Deep learning for prediction of hospitalization of dialysis patients

Niklas Hansson

Kongens Lyngby 2017

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

In this thesis the author have applied deep learning techniques in order to predict the hospitalization of dialysis patients. The goal was to predict if a patient will be hospitalized within the next 30 days. The data set consists of several records where the main data used has been collected form a multivariate time series based upon measurements during the dialysis session. Dialysis is a vital treatment for patients with reduced kidney function and today over 670 000 patients in the US need dialysis treatment to survive. Dialysis is usually received three times a week during a 3-4 hour session where the blood is cleaned outside the body. The patients are frequently hospitalized, resulting in suffering and large costs. Predicting the hospitalizations, preventive actions can be taken, suffering reduced and resources used more efficiently. As reference models a Random forest model and a k-Nearest neighbor model were implemented. In terms of neural networks, three different types of models were built and evaluated, a feedforward neural network; a recurrent neural network and a convolutional neural network. The thesis was conducted in cooperation with Lytics Health AB, in Malmö, Sweden. Deep learning was successfully implemented, and for the neural networks the best performance in terms of area under the receiver operational characteristic curve was achieved with a feedforward neural network, with AUC of 0.689. The best performance was achieved by the Random forest model with an area under the receiver operation characteristics curve of 0.692.

Keywords: Deep learning, Machine learning, Dialysis

LYTICS

Preface

This thesis was prepared at DTU Compute in fulfillment of the requirements for acquiring an M.Sc. in Electrical engineering. The thesis was conducted in collaboration with Lytics Health AB, in Malmö, Sweden. The thesis have been supervised by Prof. Ole Winther at DTU COMPUTE, Institut for Matematik og Computer Science and Dr. Fredrik Ståhl at Lytics Health AB, Malmö, Sweden.

Lyngby, 23-June-2017

Niklas Hansson

Acknowledgements

I would like to thank my supervisors Prof. Ole Winther at DTU COMPUTE and Dr. Fredrik Ståhl at Lytics for their help and feedback. I would also like to thank all the people working at Lytics for the opportunity and the warm welcome. A special thanks to Erik Gillsbro and Mattias Albinsson for all the lunch break walks and feedback. I would also like to thank Filip Thor helping with proof reading and the editorial feedback. Last but not least, I would also like to thank my family and friends for their support.

Contents

Summary **i**

Preface **iii**

Acknowledgements **v**

1 Introduction **1**
1.1 Problem 2
1.2 Outline 3

2 Theory **5**
2.1 Renal failure 5
2.2 Dialysis 6
2.3 Accuracy metrics 7
2.4 Reference methods 10
2.4.1 Random forest 10
2.4.2 k-Nearest Neighbors 10
2.5 Deep learning 10
2.5.1 Neural networks 11
2.5.2 Backpopagation 13
2.5.3 Optimization techniques 14
2.5.4 Gradient descent variants 14
2.5.5 Convolutional neural networks 18
2.5.6 Recurrent neural networks 22
2.6 Regularisation 32
2.6.1 Dropout 34
2.6.2 Batch-normalisation 35
2.6.3 Early stopping 35
2.6.4 L^2 Parameter Regularization 37

2.7 Padding 37
2.8 Hyper parameter optimisation 37
2.8.1 Hyperopt 40
2.9 Related work 40

3 Data **43**
3.1 Data 43
3.2 Pre-processing and standardization 51

4 Methods **53**
4.1 Pipeline, feature extraction and feature generation 53
4.2 Approach 57
4.3 Developed models 59
4.3.1 Random forest 59
4.3.2 kNN 60
4.3.3 Feedforward neural network 60
4.3.4 Convolutional neural network 61
4.3.5 RNN models 64

5 Results **65**

6 Discussion **73**
6.1 Future work 76

7 Conclusion **79**

Appendices **81**

A Rough estimation of significance **83**

B Receiver operation characteristics curves **85**

C RNN model without early stopping **87**

D Results for alternative feedforward model **89**

E Feature importance for Random forest **91**

Bibliography **103**

Chapter 1

Introduction

In health care large amount of data is collected but its usage is not always optimal. Utilizing technologies as machine learning and artificial intelligence could make improvements in the use of data for diagnosis accuracy, tailor-make treatments and in the long run replace work performed by clinicians [ESM17]. The development of intelligent systems within health care is related to the development of electronic health records (EHR) and the digitization of the health care. The development of EHR has increased nine times between 2008 and 2015 in the US. By digitizing the health records the expectations is to achieve safer, more intelligent, efficient, and cost efficient health care solutions [JH17]. Machine learning and artificial intelligence have made promising steps showing that it could be an key component in this development. The health care analyst, IDC predict that by 2018, 30 % of the health care providers will use cognitive analytics on health care data [ESM17].

Deep learning has been introduced with great success delivering state of the art performance in several fields like machine translation [BCB14] and speech recognition [HCC+14]. Recently, machine learning have been used to predict heart attacks with great success — beating the doctors at the task [SFW17]. In their work predicting heart attacks, F.Wang et al [SFW17] showed that using machine learning and neural networks they could predict hundreds of heart attacks that doctors miss while also rising less false alarms.

In this thesis, machine learning and neural networks will be used for predicting the hospitalization of dialysis patients. Dialysis is a medical treatment for patients with reduced kidney functions where the blood is cleaned. Hospitalization is frequent among dialysis patients, and reducing the number of hospitalization would result in both increased life quality and reduced cost [Jaa17].

1.1 Problem

For patients with reduced or malfunctioning kidneys, dialysis is a vital treatment where the cleaning of the blood is done artificially outside the body. Dialysis is an expensive medical treatment where a lot of the patients continuously have to be hospitalized due to worsening conditions and sequelae. These unscheduled hospitalization is very expansive and if prevented, the life quality of the patients could be improved, and the cost of the health care could be reduced. An average hospitalization for hemodialysis patients is $\sim$ 2700$ per day according to the Centers for dialysis Care in the US. In 2013, 117 162 new cases of end stage renal disease (end stage renal disease is defined as when dialysis is needed for survival) were reported in the US [S0216]. The number of patients that received treatment for end stage renal disease in the US was in the end of 2014, 678 33, a number that is increasing due to falling mortality rates [S0216]. In this thesis deep learning and neural networks will be used in order to predict if a patient will be hospitalized within 30 days. The thesis is conducted together with Lytics. Lytics is a start up focusing on artificial intelligence and machine learning for health care. They have been working with the data and the problem over a longer period of time. However previous investigation have not involved sequential methods or neural network which is the main goal of this thesis to investigate. In theory sequential methods should suit the problem very well since the data is a multivariate time series and the health state of the patients are expected to progress over time. The data that will be used is supplied by the Centers for dialysis Care in the US which is working together with Lytics. Previous work within health care have also found success using recurrent neural networks [LKEW15], [AM16], [CBS15]. The goal is to predict if a patient will be hospitalised within 30 days or not and is thus a classification problem. 30 days have been selected as the period in previous discussion between the doctors at the dialysis clinics and Lytics, based upon the time needed to be able to make additional investigation and take preventive measures.

1.2 Outline

The outline of the report is the following:

- The first chapter introduces the problem and why it is important.
- The second chapter includes the theory and background. Chapter two also includes a background to renal failure and dialysis.
- The third chapter introduces the data and presents some basic statistics of the data.
- The fourth chapter describes the developed models and the data pipeline.
- The fifth chapter presents the results.
- The sixth chapter includes the discussion.
- The seventh chapter includes the conclusion of the master thesis.

Chapter 2

Theory

In this chapter the background to the investigated methods, metrics, renal failure and dialysis will be presented. Related work that the author considered relevant for the work will also be briefly presented.

2.1 Renal failure

Renal failure, kidney failure or kidney insufficiency are medical conditions due to malfunction of the kidneys, resulting in insufficient filtering of the blood. This in turn results in toxic and harmful substances stay in the blood. Patients suffering from reduced kidney functions usually also collect large amount of water in the blood resulting in that the patients gets swollen with an elevated blood pressure and are in general exhausted. There are several reasons for renal failure. Some of the most common are diabetes and high blood pressure [Natb]. End-stage kidney disease is defined as when dialysis is needed for survival due to that the remaining kidney function is so severely reduced. The earlier the kidney problems are detected, the better can they be handled —resulting in slowing down or stopping the progression [Nata].

2.2 Dialysis

Dialysis is a medical process that removes waste products and excess water from the blood[Natb]. Dialysis treatments is essential for patients that have reduced or completely failed kidney functions. Since the kidneys are responsible for the cleaning of the blood and maintain the equilibrium of water and minerals in the body this process has to be done artificially when the kidney function is reduced by 85 to 90 %. [Natb]. Dialysis is used to control the amount of salt and preventing water from building up in the body. Dialysis also helps with control the blood pressure.

There are different causes to kidney failure where some conditions can improve over time making dialysis unnecessary after recovery. For chronic or end-stage renal failure dialysis treatment is needed for the rest of the patients life with the exception of transplantation where the kidney is replaced. Two types of dialysis is available, hemodialysis and peritoneal dialysis.

Hemodialysis is a type of dialysis where the blood is cleaned outside the body in a specially designed dialysis machine. In order to receive this treatment, a small channel is inserted through surgery into a blood vessel which allows the patient to be connected to the dialysis machine and the blood to be cleaned. Hemodialysis session usually lasts for 4 hours with a frequency of 3 sessions per week. It is possible to do Hemodialysis at home but then the frequency of treatments is usually higher[Natb]. The treatment schedule and amount of dialysis is set by doctors. Getting the correct amount of dialysis is critical and studies have shown that it affects hospitalization and life expectancy[Natb]. Lab test are used to set the correct amount of dialysis which is usually done with a frequency of once a month. According to The national Kidney Foundation, two of the key metrics are:

- Kt/v (which should be below 1.2) K is the average blood flow, t is the dialysis time and v is the fluid removed.
- URR (which should be at least 65). URR is urea reduction ration.

Since the kidneys play such a critical role in the human body the loss of kidney function also affect the diet. The time needed for dialysis is affected by:

- The level of kidney function
- How much fluid the patient gains between the sessions

- The size of the patient, this affect the amount of blood
- The type of hemodialysis

The alternative to hemodialysis is peritoneal dialysis. Peritoneal dialysis uses a cleansing tube that is operated in to part of the abdomen and filterers the waste products from the blood. The data set only contains patients that receives hemodialysis.

The average life expectancy for patients receiving dialysis is $5 - 10$ years. However, some patients have lived for 20 to 30 years[Natb]. Depending on the type of dialysis, the price per patient and year is up to $ 72.000 for hemodialysis and $ 53.000 for peritoneal dialysis according to the U.S Renal Data System. The average cost per day of hospitalization is $ 2700 according to the Centers for dialysis Care in US. For reference it should be noted that in the U.S there are more than 670,000 dialysis patients.

2.3 Accuracy metrics

In this section metrics for evaluation will be presented. It should be noted that the data set is imbalanced with $\approx$ 15% positive labels and $\approx$ 85% negative labels. This affects the metrics suitable to measure performance. The most common metrics for machine learning is accuracy, which is defined as:

$$Accuracy = \frac{\text{Correct classified observations}}{\text{Total number of observations}} = \frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Observation}} \tag{2.1}$$

The possible types of correct and wrong classification for a binary classifiers are illustrated by the confusion matrix, see Figure 2.1. True positive is the sequence that where classified as positive which also is the true label. True negative are samples that are classified as negative which also holds as true. False positive are samples that are classified as positive but are in reality false. False negative are the opposite of false positive and are the samples that are incorrectly classified false.

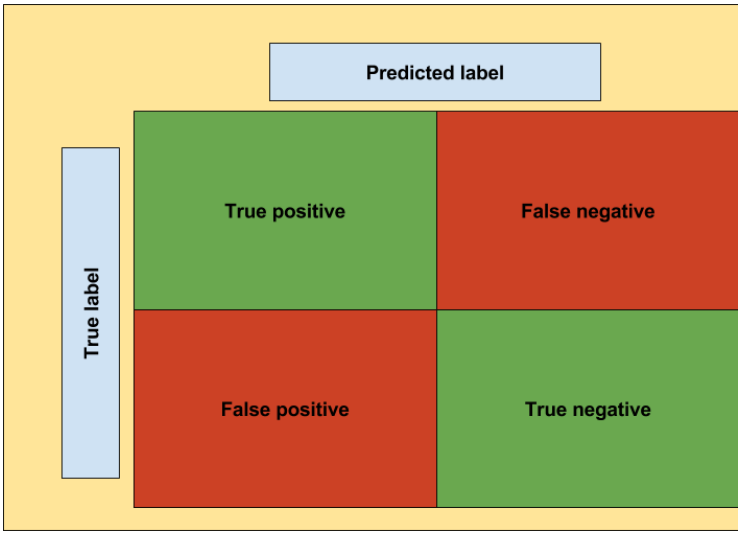


Figure 2.1: Illustration of the confusion matrix.

Accuracy doesn't take into account the misclassifications or imbalances in the data set. An alternative metric is the receiver operation characteristics(ROC). ROC measures the relation between the true positive rate and the false positive rate for varying thresholds. By looking at the distribution of the positive and negative labels for different thresholds, i.e probabilities, the ROC can be drawn, see illustration in Figure 2.2. Figure 2.3 show the probability density for three binary classifiers. The figures should illustrate how well separated two classes are based up on the probability distribution. A superior model results in a better separation of the positive and negative labels.

The area under the ROC curve is often called the area under the curve. A perfect classifier have an area of 1, randomly guessing gives and area of 0.5. The area under the ROC curve will be the primary metric for evaluation of different methods and comparing results. This is due to that it is a metric well suited for imbalanced data sets and that it is the primarily used metric for benchmark at Lytics. The area under the receiver operation characteristic curve will be abbreviated as AUC.

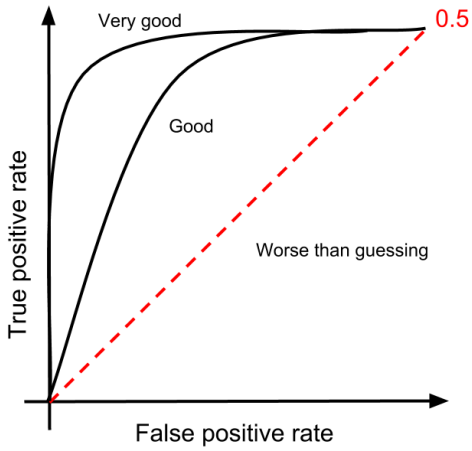


Figure 2.2: Illustration of a receiver operation characteristics curve. The dotted line indicates when the area under the curve is 0.5 and thus the model performs as good as guessing.

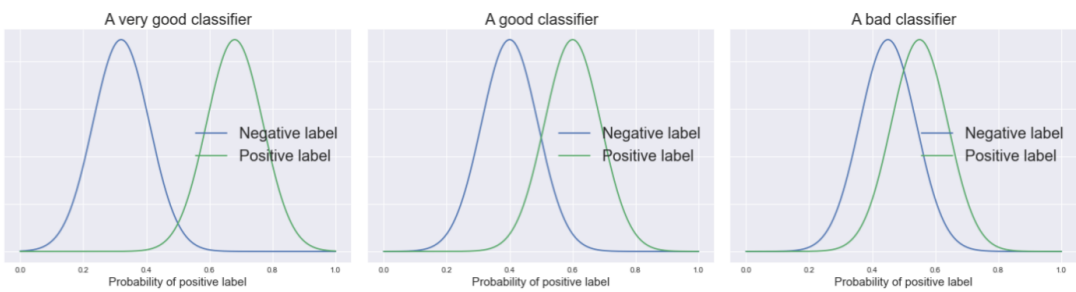


Figure 2.3: The figure should illustrate the distribution of the data for three classifiers. The distribution is plotted against the probability for a positive label. Calculating the true positive and false positive for different thresholds, the ROC curve can be draw. The better the separation of the distribution of the positive and negative labels, the better is the classifier.

2.4 Reference methods

In order to compare the methods based upon neural networks, two baseline methods from the machine learning library scikit learn in Python will be used. These two methods, Random forest and k-Nearest Neighbors will here briefly be described.

2.4.1 Random forest

Random forest is a type of bagging method which means that it averages many noisy but almost unbiased models and by doing so reduces the variance in the predictions. This is done using many models of a type known as Decision trees, hence the name Random forest. Decision trees is an algorithm which uses simple but efficient splits to partition the data in to subsets. The aim is to find split that have the ability to classify the data according to a set problem. The interested reader are referred to *The Elements of Statistical Learning* by Hastie et al [TH16] for further reading about Decision trees and Random forest. Random forest have proven to give very good results with very little tuning according to Hastie et al [TH16].

2.4.2 k-Nearest Neighbors

k-Nearest Neighbors methods uses the observations in the training closest to an observed input, x to from a prediction $\hat{y}$. k-Nearest Neighbor methods are based upon the assumption that input observations have the most in common with training observation which they are closest to in space. In terms of classification the input observation is given the class label of the closest training target or of a weighted sum of a group of the k closest neighbors. Closeness, the way the distance is measured is done based on Euclidean distance. The interested reader are referred to *The Elements of Statistical Learning* by Hastie et al [TH16] for further reading about k-Nearest Neighbor methods.

2.5 Deep learning

In this section deep learning and the different techniques used in this thesis will be discussed and explained. Deep learning is a subcategory of machine learning

focusing on neural networks. As indicated by the name, the models are "deep"; this is achieved by stacking units on top of each other to be able to model more complex and abstract relations. Deep learning uses large amounts of non-linear processing units to model complex patterns in data. In its simplest from, a processing unit in an neural network performs the following: the input data is multiplied with a weight, then a bias is added and finally a transfer function is applied before the data is passed on, see Figure 2.4 (here *tanh* is chosen as transfer function for the examples but it is just one of several transfer functions that will later be described).

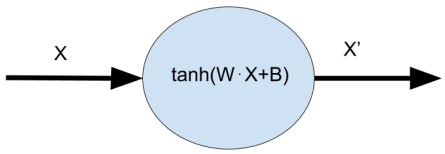


Figure 2.4: Illustration of a unit in a neural network. X is the input and X' is the output of the unit

2.5.1 Neural networks

As described earlier, neural networks combine non-linear equations in order to model data. The smallest units is often known as "units" or neurons and by combining several a layer is built, several layers together build a network. In Figure 2.5 a neural network is presented, the basic structure of the networks can be described by the three layers. The input layers fed the information in to the network. The information is fed using a simple linear equation $\mathbf{W} \cdot x + \mathbf{B}$. The input is feed to all neurons in the next layer, known as a hidden layer. The layers between the input and output layers are known as hidden layers. The input to each of the hidden units is a sum of the output from the previous layer to the corresponding hidden unit $z_j^L = \sum_k w_{jk}^L x_k^{L-1} + b_j^L$, where k is a unit in the previous layer, j is the unit in the hidden layer and L is the layer. As an example the input to h_{11} in Figure 2.5 is equal to $w_{11} \cdot x_1 + b_{11} + w_{12} \cdot x_2 + b_{12} + w_{13} \cdot x_3 + b_{13} + w_{14} \cdot x_4 + b_{14}$. The hidden layer transforms the input using a transfer function, this transfer function also allows the network to model non linear behaviours.

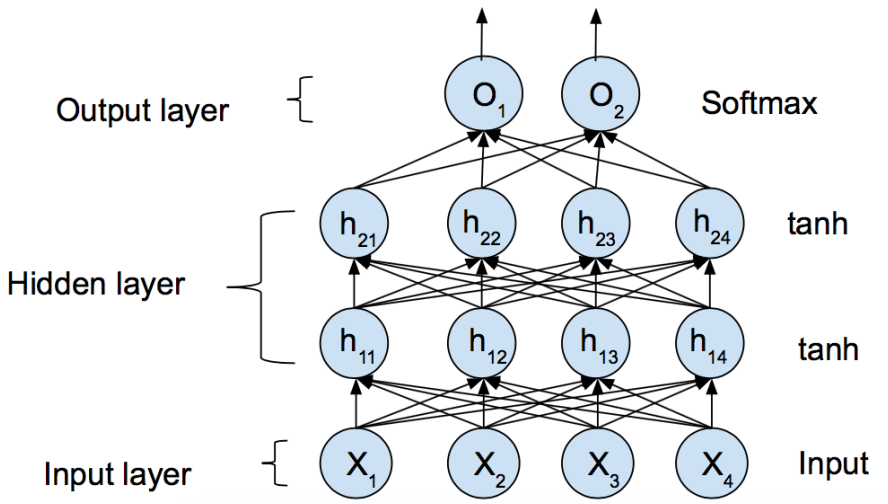


Figure 2.5: Illustration of a neural network

The commonly used transfer functions are:

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{2.2}$$

$$sigmoid(a) = \frac{1}{1+e^{-x}} \tag{2.3}$$

$$Relu, = \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases} \tag{2.4}$$

For the output layer, a transfer function known as softmax function is often used. The output from the softmax is a probability distribution. The softmax layer looks at the other units in the layer in order to calculate the output and thus separates itself from the other transfer functions. The first step is to summarize the inputs to the units as for all units the in network, $o_j = \sum_k w_{jk}^L x_k^{L-1} + b_j^L$, then the output at unit j, is equal to:

$$O_i = \frac{e^{o_i}}{\sum_{n=1}^{2} e^{o_n}} \tag{2.5}$$

Neural network are trained to minimize a cost function and thus to find the best values for the weights and biases. The cost function is set based on how good the model performs and then the error is backpropagated through the network

in order to adjust the parameters. This is described more thoroughly in section 2.5.2.

2.5.2 Backpopagation

The goal with neural networks is to, based upon a limited number of observations, find a model that generalizes well to all data for a given problem. The designer of the models sets the framework for the model by choosing the number of units, type of units, *etc*, these parameters are known as hyperparamters. The model is then fed data during which the model is trained to find the best possible weights based upon a loss function. This is achieved using the back propagation algorithm, where data is fed through the model, the error calculated and then the weights are updates in order to minimize the loss. The backpropagation algorithm can be explained by the following five steps[Nie17a]:

- **Input step**: The input is feed to the network, meaning that the values of the neurons in the input layer are put to the input vales.
- **Feedforward**: The activations, input values are feed through the network. For layers $l = 2, 3, ..., L$ the output is calculated based on $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$ where σ should represent the sigmoid function.
- **Calculate the error in the output layer**: The error is calculated based on the $\delta^L = \nabla_a C \odot \sigma'(z^L)$
- **Backpropagate the error through the network** The error at each layers is calculated, $l = L-1, L-2, ..., 2$, $\delta^l = ((w^{l+1})^T \delta^{l+1} \odot \sigma'(z^l))$
- **Calculate the gradient based on the error** The gradient step based on the cost is $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \sigma^l_j$

The error is thus propagated backwards through the network and the weights are updated. For this work the cross entropy is used as loss function:

$$C(w, b) = -\sum p(x) \log q(x) \tag{2.6}$$

Where p(x) is the true probability and q is the predicted probability. The cross entropy cost function is implemented in order to avoid the slow down of the learning which can be a problem when the root mean square error is used.

While training neural networks the performance is usually continuously evaluated during the development process. However evaluating after each backpropagation step is computationally costly. Therefore the performance is usually evaluated after the whole training data set have been feed through the network, also known a one epoch. The data set is feed through the network several epoch depending on the nature of the data and the network. The number of epochs to training the neural network for can be viewed as a hyperparameter among others.

2.5.3 Optimization techniques

Optimization within machine learning and deep learning is different from traditional optimization in terms of that the optimization is only acting indirectly on the performance measurement, the generalization error. For deep learning the optimization is based on the cost of the training data but the true performance measure(on the validations data) is expected to increase as the cost function is minimized[GBC16]. This is different from traditional optimisation where the goal is to minimize the cost function itself[GBC16].

2.5.4 Gradient descent variants

Gradient descent is the standard algorithm for training neural networks[Nie]. Based on the gradient of the cost the biases and weights are updated. This update can be done in different ways. There are three types of gradient descent with the difference between them in the amount of data used. Before the difference between them is described gradient descent in general will be described.

The first step of a learning algorithm is to use some from of coast function also known as loss function. An example of a cost function is:

$$C(w,b) = \frac{1}{2n} \sum_{i=0}^{i=n} \mid x_i - y_i \mid^2 \tag{2.7}$$

This is known as the mean square error loss function. The goal during the learning phase is to minimize the cost function by finding weights and biases such that the mean square error is minimized. Gradient descent is based on the concept of calculating the gradient and then "move in the opposite direction". Often an analogy with finding the lowest location in a valley is used where the

lowest point is the minimum of the cost function. Minimizing the function is done by taking steps in the opposite direction of the gradient and thus moving towards the minimum. The step length in the analogy is known as learning rate in machine learning. Assuming that the cost function C is depending on n variables, $v_1, v_2, v_3, ..., v_n$ the change in the cost function based on a small step $= (\Delta v_1, \Delta v_2, \Delta v_3, ...\Delta v_m)^2$ is equal to:

$$\Delta C \approx \nabla C \cdot \Delta v, \tag{2.8}$$

where ∇C is the partial derivative of C. The gradients that is calculated through the cost function is:

$$\nabla C \equiv (\frac{\partial C}{\partial v_1}, ..., \frac{\partial C}{\partial v_m})^T \tag{2.9}$$

In order to calculate the change in the variables can thus be done according to

$$\Delta v = -\eta \nabla C \tag{2.10}$$

in order to minimize the cost function we should move in the opposite direction of the gradient and thus the negative sign. η set the "length" of the step and is commonly known as learning rate. Using these steps the weights and biases are updated in order to move towards a minimum.

$$v' = v - \eta \nabla C. \tag{2.11}$$

In the following sections 2.5.4.1, 2.5.4.2 and 2.5.4.3 three variants of gradient descent are described. They offer a trade-off between accuracy and the computational burden.

2.5.4.1 Batch gradient descent

Batch gradient descent uses the complete training set in every update of the weights and parameters, $v' = v - \eta \nabla C(\Theta)$. This gives a high accuracy but makes the updates slow since the computational burden is high. This also limits the model in terms of updating the model on the go, also known as online [Rud].

2.5.4.2 Stochastic gradient descent

Stochastic gradient descent is the opposite of batch gradient descent where the parameters are update for each training sample, $v' = v - n\nabla C(\Theta, x^{(l)}, y^{(l)})$. Stochastic gradient descent have the advantage that it can be used online. The downside is that the updates have a high variance and that this causes the cost function to fluctuate.

2.5.4.3 Mini-batch gradient descent

Mini-batch gradient descent tries to combine the advantages of batch learning with stochastic gradient descent. This is done by using mini batches with a set of, n training examples, $v' = v - \eta\nabla C(\Theta, x^{l:l+n}, y^{l:l+n})$. This limits the variance in the cost function. Many deep learning libraries also have implementations that allow mini-batches optimisation very efficient [Rud].

2.5.4.4 Momentum

Stochastic gradient descent have problems when the surface is significantly steeper in one direction [Rud16]. Stochastic gradient descent have problems due to oscillations along the steeper slope when finding the minimum, this is illustrated in figures 2.6 and 2.7.

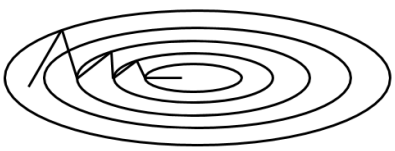


Figure 2.6
Illustration of Stochastic gradient descent with momentum. The figure should illustrate the minimization for a simple example.

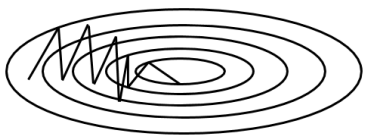


Figure 2.7
Illustration of Stochastic gradient descent and the minimization process. The figure should illustrate the minimization for a simple example.

To reduce this problem a technique known as momentum can be implemented, which add a fraction of the update vector from the past step to the current update.

$$\Delta v = \gamma \Delta v_{t-1} + \eta \nabla C \tag{2.12}$$
$$v' = v - \Delta v \tag{2.13}$$

The fraction of the previous update is determined by the gamma. Adding the fraction of the previous state gives the updating function a momentum. An analogy with a ball rolling down a hill picking up speed can be made. This reduces oscillation and a faster convergence can be gained [Rud].

2.5.4.5 Adaptive moment estimation optimization

Adaptive moment optimization (Adam) computes individual adaptive learning rates for different parameters from estimates of the first and second derivatives of the gradients [KB14]. Adam keeps individual learning rates for each weights and have a exponentially decay average of previous gradients. The algorithm for Adam optimization is describe in Algorithm 1.

Algorithm 1 Adam optimization as defined by Kingman et al. [KB14] and implemented in tensorflow. g_t^2 indicates the elementwise squared $g_t \odot g_t$. Default setting in tensorflow and recommended by Kingman et al [IS15] is $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta = 0.999$ and $\epsilon = 10^{-8}$. All the vectors are elementwise. β_1^t and β_2^t β_1 denotes β_2 to the power of t

Require: α: Step size (also known as learning rate)
Require: β_1,$\beta_2 \in$ [0,1]: Exponential decay rate for the moment estimates
Require: $f(\Theta)$: Stochastic objective function with parameters Θ
Require: Θ_0: Initial parameter vector
$m_0 \leftarrow 0$(Initialize the first moment vector)
$v_0 \leftarrow 0$(Initialize the second moment vector)
$t \leftarrow 0$(Initialize time step)
while Θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\Theta f_t(\Theta_{t-1})$(Get gradients with regards t stochastic objective function at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
    $\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$(Compute bias-corrected first moment estimate)
    $\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$(Compute bias corrected second raw moment estimate)
    $\Theta_t \leftarrow \Theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return

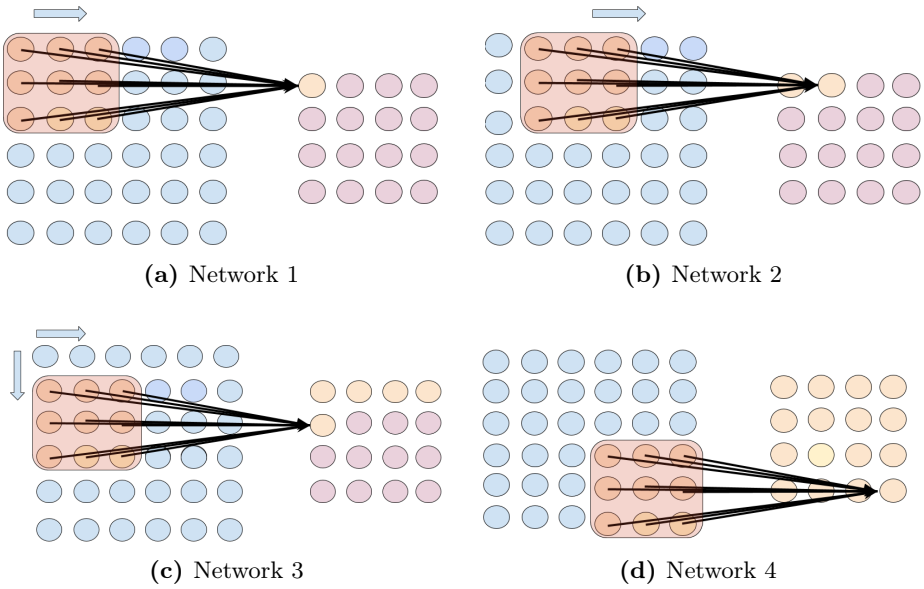
2.5.5 Convolutional neural networks

When the input contains spatial information, where the location of the data observation in relation to other observations carry information convolutional neural networks have proven successful. Feedforward neural networks treat every observation in the same way no matter how they are related in there input space. Convolutional neural networks look at the spatial information of the observed data and extract features from these data [Nie17b]. Convolutional neural network uses three different structural layers as basic building blocks, convolutional layers, pooling layers and flattening layers [Nie17b]. The output after the flattening layer is feed to a feedforward neural network. The different layers will be described in the following section. Convolutional layers use filters to extract structural information. The convolutional filters process the input in local areas and slide over the input, see Figure 2.8. The length of the steps of the sliding movement is known as stride length. The output from the filters are extracted to the hidden layers next in line in the model. The filter size are set depending upon the structure of the data and how the data are spatially related. An example using a three-by-three filter for a one channel input the output from a filter is :

$$\text{Hidden neuron} = f(\sum_{n=0}^{3}\sum_{m=0}^{3} w_{n,m}a_{(j+n),(k+m)} + b) \tag{2.14}$$

The function f is a transfer function, for example the sigmoid function. a is the input from the previous layer where j and k should represent the index of the filter location in the input layer. w is the weights of the filter and are shared for the filter. That the weights are shared result in that each filter will look for the same trend at different location in the input data. Thus several filters are needed to extract several features. The shared weights play an important role in reducing the computational burden of the model. For a convolutional layer several different filter are often used. It should be noted that each filter is sliding over all the channels simultaneously when filtering the input data, see Figures 2.9. In combination with convolutional layers pooling layers are often used, these are used to reduce the dimensionality by condensing the input. Reducing the dimensions also reduce the computational burden and also control overfitting.

There are a range of different types of pooling layers were some of the most used are max pooling, Average pooling and L2 pooling. Pooling layers are similar to convolutional filter where the filter slide over a local region extracting information to the next hidden layer. Max pooling takes the maximum input from the previous layer as output, see Figure 2.10. Average pooling look at



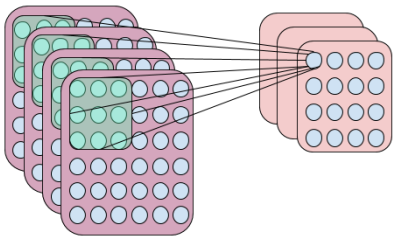
(a) Network 1 **(b)** Network 2

(c) Network 3 **(d)** Network 4

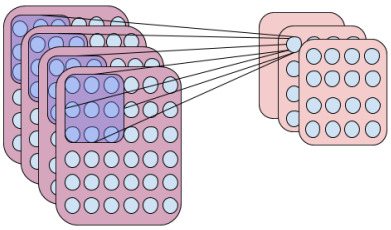
Figure 2.8: Illustration of a convolutional filter and how it slides over the observations. The filter should illustrate a $3x3$ filter with a stride of one.

the average value and outputs the average as the output. L2 pooling takes the square root of the activations in the filter region. Pooling is done for each layer individually.

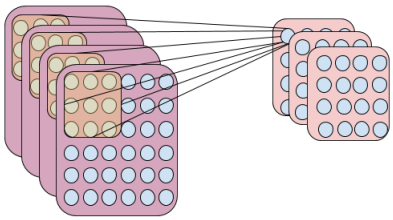
The last type of layer in a convolutional neural network is the flattening layer which flattens the matrices to a one-dimensional array. This is done in order to allow the information to be passed to a feedforward neural network. With this set up the convolutional neural network extract features from the spatial information in the data and then passes these features to a feedforward network. In similarly with feedforward neural networks the weights and biases are trained by back propagating the error. An example of a simple structure for a convolutional neural network can bee seen in Figure 2.11. For the set up in Figure 2.11 it can be noticed that the flattening layer flattens on all dimensions only keeping data separated by batch. Thus the spatial information is lost.



(a) Filter 1 out of 3, for a filter size of $3x3$



(b) Filter 2 out of 3, for a filter size of $3x3$



(c) Filter 3 out of 3, for a filter size of $3x3$

Figure 2.9: Illustration of how the filters look at all the channels when prediction the output to the next layers.

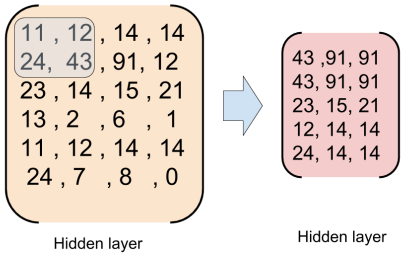


Figure 2.10: Illustration of max pooling. The pooling filter has a size of 2x2, the input dimensions is of size 6x4 and the output dimensions are 5x3

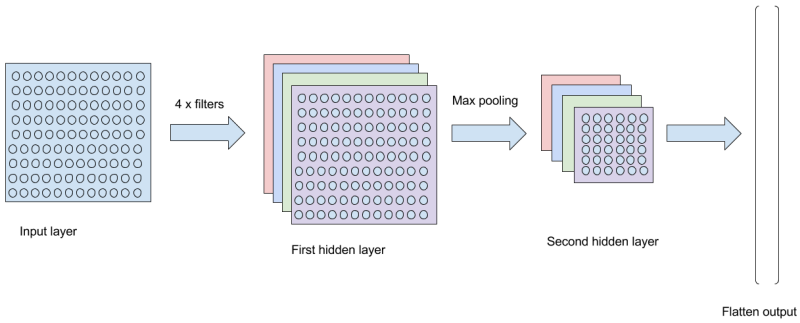


Figure 2.11: Illustration of a simple convolutional neural network. The first layers is a convolutional layer with four filters. Secondly there is a pooling layer reducing the dimensions. Lastly the hidden layer is flatten in order for the extracted features to be feed to a feedforward neural network.

2.5.6 Recurrent neural networks

Recurrent neural networks(RNN) is a type of neural network specialised on working with sequential data i.e $(x^{(1)}, x^{(2)}, ..., x^{(n)})$[GBC16]. Recurrent neural networks relay heavily on weight sharing in order to limit the computational burden. The weight sharing allows the model to be dynamic in terms of length of the sequences. This is of key importance because the weight sharing allows for sequences of different length than those trained on can be used for inference due to the shared weights[GBC16]. To illustrate and demonstrated how recurrent neural networks work an example will in this section be unfolded and described. Assume a dynamic system

$$s^t = f(s^{t-1}; \theta) \tag{2.15}$$

The equation for the system is recurrent since state s at time t depend on that state at $t-1$. With a finit number of steps a networks can be unrolled using the equation for the systems in t-1 steps. Thus unrolling a network having the dynamic system equal to equation 2.15 at $t = 3$ is thus equal to:

$$s^{(3)} = f(s^{(2)}; \theta) = f(f(s^{(1)}; \theta); \theta) \tag{2.16}$$

Completely unfolding the network result in a expression that does not involve recurrence. The same behavior is used in RNN but where a external signal drives the system. $h^{(t)} = f(h^{(t-1)}, x^t; \theta)$ is usually used for RNN[GBC16]. In this case the information about the previous states in the network is compressed into the h^t which is known as the hidden state. In order to make prediction further layers are introduced on top of the hidden state, see Figure 2.12.

Training the network with the goal to predict a future state based on the sequence information the network learns how to compress the sequence into a losy hidden state h. This hidden state is then a summary of the most important information related to make the prediction. To summarise the RNN maps the input sequence $(x^0, x1, ..., x^{t-1}, x^t)$ of a arbitrary length in to a vector of fixed size h. It should be noted that, t don't have to be time steps but any binary step in a sequence.

In Figure 2.12 the unfolded network is presented,with the size of the network is depending on the number of sequence steps. In a similar way as in equation 2.16 the unfolding of recurrence after time step t can be done. Here the unfolding is represent by a function g^t.

$$h^t = g^{(t)}(x^{(t)}, x^{(t-1)}, ..., x^{(0)}) = f(h^{(t-1)}, x^{(t)}; \theta) \tag{2.17}$$

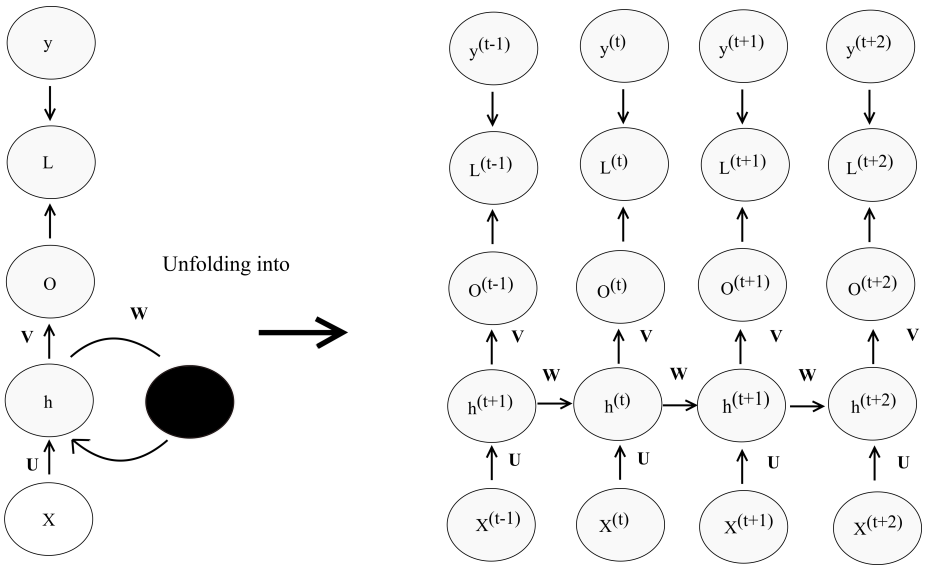


Figure 2.12: Figure to illustrate a Recurrent neural network and how it can be unfolded. The weight sharing is also illustrated in from of the weight matrices U, V and W which are shared in the unit. x is the input, h is the hidden state, o is the output(prediction), L is the loss and y is the true label. Thus the output at each sequence step is depending on the hidden state of the previous step and the input at the current step. The inputs are effected by the weights and the transfer function. For the sequence labeling softmax is used on the output from the hidden state to get the probabilities of the class labels. This figure is a recreation of figure 10.3 in [GBC16], chapter 10.

The input at each sequence, concerning previous sequence steps is constant and is not affected by the length of the sequence. Thus the network can handle sequences of varying length since the same transition functions and parameters are used at every time step. Recurrent neural networks can be designed in several different ways, see Figure 2.13, some of the design patterns are:

- At each time step the network produces an output, the state of the hidden unit is connected to the next sequence step. This is known as many-to many.
- At each time step the network produce an output and the output is forwarded as an input to the next sequence step. When all the input sequence steps are taken the model in order returns the same number of outputs as

input sequence steps. This is known as many-to-many.

- An sequence is read and the information is passed between the hidden units, the entire sequence then produce a single output. This is known as many-to-one.

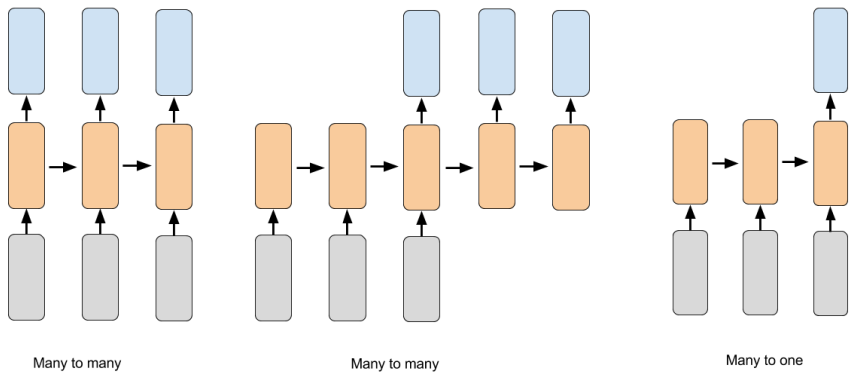


Figure 2.13: Structures of RNN illustrated. The illustration to the left is the more traditional many to many while the second illustration from the right side is a encoder decoder setup. The set up illustrating many to one is furthest to the right. Figure inspired by Andrej Karpathy [Kar15]

An example of an RNN in its folded and unfolded set up is illustrated in Figure 2.12, the figure is a recreation of figure 10.3 in [GBC16], Chapter 10. In Figure 2.12, o should represent the output at each time step. A softmax function can be implemented on the output thus assuming that the output is the unnormalised log probabilities. The loss is represented by L where the predicted sequence label and the true sequence label is compared. The $\mathbf{U}$ weight matrix is used in the connection between input and hidden state, $\mathbf{W}$ in the hidden to hidden connections, $\mathbf{V}$ matrix is used in the transfer from the hidden states to the output.

The following equations should mathematically demonstrate the functions of the RNN in Figure 2.12:

$$\mathbf{a}^{(\mathbf{t})} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \tag{2.18}$$

$$\mathbf{h}^{(\mathbf{t})} = \tanh(\mathbf{a}^{(t)}) \tag{2.19}$$

$$\mathbf{o}^{(\mathbf{t})} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)} \tag{2.20}$$

$$\hat{\mathbf{y}}^{(\mathbf{t})} = \text{softmax}(\mathbf{o}^{(\mathbf{t})}) \tag{2.21}$$

Figure 2.14: $\mathbf{a}^{(\mathbf{t})}$ takes the input to the hidden states and apply the linear transformation on the data. The weights **W** and **U** are trained during the training process. A non linear transformation function is then applied on the state. The output is the calculated by doing a linear transformation of the hidden state $h^{(t)}$ before the softmax function is applied.

In the example demonstrated in Figure 2.12 an input sequence is mapped to an output sequence of equal length. The losses can be calculated based on the miss labeling for all time steps compared to the correct labels. The loss can be evaluated in different ways but for this work the cross entropy loss function will be used to avoid the slow down in the learning. The cross entropy loss function makes the neural network learn faster when the initial weights are far from the optimal values[Nie].

2.5.6.1 Back propagation through time

The back propagation for RNN is different compared to feedforward neural networks due to the weight sharing and folding over time. The gradients at each sequence step is summed up in order to get the gradient of the weights. The gradients are as for forwards neural networks calculated using the derivatives based on the chain rule.

Based on Figure 2.15 and equations 2.21 the error at time t is used as example, E^t, where $E^t = -y_t \cdot \log\hat{y}_t$(the cross entropy loss function).

$$\frac{\partial E^t}{\partial V} = \frac{\partial E^t}{\partial \hat{y}^t}\frac{\partial \hat{y}^t}{\partial V} \tag{2.22}$$

Equation 2.22 illustrate the gradient for the weight **V**. **V** is only dependent on the true target, predicted value and the state $\mathbf{h_t}$. This is the same for the bias **c**. For the weights **W**,**U** and **b** this is different since they depend on the previous hidden state.

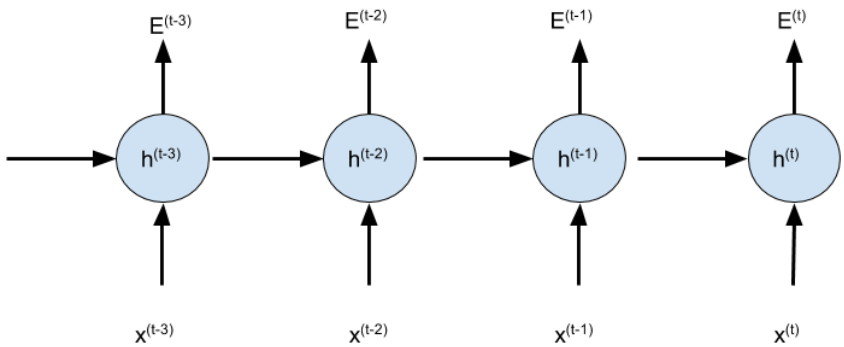


Figure 2.15: Simple illustration of a RNN E is the error at time t, h is the hidden state and x is the output.

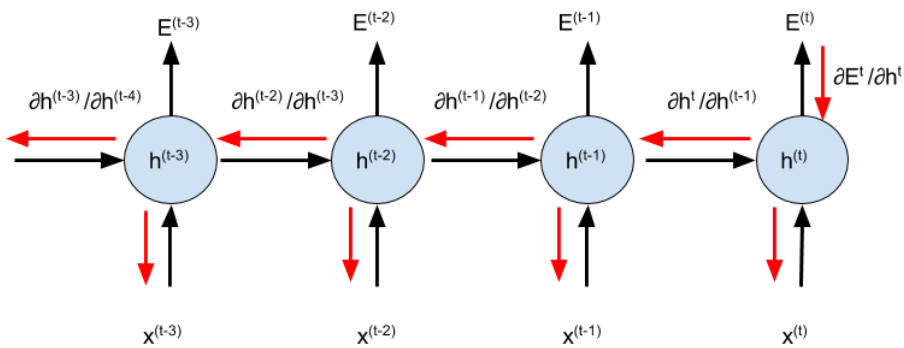


Figure 2.16: Illustration of a RNN, E is the error at time t, h is the hidden state and x is the output.

$$\frac{\partial E^t}{\partial W} = \frac{\partial E^t}{\partial \hat{y}^t} \frac{\partial \hat{y}^t}{\partial h^t} \frac{\partial h^t}{\partial W} \tag{2.23}$$

where h^t is, $h^t = \tanh(\mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)})$, this is illustrated in Figure 2.16. It should be noted that the basic idea and approach is the same as for feedforward neural networks. The difference is due to the design of RNN:s and that the

backpropagation have to be adjusted for this. The key difference is that the gradient are summarized to find the final gradient to update the weights at each sequence step.

2.5.6.2 Vanishing and exploding gradient

When training RNN two of the most common problems is the vanishing and exploding gradient problems. When training deep neural networks the earlier layers have a tendency to learn faster than neurons in the later layers. In this case, earlier refers to closer to the output layers of the neural network, see Figure 2.17. This is known as vanishing gradients due to that the gradient in the later layers in the networks becomes very small, "vanish" during the back propagation[Nie17c]. There can also be a problem if the gradients in the earlier layers become very large, explode also known as exploding gradients. Instabilities of the gradients is a common problem for all types of gradient based learning[Nie17c]. For the learning process itself the exploding or vanishing gradient is in itself not the biggest problem. Instead that it is that the gradient in earlier layers is the product of the gradient in all the later layers. Thus the weights in the network will be trained with a different speed depending on where they are in the network if not specific techniques are used.

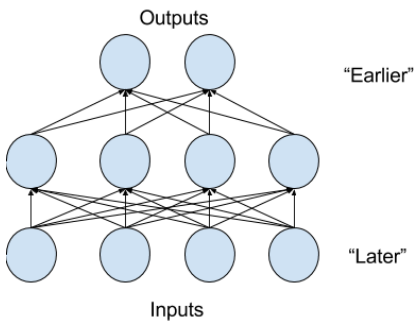


Figure 2.17: Illustration of a neural network to show what "early" and "later" refers to in the meaning of gradient instability

2.5.6.3 Long Short Term Memory

Standard RNN of the type discussed in section 2.5.6 have problems to learn long range dependencies. This has been thoroughly discussed by Bengio, et al [BSF94]. They show that the gradient based learning methods have increasing difficulty when the length dependencies to be captured by the RNN increase. A method to solve the problem and increase the capabilities of the RNN to handle long-term dependencies is to use Long Short Term Memory networks (LSTM). In Figure 2.18 an unfolding of two sequence steps of RNN is illustrated. This illustration should act as a comparison for the LSTM unit that is presented in Figure 2.19.

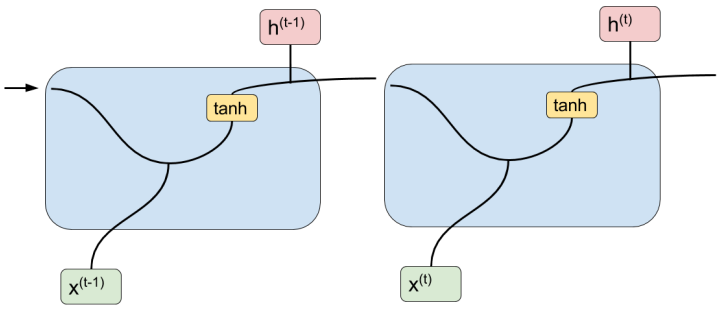


Figure 2.18: Figure illustrating the most common configuration of a RNN unit unfolded for two time steps.

The structure of the LSTM units will be further described. The key features introduced for the LSTM unit is the cell state that is illustrated in Figure 2.19 and the different gates introduce in order to control the information stored in the cell state. The gates controls the information that is stored. In the following section a more through introduction of the LSTM unit and the mechanism will be presented.

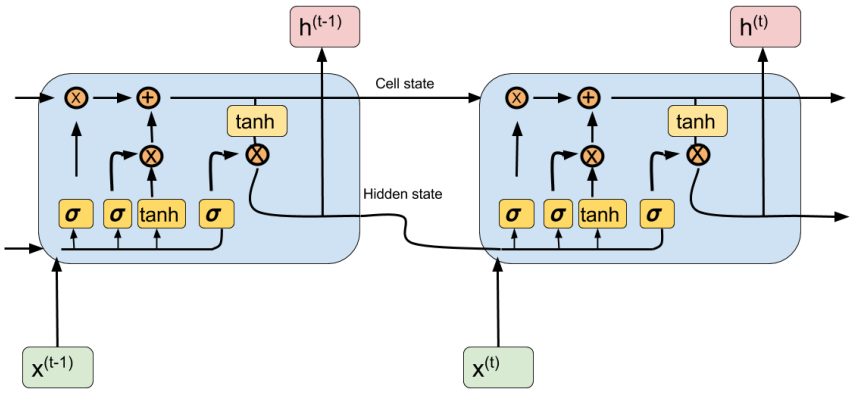


Figure 2.19: Illustration of a LSTM unit unfolded for two time steps. The figure illustrated how the units are connected, the cell state and the hidden state is marked.

The first gate controls the amount of information let through from the previous cell state, $c^{(t-2)}$, see Figure 2.20. The transfer function is depending on the concatenation of the input to the state $x^{(t-1)}$ and the previous hidden state $h^{(t-2)}$. $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$. This decision layer is also known as the forget gate layer. If the sigmoid function outputs 1 everything from the previous hidden cell state is kept, if it outputs zero nothing of the previous cell is let through.

The following step determines what information will be added to the cell state. This step has two sub-parts where the first is known as the input gate layer see Figure 2.20. The input gate layer sets which values in the cell state the will be updated. i_t controls the input gate and has the function $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$. New candidates values of the cell state are calculated based on the hidden state and the input values according to $C'^{(t-1)} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$

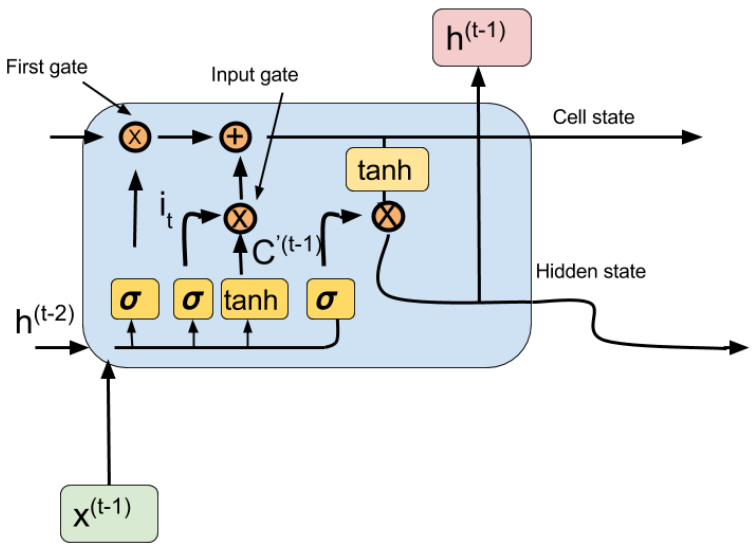


Figure 2.20: Illustration of the concepts of LSTM units. The first gate and the input gates is marked. The figure should illustrate how the cell state is updated.

The cell states is updated based on the input gate layer and the candidate values, $i_t * C'^{(t-1)}$ and thus the candidate values are scaled by the input gate layer. Last task is to calculate the output values.

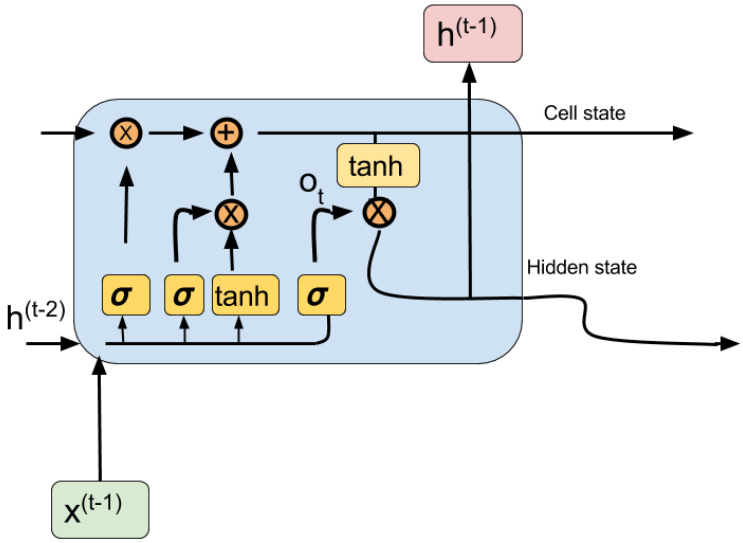


Figure 2.21: Illustration of the concepts of LSTM units. The figure should illustrate how the hidden state is updated based on the input, previous hidden state and the cell state.

The output is calculated based on the cell state but filtered by the input and the hidden state, see Figure 2.21. The filtering function is $o_t = \sigma(W_0[h_{t-1}, x_t] + b_0)$. The output is calculated based on $h_{t-1} = o_t * \tanh(C^{(t-1)})$. The output will also be the hidden state for the next layer. Figure inspired by blog post by Christopher Olah[Ola].

2.5.6.4 Attention

LSTM have resulted in increased performance and given state of the art [KDG15]. However LSTM still have some problems dealing with long range dependencies[Bri]. Attention is a techniques that have shown promising results in improving this. Attention allowed the networks to attend to certain part of the sequence at each time step of when the output is generated. The model learns which part of the sequence it should put attention to at a certain output, thus the name Attention. To generate the output, the network looks at all the states and based on the weights representing the importance of each part of the sequence at given time step the output is calculated.

2.6 Regularisation

The goal with deep learning and any machine learning technique is that based on a limited set of training data fit a model that generalize well. In order to do so the model have to learn as much as possible from the data but not to much to fit the model only to the training data and thus perform worse on new data. When the model fit to well to the training data and get worse on generalisation is known as overfitting. Regularisation are techniques to allow the model to have a better generalisation performance while extracting as much information as possible from the training data[GBC16]. Regularisation techniques have been an important area within machine learning and deep learning[GBC16]. In this section a selected range of methods will be discussed and explained. In Figure 2.22 an example of overfitting is illustrated. The data is collected during the development of the RNN model.

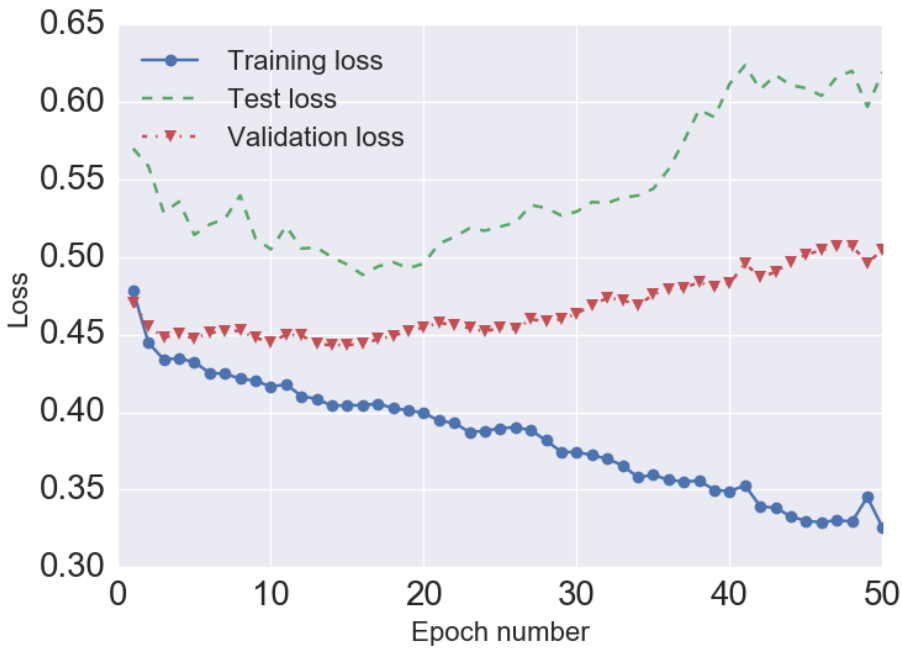
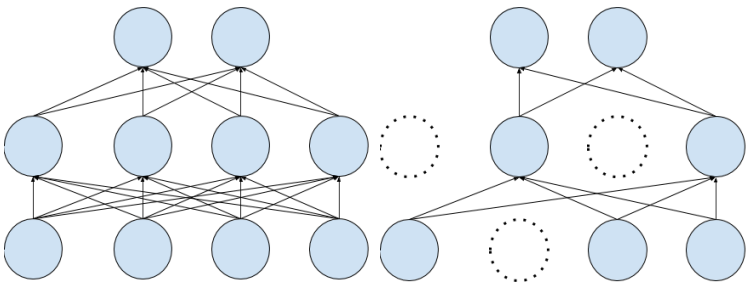


Figure 2.22: Example of overfitting. During the training the initial fits better to training, test and validation set but over several epochs the model start to fit to well to the training data and fit worse to the test and validation set. Thus the model is overfitting and perform worse in terms of generalization. The example is taken from the process of fitting the RNN model

.

2.6.1 Dropout

Dropout was first presented in 2014 by Srivastava et al [SHK+14]. The idea behind dropout is very simple yet it has shown to significantly reduce overfitting[SHK+14]. The name Dropout refers to that some of the units in the neural network is dropped out see Figure 2.6.



(a) Simple feedforward nerual network. **(b)** Simple feedforward neural network with dropout.

Figure 2.23: The figures (a) and (b) should illustrate the effect of implementing dropout.

Dropout provides a approximately combination of exponentially many different neural network architecture. The dropout is set by a hyper parameter that represent the probability keeping each individual unit in the layer. Usually the dropout probability is closer to 1 than 0.5 [SHK+14]. Applying dropout to a neural network has the effect comparable to sampling a thinned network from the original network, see Figures 2.23a and 2.23b. Where the thinned network is composed of the units that survived the dropout process. The neural network in 2.23a with n units (8) can be seen as a weighted combination of 2^n (2^8) possible thinned networks sharing weights. For each training sample one of the thinned networks is trained. This will result in that each of the networks get trained very rarely if ever through out the training of the network[SHK+14]. However at test time it is not possible to test a sample on all the possible thinned networks (2^n) instead the weight are multiplied with the probability used for the dropout, thus scaled depending on the dropout. For a traditional neural network,see Figure 2.23a the feedforward operation is as follows:

$$z_i^{(l+1)} = b_i^{l+1} + \mathbf{W_i}^{l+1}\mathbf{y}^{(l)}, \tag{2.24}$$

$$y_i^{l+1} = f(z_i^{l+1}), \tag{2.25}$$

Where $\mathbf{b}$ is the biases, $\mathbf{W}$ are the weights, $\mathbf{y}$ is the output from the previous layer of the input to the model and f is the transfer function. implementing dropout the feedforward, see Figure 2.23b operations becomes:

$$r_j^l \sim \text{Bernoulli}(p), \tag{2.26}$$

$$\mathbf{y}'^{(l)} = \mathbf{r}^l * \mathbf{y}^l, \tag{2.27}$$

$$z_i^{(l+1)} = b_i^{l+1} + \mathbf{W}^{l+1}\mathbf{y}'^{(l)}, \tag{2.28}$$

$$y_i^{l+1} = f(z_i^{l+1}), \tag{2.29}$$

The same notifications are used as in the previous equation with the added parameters p which is a probability. The Bernoulli function returns 1 or 0.

2.6.2 Batch-normalisation

Batch-normalisation is a regularization technique that allows for a higher learning rate in the model and also less attention is needed for the initialization of the weights and biases [IS15]. Using batch normalisation also reduces the need for dropout[IS15]. According to batch normalisation for a layer with $x = (x^1...x^d)$, each dimension will be normalised to:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - E[x^k]}{\sqrt{Var[x^{(k)}]}}. \tag{2.30}$$

The mean and the variance is calculated for each feature. Batch-normalisation also introduces a new transfer function for each activation function $x^{(k)}$ according to:

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}. \tag{2.31}$$

Where γ, β are trainable parameters that are trained together with the rest of the parameters in the network. This allows the network to find the best distribution of the features in the network by shifting and scaling[IS15].

2.6.3 Early stopping

Early stopping is a regularisation method that is based upon the concept of stopping the training when the model starts overfitting. For models with large

amount of parameters it can often be observed that the training error decrease steadily over time while the test error after a while start to increase [GBC16]. Thus a model with a better generalization error can be obtained by returning to the earlier parameter settings. According to Goodfellow et. al [GBC16] early stopping is probably the most common from of regularization in deep learning due to its effectiveness and simplicity. The early stopping algorithm is presented in Algorithm 2.

Algorithm 2 Early stopping algorithm as defined by GoodFellow et.al [GBC16]

```
Let n be the number of steps between evaluations.
Leta p be the number of times to observe worsening test set error before
giving up.
Let Θ_c be the initial parameters
i ← 0
j ← 0
v ← ∞
Θ* ← Θ
i* ← i
while j < p do
    Update Θ by running the training algorithm for n steps.
    i ← i + n
    v' ← Test set error(Θ)
    if v' < v then
        j ← 0
        Θ* ← Θ
        i* ← i
        v ← v'
    else
        j ← j + 1
    end if
end while
Best parameters are Θ∗, best number of training step is i∗
```

The number of epochs a neural network is trained can be viewed as a hyper parameter and thus selecting the best number of epochs is a type of hyper parameter optimization. However there are a significant differences between the number of epochs and other hyperparameters. Training a neural network several epochs are evaluated in comparison to other hyperparameters where only one set up is evaluated for each complete training session. Early stopping doesn't effect the underlying learning dynamics of the model. This is sometimes a problem for other regularization methods as weight decay(e.g L^2) where care have to be taken not to limit the network to a local minimum.

2.6.4 L^2 Parameter Regularization

L^2 regularization is a also known as weight decay and is a regularisation method that drives the weights closer to the origin by adding a norm penalty, $\Omega(\Theta) = \frac{1}{2} * \omega$.The norm penalty is usually set to only penalizes the weights, Ω and leave the biases unaffected [GBC16]. This is due to that leaving the biases unregulazied should only induce a limited amount of variance, it has also been noted that regularizing the biases could result in a significant underfitting [GBC16]. The weight decay for the regularization should ideally be fitted individually for the different layers, however this result in a larger search space during hyper optimization. It is reasonable to set the same decay at all layers and thus limit the search space [GBC16].

2.7 Padding

Recurrent neural networks can work with sequences of varying length however Tensorflow cannot handle input data of varying size within each batch. In order to solve this the sequences are padded with zeroes to have a specific length. The output from the sequence steps where the input data is the result of padding is discarded when the cost and accuracy is calculated. Thus the network is only trained on the steps where true input data is feed to the network. The padding is done in preprocessing step. The Figure below, Figure 2.24 illustrate the result of zero padding.

$$\begin{bmatrix} [1.0, 2.1] & [1.1, 1.9] & [1.0, 2.0] & \\ [1.1, 2.2] & [1.0, 2.0] & [1.1, 2.1] & [1.3, 2.2] \\ [1.0, 2.2] & & & \end{bmatrix} = \begin{bmatrix} [1.0, 2.1] & [1.1, 1.9] & [1.0, 2.0] & [0.0, 0.0] \\ [1.1, 2.2] & [1.0, 2.0] & [1.1, 2.1] & [1.3, 2.2] \\ [1.0, 2.2] & [0.0, 0.0] & [0.0, 0.0] & [0.0, 0.0] \end{bmatrix}$$

Figure 2.24: Illustration of padding.

2.8 Hyper parameter optimisation

Machine learning models are trained based on a set hyper parameters, λ to minimize the loss function in order to achieve the best performance. The process of finding good values for the hyper parameters are known as hyper parameter optimization [JB12]. Machine learning models, f are algorithms that should find a function that minimized the expected loss $\mathcal{L}(x; f)$ where x should be a sample from the true distribution of the data, G_x. In order to achieve good

performance the hyper parameters λ have to evaluated and optimized and thus there are both a "inner" and a "other" optimization problem. $\mathcal{L}$ is a machine learning model with a set of hyper parameters will be denoted A. The goal is to minimize the generalization error by setting the best hyper parameters.

$$\lambda = argmin\ E_{(x \sim G_x)}(\mathcal{L}(x; A_\lambda(x_{(train)})) \tag{2.32}$$

The following equations describe how hyper parameters optimisation is handled in practice according to Bergstra and Bengio [JB12]:

$$\lambda \approx \text{argmin}_{\lambda \in \Lambda}\ \text{mean}_{x \Lambda x_{valid}}\ \mathcal{L}(x; A_{(\lambda)}(x_{(train)}) \tag{2.33}$$

$$\equiv \text{argmin}_{\lambda \in \Lambda} \Psi(\lambda) \tag{2.34}$$

$$\approx \text{argmin}_{\lambda \in \lambda_1, \ldots, \lambda_n} \Psi(\lambda) \tag{2.35}$$

The parameter optimization described in equation 2.33, 2.34, 2.35 is based upon the use of cross-validation in order to replace an expectancy with a mean over a validation set. Equation 2.35 is known as the hyper parameter response function

Grid search over the hyper parameters space is the most common and widespread approach to hyper parameter optimization [JB12]. It is also the most straight forward approach where the parameter space is evaluated according to a set pattern. Research have shown that random search for neural networks outperform grid search at finding a model that are as good or better in a small fraction of the computational time [JB12]. Allowing random search the same computational it has proven to find better models by searching a larger but initially less promising space. There has also been investigations indicating that only a fraction of the hyper parameters really matter for the performance. However which parameter that matters varies between different data sets. Random search also poses some other advantages which are[JB12]:

- At any time the random search can be stopped and from a complete experiment
- If extra computational power is available new tests can be added to old experiments and from a new experiment
- If a single random search fails it can be abandoned with out effecting the complete experiment.
- Tests are independent of other tests and can be carried out asynchronous

Recently focused on hyperparamter optimization have been put on finding better ways of choosing the hyper parameter search based on previous searches of the hyper parameter space. Sequential model-based optimization(SMBO) which is also known as Bayesian optimization is a general technique for function optimization that have been used for optimization of neural networks[JB13]. SMBO where initially developed to be used for experiment design and oil exploration[JB13]. In general SMBO are used for minimizing functions that are costly to evaluate. In comparison to other methods for minimization, as conjugated gradient descent methods, SMBO uses more computational power to evaluate which set of parameters to use for each evaluation. This is advantageous when the evaluations become more and more expensive. Another advantage is that Bayesian optimization doesn't require derivatives. The SMBO algorithm is presented in Algorithm 3. First step in the algorithm is to build a model M_L that captures the dependencies of the loss function L based on a hyper parameter setting λ, see line 1. The following steps are iterated until the time is up(or number of computations). For each iteration the first step is to determine a promising configuration of the hyper parameters λ. Then the loss c is calculated for the set of hyper parameters λ. These values are then used to update the model M_L that should capture the dependencies for the loss in relation to the hyper parameters. In order to choose the next set of hyper parameters to use, an acquisition function a_{M_L} uses the predictive distribution of model M_L for arbitrary hyper parameters $\lambda \in \Lambda$ to quantify how useful configuration λ would be. Then this function is maximised over Λ in order to find the most useful λ to evaluate. This result in that the space is continuously evaluated by looking at the hyper parameter set up that is expected to give the most information about the hyper parameter space.

Algorithm 3 Sequential Model-Based Global Optimization

1: initialise model $M_L; H \leftarrow \Theta$
2: **while** Time budget have not been exceeded **do**
3: $\lambda \leftarrow$ candidate configuration from M_L.
4: Compute the cost function, $c = L$(Model to optimise hyper parameters for)
5: $H \leftarrow H \cup (\lambda, c)$
6: Update M_l given H
7: **end while**
8: **return** λ from H with minimal c

There are several different acquisition functions available. For this thesis hyper parameter optimization have been done using Hyperopt [JB13].

2.8.1 Hyperopt

Hyperopt is an Python library for "optimisation over awkward search spaces with real-valued,discrete and conditional dimensions"[Hyp17]. Hyperopt was originally developed for optimisation of hyper parameters for neural networks. By defining a objective function which should be optimised and an a search space, Hyperopt will use the the chosen algorithm for finding the best hyper parameters. Currently two optimisations algorithms are implemented: random search and Tree of Parzen(TPE) estimators(Bayesian optimisation).

TPE was first proposed by Bergsta et al [BBBK11] for hyper parameters optimization. It uses tree structured Parzen estimators to model the error distribution in a non-parametric way.

2.9 Related work

An article named *Learning to Diagnose with LSTM Reccurent Neural Networks* was published in 2016 by Lipton et al [LKEW15]. The authors investigate how RNN and LSTM could be used for diagnose based on clinical medical data. The report was based on a data set which contained 13 frequently but irregularly sampled clinical measurements from the intensive care unit. The goal was to predict the diagnose out of 128 different diagnosis. According to the authors this was the first article published to their knowledge about the use of RNN and LSTM for health diagnosis on a multiclass case. The RNN model was compare with a fixed window and a set of hand engineered features. The data was irregularly sampled, containing missing values. The data was down sampled to an hourly rate and the missing data was imputed. The initial rate of sampling was not mentioned in the article. The model used was of the type many to one where the input from all time steps was used to make a final prediction. However there test indicated that training the model to make the prediction of the diagnosis at each time step was superior, the authors call this method sequential target replication. According to there results the RNN model outperformed a feedforward network over a sliding window. However the difference between the sliding window approach and the RNN is below 1%. The authors also prove that combining the recurrent neural networks and the sliding window model increase the results. Interesting to notice is that the authors saw improvements of using the first 6 hours of the time series and the last 6 hours instead of the last 12 hours for the sliding window approach.

Deep learning have also been applied to predict the health status of Parkinson's

disease [NYH15]. The investigate data was multivariate continues time series. The data was collected in from the daily life of the patient in terms of the movement. For the predictions an sliding window model was used. it should be noted that the over all results of this work was very low.

In the article *Using recurrent neural network models for early detection of heart failure onset* the use of neural networks for detection of heart failure was investigated [EC16]. In the work a many to one approach was used. The developed RNN model outperformed the other methods also showing that the RNN model became better compare to the other models with a larger window/length on the input sequence. The predictions was based upon if heart failure was detected within a prediction window, where prediction windows of 3 to 9 months was investigated. For all the investigated prediction windows and observation windows the RNN outperformed the other methods investigated(Logistic regression, SVM,MLP, kNN). It should however here also be noted that the differences between the RNN and the second best models was very small below 1.6 % for all investigated cases. They observed that longer observation windows improved the results for all methods.

Adam et al have wirtten an article named *Predicting Patient State-of-Health using Sliding Window and Recurrent Classifiers* which investigate using data from intensive care units to classify the patients health status[AM16]. The input data were a multivariate time series. In there results they report slight improvement of using RNN compare to using feedforward neural networks on a sliding window model on three out of four tests. The authors also mention that combining approaches could increase the results further.

In the article *Doctor AI:Predicting Clinical Events via Recurrent Neural Networks* by E.Choi et al recurrent neural networks used for prediction based on electronic health care records [CBS15]. The goal with the designed models and their study was to " use longitudinal patient visit records to predict the physician diagnosis and medication order of the next visit. As a secondary goal we predicted the time to the patients next visit." The designed models makes multiple predictions at each time step, one prediction for each of the investigated conditions/medication categories. As input the authors uses discrete medical codes e.g from diagnosis, medication, procedures extracted from longitudinal patient visit records. Longitudinal patient record is a comprehensive record that should be based upon several records in order to give a complete picture of the patients health background.

Chapter 3

Data

In this chapter the data will be described and some statistics of it will be presented. This chapter will give a background to the variables and how the data is collected.

3.1 Data

The data set consists of approximately 5700 patients. The patients have been to dialysis treatments a number of times at which measurements on the patient and the dialysis have been collected. The patients have been treated for a different number of days and thus the number of dialysis treatments varies for the patients resulting in a data set with varying sequence lengths. The sequence lengths vary between a few sequence steps up to several hundreds. The frequency at which the patients receive dialysis treatment is also varying. Since the patients are usually not hospitalized, the data set is imbalanced towards patients that will not be hospitalized within 30 days. From each dialysis treatment the following variables are used:

- Average Blood Flow Rate - The blood flow rate during the dialysis session in ml/min.

- Start Sitting Pulse - The patients pulse before the dialysis session was started in bpm.
- Liters Processed - The number of liters processed during the dialysis session in liter.
- Time in Dialysis - The number of minutes that a patients have been connected to the dialysis in minutes.
- Fluid Removed - The amount of fluid removed during the dialysis session in L.
- Patient Temp Start - The temperature of the patients at the start of the dialysis session in Fahrenheit.
- Patient Temp End - The temperature of the patients at the end of the dialysis session in Fahrenheit.
- Start Sitting Systolic Bp - The systolic blood pressure before the dialysis session in mmHg.
- Start Sitting Diastolic BP - The diastolic blood pressure before the dialysis session in mmHg.
- End Sitting Systolic BP - The systolic blood pressure at the end of the dialysis session in mmHg.
- End Sitting Diastolic BP - The diastolic blood pressure at the end of the dialysis session in mmHg.

In addition to the measurements from the dialysis treatment sessions there are also 4 other records available, these are:

- Patient Schedule records
- Patient Demographic records
- Patient Hospitalisation records
- Patient List records

The *Patient Schedule* contains information about the dialysis schedule of the patient. This includes the number of sessions per week the patient should receive treatment, it also contains the schedule the patient is on i.e which weekdays the patient receive treatment. The *Patient Demographic* contains information such as age, ethnic background, and date of first dialysis session. The *Patient*

Hospitalisation is the record of previous hospitalizations containing information about the arrival date, stay length, reason for hospitalization and diagnose. The *Patient List* contains information about the dialysis settings, clinic and where the patient lives. The data collection process is described in Figure 3.1

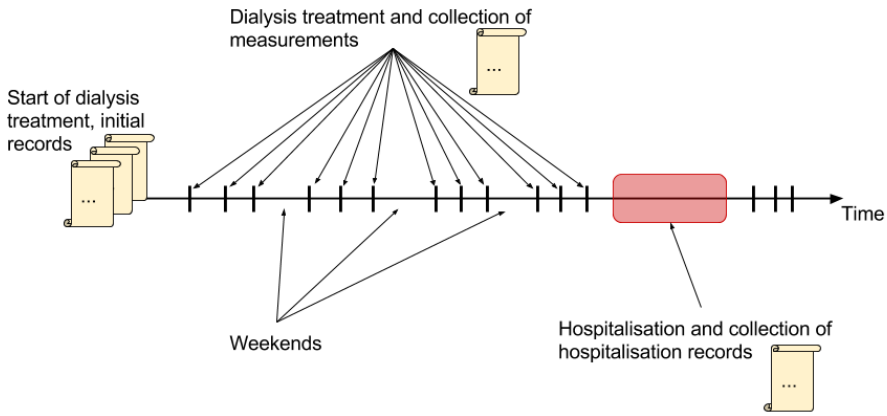


Figure 3.1: Visualization of the data collected over time for a patient. Initially, records about the patient are set up. These include Patient Schedule, Patient Demographics and Patient List. Some of the values in these are updated during the treatment, for example the number of treatments per week that the patient should receive. The illustration is based upon the most common case where the patient receive dialysis treatment three times a week. No treatments are usually given at the weekends which is also indicated in the illustration. During the dialysis treatments data is collected about the treatment and the health of the patient. During hospitalization no data about possible dialysis treatments are recovered. However records about the hospitalization and diagnoses are stored and available.

The number of treatments patients have received is varying and the distribution is presented in Figure 3.2. The mean sequence length is 296 and the median is 188. Usually the patients received dialysis treatments three times a week and thus the mean number of days that the patient been in the record is $296 * 7/3 \approx 684$. The life expectancy is 5-10 years according to the National Kidney Foundation [Natb] and thus the sequences are shorter than expected. There are several reasons for the relatively short sequence lengths. One possible reason is that the patient gets a kidney transplantation and thus doesn't need to receive dialysis any longer. The data is from clinics in USA and due to the privatization of the hospitals, a change in health insurance might result in that the patients

are forced to change clinics and thus results in shorter sequences. Important to notice is also that the records only go back to 2009, and that a lot of the patients have received dialysis long before the records started.

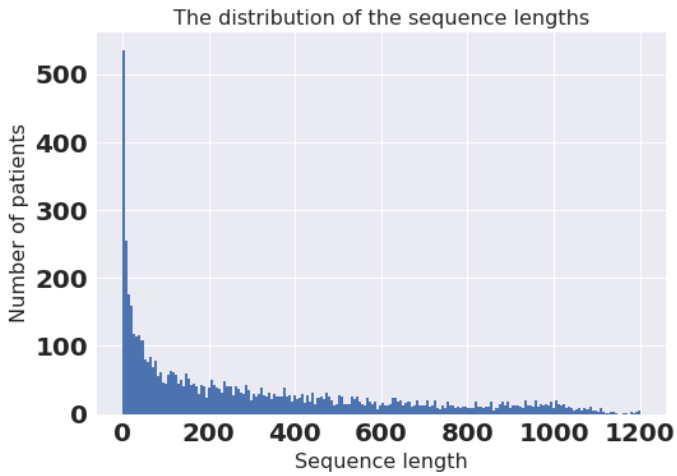


Figure 3.2: Distribution of the number of treatment the patients have been to.

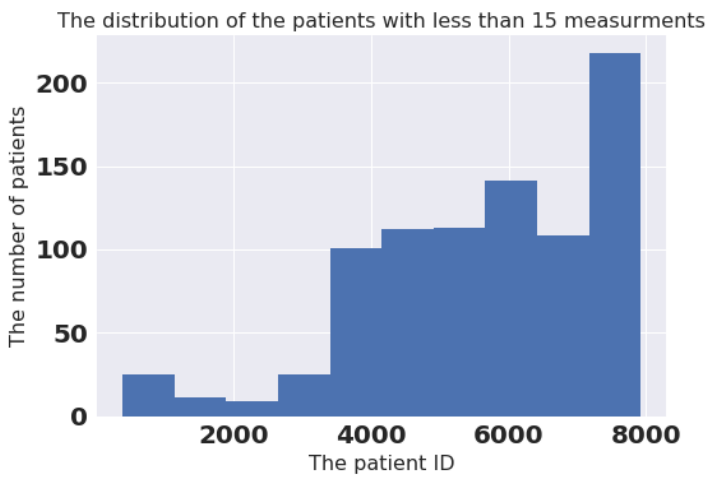


Figure 3.3: The distribution of sequences shorted than 15 sequence steps and there ID numbers. It should be noted that the ID:s are distributed in ascending order for new patients.

The distribution of sequences shorter than 15 in relation to patient ID number was investigated, see Figure 3.3. Patient ID is given in the order the patient gets registered at the clinics thus the patients receiving a ID number starting with 8xxx are the most recently submitted to dialysis. The large part of short sequences for the patients with the highest ID number is due to that they have note been admitted long enough to have been to more than 15 treatments. However, it should be noted that for patients with ID:s between 4000 and 7000, there is an over-representation of short sequences in relation to the expected lifespan of patients. The relatively large distribution of short sequences might be related to patients shifting health insurance and thus are forced to change clinics. If this is the true reason for the high representation of relatively short sequences in relation to the expected life of dialysis patients is unknown to the author. When patients shift clinics this affects the quality of the medical history available due to that previous data is unknown.

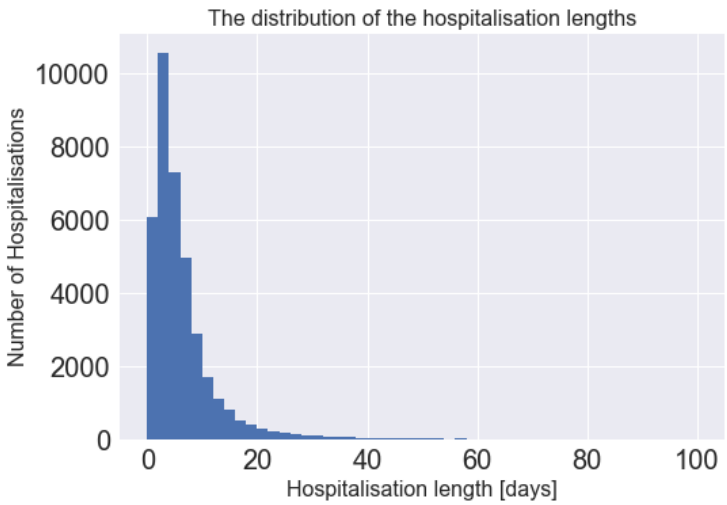
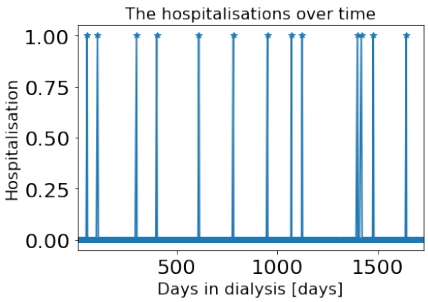
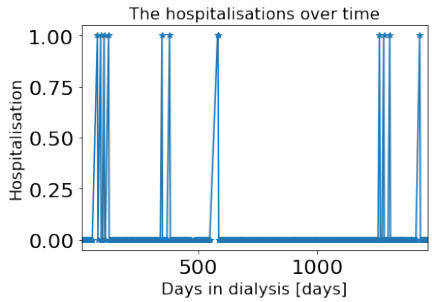


Figure 3.4: The distribution of the length of the hospitalisations.

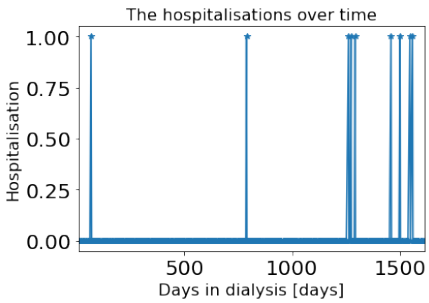
The distribution of the length of the hospitalizations can bee seen in Figure 3.4. For all the patients the median hospital length is 4 days and the mean hospitalization length is 5.9 days. The frequency for the hospitalization was also briefly investigated, see Figure 3.5.



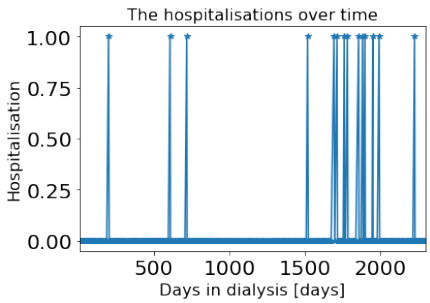
(a) Patient A



(b) Patient B



(c) Patient C



(d) Patient D

Figure 3.5: The hospitalizations plotted as binary points in time against the days the patient has been receiving dialysis treatment for four different patients A,B,C and D. The data was filtered to only include patients that have been hospitalized a minimum of two times. The four samples of the patients were randomly chosen.

It seems as though the hospitalization often come in groups, see Figure 3.5. The number of hospitalization doesn't show any obvious pattern to increase over time. The hospitalizations do not seem to be evenly distributed over time, but rather the patients have periods of higher and lower hospitalization frequencies. In Figures 3.6 and 3.7 the variables based on the measurements during dialysis is plotted to show their distribution.

In order to investigate if there were some obvious patterns in the data an PCA analysis was conducted for the data. The labels are indicated by red and green. As can be seen in Figure 3.8 no obvious pattern is presented in the data based upon the PCA.

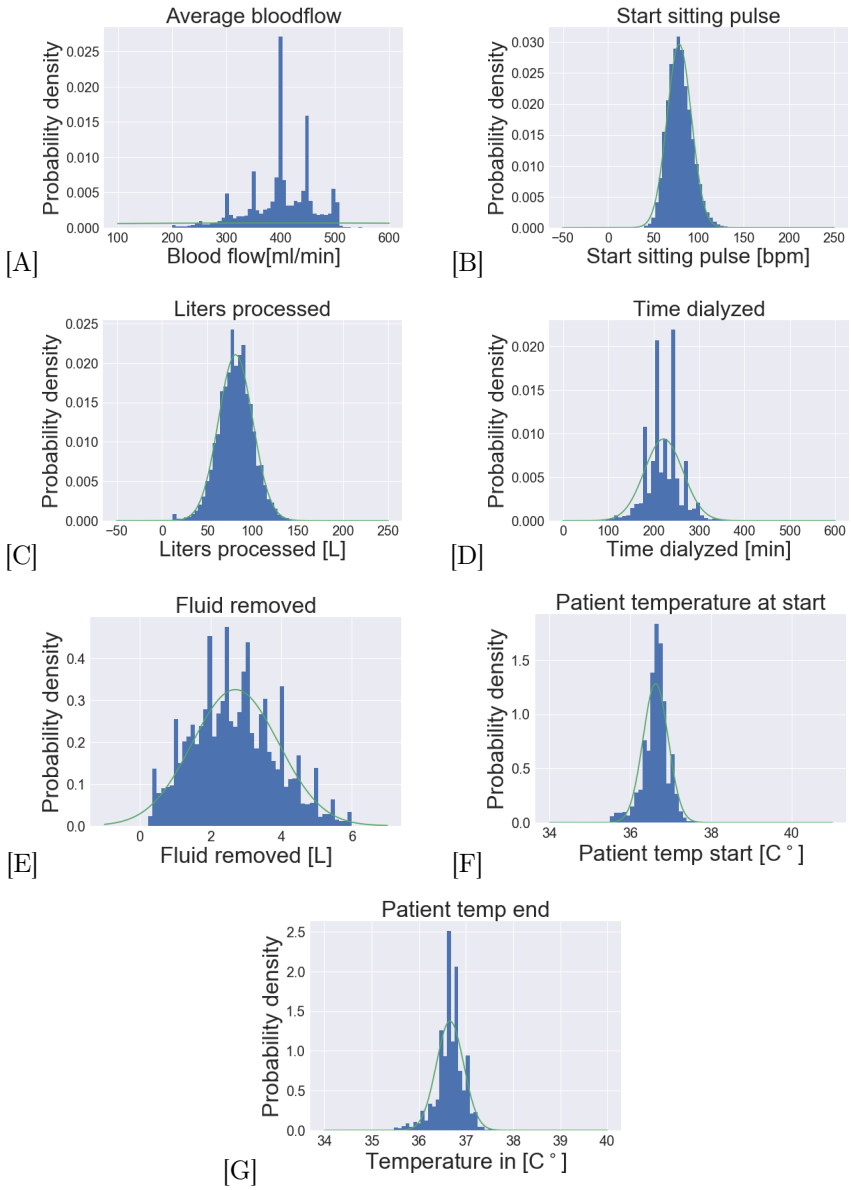


Figure 3.6: The distribution for seven of the parameters that are continuously measured for the dialysis patients. The distributions are plotted and normal distribution curves are plotted for comparison.

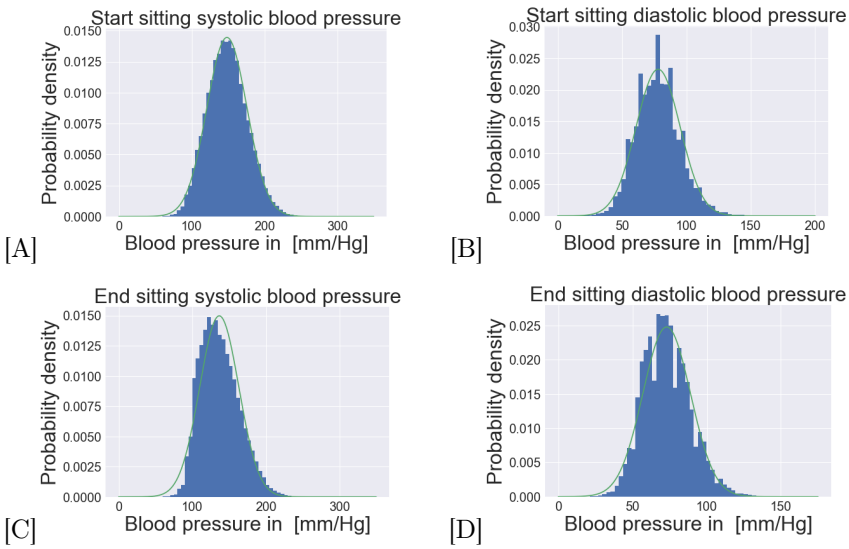


Figure 3.7: The distribution for four of the parameters that are continuously measured for the dialysis patients. The distributions are plotted and normal distribution curves are plotted for comparison.

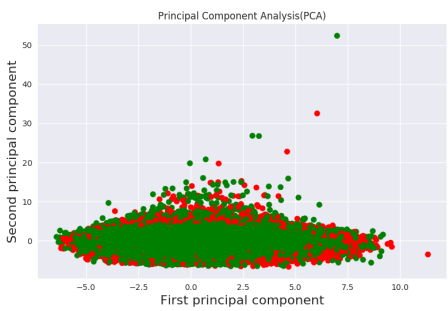


Figure 3.8: Principal component analysis of the original 11 features. The data was plotted for the first two principal components where the green and red should indicate the two labels. Based on the two first principal components it is not possible to make any successful classification.

3.2 Pre-processing and standardization

The data was preprocessed to remove outliers and measurements that have shown very abnormal behavior. This step was done before the data was received by two master thesis students at Lytics. Therefore the first step in the preprocessing was to standardize according to common practice [Gra12] so that the mean, m_i is 1 and that the standard deviation, σ_i is 1 for each feature.

$$m_i = \frac{1}{n} \sum_{i=0}^{i=n} x_i \tag{3.1}$$

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=0}^{i=n} (x_i - m_i)^2} \tag{3.2}$$

$$\hat{x}_i = \frac{x_i - m_i}{\sigma_i} \tag{3.3}$$

The new standardized feature is represented by $\hat{x}_i$ and the input feature is represented by x_i. By standardizing the data the performance is increased due to that the values are in a more suitable range for the common used activation functions [Gra12].

Chapter 4

Methods

The different methods implemented and how they where designed will here be presented. The pipeline of the feature extraction/feature engineering will also be presented and the approaches to the problem.

4.1 Pipeline, feature extraction and feature generation

Based on the different record there is a possibility to create features beyond the continuously measured dialysis variables, features that could carry a lot of predictive information. The work with creating these features, and why these features were chosen will be explained in this section. It should be noted that the features generated, and which features to generate were discussed with a specialist nurse working at Lytics, who has knowledge in which features are used by doctors and which features could prove to have an effect on the predictions. However, the goal was not only to generate previously known indicators, but also to create new features to improve and extend the previous results. As mentioned in the introduction to the 3.1 Data, simultaneously to this thesis work, another thesis project was conducted at Lytics focusing on imputation of missing data, thus investigating missing data and outliers will not be conducted,

instead the data used for the predictive models will be preprocessed according to their findings [MA17]. An illustration of the pipeline is presented in Figure 4.1. In the figure, the light red rectangle to the left illustrates the part of the pipeline that was conducted by the other master thesis students. The green rectangle illustrates the work conducted by the author.

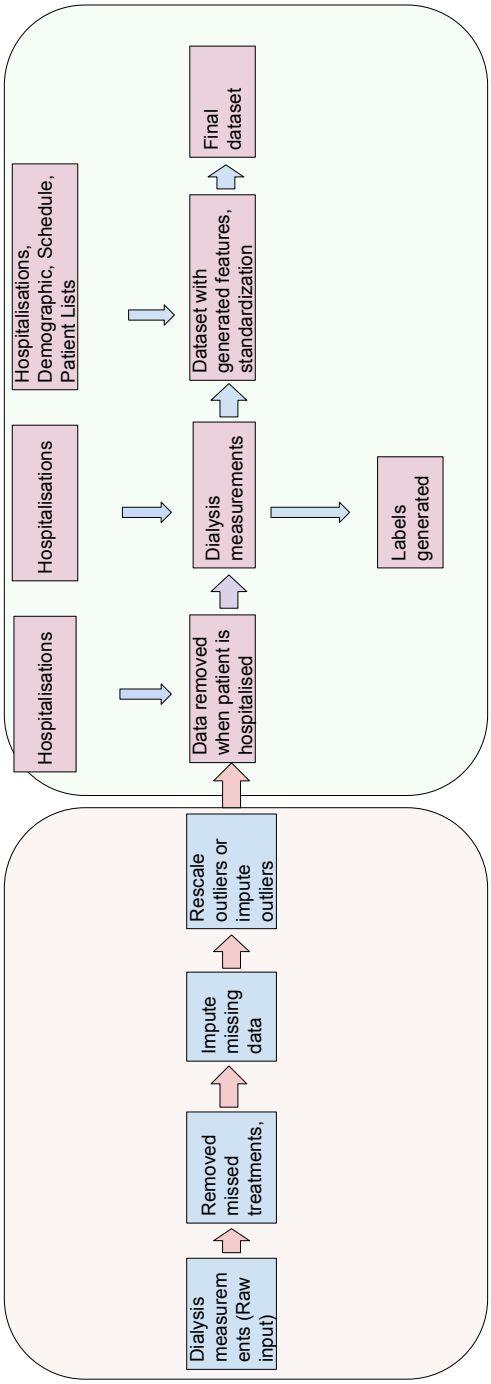


Figure 4.1: Illustration of the pipeline and the collaboration with a master thesis group working on imputation of missing data

The generated features are described in Table 4.1 and 4.2

Generated features		
Variable name	Explanation	Variable type
Days since the last 20 hospitalizations; **Days since last hospitalisation**, **Days since second last hospitalisation**, ..., **Days since twentieth hospitalisation**	The number of days since the last 20 hospitalizations. Days since each of the 20 last hospitalisations is generated as an individual feature.	Discrete, ratio
Length of the 20 last hospitalisations; **length of the last hospitalisation**, **length of the second last hospitalisation**, ..., **length of the 20:th last hospitalisation**	Number of days for each of the 20 last hospitalisations. The length of each hospitalisations is generated as an individual feature.	Discrete, ratio
Number of previous hospitalisations	The number of times the patients have been hospitalised	Discrete, ratio
Number of days the patient received dialysis.	The number of days the patient have been receiving dialysis treatment	Discrete, ratio
Schedule of the dialysis treatment	The schedule for the dialysis treatment. This is one-hot-encoded. Thus each possible schedule is generated as a feature of its own	Discrete, nominal
Number of scheduled dialysis treatments per week	The number of dialysis treatments per week that the doctors have scheduled the patient for	Discrete, ratio

Table 4.1: Table describing the generated features.

Generated features		
Variable name	Explanation	Variable type
Days since last dialysis session	The number of days since last dialysis session	Discrete, ratio
Sex	The sex of the patient is one-hot-encoded thus one feature for each sex is generated	Discrete, nominal
Age	The age of the patient in years	Discrete, ratio
Clinic Id	The clinic Id for the clinic the patients is receiving treatment at	Discrete, nominal

Table 4.2: Table describing the generated features.

Most of the generated features are related to time and how the sequence steps are related in terms of time. The generated features also include medical history of the patient. Life expectancy for dialysis patients is short, between 5-10 years and thus the time in dialysis, and the number of previous hospitalization could potentially include information about the probability for new hospitalizations.

In order to speed up the feature generation for the data sets, which became very time consuming, parallelization of the calculations was implemented utilizing more of the computational power at the server at which the calculations were done.

4.2 Approach

Two separate approaches to the predictions were taken. The first one taken was to have a sliding window where the models were fed a window of n sequence steps and the label for the last sequence step. The model should then used the window to predict the label at the last time step, see Figure 4.2. This allows the model not only to account for the current sequence step but also previous steps. This allows the model to learn patterns over time and thus detect patterns for the patients rather than only look at the values at the present

sequence step. This approach have been used previously for feedforward neural networks on predicting the Respiratory Motion Predictions of Lung Cancer Patients [TBP15], classify cases of gout, leukemia [TAL13], and critical illness [CKL+15]. Sliding window has also been used as reference method to RNN for diagnose in several articles [EC16], [AM16], [BSF94]. It can also be noticed that trends are today monitored by the dialysis clinics as an indication of the health status of the patients. This was done over a three months period, which equals 36 dialysis treatment for a patient receiving treatment three times a week, which is most common. This would indicate that having larger windows would be advantageous. The sliding window approach will be used for feedforward neural networks, Random Forest, kNN and a convolutional neural network. No previous work have been found by the author of using CNN for predictions on health care time series data. CNN should have an advantage finding trends and relations between variables.

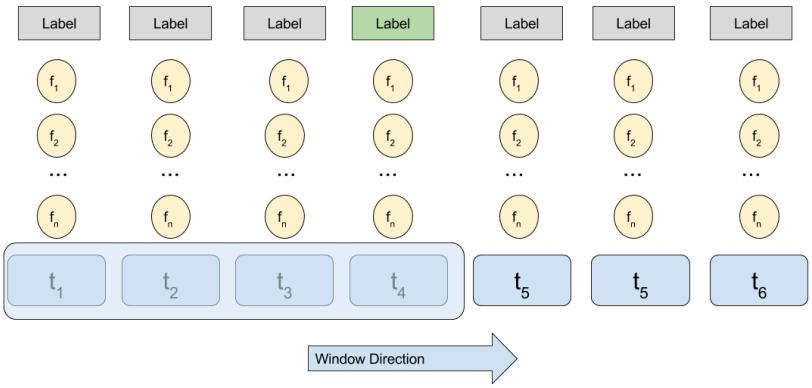


Figure 4.2: Illustration of the sliding window approach, where the model is given a window of data and should predict the label for the last sequence step.

The second approach that was investigated was to look sequentially at the time series, similarly like a doctor, see Figure 4.3. Ideally this should allow the models to look at the complete history of the patient which potentially would allow for improved results. Compared to the sliding window, this approach could be illustrated as a sliding window with a growing window size for each sequence step. However there are obvious computational limitations with learning behaviours from long sequences.

The raw data is unevenly spaced with the most common frequency of 3 measurements per week. However, no special care was taken towards making the

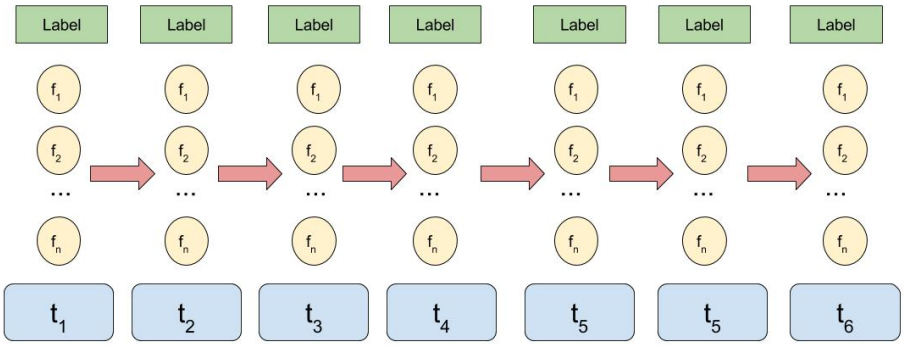


Figure 4.3: Illustration of the the principal with the sequential model predicting the current output on the current input and all previously known inputs.

time series evenly space, with the expectation that the models should be able to handle this. It should be noticed that the labels at each time step is set according to if the patient will be hospitalized within 30 days from that date.

4.3 Developed models

Three different methods based upon neural networks were developed; a recurrent neural network, and convolutional neural network, and a feedforward neural network. To compare two baseline methods from the Python machine learning library sci-kit learn was also used, a Random forest model and an k-Nearest Neighbor classifier. In this section, the design of the different models will be presented. Hyperparamters are chosen based validation set performance. For the neural networks, Tensorflow was used for the implementation. Tensorflow is an open source library developed by Google for machine learning.

4.3.1 Random forest

A Random forest model was implemented based on the sci-kit learn package for Python. The model was based upon the sliding window approach discussed earlier. The final model used 700 trees to make the prediction. The hyperparameters for the Random forest model were chosen based on a grid search. No increase in AUC score was found with a larger number of trees than 700. The

Random forest was optimized based on the number of trees, maximum number of features to try for each tree and minimum sample leaf size.

4.3.2 kNN

The kNN model was very computationally demanding — a problem that is well known for large data set since the distance between all data points in the test data and the training data have to be calculated. In order to run the models on a reasonable time, a reduced data set was used. The reduced data set was based upon a random selection of $\sim 1/3$ of the data. The models was fit based upon the 3 closest neighbours.

4.3.3 Feedforward neural network

A feedforward network was built using a sliding window approach in order to let the model get data over a certain sequence window. This was done since it is assumed that it is not only the absolute values but also the trends over time that is important for predicting the hospitalization. That the trend is of importance is known since earlier tests, and is currently practiced at the dialysis clinics which Lytics cooperates with. The model was originally built using two hidden layers with dropout layers introduced between the layers in order to avoid overfitting. Initially a model was developed with $\sim$ 400 000 trainable parameters. The initial model was developed with a larger number of parameters to allow the model to model more complex relations in the data. This model was later further developed and shrunken to $\sim$ 20 000 trainable parameters. It was found that a lower number of parameters could give the same results in terms of AUC with a lower computational burden. The hyperparameter optimization packaged Hyperopt was partly used for optimization of the model. Using the hyper optimization packaged should be used with care, which will be discussed more in the discussion. An illustration of the model is presented in Figure 4.4, the final number of units is indicated in the model. Dropout was implemented between hidden layer 1 and hidden layer 2 with a probability of 0.95. Dropout was also implemented between hidden layer 2 and the output layers with a probability of 0.9. Dropout was chosen due to that it performed great and was relatively easy to tune. As transfer function the relu function was used. The output is a softmax output layer. The total amount of trainable parameters in the final model is 19813. For the model, Adam optimization was implemented. Adam optimization was chosen since it should in theory allow for less optimization of the learning rate and improved learning compare to

stochastic gradient descent due to use of momentum. A learning rate of 0.01 was used.

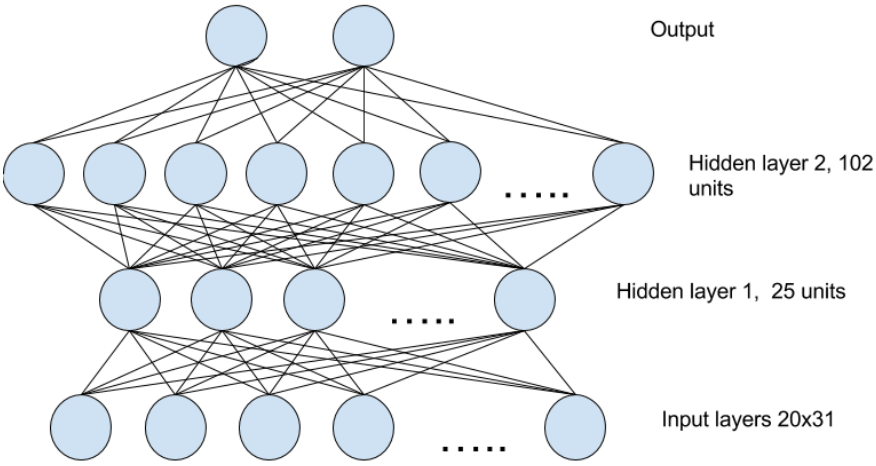


Figure 4.4: Illustration of the developed feedforward neural network model.

4.3.4 Convolutional neural network

A convolutional neural network was built and evaluated on the hypothesis that it could be superior on finding trends in the data which could allow for better predictions. It should also have the possibility to find trends between different features. The input data was fed based upon a sliding window of 20 sequence steps. The input was fed as a matrix where each column represented a sequence step and thus each row is the input from one feature for 20 sequence steps, see Figure 4.5.

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,20} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,20} \\ \dots & \dots & \dots & \dots & \dots \\ x_{31,1} & x_{31,2} & x_{33,3} & \dots & x_{31,20} \end{bmatrix}$$

Figure 4.5: Illustration of the input for the CNN model.

The design of the convolutional neural network is presented in Figure 4.6. The first convolutional layer was set to filter all the 31 features simultaneously. For this, a relatively high number of filters were chosen — 24 different filters. This was done in order to be able to capture different trends between the different

features. No spatial relation was assumed between any specific features and the first filter was thus set to filter based upon all features. The general design is inspired by the design of convolutional neural networks for image analysis. The model also uses a 1x1 convolutional layer. The model contained 1334828 trainable parameters. For the model, Adam optimization was implemented with a learning rate of 0.05.

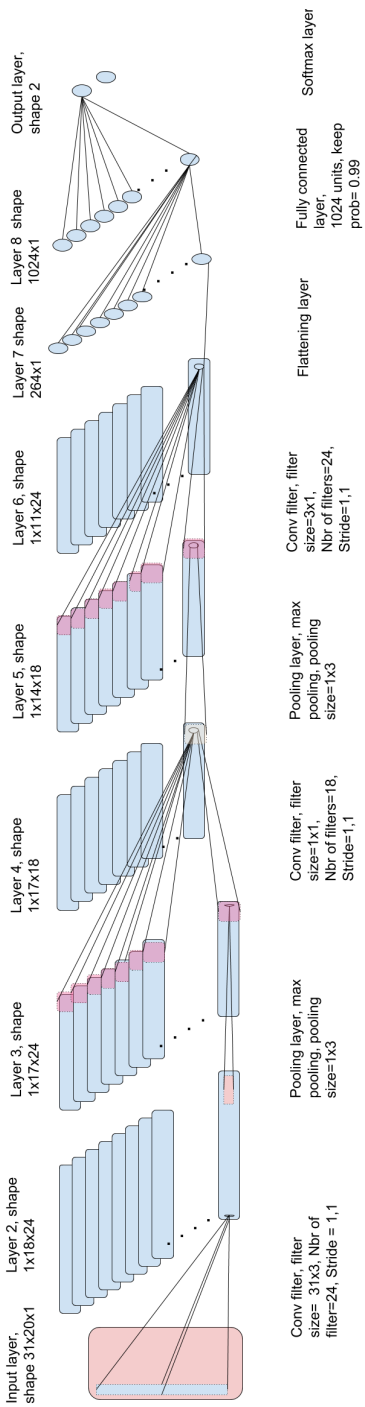


Figure 4.6: Illustration of the developed convolutional neural network.

4.3.5 RNN models

The developed RNN models proved hard to optimise, and initially the author struggled with getting a good fit. Several different types of regularization were tested; dropout, batch normalisation, L2 regularization, but in the end a combination of dropout and early stopping was implemented in order to avoid overfitting. Dropout and early stopping was selected based upon that they performed the best on the problem and were relative easy to tune. The model uses 14322 trainable parameters. The number of units was selected through optimization of the model. Tensorflow unfolds the recurrent neural network when performing the calculations. In order to represent a training batch as a single tensor the shape has to be (batch size x max length x nbr features). In order to achieve this the shorter sequences were padded in order to have similarly length. The padded sequence steps should not be evaluated, only evaluating the actual data. By using a binary mask based on the true sequence length this was achieved, where 1 represent true states and 0 for padded sequence steps. The binary mask was multiplied with the predictions to reduce the padded steps to zero. For the setup, the binary mask was generated in the preprocessing step and fed to the Tensorflow. For the predictions, LSTM units were used. Due to problems with the models, there was sadly no time to implement batch padding which would have reduced the computational time, by only padding the sequences based upon the longest sequence in each batch. For the model, Adam optimization was implemented with a learning rate of 0.001. RNN was evaluated using LSTM units, standard RNN units, and Attention. The size of the model was kept constant for the different variations of RNN models, see Figure 4.7.

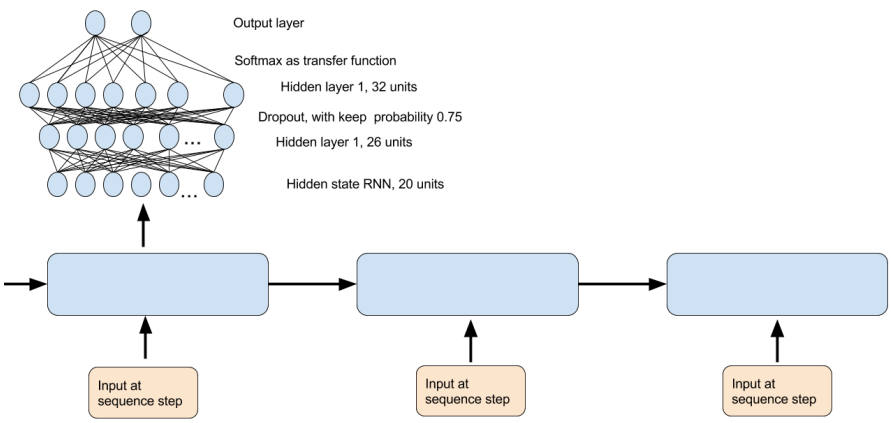


Figure 4.7: Illustration of the developed recurrent neural network model.

Chapter 5

Results

In this section the results from the implemented models is presented. The data set was initially split to data for training(80) and test(20%). The split was done based on the patient ID's in order to make sure that the RNN and the sliding window would be validated against the same data. It was also done in order to not break the causality of the data. The training data was split into training data(80%) and validation data(20%). For the investigate models a window size of 20 sequence steps was used if not stated otherwise. This was selected due to that the hospitals look at trends over 3 months ($\sim$ 40 sequence steps). However, a large amount of the sequences are relative short see Figure 3.4 and thus in order to be able to utilize as much of the data as possible the window size was set to 20 sequence steps.

Due to lack of time and problems with the models, a selection of the generated features was done. For the evaluation of the models, 31 features were used. The 11 continuous features discussed in section 3.1 were used, together with the days since the last 20 hospitalizations which is discussed in section 4.1. The 11 continuous features were selected since they had previously been found by Lytics to be very important to describe the health status of the patients. The previous hospitalization parameters were chosen based upon that it has been reported that repeated hospitalizations might be common for dialysis patients [PJ09]. Repeated hospitalizations refer to hospitalizations that occur within a short period (couple of weeks) following a previous hospitalizations. Groups of

hospitalizations were also identified when previous hospitalizations were plotted agings time, see Figure 3.5.

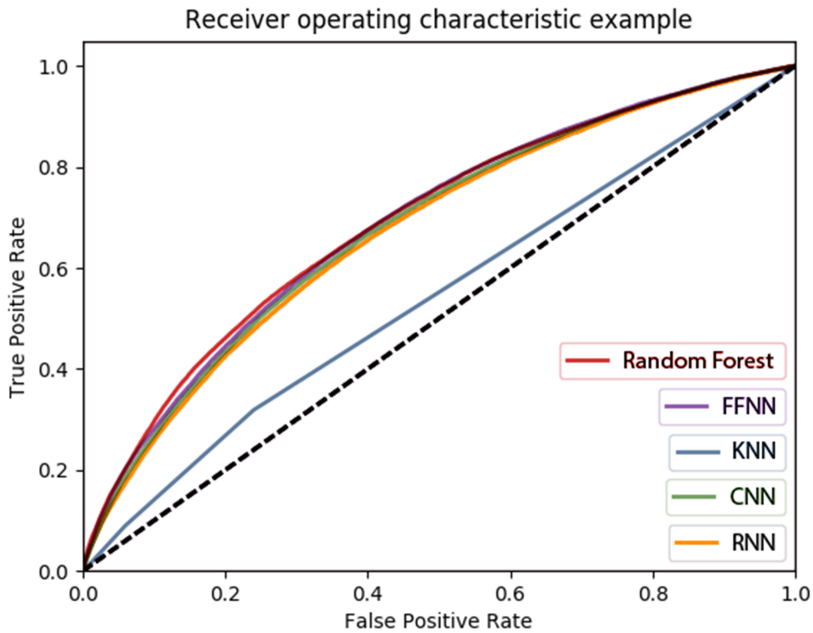
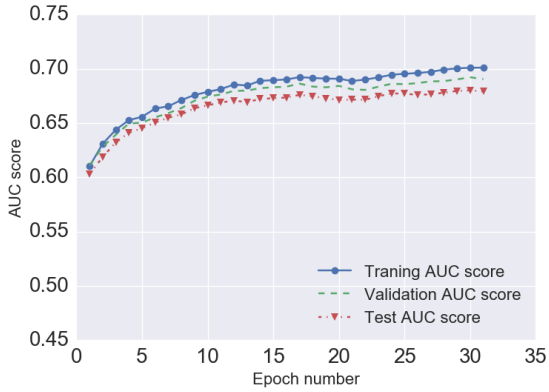


Figure 5.1: ROC-curve for the investigated models. The ROC-curves are plotted individually in Appendix B

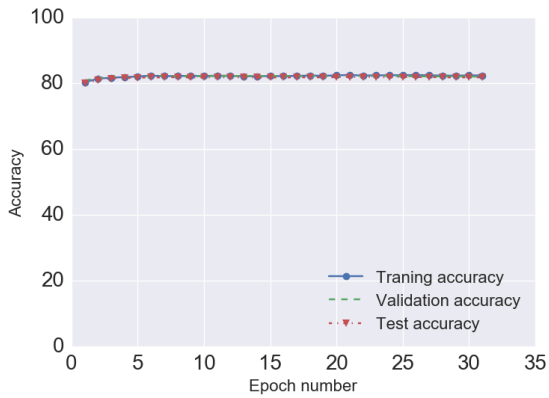
Model	AUC score
kNN	0.545
Random forest	0.692
FFNN	0.689
RNN	0.650
RNN(LSTM)	0.675
RNN(LSTM) with Attention	0.680
CNN	0.678

Table 5.1: AUC score for the developed models

Based upon the developed models, the best performance was achieved with a Random forest model. The AUC score for the different models are presented in Table 5.1. The receiver operation characteristics curves are presented in Figure 5.1. It should be noted that the CNN model is significantly larger (in terms of trainable parameters) than the FFNN and the RNN models, in that the CNN model proved to give the best results. The loss, AUC-score, and accuracy for the CNN, FFNN and RNN model using LSTM units and Attention is presented in Figures 5.4, 5.3 and 5.2. As can be seen in **(a)** and **(c)** in Figures 5.4, 5.3 and 5.2 the RNN model achieved a lower accuracy and higher loss than the CNN and FFNN. A feedforward neural networks with a larger number of units in the hidden layers was created for comparison, see Appendix D. However, the results where not improved, thus the smaller model was selected since the computational burden is smaller.



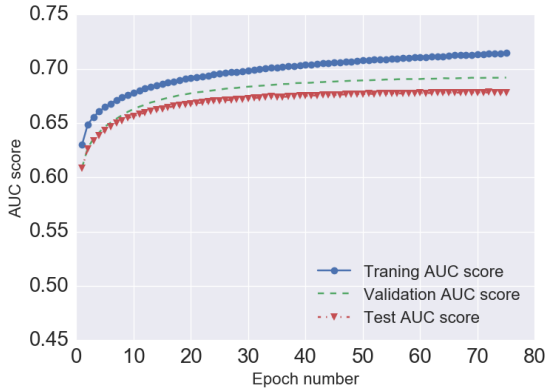
(a)



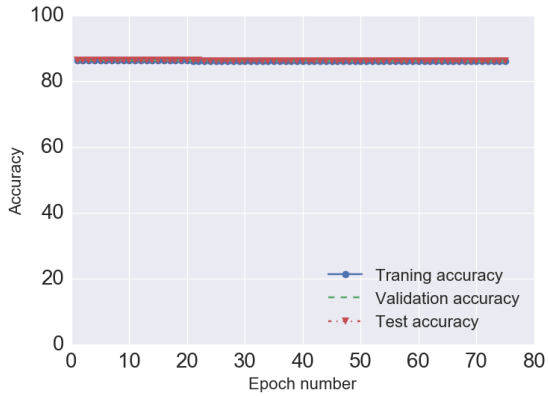
(b)

(c)

Figure 5.2: Results for the RNN model. (a) Is the AUC score plotted against the number of epochs. (b) Is the accuracy plotted as a function of the number of epochs and (c) is the loss plotted against the number of epochs.



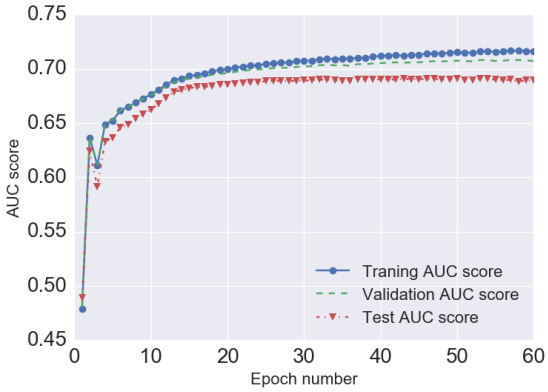
(a)



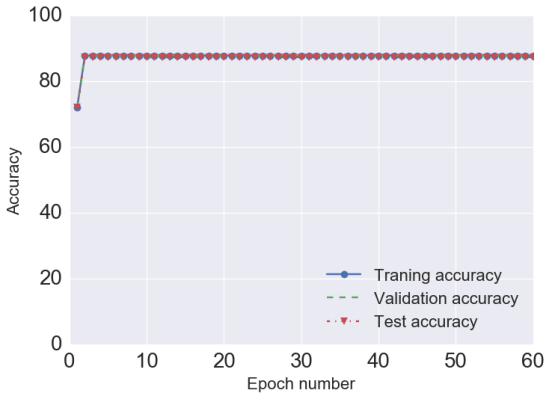
(b)

(c)

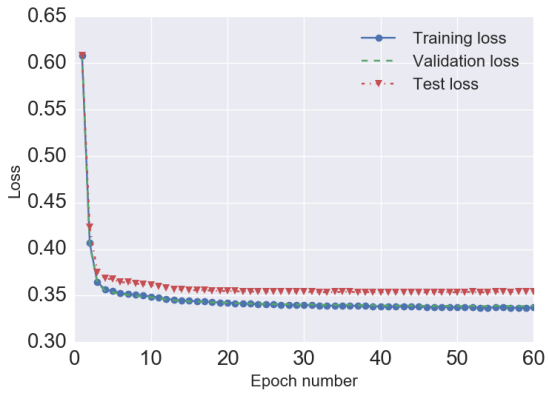
Figure 5.3: Results for the CNN model. (a) Is the AUC score plotted against the number of epochs. (b) Is the accuracy plotted as a function of the number of epochs and (c) is the loss plotted against the number of epochs.



(a)



(b)



(c)

Figure 5.4: Results for the FFNN model. (a) Is the AUC score plotted against the number of epochs. (b) Is the accuracy plotted as a function of the number of epochs and (c) is the loss plotted against the number of epochs.

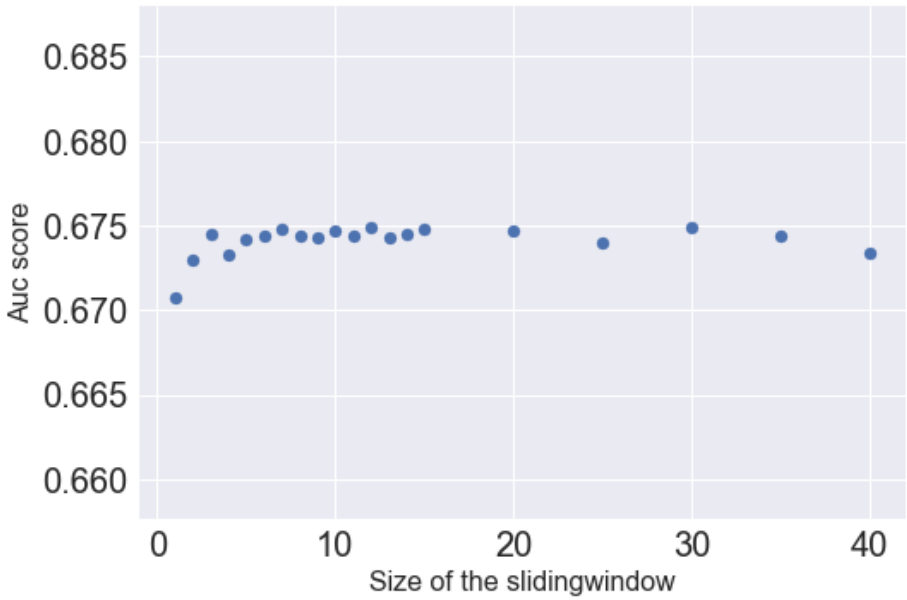


Figure 5.5: The AUC score for different lengths of the window for the Random forest model. As can be seen in the figure the model is very insensitive to the length of the sliding window. It should be noted that the performance for a window of 20 sequence steps is lower than the performance presented in Table 5.1. This is due to differences in the data used. For the comparison of the sliding window size only sequences longer than 60 is used where as for the data used in Table 5.1 uses sequences with a minimum of 20 sequence steps.

For the Random forest model, the feature importance was briefly investigated. The most importance features were found to be the previous hospitalizations parameters. The complete results form the feature importance is presented in Appendix E. That the previous hospitalizations was the most important features might indicate that it is common with repetitive hospitalizations. For the Random forest model the sliding window size was also investigated briefly, see Figure 5.5.

Chapter 6

Discussion

With an aging population, shortage of physicians and a growing number of chronic diseases the health care cost are expected to increase [Mon16]. Thus new more intelligent systems and tools are needed to keep the costs in check. Data analytics and Machine learning are one of many tools that have great potential to do that. According to the Wall Street Journal the pressure to deliver more with less resources have resulted in that insurers, pharmacy benefit managers and health-care providers are pushing the development in data analysis [Tib13]. Thus the future for machine learning and AI within health care looks very promising. By utilizing data analytics better care can be provided for a lower cost.

The investigated models showed that neural network can perform well on the problem. According to the investigated models the best performance was achieved with a Random forest with a AUC of 0.692. For the neural networks the best performance was achieved with a feedforward neural network with a AUC score of 0.689. A rough estimation of significance was done based on the error rate, see appendix A, which indicates that there might be a significant difference. However this has to be further investigated. It should be noted that the results from the kNN model could be questioned due to that only a limited amount of data was used to train the model.

Health care data are in some ways different compared to other fields. Where

decisions might be about life or death for a patient, it makes having robust algorithms very important[DS17]. Data sets for health care are also often imbalanced, since most people are healthy and some diseases are rare. The problem with the imbalance is present in this problem where the data set has an imbalance of $\sim$ 15 % positive labels and $\sim$ 85 % negative labels. Verification also becomes very important since it is a matter of peoples life and well being [DS17].

Machine learning have the advantage of outperforming humans in terms of speed and accuracy. However, the logic behind the decision are often hard to understand. This is especially a problem for neural networks where even experts are uncertain how a network draws conclusions [DC16]. The process of understanding the logic is important in terms of incorporating intelligent systems in health care due to legal liabilities and regulatory processes as well as for getting acceptance of the technology by professionals and patients [STC16]. To make mistakes is human, but machine learning algorithms also make mistakes (wrong predictions which in it self might not be classified as an error). In the interaction between humans and intelligent systems the issue of accountability becomes ambiguous and who should be held accountable is not clear [Sin16]. It has also been found that if the professionals and the patients don't understand how the data is used for predictions it may be difficult to achieve acceptance of the system [STC16]. However it could be argued that good results in combination with low risk are most important and not how the algorithms operate [Rei16].

Neural networks have reported state of the art performances in many fields. However, in comparing neural networks to other models such as Random forest and kNN, they require more knowledge and are harder to develop and optimize. For projects with a limited time the difference in results might be bigger due to better engineered features than to investigate more complex methods. To build neural networks is a craft where finding the best parameters is a trick of the trade. Using optimization tools might reduce the stepping stone. However, the optimization process is costly in terms of computational burden and the better the original model and the search space is defined the less optimization is required.

Hyperopt was initially used for hyperparamter optimization. Initial care were not taken and to high expectations was put to the software. When using hyperparamter optimization software only the final loss is evaluated which result in that only part of the models behavior is evaluated. The author used Hyperopt to early on in the process which resulted in heavily overfitting of the models when the complete behaviour was evaluated on a late stage. This resulted in that the models had to be redesigned. The conclusion from this is that hyper parameter optimization tools should be used with care on models that behave properly, in terms of train and validation losses, before automated

hyperparameter optimization is used.

For the given problem data were available from several records. This makes it hard to look over the data and also to understand the relations. Data is not only data, by understanding the problem and the relations in the data, time can be saved. Neural networks perform great on the raw input data but when the problems grow inputting all the raw data might not be the best solution. When the input data grows the required computational power also grows, in terms of memory required, the time to train the models and thus the process of optimizing the models becomes slower. Therefore the author beliefs that it is important to be critical and try to only incorporate data that is relevant to make better predictions. However by starting with understanding the problem, the data and gather domain knowledge better choices in terms of feature selection, generation and extraction can be done, resulting in higher performance, faster optimization and overall better results.

For the random forest model the size of the sliding window was investigated, see Figure 5.5. The results indicate that the performance doesn't increase for windows larger than 3 sequence steps. This behaviour is unexpected since diseases are developed over time and doctors at the clinics have found correlations between trends and hospitalizations. These trends are measured over 3 months of time which should correlate with ~ 40 sequence steps (based on three dialysis sessions a week, which is the most common). This might also be the reason for why the sliding window model outperforms the RNN model. Due to a limited amount of time this is left for future work. However it could be noticed that the behaviour of the other models might be different.

Several other similar studies where investigated in 2.9 Related work. The difference between the sliding window and the RNN models where relatively small for these investigated cases, $\sim 1 - 1.6$ %. This could indicate that even though RNN in theory should be well suited for multivariate time series the performance increase might be relatively small compared to a sliding window approach for FFNN.

Neural networks perform very well working with raw data, so in utilizing more of the available raw data, the neural networks might be able to achieve better results compare to other models with less feature engineering. For the current predictions the Random forest put a lot of importance to the engineered features, see Appendix E .

6.1 Future work

In order to further improve upon the performance of the models the obvious first step would be to utilize more of the available data. Several records were used during the feature generation process but more data is available, such as medication reacords, hospitalization reasons, demographics etc. It could also be noticed that a large amount of the generated features where never used in the developed models. The data sources are hard to combine and the author experienced that the EHR where built for storing data and not designed to be used for analyzes. This makes it harder to work with, and more work is needed to combine the records in a proper way. This would also require a better understanding of how the different records are related and how the data gathering process is done in order to utilized the data in the best possible way.

Depression and anxiety are among the most common comorbid illnesses for patients with end stage renal disease[FMRP+10]. Patients with end stage renal disease face many challenges related to their illness that increase the likelihood to develop depression or anxiety. This is related to a general feeling by patients of unwellness caused by the disease it self or by the treatment and the major disruption of the lifestyle [FMRP+10]. These disruptions includes complying with the treatments, dialysis schedules, diets, water restrictions, hospitalization, fear of disability, morbidity, and shortened lifespan [FMRP+10]. Medication given to dialysis patients might also be a potential source of psychiatric disorders, sterioides and various glomerulopathies which are sometimes given after transplantaiton have been linked to depression, mania, and psychotic symptoms [J07]. Kimmel et al [FMRP+10] showed that patients with chronic kidney diseases had to be hospitalized for psychiatric disorders 1.5 to 3 times more than individuals with other chronic diseases. To collect data about the mental health of the patients could thus also improve the predictions since potentially a large number of the hospitalizations are linked to mental health.

From a medical standpoint the data set could potentially be used for other predictions as well, such as which treatments could be recommended, optimal dialysis settings, medication amount and type of medication. Utilizing the data on other predictions could potentially improve the health care for the patients.

In terms of the developed models evenly sampled batches were briefly investigated, however this could also be done more thoroughly. Combining the different models to an ensemble could also be investigated to possible boost the performance. Also investigating which hospitalizations the models are able to predict would be interesting to understand to better know which actions should be taken for the patients that are expected to be hospitalized within the next 30 days. Padding of the sequences were done based on the length of the longest sequence

in order to have equal length of all sequences. The computational burden could have been reduced by using bucketing. Bucketing is a method to handle sequences of different lengths in an efficient way[Ten]. Bucketing uses the concept of sorting out the sequences in to "buckets" of similar length. Then padding is applied within each bucket to make all the sequences have the same sequence length as the other sequences in the bucket. Bucketing allow for a faster training and predictions since fewer unwanted predictions are made. Investigating if the models are too late or early in there predictions would also be interesting and thus maybe a better size of the prediction horizon could be developed together with the hospitals. The used sliding windows of 20 sequence step is an assumption that also should be challenged and for future work the predictions for different window sizes should be evaluated. Having large windows could have the potential to improve the predictions. However, the brief investigation of window size that was conducted indicated that longer time windows than 3 sequence steps did not improve the results for a Random forest model. Further work could also be put to handle the irregular sampling of the sequences.

Chapter 7

Conclusion

The hospitalization of dialysis patients where successfully predicted and Convolutional, Feedforward, and Recurrent neural networks successfully implemented for the problem. The best results where achieved with a Random forest model with a AUC score of 0.692. Of the neural networks the feedforward model performed the best with a AUC score of 0.689. To further improve the performance more of the available data should be utilized. In this work it has also been showed that machine learning has the potential to be a great tool within health care.

Appendices

Appendix A

Rough estimation of significance

Assuming independent test examples and binary classification. The observed test error is:

$$\epsilon_{test} = \frac{n_{wrong}}{N_{test}} \tag{A.1}$$

Assuming that the number of wrong predictions, n_{wrong} have a binomial distribution with a unknown mean of $N_{test} \cdot \epsilon$ and variance $N_{test} \cdot (1 - \epsilon)$ gives that ϵ_{test} will have a mean of:

$$\frac{N_{test} \cdot \epsilon}{N_{test}} = \epsilon \tag{A.2}$$

and variance of

$$N_{test} \cdot \epsilon \cdot \frac{(1 - \epsilon)}{N_{test}^2} = \epsilon \cdot \frac{(1 - \epsilon)}{N_{test}} \tag{A.3}$$

Which gives that

$$\epsilon_{test} \approx \epsilon \pm \sqrt{\frac{\epsilon \cdot (1-\epsilon)}{N_{test}}} \tag{A.4}$$

ϵ is unknown but is approximately replaced with the observed value for the standard deviation

$$\sqrt{\frac{\epsilon \cdot (1-\epsilon)}{N_{test}}} \approx \sqrt{\frac{\epsilon_{test} \cdot (1-\epsilon_{test})}{N_{test}}} \tag{A.5}$$

Equation A.5 is the uncertainty of the error rate and is used as a rough estimation of the error rate of the AUC score, see Table A.1. The test set consists of 1100 patients with the mean length of 296 sequence steps.

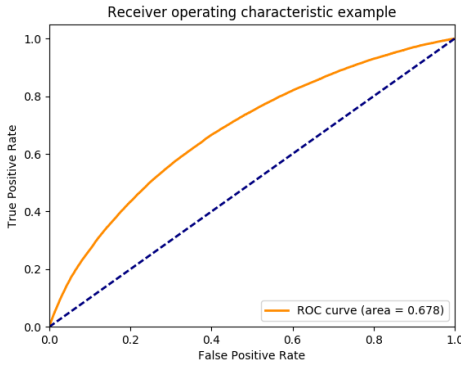
Method	AUC	AUC uncertainty
kNN	0.545	± 0.00088
Random Forest	0.692	± 0.00081
FFNN	0.689	± 0.00082
RNN	0.650	± 0.00084
RNN(LSTM)	0.675	± 0.00083
RNN(LSTM) with Attention	0.680	± 0.00082
CNN	0.678	± 0.00082

Table A.1: The AUC score and uncertainty for the AUC score.

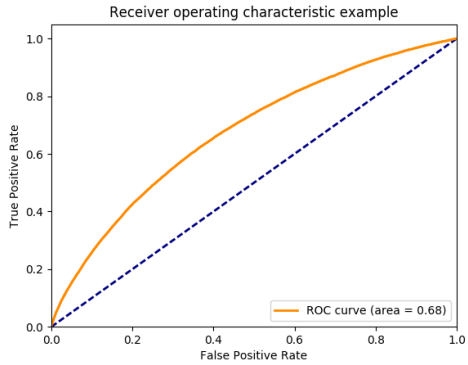
Appendix B

Receiver operation characteristics curves

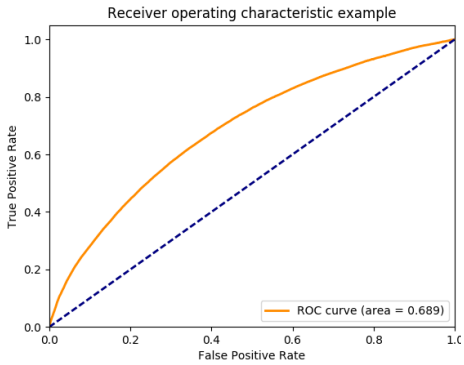
In Figure 5.1 the ROC curves are plotted together. In Figure B.1 the ROC curves are plotted individually.



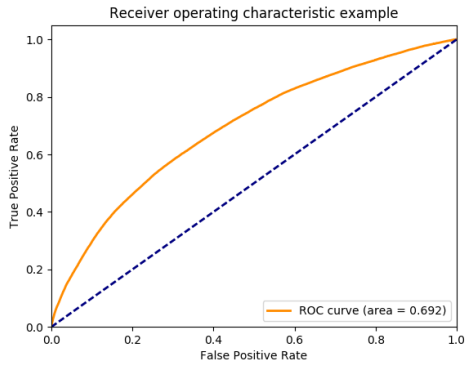
(a) Receiver operating characteristic curve for the CNN model.



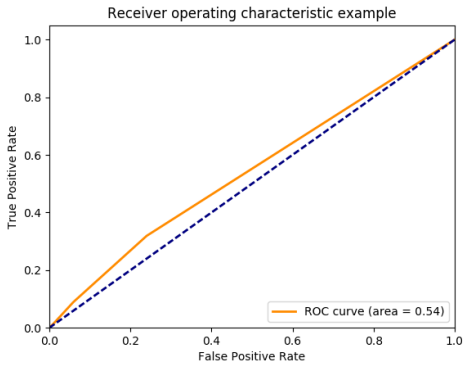
(b) Receiver operating characteristic curve for the RNN model.



(c) Receiver operating characteristic curve for the FFNN model.



(d) Receiver operating characteristic curve for the Random forest model.



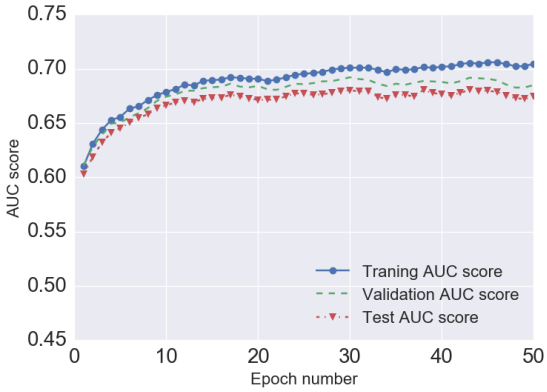
(e) Receiver operating characteristic curve for the kNN model.

Figure B.1: The receiver operating characteristic curves for the different developed models.

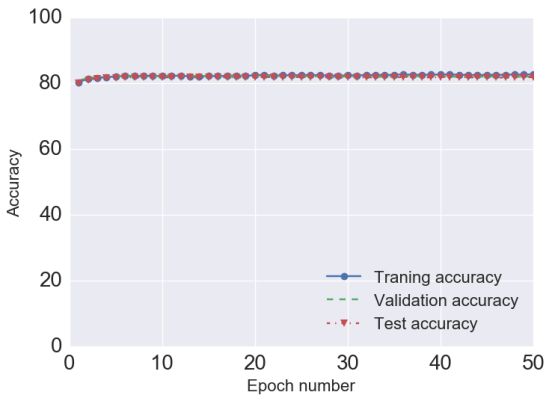
Appendix C

RNN model without early stopping

For the RNN model early stopping was implemented. Early stopping found the optimal performance to be achieved after 30 epochs. The results for 50 epochs are presented in Figure C.1. The overfitting of the model is visible after 30 epochs in plot **(c)** in Figure C.1.



(a)



(b)

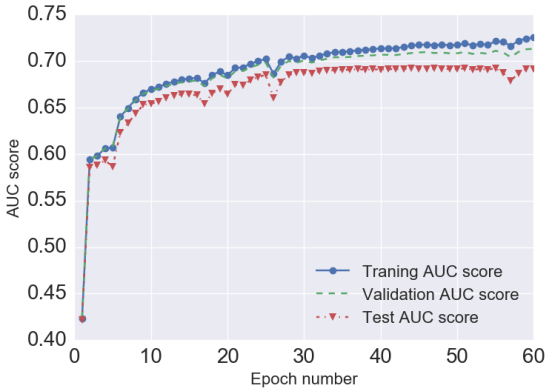
(c)

Figure C.1: This is the result from the RNN model when early stopping is not implemented and the model is allowed to run for 50 epochs. (a) Is the AUC score plotted against the number of epochs. (b) Is the accuracy plotted as a function of the number of epochs and (c) is the loss plotted against the number of epochs.

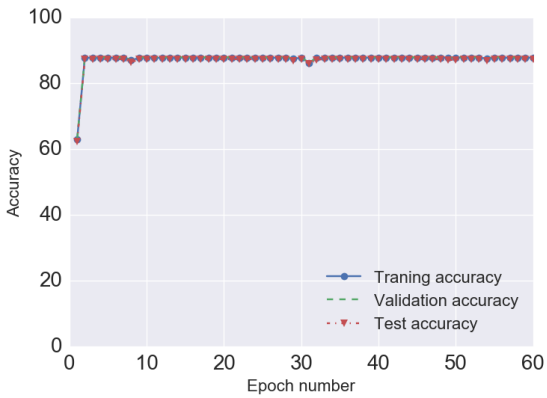
Appendix D

Results for alternative feedforward model

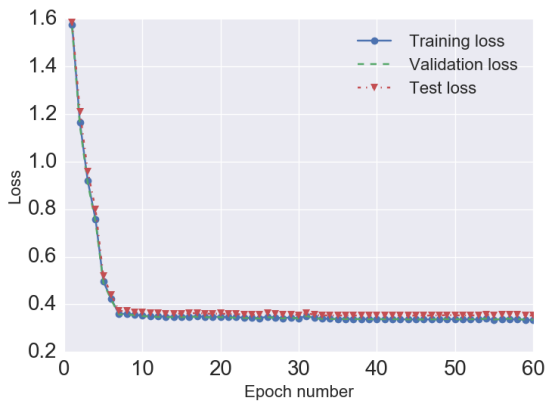
In comparison to the final feedforward neural network a larger model in terms of number of trainable parameters was developed. The results from this model is presented in Figure D.1. The smaller model was selected since the validation performance was equivalent and the computational cost was far less due to the reduced number of parameters.



(a)



(b)



(c)

Figure D.1: This is the result from the larger feedforward neural network. The model was redeveloped due to its large size and was for the final model shrunken from $\sim 400\,000$ to $\sim 20\,000$ trainable parameters. (a) Is the AUC score plotted against the number of epochs. (b) Is the accuracy plotted as a function of the number of epochs and (c) is the loss plotted against the number of epochs.

Appendix E

Feature importance for Random forest

The feature importance for the Random forest model was evaluated based on the built in function for feature importance in sci-kit learn, see Figure E.1a and E.1b. The Random forest model used the sliding window approach. The size of the sliding window is 20 sequence steps where each sequence step have 31 features, which gives a total of 620 features. The latest sequence step has the highest feature numbers, see Table E.1.

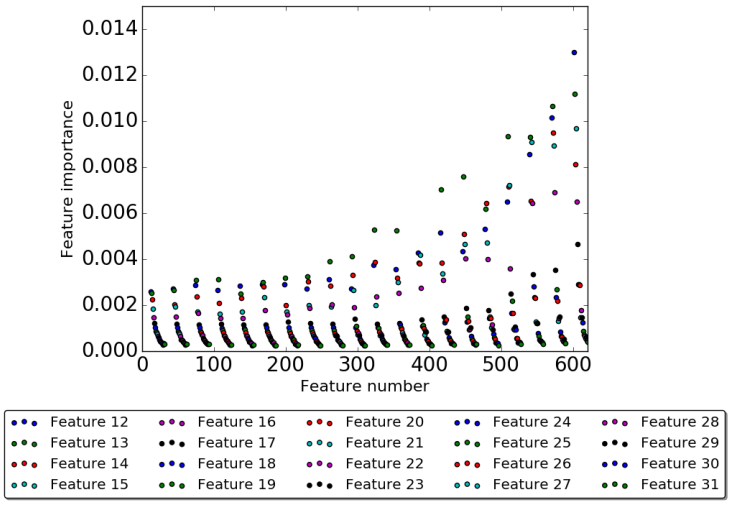
Time step	**Feature numbers**
t	588-619
t-1	557-588
⋮	⋮
t-19	0-30

Table E.1: Table that describes how the feature numbers are asserted.

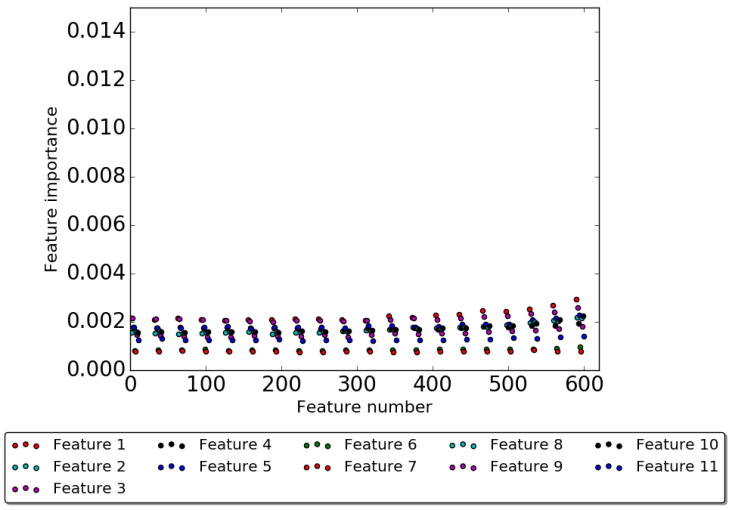
The 31 feature at each time step are explained in Table E.2.

Feature	Represents
Feature 1 at time t	Average blood flow
Feature 2 at time t	Start sitting pulse
Feature 3 at time t	Liters processed
Feature 4 at time t	Time in Dialysis
Feature 5 at time t	Fluid removed
Feature 6 at time t	Patient temp start
Feature 7 at time t	Patient temp end
Feature 8 at time t	Start sitting systolic Bp
Feature 9 at time t	Start sitting diastolic Bp
Feature 10 at time t	End sitting systolic Bp
Feature 11 at time t	End sitting diastolic Bp
Feature 12 at time t	Days since last hospitalization
Feature 13 at time t	Days since second last hospitalization
Feature 14 at time t	Days since third last hospitalization
Feature 15 at time t	Days since fourth last hospitalization
Feature 16 at time t	Days since fifth last hospitalization
Feature 17 at time t	Days since sixth last hospitalization
Feature 18 at time t	Days since seventh last hospitalization
Feature 19 at time t	Days since eighth last hospitalization
Feature 20 at time t	Days since ninth last hospitalization
Feature 21 at time t	Days since tenth last hospitalization
Feature 22 at time t	Days since eleventh last hospitalization
Feature 23 at time t	Days since twelfth last hospitalization
Feature 24 at time t	Days since thirteenth last hospitalization
Feature 25 at time t	Days since fourteenth last hospitalization
Feature 26 at time t	Days since fifteenth last hospitalization
Feature 27 at time t	Days since sixteenth last hospitalization
Feature 28 at time t	Days since seventeenth last hospitalization
Feature 29 at time t	Days since eighteenth last hospitalization
Feature 30 at time t	Days since nineteenth last hospitalization
Feature 31 at time t	Days since twentieth last hospitalization

Table E.2: The features and how they are numbered.



(a)



(b)

Figure E.1: (a) The feature importance for the 20 last hospitalization, for a window of 20 sequence steps. (b) The feature importance for the 11 continuously measured variables for the same window. The feature importance is plotted in two graphs for easier comparison of the 11 continues variables and the days since the previous hospitalizations. It should be noticed that the Random forest is trained based on all 31 features for a window size of 20 sequence steps.

The feature importance is presented in the table below. The most important features are the feature about previous hospitalizations.

1. feature 600 (0.013004)
2. feature 601 (0.011178)
3. feature 570 (0.010660)
4. feature 569 (0.010153)
5. feature 603 (0.009689)
6. feature 571 (0.009510)
7. feature 508 (0.009338)
8. feature 539 (0.009323)
9. feature 541 (0.009079)
10. feature 572 (0.008934)
11. feature 538 (0.008558)
12. feature 602 (0.008117)
13. feature 446 (0.007592)
14. feature 510 (0.007223)
15. feature 509 (0.007161)
16. feature 415 (0.007036)
17. feature 573 (0.006903)
18. feature 540 (0.006525)
19. feature 507 (0.006498)
20. feature 604 (0.006491)
21. feature 478 (0.006450)
22. feature 542 (0.006448)
23. feature 477 (0.006178)
24. feature 476 (0.005319)
25. feature 322 (0.005279)
26. feature 353 (0.005253)
27. feature 414 (0.005156)
28. feature 447 (0.005089)
29. feature 479 (0.004733)
30. feature 605 (0.004668)
31. feature 448 (0.004662)
32. feature 445 (0.004349)
33. feature 383 (0.004294)
34. feature 386 (0.004182)
35. feature 291 (0.004116)
36. feature 449 (0.004031)
37. feature 480 (0.003996)
38. feature 260 (0.003912)
39. feature 323 (0.003889)
40. feature 416 (0.003859)
41. feature 384 (0.003850)
42. feature 385 (0.003807)
43. feature 321 (0.003756)
44. feature 511 (0.003591)
45. feature 352 (0.003550)
46. feature 574 (0.003532)
47. feature 417 (0.003378)
48. feature 543 (0.003329)
49. feature 292 (0.003302)
50. feature 229 (0.003242)
51. feature 198 (0.003190)
52. feature 354 (0.003175)
53. feature 105 (0.003140)
54. feature 259 (0.003125)
55. feature 74 (0.003097)
56. feature 418 (0.003084)
57. feature 230 (0.003018)
58. feature 355 (0.003015)
59. feature 167 (0.003002)
60. feature 589 (0.002924)
61. feature 606 (0.002915)
62. feature 197 (0.002903)
63. feature 166 (0.002902)
64. feature 607 (0.002891)
65. feature 73 (0.002875)
66. feature 608 (0.002865)
67. feature 261 (0.002840)
68. feature 135 (0.002828)
69. feature 544 (0.002808)
70. feature 168 (0.002802)
71. feature 387 (0.002754)
72. feature 42 (0.002733)
73. feature 228 (0.002732)
74. feature 290 (0.002707)
75. feature 576 (0.002699)
76. feature 558 (0.002674)
77. feature 43 (0.002669)
78. feature 293 (0.002663)
79. feature 104 (0.002661)
80. feature 11 (0.002586)
81. feature 591 (0.002582)
82. feature 527 (0.002546)

83. feature 356 (0.002528)
84. feature 12 (0.002521)
85. feature 512 (0.002487)
86. feature 136 (0.002486)
87. feature 465 (0.002460)
88. feature 496 (0.002443)
89. feature 560 (0.002416)
90. feature 75 (0.002375)
91. feature 325 (0.002368)
92. feature 545 (0.002355)
93. feature 575 (0.002353)
94. feature 529 (0.002336)
95. feature 169 (0.002334)
96. feature 434 (0.002326)
97. feature 137 (0.002324)
98. feature 546 (0.002315)
99. feature 593 (0.002285)
100. feature 403 (0.002282)
101. feature 498 (0.002263)
102. feature 13 (0.002250)
103. feature 598 (0.002247)
104. feature 341 (0.002243)
105. feature 467 (0.002229)
106. feature 577 (0.002187)
107. feature 372 (0.002186)
108. feature 514 (0.002184)
109. feature 590 (0.002172)
110. feature 596 (0.002171)
111. feature 436 (0.002166)
112. feature 374 (0.002165)
113. feature 0 (0.002165)
114. feature 2 (0.002163)
115. feature 62 (0.002158)
116. feature 562 (0.002157)
117. feature 64 (0.002134)
118. feature 217 (0.002134)
119. feature 248 (0.002122)
120. feature 33 (0.002121)
121. feature 567 (0.002109)
122. feature 531 (0.002108)
123. feature 343 (0.002101)
124. feature 31 (0.002099)
125. feature 95 (0.002096)
126. feature 93 (0.002093)
127. feature 186 (0.002093)
128. feature 279 (0.002089)
129. feature 405 (0.002089)
130. feature 106 (0.002088)
131. feature 155 (0.002085)
132. feature 310 (0.002073)
133. feature 250 (0.002068)
134. feature 126 (0.002062)
135. feature 124 (0.002060)
136. feature 312 (0.002059)
137. feature 559 (0.002048)
138. feature 219 (0.002048)
139. feature 44 (0.002041)
140. feature 565 (0.002039)
141. feature 157 (0.002038)
142. feature 281 (0.002024)
143. feature 263 (0.002018)
144. feature 534 (0.002015)
145. feature 188 (0.002006)
146. feature 324 (0.001990)
147. feature 231 (0.001989)
148. feature 199 (0.001987)
149. feature 528 (0.001961)
150. feature 536 (0.001948)
151. feature 262 (0.001931)
152. feature 45 (0.001929)
153. feature 592 (0.001929)
154. feature 469 (0.001920)
155. feature 438 (0.001908)
156. feature 500 (0.001905)
157. feature 294 (0.001901)
158. feature 232 (0.001889)
159. feature 497 (0.001877)
160. feature 450 (0.001873)
161. feature 505 (0.001858)
162. feature 530 (0.001852)
163. feature 503 (0.001845)
164. feature 14 (0.001843)
165. feature 474 (0.001836)
166. feature 345 (0.001833)
167. feature 561 (0.001831)
168. feature 314 (0.001830)
169. feature 407 (0.001823)
170. feature 466 (0.001820)

171. feature 468 (0.001810)
172. feature 597 (0.001809)
173. feature 128 (0.001800)
174. feature 435 (0.001796)
175. feature 221 (0.001795)
176. feature 376 (0.001791)
177. feature 499 (0.001790)
178. feature 170 (0.001785)
179. feature 472 (0.001785)
180. feature 97 (0.001784)
181. feature 481 (0.001783)
182. feature 441 (0.001776)
183. feature 252 (0.001773)
184. feature 127 (0.001771)
185. feature 610 (0.001771)
186. feature 3 (0.001768)
187. feature 65 (0.001768)
188. feature 190 (0.001768)
189. feature 4 (0.001767)
190. feature 373 (0.001766)
191. feature 437 (0.001765)
192. feature 283 (0.001763)
193. feature 66 (0.001759)
194. feature 404 (0.001759)
195. feature 34 (0.001758)
196. feature 412 (0.001758)
197. feature 96 (0.001755)
198. feature 375 (0.001749)
199. feature 35 (0.001746)
200. feature 443 (0.001743)
201. feature 313 (0.001740)
202. feature 158 (0.001740)
203. feature 189 (0.001738)
204. feature 220 (0.001735)
205. feature 251 (0.001726)
206. feature 76 (0.001724)
207. feature 379 (0.001715)
208. feature 138 (0.001715)
209. feature 566 (0.001715)
210. feature 344 (0.001709)
211. feature 159 (0.001708)
212. feature 410 (0.001707)
213. feature 200 (0.001705)
214. feature 381 (0.001699)
215. feature 342 (0.001698)
216. feature 350 (0.001691)
217. feature 406 (0.001681)
218. feature 311 (0.001670)
219. feature 282 (0.001661)
220. feature 319 (0.001658)
221. feature 535 (0.001657)
222. feature 348 (0.001654)
223. feature 77 (0.001651)
224. feature 513 (0.001649)
225. feature 515 (0.001646)
226. feature 107 (0.001635)
227. feature 317 (0.001633)
228. feature 226 (0.001625)
229. feature 38 (0.001619)
230. feature 280 (0.001618)
231. feature 504 (0.001614)
232. feature 288 (0.001612)
233. feature 133 (0.001603)
234. feature 162 (0.001602)
235. feature 7 (0.001600)
236. feature 156 (0.001598)
237. feature 201 (0.001596)
238. feature 473 (0.001595)
239. feature 286 (0.001593)
240. feature 102 (0.001591)
241. feature 193 (0.001589)
242. feature 257 (0.001587)
243. feature 40 (0.001586)
244. feature 255 (0.001583)
245. feature 164 (0.001581)
246. feature 131 (0.001579)
247. feature 224 (0.001576)
248. feature 9 (0.001570)
249. feature 249 (0.001570)
250. feature 218 (0.001560)
251. feature 71 (0.001560)
252. feature 1 (0.001560)
253. feature 195 (0.001559)
254. feature 125 (0.001557)
255. feature 100 (0.001556)
256. feature 411 (0.001543)
257. feature 69 (0.001542)
258. feature 32 (0.001542)

259. feature 442 (0.001538)
260. feature 349 (0.001524)
261. feature 94 (0.001522)
262. feature 380 (0.001511)
263. feature 63 (0.001509)
264. feature 46 (0.001508)
265. feature 580 (0.001507)
266. feature 187 (0.001505)
267. feature 452 (0.001505)
268. feature 419 (0.001490)
269. feature 483 (0.001487)
270. feature 611 (0.001473)
271. feature 609 (0.001466)
272. feature 8 (0.001464)
273. feature 256 (0.001457)
274. feature 15 (0.001454)
275. feature 318 (0.001453)
276. feature 484 (0.001448)
277. feature 579 (0.001447)
278. feature 139 (0.001440)
279. feature 108 (0.001440)
280. feature 482 (0.001438)
281. feature 163 (0.001437)
282. feature 39 (0.001432)
283. feature 295 (0.001416)
284. feature 225 (0.001412)
285. feature 70 (0.001404)
286. feature 599 (0.001403)
287. feature 194 (0.001399)
288. feature 132 (0.001389)
289. feature 101 (0.001389)
290. feature 388 (0.001385)
291. feature 568 (0.001377)
292. feature 287 (0.001371)
293. feature 422 (0.001361)
294. feature 421 (0.001336)
295. feature 506 (0.001334)
296. feature 453 (0.001324)
297. feature 578 (0.001322)
298. feature 537 (0.001322)
299. feature 41 (0.001314)
300. feature 547 (0.001294)
301. feature 451 (0.001289)
302. feature 475 (0.001288)
303. feature 202 (0.001281)
304. feature 444 (0.001274)
305. feature 196 (0.001272)
306. feature 413 (0.001263)
307. feature 134 (0.001260)
308. feature 10 (0.001257)
309. feature 72 (0.001253)
310. feature 420 (0.001252)
311. feature 351 (0.001251)
312. feature 258 (0.001251)
313. feature 165 (0.001247)
314. feature 612 (0.001247)
315. feature 103 (0.001243)
316. feature 382 (0.001239)
317. feature 289 (0.001238)
318. feature 357 (0.001233)
319. feature 549 (0.001232)
320. feature 227 (0.001231)
321. feature 233 (0.001219)
322. feature 16 (0.001216)
323. feature 548 (0.001213)
324. feature 320 (0.001213)
325. feature 326 (0.001202)
326. feature 47 (0.001196)
327. feature 264 (0.001183)
328. feature 109 (0.001177)
329. feature 358 (0.001157)
330. feature 78 (0.001150)
331. feature 171 (0.001147)
332. feature 486 (0.001147)
333. feature 140 (0.001141)
334. feature 390 (0.001138)
335. feature 389 (0.001106)
336. feature 297 (0.001098)
337. feature 518 (0.001071)
338. feature 79 (0.001045)
339. feature 203 (0.001045)
340. feature 296 (0.001042)
341. feature 485 (0.001040)
342. feature 456 (0.001038)
343. feature 172 (0.001038)
344. feature 234 (0.001032)
345. feature 327 (0.001027)
346. feature 17 (0.001026)

347. feature 359 (0.001025)
348. feature 48 (0.001024)
349. feature 265 (0.001016)
350. feature 110 (0.001011)
351. feature 141 (0.001010)
352. feature 391 (0.001007)
353. feature 328 (0.000991)
354. feature 455 (0.000970)
355. feature 594 (0.000968)
356. feature 360 (0.000964)
357. feature 516 (0.000956)
358. feature 517 (0.000951)
359. feature 487 (0.000936)
360. feature 454 (0.000931)
361. feature 142 (0.000930)
362. feature 519 (0.000923)
363. feature 266 (0.000910)
364. feature 563 (0.000910)
365. feature 235 (0.000894)
366. feature 111 (0.000891)
367. feature 49 (0.000890)
368. feature 393 (0.000889)
369. feature 470 (0.000889)
370. feature 532 (0.000888)
371. feature 439 (0.000887)
372. feature 173 (0.000886)
373. feature 18 (0.000883)
374. feature 204 (0.000880)
375. feature 80 (0.000879)
376. feature 501 (0.000878)
377. feature 98 (0.000871)
378. feature 408 (0.000871)
379. feature 613 (0.000869)
380. feature 423 (0.000868)
381. feature 298 (0.000866)
382. feature 67 (0.000857)
383. feature 377 (0.000854)
384. feature 329 (0.000847)
385. feature 36 (0.000843)
386. feature 160 (0.000842)
387. feature 315 (0.000842)
388. feature 425 (0.000837)
389. feature 394 (0.000835)
390. feature 581 (0.000833)
391. feature 284 (0.000832)
392. feature 346 (0.000832)
393. feature 191 (0.000830)
394. feature 533 (0.000829)
395. feature 143 (0.000827)
396. feature 129 (0.000825)
397. feature 5 (0.000822)
398. feature 112 (0.000821)
399. feature 50 (0.000814)
400. feature 222 (0.000814)
401. feature 424 (0.000812)
402. feature 19 (0.000812)
403. feature 174 (0.000812)
404. feature 81 (0.000811)
405. feature 68 (0.000811)
406. feature 236 (0.000809)
407. feature 361 (0.000807)
408. feature 253 (0.000806)
409. feature 564 (0.000796)
410. feature 267 (0.000792)
411. feature 595 (0.000786)
412. feature 300 (0.000785)
413. feature 409 (0.000784)
414. feature 502 (0.000783)
415. feature 192 (0.000782)
416. feature 205 (0.000781)
417. feature 6 (0.000777)
418. feature 550 (0.000777)
419. feature 130 (0.000776)
420. feature 161 (0.000773)
421. feature 316 (0.000773)
422. feature 99 (0.000772)
423. feature 37 (0.000772)
424. feature 285 (0.000771)
425. feature 471 (0.000767)
426. feature 440 (0.000767)
427. feature 378 (0.000765)
428. feature 223 (0.000760)
429. feature 347 (0.000759)
430. feature 254 (0.000745)
431. feature 331 (0.000744)
432. feature 20 (0.000744)
433. feature 488 (0.000741)
434. feature 51 (0.000740)

435. feature 82 (0.000729)
436. feature 299 (0.000728)
437. feature 362 (0.000726)
438. feature 175 (0.000721)
439. feature 237 (0.000721)
440. feature 614 (0.000713)
441. feature 392 (0.000707)
442. feature 113 (0.000700)
443. feature 301 (0.000699)
444. feature 144 (0.000694)
445. feature 269 (0.000693)
446. feature 83 (0.000689)
447. feature 330 (0.000689)
448. feature 268 (0.000688)
449. feature 332 (0.000677)
450. feature 207 (0.000671)
451. feature 363 (0.000666)
452. feature 582 (0.000666)
453. feature 270 (0.000664)
454. feature 551 (0.000664)
455. feature 615 (0.000659)
456. feature 238 (0.000658)
457. feature 457 (0.000657)
458. feature 206 (0.000657)
459. feature 145 (0.000654)
460. feature 21 (0.000650)
461. feature 176 (0.000650)
462. feature 52 (0.000648)
463. feature 114 (0.000648)
464. feature 617 (0.000629)
465. feature 583 (0.000623)
466. feature 489 (0.000603)
467. feature 239 (0.000597)
468. feature 177 (0.000591)
469. feature 22 (0.000589)
470. feature 426 (0.000581)
471. feature 84 (0.000572)
472. feature 146 (0.000567)
473. feature 115 (0.000564)
474. feature 616 (0.000558)
475. feature 521 (0.000558)
476. feature 208 (0.000556)
477. feature 520 (0.000549)
478. feature 240 (0.000549)
479. feature 395 (0.000547)
480. feature 364 (0.000546)
481. feature 302 (0.000541)
482. feature 333 (0.000532)
483. feature 586 (0.000532)
484. feature 458 (0.000530)
485. feature 490 (0.000517)
486. feature 53 (0.000516)
487. feature 116 (0.000515)
488. feature 147 (0.000514)
489. feature 210 (0.000510)
490. feature 365 (0.000510)
491. feature 552 (0.000508)
492. feature 584 (0.000504)
493. feature 23 (0.000501)
494. feature 271 (0.000497)
495. feature 85 (0.000495)
496. feature 178 (0.000494)
497. feature 54 (0.000491)
498. feature 427 (0.000489)
499. feature 428 (0.000480)
500. feature 272 (0.000478)
501. feature 396 (0.000475)
502. feature 179 (0.000475)
503. feature 459 (0.000474)
504. feature 55 (0.000473)
505. feature 86 (0.000473)
506. feature 334 (0.000469)
507. feature 117 (0.000466)
508. feature 618 (0.000458)
509. feature 553 (0.000457)
510. feature 209 (0.000455)
511. feature 303 (0.000453)
512. feature 554 (0.000448)
513. feature 24 (0.000445)
514. feature 585 (0.000443)
515. feature 241 (0.000440)
516. feature 366 (0.000440)
517. feature 242 (0.000433)
518. feature 397 (0.000426)
519. feature 118 (0.000425)
520. feature 273 (0.000423)
521. feature 523 (0.000420)
522. feature 555 (0.000417)

523. feature 148 (0.000417)
524. feature 335 (0.000414)
525. feature 56 (0.000413)
526. feature 304 (0.000412)
527. feature 211 (0.000411)
528. feature 180 (0.000407)
529. feature 87 (0.000406)
530. feature 619 (0.000406)
531. feature 149 (0.000405)
532. feature 25 (0.000404)
533. feature 524 (0.000403)
534. feature 274 (0.000399)
535. feature 26 (0.000391)
536. feature 491 (0.000389)
537. feature 556 (0.000387)
538. feature 181 (0.000387)
539. feature 57 (0.000384)
540. feature 367 (0.000383)
541. feature 587 (0.000383)
542. feature 461 (0.000381)
543. feature 182 (0.000381)
544. feature 522 (0.000380)
545. feature 368 (0.000376)
546. feature 243 (0.000376)
547. feature 150 (0.000373)
548. feature 430 (0.000373)
549. feature 525 (0.000371)
550. feature 336 (0.000370)
551. feature 88 (0.000367)
552. feature 119 (0.000366)
553. feature 492 (0.000360)
554. feature 460 (0.000359)
555. feature 213 (0.000357)
556. feature 58 (0.000354)
557. feature 494 (0.000353)
558. feature 244 (0.000353)
559. feature 429 (0.000351)
560. feature 275 (0.000351)
561. feature 89 (0.000346)
562. feature 27 (0.000345)
563. feature 588 (0.000343)
564. feature 306 (0.000343)
565. feature 398 (0.000339)
566. feature 399 (0.000339)
567. feature 493 (0.000339)
568. feature 337 (0.000335)
569. feature 120 (0.000333)
570. feature 151 (0.000330)
571. feature 29 (0.000330)
572. feature 305 (0.000329)
573. feature 431 (0.000328)
574. feature 60 (0.000327)
575. feature 462 (0.000323)
576. feature 557 (0.000323)
577. feature 432 (0.000323)
578. feature 338 (0.000321)
579. feature 91 (0.000320)
580. feature 212 (0.000319)
581. feature 400 (0.000317)
582. feature 61 (0.000312)
583. feature 30 (0.000311)
584. feature 92 (0.000306)
585. feature 526 (0.000305)
586. feature 464 (0.000303)
587. feature 307 (0.000301)
588. feature 245 (0.000300)
589. feature 90 (0.000298)
590. feature 123 (0.000296)
591. feature 121 (0.000286)
592. feature 370 (0.000286)
593. feature 154 (0.000286)
594. feature 216 (0.000285)
595. feature 463 (0.000284)
596. feature 122 (0.000284)
597. feature 369 (0.000282)
598. feature 59 (0.000282)
599. feature 28 (0.000281)
600. feature 371 (0.000281)
601. feature 277 (0.000279)
602. feature 340 (0.000276)
603. feature 276 (0.000276)
604. feature 339 (0.000274)
605. feature 183 (0.000274)
606. feature 153 (0.000272)
607. feature 433 (0.000271)
608. feature 214 (0.000269)
609. feature 401 (0.000268)
610. feature 184 (0.000264)

611. feature 152 (0.000264)
612. feature 495 (0.000261)
613. feature 215 (0.000259)
614. feature 309 (0.000258)
615. feature 278 (0.000256)
616. feature 402 (0.000256)
617. feature 246 (0.000256)
618. feature 308 (0.000255)
619. feature 247 (0.000253)
620. feature 185 (0.000246)

Bibliography

[AM16] Williams Adam McCarthy, Christopher K.I. Predicting patient state-of-health using sliding window and reccurent classifiers. *arXiv*, eprint arXiv:1612.00662, December 2016.

[BBBK11] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc., 2011.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[Bri] Denny Britz. Attention and memory in deep learning and nlp – wildml. `http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/`. Accessed in Feb, 2017.

[BSF94] 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, Mar 1994.

[CBS15] Edward Choi, Mohammad Taha Bahadori, and Jimeng Sun. Doctor AI: predicting clinical events via recurrent neural networks. *CoRR*, abs/1511.05942, 2015.

[CKL+15] Zhengping Che, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. Deep computational phenotyping. In *Proceedings of the 21th ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining, KDD '15, pages 507–516, New York, NY, USA, 2015. ACM.

[DC16] Nature Davide Castelvecchi. Can we open the black box of ai? `http://www.nature.com/news/can-we-open-the-black-box-of-ai-1.20731`, oct 2016. Accessed in May, 2017.

[DS17] Massachusetts Institute of Technology David Sontag. Machine learning for healthcare, lecture 1 : What makes healthcare unique?, jan 2017.

[EC16] Walter F Stewart Jimeng Sun Edward Choi, Andy Schuetz. Using reccurent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association*, 292, 2016.

[ESM17] Steven Escaravage Ernest Sohn, Joachim Roski and Kevin Maloy. Four lessons in the adoption of machine learning in health care. *Health Affairs blog*, 2017.

[FMRP+10] U. Feroze, D. Martin, A. Reina-Patton, K. Kalantar-Zadeh, and J. D. Kopple. Mental health, depression, and anxiety in patients on maintenance dialysis. *Iran J Kidney Dis*, 4(3):173–180, Jul 2010.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[Gra12] Alex Graves. *TSupervised Sequence Labelling with Recurrent Neural Networks*. Springer-Verlag Berlin Heidelberg, 2012. An optional note.

[HCC+14] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[Hyp17] Hyperopt. What is hyperopt? `https://www.hyperopt.github.io/hyperopt`, 2017. Accessed in April, 2017.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[J07] Butler J. Handbook of liaison psychiatry. 1st:p.506–26, 2007.

[Jaa17] Laura C. Bernard G. Jaar. Preventing repeat hospitalizations in dialysis patients: a call for action. 76(3):249–251, 2017.

[JB12] Yoshua Bengio James Bergstra. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13 (2012) 281-305, Submitted 3/11; Revised 9/11; Published 2/12.

[JB13] David D. Cox James Bergstra, Dan Yamins. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. *PYTHON IN SCIENCE CONF.*, PROC. OF THE 12th, (SCIPY 2013).

[JH17] PhD; Talisha Searcy MPA MA; Vaishali Patel PhD MPH JaWanna Henry, MPH; Yuriy Pylypchuk. Adoption of electronic health record systems among u.s. non-federal acute care hospitals: 2008-2015. ONC Data Brief 35, 2017.

[Kar15] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`, May 21, 2015.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[KDG15] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *CoRR*, abs/1507.01526, 2015.

[LKEW15] Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wetzel. Learning to diagnose with LSTM recurrent neural networks. *CoRR*, abs/1511.03677, 2015.

[MA17] Erik Gillsbro Mattias Albinsson. Imputation methods in dialysis data. `http://www.lunduniversity.lu.se/lup/publication/8912742`, May,2017. Accessed June, 2017.

[Mon16] Mike Montgomery. The future of health care is in data analytics. `https://www.forbes.com/sites/mikemontgomery/2016/10/26/the-future-of-health-care-is-in-data-analytics/#661a01323ee2`, Oct 2016. Accessed June,2017.

[Nata] Glomerular filtration rate (gfr). `https://www.kidney.org/atoz/content/gfr`. Accessed in May, 2017.

[Natb] A to z health guide, national kidney foundation. `https://www.kidney.org/atoz/content/dialysisinfo`. Accessed in March, 2017.

[Nie] Michael Nielsen. Neural networks and deep learning. http://neuralnetworksanddeeplearning.com. Accessed in May 2017.

[Nie17a] Michael Nielsen. How the backpropagation algorithm works. http://neuralnetworksanddeeplearning.com/chap2.html, jan 2017. Accessed in Feb, 2017.

[Nie17b] Michael Nielsen. Introducing convolutional networks. http://neuralnetworksanddeeplearning.com/chap6.html, jan 2017. Accessed in March, 2017.

[Nie17c] Michael Nielsen. Why are deep neural networks hard to train? http://neuralnetworksanddeeplearning.com/chap5.html, jan 2017. Accessed in March,2017.

[NYH15] Peter Andras Lynn Rochester Richard Walker Thomas Ploetz Nils Yannick Hammerla, James Fisher. Pd disease state assessment in naturalistic environments using deep learning. *AAAI Publications,*, Twenty-Ninth AAAI Conference on Artificial Intelligence, February 18, 2015.

[Ola] Christopher Olah. Understanding lstm networks – colah's blog. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed in Feb, 2017.

[PJ09] Laura C. Plantinga and Bernard G. Jaar. Preventing repeat hospitalizations in dialysis patients: a call for action. *Kidney International*, 76(3):249 – 251, 2009.

[Rei16] Matthew J Reid. Black-box machine learning: implications for healthcare. https://www.polygeia.com/single-post/2017/04/06/Black-box-machine-learning-implications-for-healthcare, april 2016. Accessed in April,2017.

[Rud] Sebastian Ruder. An overview of gradient descent optimization algorithms. http://sebastianruder.com/optimizing-gradient-descent/index.html#gradientdescentvariants. Accessed in Feb, 2017.

[Rud16] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[S0216] Us renal data system 2015 annual data report: Epidemiology of kidney disease in the united states. *American Journal of Kidney Diseases*, 67(Supplement 1):A4, 2016.

[SFW17] Joe Kai Jonathan M. Garibaldi Nadeem Qureshi Stephen F. Weng, Jenna Reps. Can machine-learning improve cardiovascular risk prediction using routine clinical data? *PLoS ONE*, 12(4), April 4, 2017.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[Sin16] Jatinder Singh. The tech-legal aspects of machine learning:considerations for moving forward. 2016.

[STC16] Science and UK parliament Technology Committee, House of Commons. Science and technology committee (house of commons), report on robotics and artificial intelligence, 2016.

[TAL13] Mia A. Levy Thomas A. Lasko, Joshua C. Denny. Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data. *PLOS ONE*, 8(8)), June 24, 2013.

[TBP15] P. T. Teo, N. Bruce, and S. Pistorius. *Application and Parametric Studies of a Sliding Window Neural Network for Respiratory Motion Predictions of Lung Cancer Patients*, pages 595–598. Springer International Publishing, Cham, 2015.

[Ten] Tensorflow. Sequence-to-sequence model. `https://www.tensorflow.org/tutorials/seq2seq#bucketing_and_padding`. Accessed in April,2017.

[TH16] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning, Datamining, Inference, and prediction*, volume Second Edition. Springer Series in Statistics, 2016.

[Tib13] Shara Tibken. Numbers, numbers and more numbers. `https://www.wsj.com/articles/SB10001424052702304692804577285821129341442`, Feb. 14, 2013. Accessed in May, 2017.