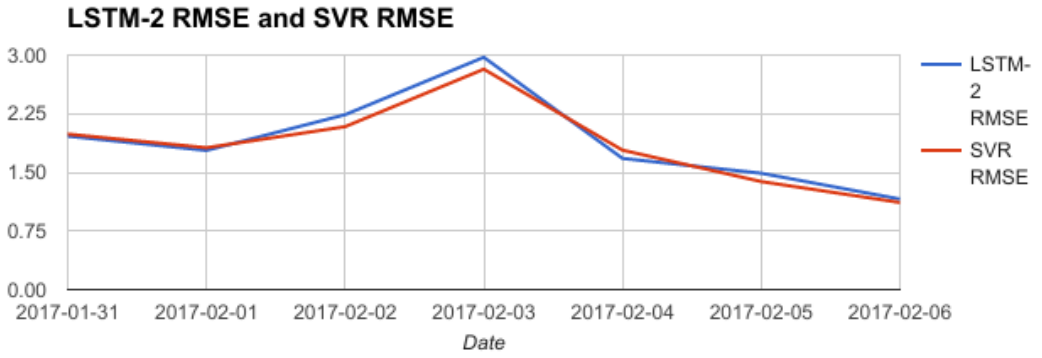

Figure A.7: The average 60 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #1.

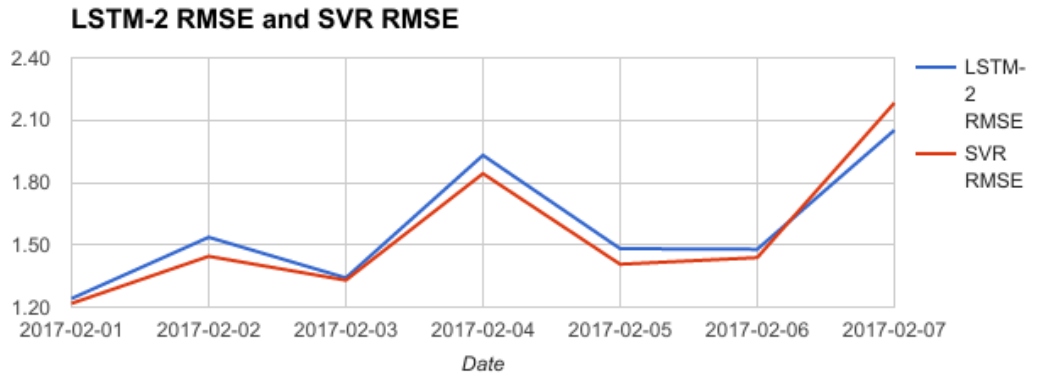

Figure A.8: The average 60 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #2.

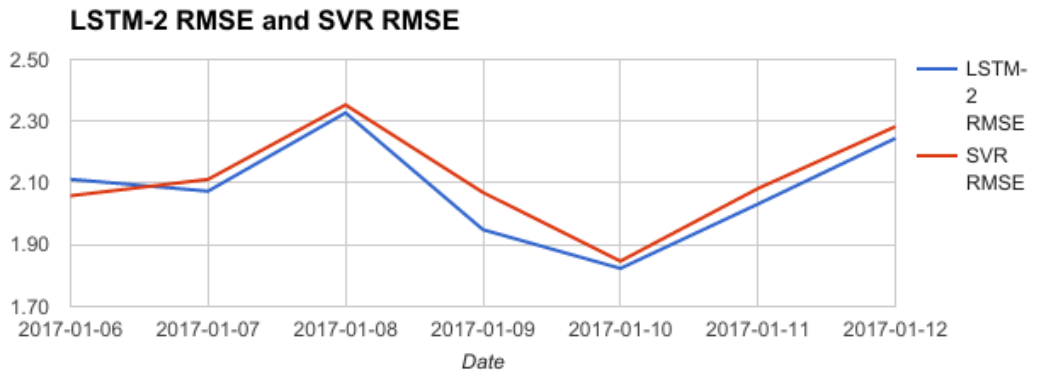

Figure A.9: The average 60 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #3.

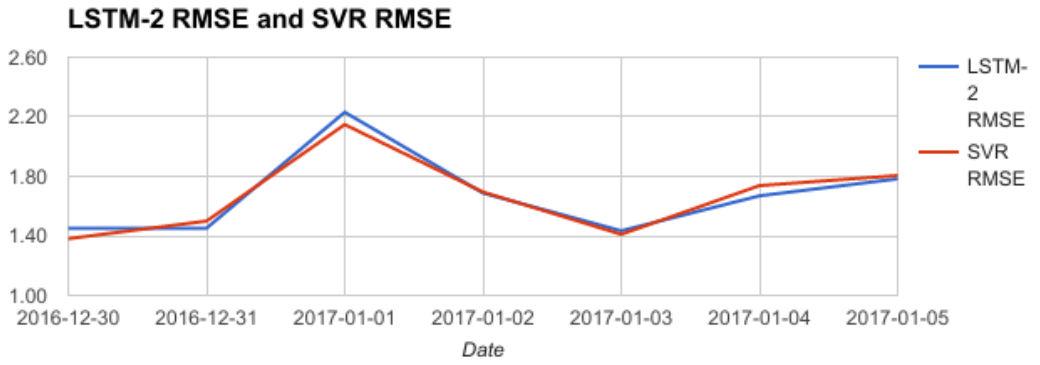

Figure A.10: The average 40 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #4.

A.4.3 90 minute prediction horizon

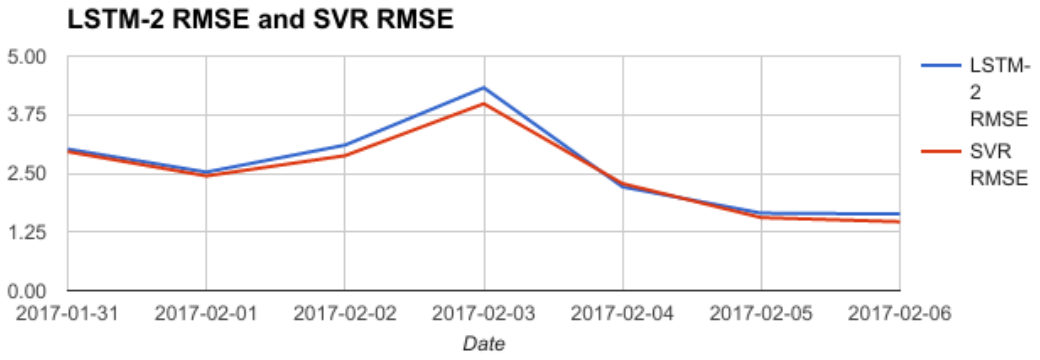

Figure A.11: The average 90 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #1.

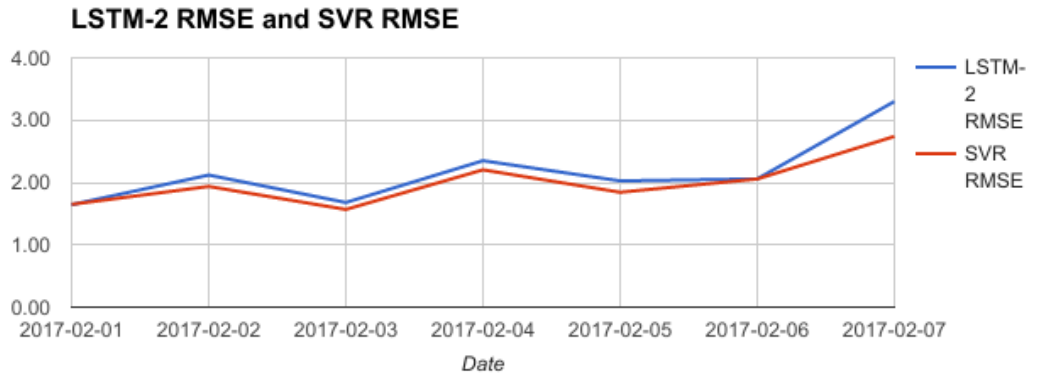

Figure A.12: The average 90 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #2.

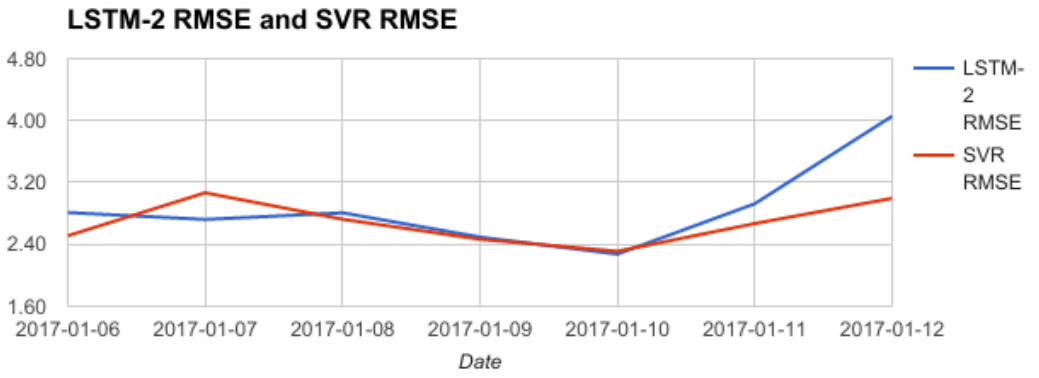

Figure A.13: The average 90 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #3.

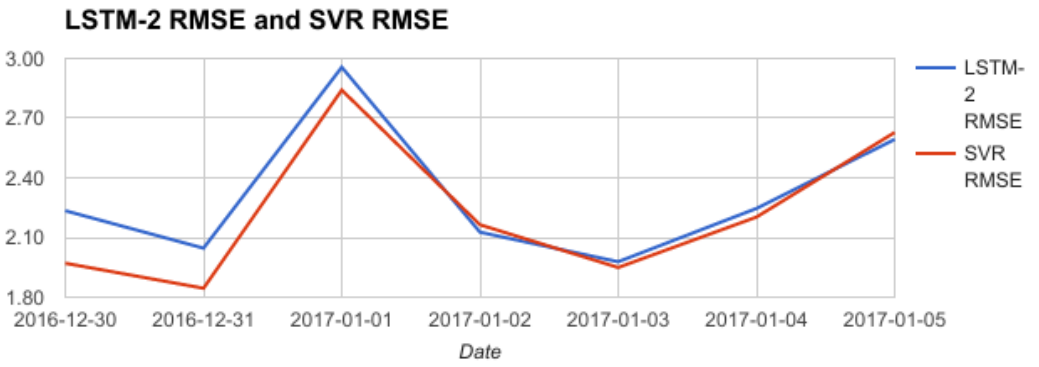

Figure A.14: The average 90 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #4.

A.4.4 120 minute prediction horizon

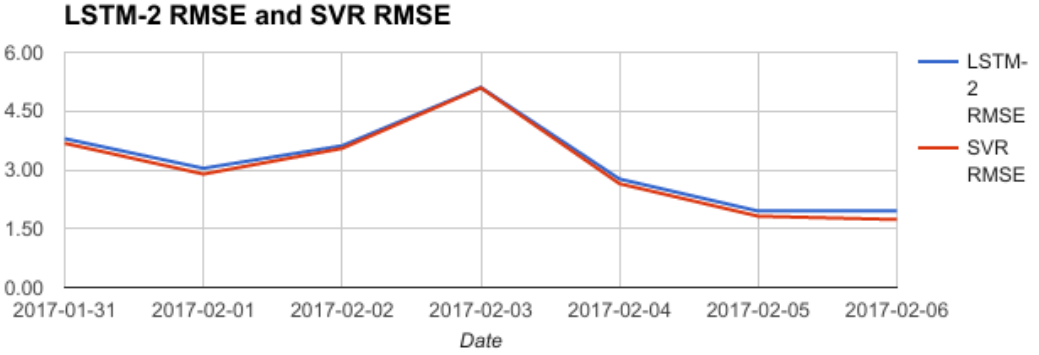

Figure A.15: The average 120 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #1.

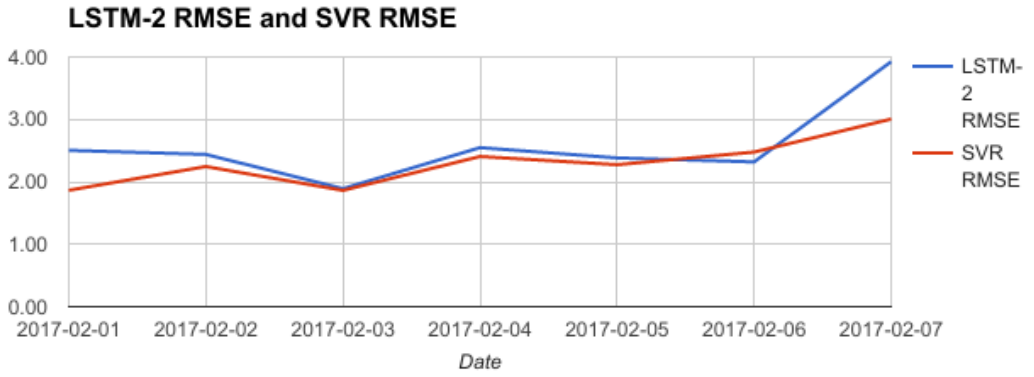

Figure A.16: The average 120 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #2.

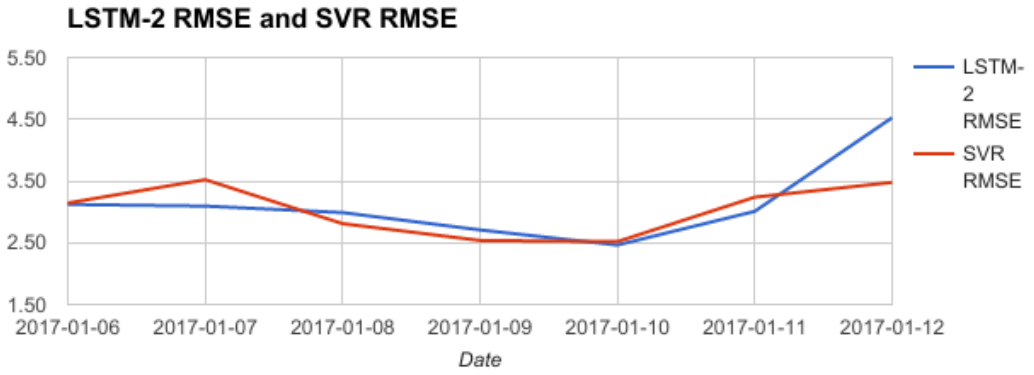

Figure A.17: The average 120 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #3.

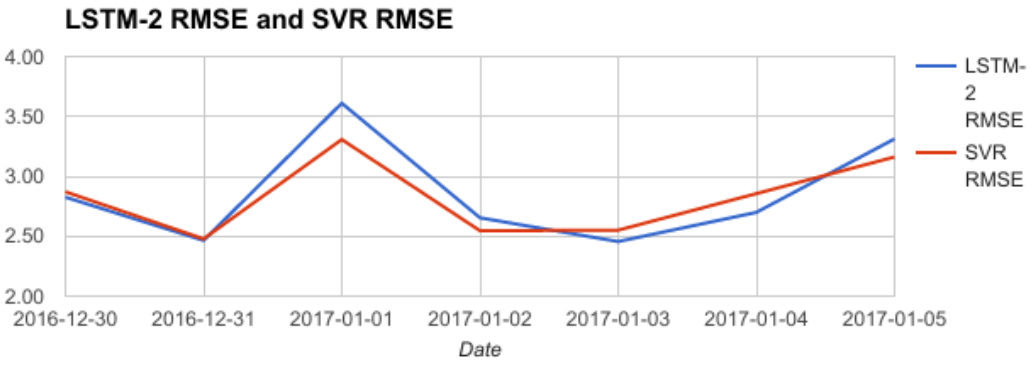

Figure A.18: The average 120 minute daily prediction error for SVR (red) versus LSTM-2 (blue) over all test days for patient patient #4.