VRIJE UNIVERSITEIT AMSTERDAM

RESEARCH PAPER MSC. BUSINESS ANALYTICS

Predicting drug serum concentrations using a Long Short-Term Memory network

Vancomycin Therapeutic Drug Monitoring

Author:
D.J.M. VAN WEERDENBURG
(2554298)

Supervisor:
Dr. M. HOOGENDOORN

February 27, 2019

Vrije Universiteit Amsterdam
Faculty of Science
De Boelelaan 1085
1081HV Amsterdam

Abstract

Vancomycin is widely used for infections with methicillin-resistant *Staphylococcus aureus* (MRSA) [1] and therefore highly evaluated in research. However, little research has been done to the use of Long Short-Term Memory (LSTM) networks for predicting the vancomycin serum concentrations.

In this research, an LSTM model was built to predict the vancomycin serum concentrations of the first 24 hours after a vancomycin drug administration. Each dose contains a history; the measurements since the previous dose, and a future; the vancomycin serum concentrations for the next 24 hours. Therefore, the input of the model is a set of sequences of measurements of variable length and the output of the model is a single sequence with 48 time steps (30 minutes per time step).

The model was trained on 241 patients containing 3,231 vancomycin doses in total and tested on 564 patients containing 6,733 doses in total. Since the training data contains only 749 observations of the vancomycin serum concentrations, for most predictions the real serum concentration was unknown. To prevent unrealistic predictions, not only the error on the observations was calculated, but also an error on the shape of the predicted sequence in which extra fluctuations and negative predictions are penalized. NSGA-II, a Multi-objective Evolutionary Algorithm, was chosen to minimize these two error functions.

A grid search was performed on a subset of the training data to find the best model parameters. The final model was applied to the basic feature set (time step, time since the previous dose, number of already administrated doses, current dose size, serum creatinine concentration, admission body weight, admission age and sex) with 4 neurons in the first LSTM layer and 2 in the second.

The final model was evaluated against a benchmark, which was equal to the mean serum concentration (16.5 mg/L). The results showed that the LSTM model was not significantly better than the benchmark for the error function on the observed vancomycin serum concentrations. The most important reason for the low performance suggested in this research is the low number of observed vancomycin serum concentrations in the training data. Therefore, the model was not able to detect the correlations between the features and the serum concentrations correctly.

Keywords: Long Short-Term Memory network, Evolutionary Algorithm, NSGA-II, Therapeutic Drug Monitoring, Vancomycin, Pharmacokinetics, critically ill patients, Deep Learning, Peak and Trough Levels

Acknowledgements

This research is part of the curriculum of the master *Business Analytics* at the Vrije Universiteit Amsterdam. Students are supposed to conduct one month of full-time research. This research is established by supervision of Mark Hoogendoorn on behalf of the Vrije Universiteit and Paul Elbers and Patrick Thoral on behalf of the VU medical center (VUmc).

This research is part of bigger research to the use of Machine Learning in pharmacokinetics, which is focused on obtaining insights in which features influence the elimination of a drug by the human body and getting better predictions of the serum concentrations of medicines.

I would like to thank Mark Hoogendoorn for his ideas and guidance throughout the research. He helped me a lot by providing ideas and techniques to solve the problems I encountered. Furthermore, I would like to thank Paul Elbers and Patrick Thoral, intensivists of the VU medical center (VUmc) in Amsterdam, for their time, the provided medical information, their fast and detailed responses and thinking along for data possibilities.

Contents

List of Figures

List of Tables

List of Abbreviations

ICU	**I**ntensive **C**are **U**nit
MRSA	**M**ethicillin-**R**esistant **S**taphylococcus **A**ureus
TDM	**T**herapeutic **D**rug **M**onitoring
PK	**P**harmacokinetics
PD	**P**harmacodynamics
ANN	**A**rtificial **N**eural **N**etwork
FNN	**F**eedforward **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort-**T**erm **M**emory
EA	**E**volutionary **A**lgorithm
NSGA	**N**on-dominated **S**orting **G**enetic **A**lgorithm

1 Introduction

Vancomycin serum levels need to be within a certain range for patients at the Intensive Care Units (ICU) of hospitals. Vancomycin is widely used for infections with methicillin-resistant *Staphylococcus aureus* (MRSA) [1], which is a bacteria infection. When the concentration of the medicine is too high, it is harmful to the patient, but when it is too low it is not effective.

Choosing optimal doses requires knowledge of pharmacokinetics (the process of absorption, distribution, metabolism and excretion of drugs within the human body) and pharmacodynamics (the drug action: how the body is affected by the drug). According to Marsot et al., several pharmacokinetic studies in various patient populations use in particular nonlinear mixed-effects modelling, a commonly used population-based modelling approach, to identify the covariates that could influence the dose-concentration relationship [2]. In 2000, Tolle et al. suggested artificial neural networks as an effective replacement for those "complex computation models and/or cumbersome statistical prediction applications" [3]. They concluded that their predictions for concentrations of the drug *tobramycin* by a neural network were as good as or better than the predictions by statistical analysis methodologies done with NONMEM®, the current industry standard application for pharmacokinetic data analysis [3].

In recent studies, the effectiveness of Machine Learning models for predicting serum concentration levels is performed for vancomycin too. For example, Hu et al. propose Machine Learning models (SVM and M5) to predict the vancomycin *peak* (maximum concentration after dose administration, approximately examined after 1-2 hours) and *trough* (minimum concentration before next dose is administrated) concentrations [4]. Since they limit their research to the *peak* and *trough* concentrations, they leave out all other observations.

The goal of this research is to predict the vancomycin serum concentration in the first 24 hours after administration at each moment in time, not limited to the *peak* and *trough* concentrations. The motivation for this study is to give a complete insight into the pharmacokinetics of vancomycin by a model for which no prior knowledge is required and to investigate promising Machine Learning techniques for pharmacokinetic analysis.

For this research, a data set from the VU medical center (VUmc) in Amsterdam was made available, which contains information about 800 patients. For each patient, there is information about the administrated vancomycin dose and the measured vancomycin serum concentration level. The number of doses and the number of measurements are variable among the patients, as is the time between the doses and between the measurements. Typical decisions made by clinicians include the daily dose of the medicine and the time interval between administrations [4].

In this paper, a Long Short-Term Memory network is used to predict the serum concentration level of vancomycin. This model takes time dependency into account: it "remembers" when the doses were given and is able to detect correlations in features that change over time. Therefore is LSTM a promising model for predicting drug concentration levels of critically ill patients at the intensive care.

Because of the data quantity and structure, the model cannot be trained with *back propagation*, which is the usual way of training a neural network. This will be explained in more detail in Section 5.4. An alternative to finding the weights of the network that is used in this research is an Evolutionary Algorithm (EA), as suggested by Risi et al. in 2015 [5] and Salimans et al. in 2017 [6].

To conclude if the proposed model is able to learn the change in vancomycin serum concentration after administration of the antibiotic, the predictions of the model are compared with the average serum concentration. This results in the following concrete research question:

> Is it possible to predict vancomycin serum concentrations more accurately with a Long Short-Term Memory network than taking the average serum concentration?

In the next chapter, some medical background is given and the current model is explained. After those preliminaries, we dive into the related work that is done in Deep Learning and in the medical field. In addition, we look at applications of Evolutionary Algorithms. In Section 4, the data is analysed and transformed. After that, the LSTM model and the Evolutionary Algorithm are explained in Section 5. The experimental setup is given in Section 6 and the results are shown in Section 7. Finally, the results are discussed and conclusions are made in Sections 8 and 9 respectively.

2 Preliminaries

2.1 Therapeutic Drug Monitoring

When there is a very narrow line between the effectiveness of medicine and when it becomes toxic (a narrow therapeutic window), it is very important to apply Therapeutic Drug Monitoring (TDM). TDM refers to individualized drug doses in order to keep the drug concentration level within the therapeutic window [7]. In TDM knowledge of pharmacokinetics (PK) and pharmacodynamics (PD) is combined. PK is the study of the processes of absorption, distribution, metabolism, and excretion of drugs within the human body. In contrast, PD is the drug action: how the body is affected by the drug. The goal of TDM is to use appropriate concentrations of difficult-to-manage medications to optimize clinical outcomes in patients in various clinical situations [8].

2.2 Pharmacokinetics

The serum concentration of the drug depends on when the dose was given to the patient. First, there will be a *peak* (maximum concentration level after drug administration) and after that, the concentration will decrease until a new dose is given. The minimal serum concentration reached before a new drug administration is called the *trough* concentration. Figure 2.1 makes this more clear.

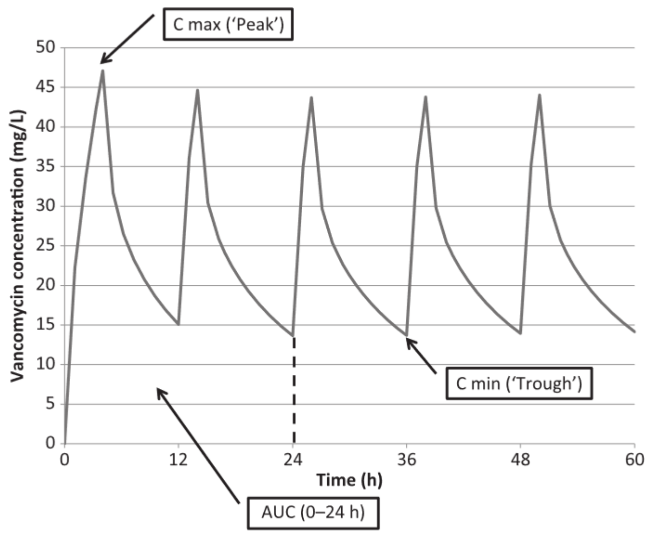


FIGURE 2.1: Pharmacokinetic parameters. Simulated vancomycin concentration versus time graph in hypothetical 60-year-old male (bodyweight 70 kg, creatinine 80 μmol/L) following 2 g intravenous (IV) loading dose and 1 g IV 12-hourly. C_{max}, maximum concentration; C_{min}, minimum concentration; AUC_{0-24}, area under the curve (24-h dosing interval). [1]

According to Rybak et al., the reference range for vancomycin trough levels is 10-20 μg/ml (15-20 μg/ml for complicated infections) and the reference range for vancomycin peak levels is 25-50 μg/ml [9]. The vancomycin peak levels are collected one or two hours after completion of the intravenous vancomycin dose, and vancomycin trough levels are collected just before the next dose is given [10].

Besides *peak* and *trough*, *half-life* is another important medical term. A serum *half-life* is the time that is required to eliminate half of the initial concentration of the medicine. Burton et al. stated: "in patients with normal renal function, the usual serum half-life of vancomycin is 6 to 10 hours, whereas in patients with end-stage renal disease, the half-life may approach 7 days" [11]. Typically the creatinine clearance and glomerular filtration rate (GFR) are measured in order to estimate the renal function [12, 13].

2.3 NONMEM®

NONMEM® is the standard for predicting PK parameters [3]. This is also the computer program that is used by the VUmc. NONMEM® is designed to fit general statistical (nonlinear) regression-type models to data [14], such as the nonlinear mixed-effects model. It is widely used for administration of drugs. This method "is based on the principle that the individual pharmacokinetic parameters of a patient population arise from a distribution that can be described by the population mean and the interindividual variance" [15].

3 Related Work

3.1 Nonlinear mixed-effects models

As already mentioned in the introduction, nonlinear mixed-effects models are used in several pharmacokinetic (PK) studies. A nonlinear mixed-effects model incorporates both *fixed effects* as *random effects*. Fixed effects are parameters that are based on the population and are assumed to stay the same for each prediction. In contrast, random effects are specific for each individual of the population. [16]

The advantage of nonlinear mixed-effects models is that they can cope with small samples sizes and sparse data sets [16]. However, a disadvantage is that you need to specify a structural model, which could be a one-, two-, or multicompartment model. Such a compartment model defines how the drug is transmitted within the human body. The number of compartments in the model represent the number of sections in the body where the concentration is assumed to be uniformly equal.

Marsot et al. reviewed 25 articles about the pharmacokinetics of vancomycin, which all use a nonlinear mixed-effects model [2]. The scope of 15 articles was pediatric patients and the scope of the other 10 articles was adults. They concluded that "in neonates and infants, the pharmacokinetics of vancomycin were mainly described by a one-compartment model, whereas in adults, a two-compartment model was preferentially used" [2]. Furthermore, many covariates were tested, but in almost all models were only three covariates found to be important: age, creatinine clearance and body weight.

In 2011, Roberts et al. suggested a nonlinear mixed-effects model, which consisted of a one-compartment linear model with combined proportional and additive residual unknown variability [17]. The model was implemented in NONMEM®. They used a bootstrap approach in which the parameters *creatinine clearance* (L/h) and the *volume of distribution*(L/kg) were bootstrapped. The volume of distribution was described by the total body weight and clearance by 24-hour urinary creatinine clearance, normalized to body surface area. Their scope was the same as the scope of this research: a population of critically ill patients. They showed promising results with an r^2 of 0.60. Therefore, this model is one of the models VUmc is currently using to predict the vancomycin serum concentrations.

3.2 Machine Learning

Various Machine Learning models are already used in the medical domain. For example, Long Short-Term Memory (LSTM) networks are used to predict heart failure onset by Choi et al. [18] and to classify 128 diagnoses by Lipton et al. [19] in 2017. According to Lipton et al., clinical medical data, especially in the Intensive Care Unit (ICU), consist of a multivariate time series of observations. They transformed the data by re-sampling the time series to an hourly rate and used forward- and back-filling to fill the gaps created by the window-based re-sampling [19].

Less Machine Learning models are used to predict vancomycin serum concentrations. One research was already mentioned in the introduction: Hu et al. use a support vector machine (SVM) and a model-tree-based regression technique (M5), with and without bagging, to predict patients' *peak* and *trough* concentrations [4].

For their study, data of 1,099 clinical cases were collected from a major tertiary medical center in southern Taiwan. The benchmark in their research consists of estimated *peak* and *trough* concentrations of a one-compartment model. According to their conclusions, both models achieve a prediction accuracy significantly higher than that by the one-compartment model.

Machine Learning techniques are more extensively used for predicting tobramycin concentration levels, which is another medicine for the treatment of bacterial infections. For example, Tolle et al. [3] and Chow et al. [20] estimated the concentration levels of the drug tobramycin by applying Artificial Neural Networks. Both papers used a statistical regression-based model in NONMEM® as a benchmark. For the study of Tolle et al. [3], 622 data points were available, each containing a serum concentration level of the drug tobramycin. The best results were obtained for predicting *peak* and *trough* concentrations separately. Their final model was better than the benchmark model, however, the difference was not significant. In contrast, the study of Chow et al. did show a significant difference. They concluded that their model was better than NONMEM®because the average absolute errors of the Neural Networks were 33.9% and 37.3% and 39.9% for NONMEM® [20].

Furthermore, Brzaković et al. predicted serum concentration levels of the drug topiramate [21]. They used Counter-Propagation Artificial Neural Networks (CPANNs) combined with a Genetic Algorithm (GA). Their scope was limited to 78 adult epileptic patients with 118 topiramate serum levels in total. In addition, those patients were at least 7 days on stable dosing regimen and therefore steady-state was assumed. They concluded that topiramate dose, renal function (eGFR) and carbamazepine dose significantly influence the topiramate serum levels by respectively a relative importance of 0.7500, 0.2813 and 0.0625.

A lot research is done to the PK of several medicines. However, we did not find any PK analysis with a LSTM network, which makes this research special.

4 Data Exploration and Transformation

The data used in this research is supplied by the VU medical center (VUmc) in Amsterdam. The data contains records about the admission of a patient at the ICU (one record per patient) and records of measurements over time (zero, one or multiple per patient). There is data available about 1,083 patients in total.

4.1 Filters

278 unique patients needed to be filtered out of the dataset. They were removed because at least one of the following conditions held, see Table 4.1.

Reason	Number of patients	
Dose was not intravenous	29	(2.7%)
Extreme low dose administrated (less than 100 mg)	11	(1.0%)
Extreme high dose administrated (more than 9,999 mg)	1	(0.1%)
No observed vancomycin serum concentrations	127	(11.7%)
Vancomycin serum concentration observed before first dose	170	(15.7%)
Younger than 18	4	(0.4%)
Missing value for gender	12	(1.1%)
The body length (cm) smaller than the body weight (kg)	20	(1.8%)

TABLE 4.1: Applied filters.

Since this research is specific about predicting the vancomycin serum concentration after intravenous dose administration, all patients with non-intravenous doses were removed from the dataset. According to the intensivists of the VUmc, doses of less than 100 mg are very likely falsely marked as intravenous and are therefore removed too.

The third condition (extreme high doses) was used, because such high doses are very unlikely to happen according to the intensivists of the VUmc, and it is not clear if they actually happened, happened with another quantity or did not happen at all.

Fourthly, since we aim to predict the serum concentrations, all patients without observations are removed from the dataset, because we cannot evaluate the performance of the model on these patients. Besides, patients with an observed serum concentration value before the first dose was given are removed too, because this indicates a previous administrated dose which is not in the dataset.

On top of that, patients younger than 18 are removed too because there are only a few (4) and the pharmacokinetics of vancomycin can be different in children [22]. In addition, patients with a missing value for sex are excluded from the dataset, because according to Beierle et al. it is "undoubtedly necessary" to include the difference between men and women in the clinical drug development process [23].

Finally, there were some errors made in the body length and body weight fields of the dataset. Since it is very unlikely to have a smaller body length (cm) than body weight (kg), and since the size of the body does affect the pharmacokinetics of vancomycin [23], these patients are removed from the dataset too.

After applying the above mentioned filters, 805 patients are left in the dataset.

4.2 Feature overview

83 variables are included in the provided dataset of the VUmc. To decrease the number of features for the model, possibly relevant features were selected in collaboration with the intensivists of the VUmc. This resulted in a set of 12 features to predict 1 target feature (vancomycin serum concentrations), see Tables 4.2 and 4.3 for an overview.

Binary features				
Attribute name	**Yes**		**No**	
Sex = male	522	64.75%	283	35.25%

TABLE 4.2: Basic statistics of the binary features.

Numerical features					
Attribute name	**unit**	**mean**	**standard deviation**	**range**	**mean number of records per patient**
Age	*years*	62.64	14.66	18 - 92	1
Body weight	*kg*	78.33	14.66	35 - 160	1
Vancomycin dose	*mg*	912.75	245.86	100 - 2000	16.77
Serum creatinine	$\mu mol/L$	128.10	95.03	1 - 1192	37.36
CVVH-substitute	ml/h	1787.83	1048.55	0 - 4700	7091.83
CVVH-filtrate	*ml*	168.96	137.52	0.07 - 2660	159.56
Dialysis filtrate	*ml*	1343.09	903.73	40 - 4000	2.58
Urine volume	*ml*	132.49	111.89	1 - 4000	375.51
Glasgow Coma Scale		8.70	3.84	3 - 15	31.20
PEEP	cmH_2O	8.86	3.28	0 - 38	4560.50
O2%		43.79	11.56	0 - 205	4566.40
Vancomycin serum concentration	mg/L	16.12	8.67	1.1 - 77.4	2.84

TABLE 4.3: Basic statistics of the numerical features.

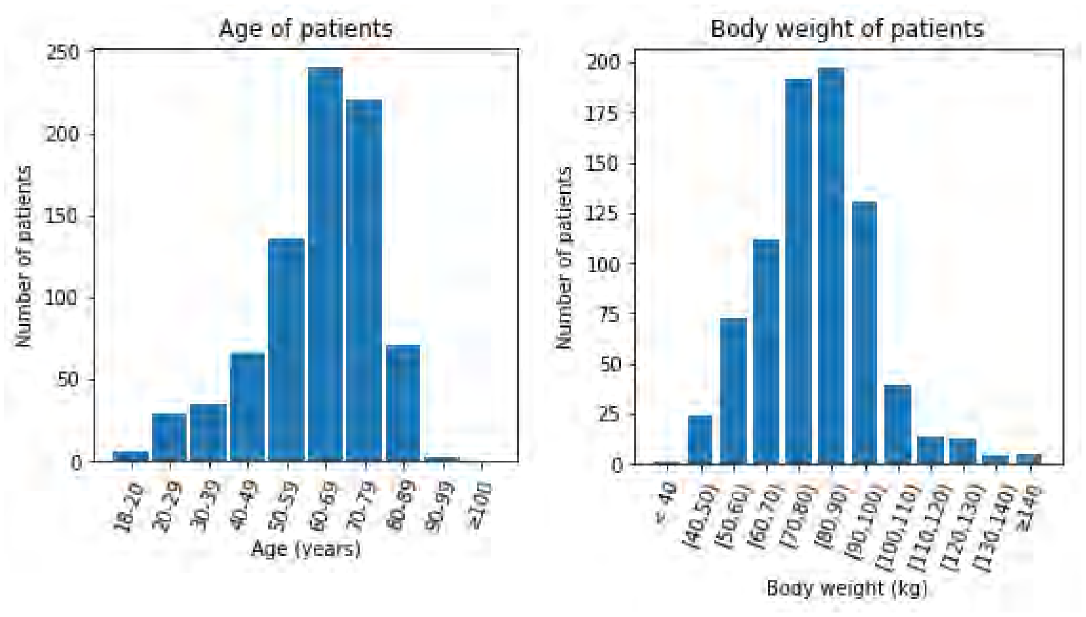
From the 12 features are the three features *sex*, *age* and *body weight* recorded at the moment of admission to the ICU. The feature *vancomycin dose* indicates how much and when vancomycin was administrated. The *serum creatinine* is regularly measured: 37 times per patient on average.

In addition, features about the renal function were included, because these features are important for the elimination of a drug from the body. For example *CVVH-substitute* and *CVVH-filtrate* represent respectively the substitution fluid and the ultrafiltrate of the CVVH (continuous veno-venous hemofiltration) process. However, an alternative to hemofiltration is hemodialysis, which results in the next variable (*dialysis filtrate*). The last feature about the renal function is the amount of urine that was collected by for example a catheter.

Finally, the last three features are selected to give a broader view of how ill a patient is. The *Glasgow Coma Scale* is a neurological scale to measure objectively how conscious a patient is. Secondly, *PEEP* and *O2%* are features about how much a patient is assisted with breathing. PEEP (positive end-expiratory pressure) assists the respiration by supporting the regulation of the respiratory rate and *O2%* is the supplied amount of oxygen during ventilation.

4.3 Statistics

Table 4.2 shows that approximately 65% of the patients are men and 35% are women. These patients are between 18 and 92 years old, see Table 4.3. However, the majority is between 50 and 80 years old, see Figure 4.1a. Therefore, we can conclude that the dataset is biased to older patients.



(A) The distribution of age. (B) The distribution of body weight.

FIGURE 4.1: Age and body weight of the patients in the dataset.

According to Beierle et al. [23], body weight is one of the important features for the pharmacokinetics of vancomycin. Therefore, it is interesting to see how body weight is distributed for the patients in the dataset. This is shown in Figure 4.1b. The distribution looks approximately normally distributed with a mean around 80 kg. In order to test this hypothesis, a normality test was performed. The p-value of the Shapiro-Wilk test is $4.7 * 10^{-11}$ which is smaller than alpha 0.05, therefore the hypothesis that the distribution of body weight is normally distributed is rejected.

4.3.1 Length of stay

For all patients we can calculate the time between the first dose administration and the last measured variable, to indicate the length of stay. The results are visualized in Figure 4.2.

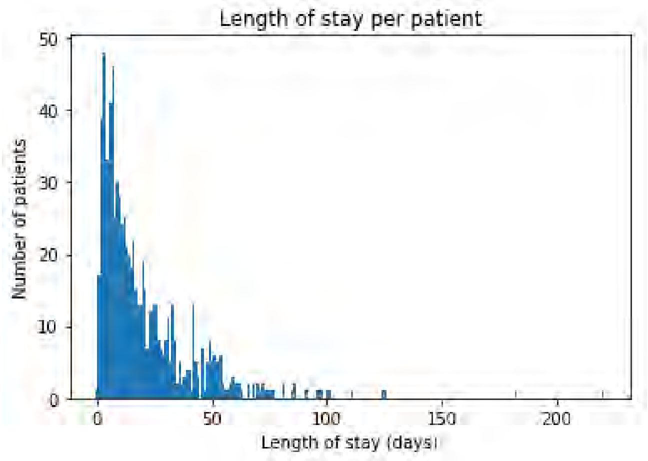
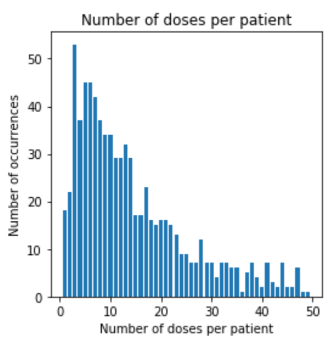


FIGURE 4.2: Distribution of the length of stay since the first administrated dose vancomycin.

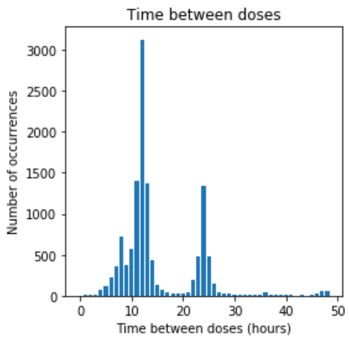
The average length of stay is 19.4 days. However, the distribution is very right-skewed (Figure 4.2), which means that most patients stay for a shorter period than 19.4 days. Moreover, 31% of the patients leave within one week, 53% within two weeks and 67% within three weeks.

4.3.2 Doses and measurements

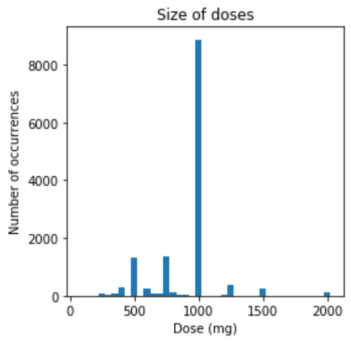
Before a vancomycin serum concentration can be observed, a vancomycin dose is given to the patient. On average 17 doses are given per patient, but 50% of the patients got 12 doses or less, see Figure 4.3a.



(A) The distribution of the number of doses per patient.



(B) The distribution of the time between doses.

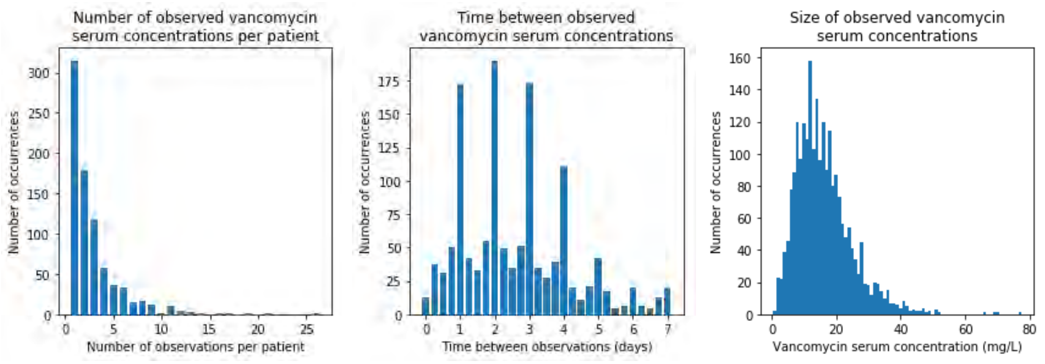


(C) The distribution of the size of the vancomycin doses.

FIGURE 4.3: Distributions of the records about vancomycin doses.

The size of the doses are between 100 and 2,000 mg, but 1,000 mg is most commonly used (65.70%), see Figure 4.3c. In Figure 4.3b, the time between doses is shown. The time between doses is on average 20 hours and the median is 12 hours. Furthermore, we can clearly see a pattern in the time between doses: there is a peak at 12, 24, 36 and 48 hours.

Since the goal of this research is to predict the vancomycin serum concentrations, it is relevant to see how many observations are available in the dataset. In Figure 4.4a is shown that most patients got very few observations of the vancomycin serum concentration. One observation per patient occurs most frequently, and 50% of the patients have two or fewer observations. On average, there are 2.84 observed values available per patient.



(A) The distribution of the number of observations per patient.
(B) The distribution of the time between observations.
(C) The distribution of the size of the observed vancomycin serum concentrations.

FIGURE 4.4: Distributions of the records about the real vancomycin serum concentrations.

In the distribution of the time between the observations is again a clear pattern shown: a peak is shown every 24 hours, see Figure 4.4b. In Figure 4.4c, the size of the observed vancomycin serum concentrations is shown. The average observed serum concentration is 16.1 mg/L.

The measurement of the vancomycin serum concentration takes most often place after approximately 12 or 24 hours, see Figure 4.5. This figure is very similar to Figure 4.3b with the time between doses. Therefore we can conclude that most measurements in the data correspond with vancomycin trough levels: obtained just before the next drug administration.

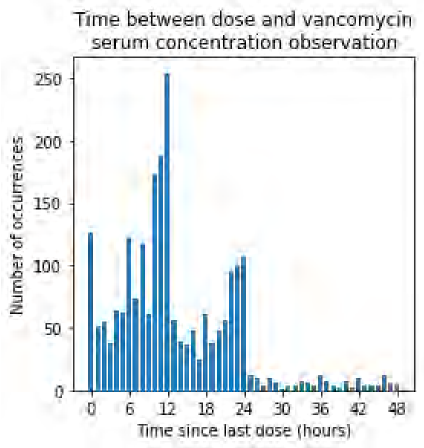


FIGURE 4.5: The distribution of the time between a vancomycin dose and a serum concentration.

4.4 Dataset preparation

The goal of this research is to predict the vancomycin serum concentrations for the next 24 hours after the drug is administrated. Therefore we can split the dataset in a historical set (the input of the model), which contains all measurements that are available at the moment of the administration, and a future set (the output of the model), which contains the observed serum concentrations after the dose administration.

Since all measurements are obtained at different moments in time, the records are split into intervals of 30 minutes [24]. This is the same approach as used by Lipton et al. [19]. For the historical dataset, time steps of -30 are taken from the moment of the current dose administration, until the previous dose administration. And for the future dataset, all observed serum concentrations related to the doses in the historical dataset are taken and split by time steps of +30 minutes since the drug administration. This is visualized in Figure 4.6. By creating the time steps in this way, we ensure that there is no overlap between the historical and the future dataset.

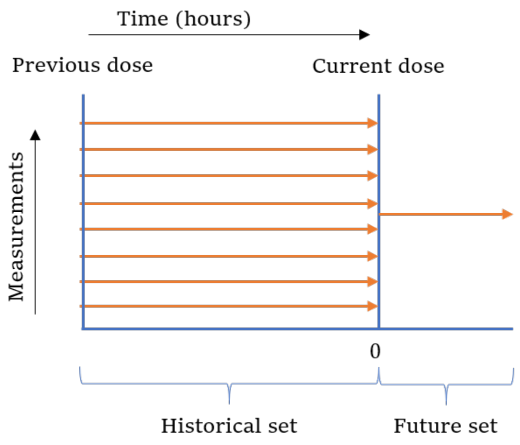


FIGURE 4.6: Abstract visualization of the historical and future dataset.

The records about the admission of the patient in the intensive care, for example, the age and the weight, are the same at each time step for a patient and are therefore duplicated for each time step in the historical dataset. For the time-dependent features, it is possible that there are multiple measurements within an interval of 30 minutes. If this occurs, the mean value of these measurements is taken.

All time steps with a missing value can be split into two groups. The first group contains values that are missing because they do not exist, for example, values of the vancomycin dose can be missing when the patient did not get it at that moment in time. These values are imputed by zero. The second group are values that are missing because they were not observed in that time step, for example, the serum creatinine observations. These values are linearly interpolated per patient [19]. However, when there is no measurement of a certain feature for a patient, it cannot be interpolated. If this occurs, the median value of all known values is taken. This seems a reasonable choice: most missing values exist because the clinician did not expect this value to be abnormal. There is explicitly chosen for the median, and not for the mean because the value should be robust for extremes in the dataset.

4.4.1 Sequences

By creating the historical and future datasets, data were grouped per drug administration per patient. In this way, for each dose, we can find records of known values in the historical dataset, and records we would like to predict in the future dataset. Remark that the records in the historical and future datasets are sequences.

Each sequence in the historical dataset and in the future dataset is related to a vancomycin dose. Therefore, the same number of sequences are available in the two datasets as the total number of doses, which is equal to 13,503. However, for this research, we are only interested in sequences (or doses) that happen before the serum concentration was observed. Therefore, the data was cut off after the last observed serum concentration of each patient, which resulted in a decrease of 3,539 sequences without removing observed vancomycin serum concentrations. The final datasets contain information about 9,964 sequences (or doses), which is 12.38 sequences per patient on average. Of these 9,964 sequences contains approximately 78% (7800) zero observations, 21% (2054) one observation and 1% (110) more than one observation for the vancomycin serum concentration.

4.4.2 New features

By creating the time intervals of 30 minutes within the sequences, a new feature called *sequence time* was created. For the historical dataset, this feature includes only negative numbers and zero, and for the future dataset only positive numbers.

Besides the already mentioned features in Tables 4.2 and 4.3, other historical features can be created. The *time since the previous dose* might help the model to understand the relation with the previous sequence. In addition, *count doses* might help the model to understand how many sequences (or doses) occurred before. This results in a historical dataset of 9,964 sequences with 15 features. In each sequence, one record of data corresponds to one time interval of 30 minutes. This results in 357,973 records in total, which is 35 records (17.5 hours) per sequence on average.

The future dataset contains 9,964 sequences too. However, this dataset contains for each time interval of 30 minutes within one sequence only one feature: the vancomycin serum concentration. Since there are only a few observations per patient available, most of these intervals contain a missing value.

5 Models

The goal of this research is to predict the vancomycin serum concentration at each point in time in the first 24 hours after a dose administration. The predictions are therefore time-dependent. In addition, the provided input data consists of multivariate time series, as for example the measurements of serum creatinine. Since a Long Short-Term Memory network takes time dependency of variables into account, this model is a good candidate for predicting vancomycin serum concentrations.

LSTM networks are a special type of neural networks. In order to make clear what LSTM networks are, artificial neural networks and recurrent neural networks will be introduced first.

5.1 Artificial Neural Network

An artificial neural network (ANN) is the traditional neural network which contains an input layer, one or more hidden layers and an output layer [25]. Each layer contains nodes, or *neurons*. Each node in each layer is connected to each node in the next layer [25], this is called full connection between layers. Each connection (or arc) has a numerical weight that specifies the influence between two neurons. While training an ANN, the input and output values are known. Therefore we can set the weights of the model in such a way that the predicted value by the model is closest to the actual value.

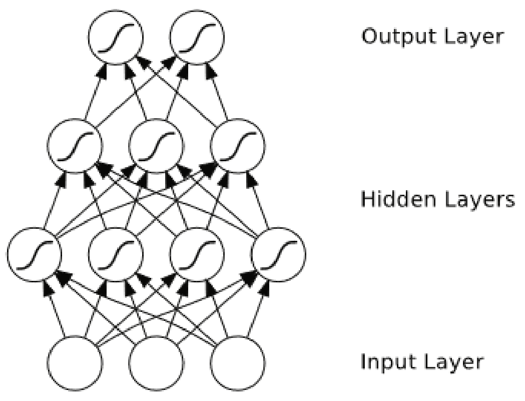


FIGURE 5.1: A multilayer perceptron, as an example of a feedforward neural network [26].

There are many variants of ANNs. ANNs with cycles are referred to as feedback, recursive, or recurrent, neural networks and ANNs without cycles are referred to as feedforward neural networks (FNNs) [26]. In an FNN, as the name suggests, the input patterns are (implicitly) connected with the output by only forward connections. A well known example of a FNN is the *multilayer perceptron* [26], of which an example is shown in Figure 5.1.

5.1.1 Activation Functions

Each neuron has input signals and an output signal. The output of a layer is used as input for the next layer, see Figure 5.1. Therefore, each neuron has to combine the input signals to an output signal. This is done by calculating the weighted sum of the input values plus a *bias* and applying an *activation function* ϕ [27]. This is given by the following formula (Formula 5.1).

$$Y_k = \phi(\sum_i (w_{ik} * x_i) + b_k) \tag{5.1}$$

In this formula Y_k and b_k represent respectively the output signal and the bias of neuron k, w_{ik} is the weight of the connection between input i and neuron k and x_i is the value of input i.

The simplest activation function is linear: $\phi(x) = x$. This means actually that no activation function is used. A linear equation is easy to solve, but it is very limited in complexity. Other, more common, activation functions are shown in Table 5.1 and in Figure 5.2.

Name	Formula	Range
linear	x	(-∞, ∞)
hyperbolic tangent	$tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$	(-1, 1)
logistic sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	(0, 1)
rectified linear unit	$\max(0, x)$	[0, ∞)

TABLE 5.1: Formulas of activation functions. [27]

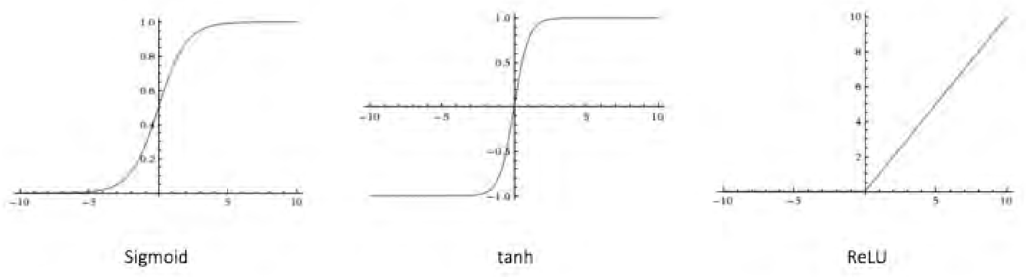


FIGURE 5.2: Different neural networks activation functions. [27]

For more information on activation functions, please refer to the work of A. Graves [26] and U. Karn [27].

5.2 Recurrent Neural Network

ANNs, have no persistence, which seems like a major shortcoming according to C. Olah [28]. Recurrent neural networks (RNNs), a type of ANNs, are designed to use the sequential information between observations, for example, time-dependent observations. To accomplish this, the RNN has a recurring connection to itself [25]. In this way the nodes are interconnected [29] and it gives the RNN a sense of time context [25]. To picture this, we can *unroll* the recurrent connection, which is shown in Figure 5.3 [28].

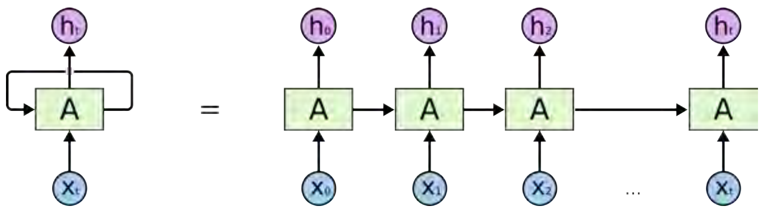


FIGURE 5.3: Structure of a recurrent neural network (left) and structure of an unrolled recurrent neural network (right). [28]

In Figure 5.3 X_t represents the inputs of time step t. The A's represent a chunk of a neural network and h_t the output of time step t [25, 28]. The arrow between two time steps, the recurring connection, indicates the hidden state of the model. Because of the recurring connection, the model is able to detect sequential patterns in the historical observations and will use this knowledge in the prediction of the next time step.

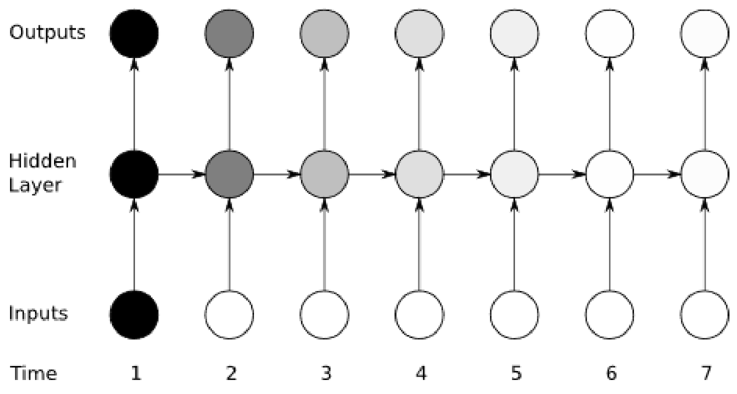


FIGURE 5.4: The vanishing gradient problem for recurrent neural networks. The shading of the nodes in the unfolded network indicates their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decays over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs. [26]

This brings us to the fundamental problem of RNNs: when the time between the historical event and the current time step grows, the RRN does not succeed to connect this information properly [28]. This *vanishing gradient problem* arises because the derivatives of the activation functions within the network are often between 0 and 1, which goes to 0 when they are multiplied many times, which occurs while applying the chain rule in the back propagation method [25]. This causes earlier layers to learn very slowly compared to later layers [25], see Figure 5.4. Information on back propagation can be found in the work of Goodfellow et al. [30].

5.3 Long Short-Term Memory Network

LSTM networks are a special kind of recurrent neural networks. The repeating chunks shown in Figure 5.3 by A, are almost the same as the neurons in an ANN: the weighted sum of the *recurrent inputs* is added to the weighted sum of the inputs and bias of Equation 5.1 [28]. However, the repeating chunks, *LSTM blocks*, in an LSTM network are more complex. An LSTM block is developed to overcome the problems faced by standard RNNs. This is done by the introduction of a new state called *cell state* [25], which is sometimes referred to as *memory cell*. The internal structure of an LSTM block is shown in Figure 5.5. Besides this cell state, the figure shows three kinds of *gates* and a block input and output. These components are briefly described below.

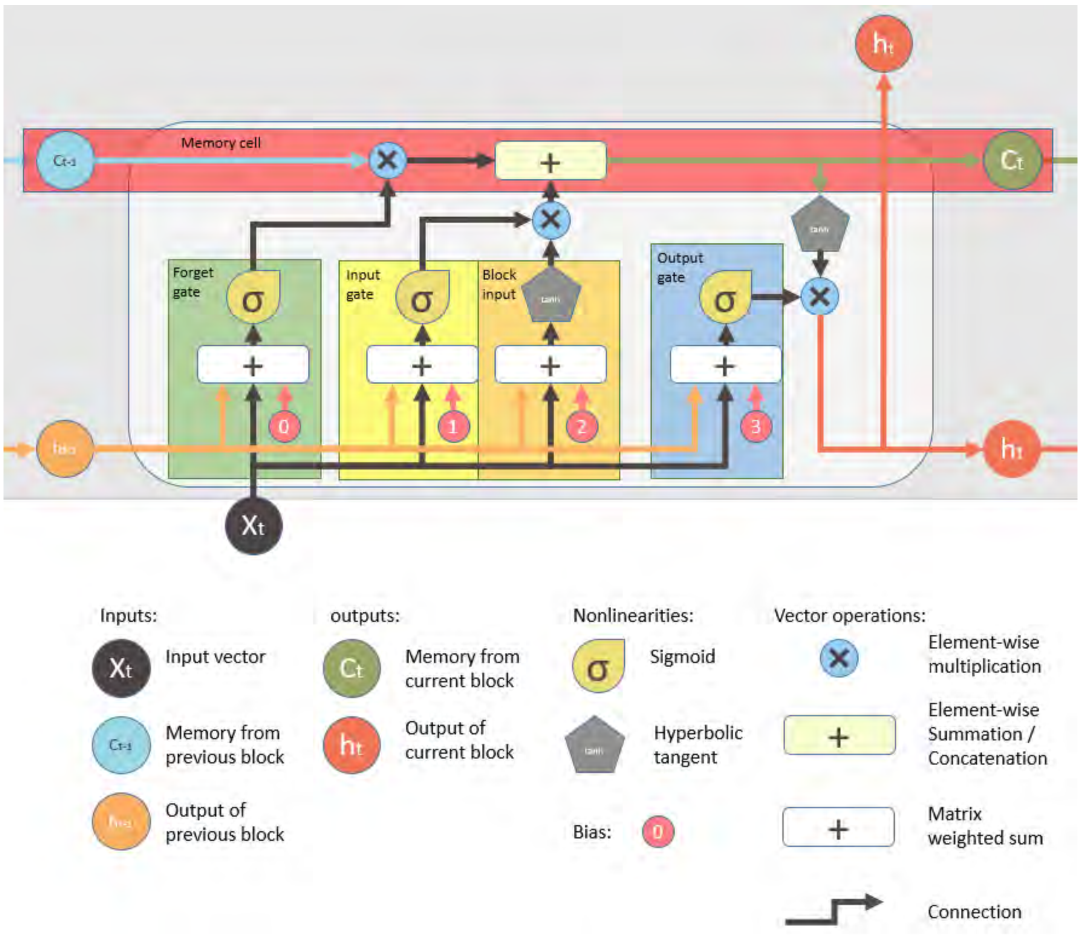


FIGURE 5.5: LSTM building block.
Adaptation of the diagram of S. Yan [31].

5.3.1 Forget gate

Each *gate* has the same structure as the calculations in a hidden layer of an ANN: it calculates an output signal by applying an activation function over the input signals. The activation function "activates" the neuron with a value close to one and does not "activate" it with a value close to zero.

Figure 5.5 shows how the previous cell state (memory cell) is updated by the multiplication with the output of the *forget gate*. The forget gate is the special part of the LSTM block because it enables the network to "reset" [32]. When the output of the forget gate is close to zero, "little memory" will go through because of the multiplication. In contrast, when the output of the forget gate is close to one, almost everything will be kept in memory.

The *forget gate* gets the input of the current time step t, x_t and the output of the previous time step $t-1$, h_{t-1} as input, see Figure 5.5. The weighted sum of these inputs are taken and a bias (b_f) is added. After that the logistic sigmoid activation function (σ) is applied, which is mentioned in Section 5.1.1. This results in Function 5.2 [31, 32], with W_{xf} input weights and W_{hf} recurrent weights:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{5.2}$$

Note that f_t is between 0 and 1, because of the logistic sigmoid activation function. This is in line with its purpose as explained: close to zero for forgetting and close to one for remembering.

5.3.2 Input gate

Besides forgetting, the memory cell is able to remember new information. This is done by the input gate and the block input. The block input is the information to be added and the input gate decides how much of this information should be added to the memory.

The *input gate* gets the same input as the forget gate (x_t and h_{t-1}). The weighted sum of these inputs is taken and a bias b_i is added. After that, the logistic sigmoid activation function (σ) is again applied, which ensures an output range of (0, 1). Almost no new information will be added to the memory when the output of the input gate is close to zero and when the output is close to one, almost all information will be added to the memory. This results in Function 5.3 [31, 32], with W_{xi} input weights and W_{hi} recurrent weights:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{5.3}$$

Before the block input reaches the merge with the input gate, the hyperbolic tangent activation function (tanh) is applied, which *squashes* the input to the range (-1, 1) [32, 33], this ensures the stability of the memory cell and the possibility to add negative changes to the cell state [34]. This results in Formula 5.4 [31, 32], with W_{xz} input weights and W_{hz} recurrent weights:

$$z_t = tanh(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{5.4}$$

The new memory state is a combination of the memory that is not forgotten ($f_t c_{t-1}$) and the new information ($i_t z_t$). This is shown by Equation 5.5 [31, 32]:

$$c_t = f_t c_{t-1} + i_t z_t \tag{5.5}$$

The summation in Equation 5.5, instead of multiplication, makes the LSTM block not suffering from the vanishing gradient problem, but "remembering" over a long distance.

5.3.3 Output gate

After updating the cell state, the hyperbolic tangent (tanh) is again applied, which makes the range again (-1, 1). This output is multiplied with the output of the *output gate*, which structure is again similar to the structure of the previous gates, see Equation 5.6 [31, 32]. This calculation results in the final block output, which is shown in

Equation 5.7 [32].

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{5.6}$$

$$h_t = o_t tanh(c_t) \tag{5.7}$$

5.4 Neuroevolution

In order to make the LSTM network useful, the weights in the LSTM blocks should be optimized during the training phase of the model. In Section 4, we created a historical and future dataset of sequences, both with one row per 30 minutes. In addition, we mentioned that not every row in the future dataset has an observed value for the vancomycin serum concentration. So, the target is not known in every row, which is problematic for the standard method for optimizing the weights of a neural network: *backpropagation* [6, 30]. Backpropagation is therefore not a good method for this specific case.

5.4.1 Evolutionary Algorithms

The basic idea behind *neuroevolution* is to train the network with an Evolutionary Algorithm (EA), which is a class of stochastic, population-based search methods inspired by Darwinian evolution [5, 35].

Algorithm 1 General outline of an evolutionary algorithm [36]

```
Initialization
repeat
    Recombination
    Mutation
    Evaluation
    Selection
until Termination criterion fulfilled
```

The general framework of Evolutionary Algorithms consists of initialization, recombination, mutation, evaluation and selection [36], see Algorithm 1. During the initialization the first generation is created, which consists of one or more individuals. For each individual the objective function value, the *fitness*, is evaluated. After initialization, the so-called evolution loop is entered, which consists of the recombination, mutation, evaluation and selection operators [36]. In this loop, new individuals (offspring) from the parent population are created by recombination. Mutation is used to create variation among the new individuals. Finally, the offspring is evaluated and individuals are selected to be the parent generation of the next iteration of the evolution loop.

Search space

Since the Evolutionary Algorithm is used to optimize the weights of the LSTM network, individuals in the EA will represent one possible set of weights for the LSTM network. All possible solutions for the weights of the LSTM network define the *search space*. In order to limit the search space, a minimum and maximum value for the weights must be given. Remark that the size of the search space is equal to S^n with n the number of weights in the LSTM network and with $S \subset \mathbb{R}$ between the minimum and maximum value. So, the complexity of the problem increases exponentially while adding weights to the LSTM network, for example by adding neurons or extra layers.

Objective space

The goal of this research is to predict the vancomycin serum concentrations for every 30 minutes between the moment of a drug administration until 24 hours afterwards. Since there are only a few observed serum concentrations per patient available, and we know that the predicted concentration is only allowed to go up after the administration, and when reaching the peak level it is only allowed to decrease, but can never become negative, we could use two objective functions: one for the fit on the observed values (F_1) and one for the shape of the predicted sequence (F_2). This results in a two-dimensional *objective space*.

Definition *Domination* [37]:

A solution $x^{(1)}$ is said to dominate the other solution $x^{(2)}$, if both the following conditions are true:

1. *The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives. Thus, the solutions are compared based on their objective function values (or location of the corresponding points on the objective space).*
2. *The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in atleast one objective.*

Above another term is introduced: *domination*, as defined by Deb in 2011 [37]. For each individual, the corresponding point in the objective space is calculated by using the objective functions (F_1) and (F_2). Figure 5.6 visualizes these points and the area they dominate. If the point of the individual is not dominated by any other individual, the point is in the set of *non-dominated points*, shown by the orange points in Figure 5.6.

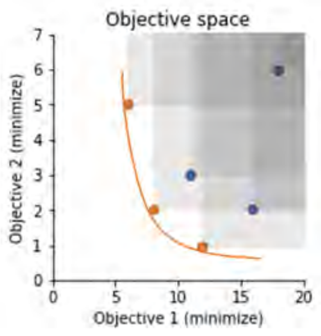


FIGURE 5.6: Objective space defined by two minimization objectives (*objective*$_1$ and *objective*$_2$). The orange curve represents the Pareto-optimal front. The non-dominated points are shown in orange and the points they dominate are shown in blue.

Since F_2, the objective for the shape of the predicted sequence limits the model in the way it can fit the observations (F_1), there is some kind of trade-off between the two objectives. This results in the so-called *Pareto-optimal front*, which contains only non-dominated points in the objective space, see orange line in Figure 5.6. The idea behind this front is that there is no better solution in one direction without the costs of the other. For more details of Pareto optimality, please refer to the work of Ehrgott in 2005 [38] and Miettinen in 2012 [39].

5.4.2 Multi-objective algorithms

Multi-objective algorithms are designed to optimize (minimize or maximize) multiple objective functions. The goal of such an algorithm is to find both a set of solutions which lie on the Pareto-optimal front and a set of solutions which are diverse enough to represent the entire range of the Pareto-optimal front [37].

NSGA

The non-dominated sorting genetic algorithm (NSGA) [40] was one of the first Evolutionary Algorithms with the ability to find multiple Pareto-optimal solutions in one single simulation run. However, the main disadvantages were (1) a high computational complexity of non-dominated sorting which was used in every generation and (2) the lack of elitism (which can help preventing loss of good solutions once they are found) [41].

NSGA-II

In 2002, Deb et al. proposed an improved version of NSGA: the NSGA-II [41]. NSGA-II uses elitism and an explicit diversity preserving mechanism, and it emphasizes non-dominated solutions by using fast non-domination sorting. The advantage of the explicit diversity preserving mechanism is that it does not require any user-defined parameter for maintaining diversity among population members, for which parameter σ_{share} was needed in NSGA. First, the mechanism defines a density estimation: *the crowding distance value*. The crowding distance value d_i is the distance in the objective space around point i, which is not occupied by any other solution in the population [37]. Therefore, the lower d_i, the more "crowded". Because of the second goal of multi-objective algorithms, diversity, solutions with a higher d_i are preferred.

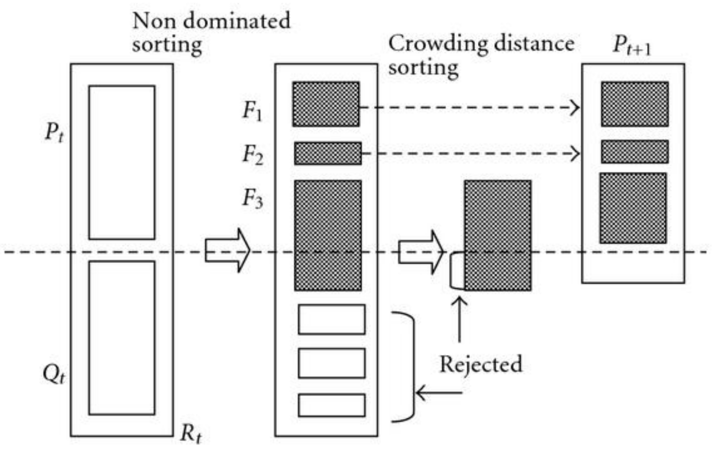


FIGURE 5.7: NSGA-II procedure [41].

Main loop The initial population (P_t) is randomly generated. After that, the first offspring (Q_t) is created from it by parent selection (*binary tournament selection*), recombination (*simulated binary crossover*) and mutation (*polynomial mutation*) [41]. P_t and Q_t are combined into R_t, which has a size of two times the initial population size. R_t is sorted by the fast non-dominated sorting procedure, where all members are classified and put into fronts. The first front contains the non-dominated points, the second front contains the non-dominated points when removing the first front, and so on. Now, the size of R_t will be reduced to the initial population size, by selecting the fronts one by one. If too many members belong to the next selected front, the members in that front are sorted according to their crowding distance in descending order (preferring diversity), and the top members are taken such that the size of R_t is again equal to the initial population. The visualization in Figure 5.7 makes this more clear.

The NSGA-II is one of the most popular Evolutionary Algorithms for solving multi-objective problems [42]. It has only 6 parameters: (1) the population size, (2) the number of generations to evolve, (3) the crossover probability, (4) the distribution index for crossover, (5) the mutation probability and (6) the distribution index for mutation. The number of generations is the termination criteria of the Evolutionary Algorithm, see Algorithm 1. The population size is the number of individuals in the population at the end of each evolution loop in Algorithm 1. The crossover probability (in general between 90% and 100% [43]) refers to the probability with which parents are recombined to create new individuals. The distribution index for crossover controls the perturbation in the recombination operator. The smaller the value of this control parameter, the larger the perturbation and vice versa [43]. Usually, this parameter is set to 10.0 [43]. The same parameters are present for the mutation operator. For the mutation probability, a value of 20% or less is commonly used [42, 44, 45], and for the distribution index for mutation, a value between 5 and 50 [41, 46].

5.4.3 Fitness function

As already mentioned, in the EA two objective functions, or *Fitness functions*, are optimized. One for minimizing the difference between the observed values and the predicted values by the LSTM network, and one for minimizing the error on the shape of the predicted sequence. In this way, two errors E_1 (for the difference with the observations) and E_2 (for the shape of the sequence) can be calculated for each predicted sequence.

There are several measures of the goodness of fit which represent how good the model performs. The mean absolute error and the mean square error are most widely used.

Mean Absolute Error

The mean absolute error (MAE) is a simple error function in which the mean is taken over all absolute errors. The Formula 5.8 shows the MAE in which e_i is the i^{th} error made.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{5.8}$$

Mean Square Error

The mean square error (MSE) is very similar to the MAE, but now the square of the error is taken instead of the absolute value, see Formula 5.9. For an unbiased estimator, the MSE represents the variance of the error.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(e_i)^2 \tag{5.9}$$

By calculating the MSE (Formula 5.9), the square of the error is taken before the mean. This results in bigger weights for bigger errors and smaller weights for smaller errors. In contrast to the MAE, where all errors have the same weight.

In this research, it was chosen to define the errors E_1 and E_2 in the following way. E_1 is defined as the MSE of the difference between the real serum concentrations and the predicted serum concentrations. The MSE was chosen, because big errors on these predictions can have serious consequences for the patient, and are therefore emphasized more in the error function. E_2 is defined as a summation of multiple types of errors, which are all related to the shape of the predicted sequence.

The first error type of E_2 is the *sign error*. This error refers to the number of changes in the sign (positive or negative) of the change in the predicted serum concentrations over time. Since we expect that the serum concentration will increase after a drug administration (positive sloop) and after the peak concentration only will decrease (negative sloop), the number of changes in sign should be one. If this is not the case, the sign error will be equal to the number of times it is more or less. In other words: the absolute value of the difference between the number of changes and 1.

The second error type of E_2 is the *positive after negative error*. This error refers to the number of times that the sloop of the predicted serum concentrations increases after it started decreasing. The predicted sequence should be a smooth function that only increases in the beginning and decreases at the end. Therefore the positive after negative error counts the number of positive sloops after the first negative sloop in the predicted sequence.

The third error type of E_2 is the *negative before positive error* and is very similar to the positive after negative error. The only difference is that this metric counts the number of negative sloops before the first positive sloop occurs in the predicted sequence. This error is calculated because after an administration the serum concentration should immediately go up.

The last error type of E_2 is the *negative error*. This metric counts the number of negative serum concentrations in the predicted sequence. This error captures our knowledge that the vancomycin serum concentration of a patient can never become negative.

Error E_2 is calculated by summing all error types together. This is done, because these error types all represent fundamental errors in the predictions, and therefore all have the same weight. Finally, to get the total error over all sequences F_1 and F_2, we can simply take the mean over the errors E_1 and E_2 of all sequences.

5.4.4 Metrics of performance

Since multiple models will be compared for parameter optimization, a global metric needs to be defined which combines F_1 and F_2. A metric that is often used for this purpose is the *hypervolume indicator*. This metric calculates the volume of the area that is dominated by the non-dominated front of the final population in the Evolutionary Algorithm, see the grey area in Figure 5.6.

Since both F_1 and F_2 can grow until infinity by definition, we need a reference point that is not exceeded by any individual of any model. In Figure 5.6 this reference point could be for example (20, 7). It is important to use the same reference point for all models to ensure comparability of the hypervolumes.

6 Experimental Setup

The goal of this research is to predict the vancomycin serum concentrations in the first 24 hours after a vancomycin drug administration by an LSTM network. The predictions are based on all the historical records of the patient before the current dose. For this purpose, the historical dataset was created in Section 4. The predictions of the model will be compared with the real values in the future dataset, which was also created in Section 4.

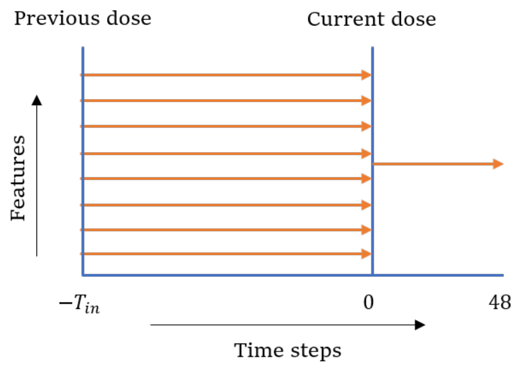


FIGURE 6.1: Abstract visualization of multiple input features of variable length, and a single output feature of static length.

Since the length of the input sequences (historical dataset) depends on the time between vancomycin drug administrations, it is not a static value. However, the number of predictions is set to 48 (24 hours by steps of 30 minutes), which is a static number. Therefore, the input sequence length (historical dataset) and the output sequence length (the predictions) are usually not the same, see Figure 6.1. In addition, the input sequence consists of multiple features, while the output has only one feature: the vancomycin serum concentrations.

6.1 Training and test set

As mentioned in Section 4, there are in total 9,964 sequences available. Some of these sequences will be kept separated in a test set, the others will be used for training the model. Since the sequences of the same patient relate to each other, the train/test split is made on the patients.

	Number of patients		Number of sequences	Number of observations	Number of historical records
Training set	241	30%	3,231	749	125,067
Test set	564	70%	6,733	1,539	232,906
Total	805	100%	9,964	2,288	357,973

TABLE 6.1: Training and test records.

There are 805 patients in the dataset in total. These patients are split into a training set and a test set by respectively 30% and 70%. These percentages are chosen in this way because the first runs indicated a long running time. To be able to finish the research in time, the number of patients in the training set was decreased to only 30% of the data. See Table 6.1 for the real numbers.

The splitting was done is a stratified fashion, where the patients were selected based on their average serum creatinine level. According to the intensivists of the VUmc, the serum creatinine plays the most important role in the clearance of vancomycin. To be able to stratify based on a numerical feature, the average serum creatinine levels were split into intervals of 50 $\mu mol/l$.

6.1.1 Exploration and validation set

Besides the test dataset, another independent dataset is needed for the evaluation of different parameter settings. Therefore, the training set was split in a smaller training set, which we will call the exploration set, and a validation set. For this the same approach is used as for the train/test split, however now only 5% will be used in the exploration set, and from the other 95% will only ten patients be selected for the validation set. This results in 206 and 105 sequences respectively, see Table 6.2. These small numbers were chosen because there is limited time available for this research.

	Number of patients		Number of sequences	Number of observations	Number of historical records
Exploration set	12	1.5%	206	46	7,165
Validation set	10	1.2%	105	31	3,943

TABLE 6.2: Exploration and validation records.

6.2 Features

15 features were selected in Section 4 based on the suggestions of the intensivists of the VUmc. The features dose size, serum creatinine concentration, admission weight, admission age and sex are present in almost every model [2, 23] and are therefore included in the *basic* feature set (Table 6.3). In addition, the sequence time, the time since the previous dose and the number of doses already administrated were included too, because these features can help the model to detect the sequential patterns.

The other features were selected in the *extended* feature set, because they give a broader view on how ill the patient is. Since we do not know if these features can help the model in predicting the vancomycin serum concentrations, the features are kept separated from the *basic* feature set. While optimizing the parameters of the model, the best feature set will be selected. An overview of the *basic* and *extended* feature set is given in Table 6.3.

Basic feature set	**Extended feature set**
Sequence time	Sequence time
Time since previous dose	Time since previous dose
Number of doses administrated	Number of doses administrated
Vancomycin dose size	Vancomycin dose size
Serum creatinine	Serum creatinine
Body weight	Body weight
Age	Age
Sex	Sex
	CVVH-substitute
	CVVH-filtrate
	Dialysis filtrate
	Urine volume
	Glasgow Coma Scale
	PEEP
	O2%

TABLE 6.3: Basic and extended feature set.

6.3 Scaling

According to LeCun et al., convergence in fitting the weights of a neural network is usually faster if the average of each input variable over the training set is close to zero and the variance is close to one [47]. The *z-score* is one of the most commonly used techniques for scaling and does exactly this. The formula is shown by Equation 6.1, with x the unscaled value, u the average value, s the variance of the values and z the scaled value. The scaling is applied for all numerical features, including the target feature, the observed vancomycin serum concentrations.

$$z = \frac{x - u}{s} \tag{6.1}$$

6.4 The model

The model consists of an LSTM network. As mentioned in Section 5, the weights of the network were fit to the training data using a Multi-objective Evolutionary Algorithm.

6.4.1 The LSTM network

The LSTM network should be able to map a multivariate time series of variable length to a single feature sequence of 48 time steps. This is done by the use of two LSTM layers and a dense layer. The first LSTM layer gets as input the multivariate time series of variable length. This layer uses the sequential information to predict the important information for the next time step in the dimension equal to the number of neurons in this LSTM layer. In this part, the model should learn which sequential information is necessary for the predictions of the next 24 hours. The number of neurons will be chosen in a grid search.

The important information found by the first LSTM layer needs to be scaled up to the output length, which is done by repeating the information over 48 time steps. Now all information is in the right place, a second LSTM layer is used to estimate the right pattern in the predicted sequence. By definition, the LSTM network uses the sequential information over the predictions, so it uses its prediction of the previous time step, to predict the next time step. This layer combines the information of the first LSTM layer and the sequential information to a sequence with at each time step the dimension of the number of neurons of the second LSTM layer. Finally, an output layer combines this information to only one prediction per time step.

This whole procedure is visualized in Figure 6.2, with $N1$ the number of neurons in the first LSTM layer, $N2$ the number of neurons in the second LSTM layer and T_{in} the number of time steps between the previous and the current dose.

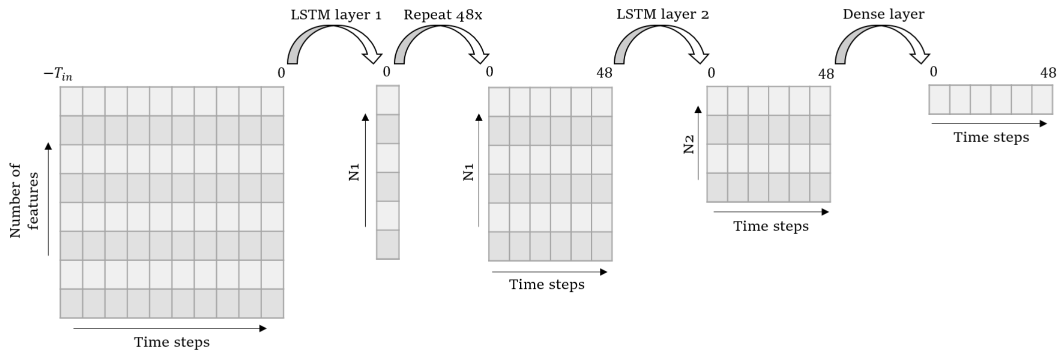


FIGURE 6.2: Model procedure.

To use the full history of a patient, the hidden state of the first LSTM layer is only reset after completing all sequences of the patient. This is not the case for the second LSTM layer, because the predicted time steps are not necessarily contiguous. When T_{in} of the next sequence is smaller than 48 time steps, the predictions of this and the next sequence will overlap, and when T_{in} of the next sequence is bigger than 48 time steps, there will be a gap between the predictions of this and the next sequence. This is not problematic while resetting the hidden state of the second LSTM layer after each prediction.

6.4.2 The Evolutionary Algorithm

In this research no attempt is made to find the best parameters of the NSGA-II algorithm, instead, the default parameters were used [48], which are in line with the recommended parameter values described in Section 5.4.2. The selected parameter values are shown in Table 6.4.

Parameter name	**Value**
Crossover probability	0.95
Distribution index for crossover	10
Mutation probability	0.01
Distribution index for mutation	50

TABLE 6.4: NSGA-II parameter settings.

The number of individuals per generation is set to 40 because it should be big enough to have diversity in the population but it should be small enough to limit the computation time.

The number of generations of the Evolutionary Algorithm will be investigated by comparing the results on the exploration and validation set for each generation between 0 and 1000 generations for both objective functions on a single run. Based on these results, the number of generations will be selected as the minimal number that the algorithm needs to converge on the validation set because there is a limited amount of time for this research. This number of generations will both be used in the grid search and in the final model.

6.5 Grid search

To select the best parameters of the model, a grid search was performed. The following parameter settings will be trained on the exploration set and evaluated on the validation set, see Table 6.5. Since the Evolutionary Algorithm is stochastic, each grid search setting will be used five times. The parameters with the biggest mean *hypervolume* will be chosen, as explained in Section 5.4.4.

Parameter name	**Evaluated values**
Feature set	basic, extended
Number of neurons in first LSTM layer	2, 4, 8
Number of neurons in second LSTM layer	2, 4, 8

TABLE 6.5: Grid search parameters and evaluated values.

Only small numbers are selected for the number of neurons in the LSTM layers because the complexity of the multi-objective problem for the Evolutionary Algorithm increases exponentially while the number of weights increases in the network, as explained in Section 5.4.1. By limiting the number of neurons, the complexity of the model is limited and the running time is kept manageable.

6.6 The benchmark

The best parameters found by the grid search are selected in the final model. In addition, the final model will be trained on the full training set and evaluated on the test set. To conclude how the final model performs, the results are compared with the results of a benchmark. The benchmark used in this research predicts always the mean vancomycin serum concentration for each time step. In this case, the mean serum concentration is calculated over the training set and evaluated on the test set.

6.7 Implementation

All implementation is done in Python^{TM}, which is a common programming language in data science. For the implementation of the Evolutionary Algorithm, the package *Pygmo* is used [48].

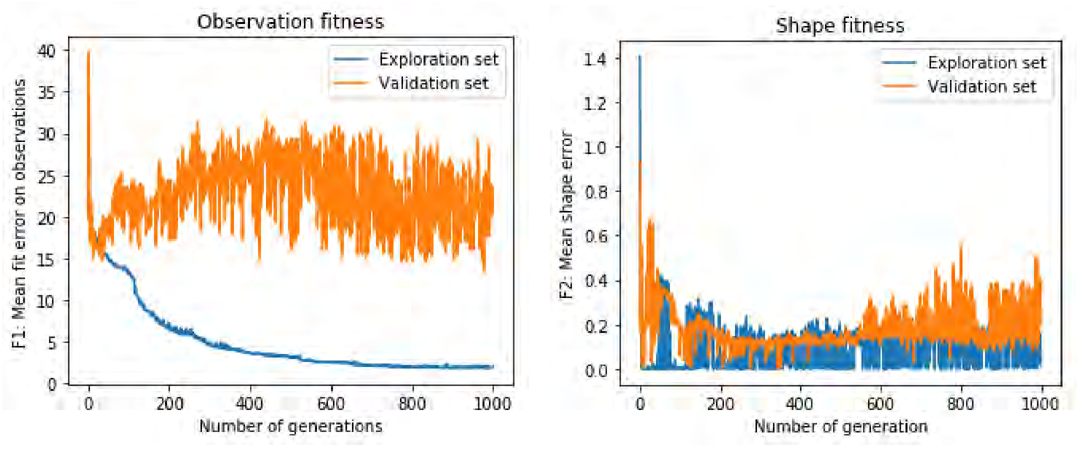
Everything is implemented on one Windows 10 laptop with the following specifications: 2,20 GHz Intel® Core^{TM} i7-8750H with 16GB RAM.

7 Results

As described in Section 6, first the number of generations of the Evolutionary Algorithm is investigated. After that, the grid search is performed and the best model parameters are selected. Finally, the model with the best parameters is compared with the benchmark.

7.1 Number of generations

The smallest error per generation, based on one run, on the observations and on the sequence shape are shown on both the exploration set and validation set in Figures 7.1a en 7.1b respectively. The used model was arbitrarily selected and uses the extended feature set and has four neurons in both the first and second LSTM layer.



(A) F_1 (B) F_2

FIGURE 7.1: Fitness on exploration and validation set over the generations.

There is a lot of noise in the observation-fitness on the validation set, see Figure 7.1a. For almost all generations, the mean square error on the observations fluctuates between 15 and 30 mg/L for each sequence on average. The error on the shape of the predicted sequences in the validation set remains approximately the same after 250 generations. Therefore, the number of generations is set to 250.

7.2 Grid search

As already mentioned in Sections 5 and 6, the performance of the parameters in the grid search are compared by the mean hypervolume over five runs. The results can be found below.

Feature set	N1	N2	mean hypervolume (scaled to 0-1)	standard deviation hypervolume (scaled to 0-1)
basic	2	2	0.9957	0.00203
basic	2	4	0.9884	0.00359
basic	2	8	0.9553	0.03662
basic	4	2	0.9963	0.00115
basic	4	4	0.9395	0.06263
basic	4	8	0.9802	0.00611
basic	8	2	0.9926	0.00748
basic	8	4	0.9841	0.01582
basic	8	8	0.9822	0.00819
extended	2	2	0.9945	0.00254
extended	2	4	0.9717	0.04583
extended	2	8	0.9848	0.01610
extended	4	2	0.9875	0.01383
extended	4	4	0.9887	0.00973
extended	4	8	0.9810	0.01136
extended	8	2	0.9941	0.00295
extended	8	4	0.9863	0.01005
extended	8	8	0.9251	0.03642

TABLE 7.1: Results after 250 iterations and 5 runs each.

Table 7.1 shows no significant improvement on the mean hypervolume for the extended feature set compared to the basic feature set. According to the results for the different number of neurons in the second layer, a lower number (2) seems to be better than a high number (4 or 8). For the number of neurons in the first layer, this is less clear. According to Table 7.1, the highest mean hypervolume is obtained for the basic feature set with four neurons in the first LSTM layer and two in the second LSTM layer. Therefore, these parameters are chosen in the final model.

7.3 Final model

The final model is trained on the full training set (30% of the patients) for 250 generations and will be evaluated on the test set (70% of the patients). For the benchmark, the average is taken over all observed vancomycin serum concentrations in the training set, which happens to be approximately 16.30 mg/L. The results are shown in Figure 7.2.

Since the final training of the weights of the LSTM model results in 40 fits in the last generation in the Evolutionary Algorithm, the best fit should be chosen based on the preference of the objectives. In this research, a preference for the fitness of the shape is assumed, because all errors on the shape are not possible in practice. Therefore, the model with the minimal shape error is chosen, see the blue dot in Figure 7.2.

According to Figure 7.2 and Table 7.2, both objective values (the prediction error and shape error) are smaller for the LSTM model, which suggests that the LSTM model outperforms the benchmark. In order to test the hypothesis that the mean prediction error on the sequences is smaller for the LSTM model than for the benchmark, a t-test can be performed. This test is applied with threshold level of α=0.05.

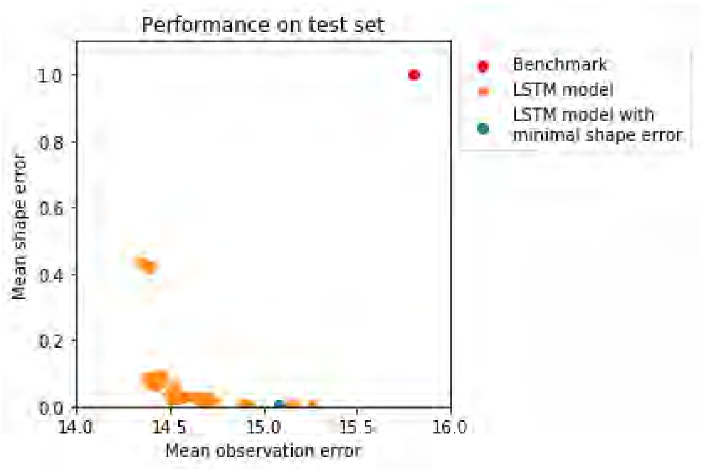
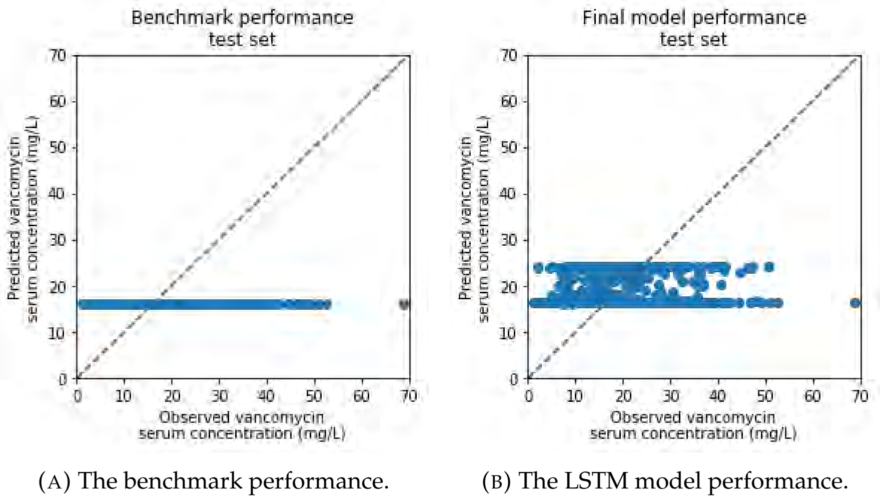


FIGURE 7.2: Results of the final LSTM model and benchmark on the test set.

The p-value was found to be 0.6211, which is bigger than α. Therefore, the null-hypothesis that the means are the same is not rejected, so the LSTM model is not significantly better in predicting the vancomycin serum concentrations than the benchmark. Despite the fact that the shape error is less for the LSTM model, the model performs no better on the observations than predicting the mean vancomycin serum concentration for every time step.

Model	Observation error	Shape error	Hypervolume (scaled to 0-1)
Benchmark	15.79834	1	0.98394
LSTM	15.07980	0.00238	0.99698

TABLE 7.2: Results of the final LSTM model and benchmark on the test set (6,744 sequences).



(A) The benchmark performance. (B) The LSTM model performance.

FIGURE 7.3: Predicted vancomycin serum concentrations versus real observations. The diagonal indicates perfect predictions.

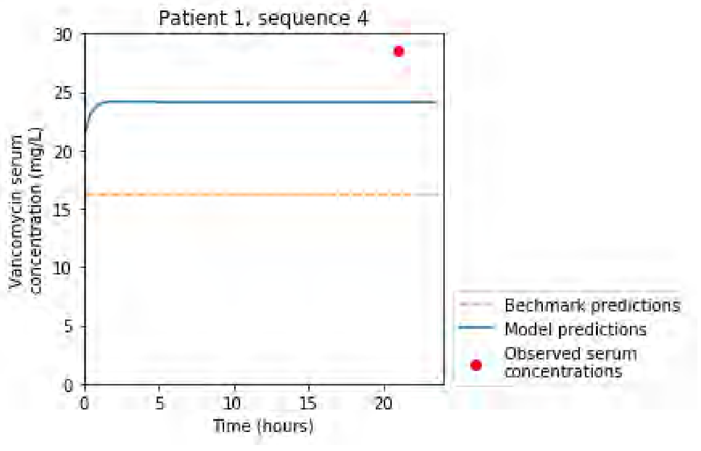


FIGURE 7.4: Predicted sequence of patient in test set.

For perfect predictions, all dots should be located on the diagonal of Figures 7.3a and 7.3b. However, the dots of the LSTM model seem to show the same kind of horizontal lines as the benchmark, which suggests predicting mean serum concentrations per sequence 'type'. In Figure 7.4, besides the observed vancomycin serum concentrations, the benchmark and model predictions are shown. This Figure confirms the suspicion that the LSTM model predicts a mean serum concentration per sequence 'type', instead of a serum concentration per time step. Only in the first three hours, there is some change in predicted serum concentration, but after that, the predictions remain constant.

By looking at Figure 7.3b, two 'types' of sequences can be distinguished. The first 'type' results in predictions of approximately 25 mg/L, and the second 'type' results in predictions of approximately 15 mg/L. When we split the sequences in sequences with a mean serum concentration bigger than 20 mg/L (sequence 'type' 1) and smaller or equal to 20 mg/L (sequence 'type' 2), we find the following differences, see Table 7.3.

Parameter	Sequence 'type' 1	Sequence 'type' 2
Number of sequences	6187	546
Mean predicted serum concentration	16.519	23.424
Mean dose size	939.567	767.125
Mean body weight	80.077	67.904
Percentage male	69.64%	39.93%
Mean age	60.603	65.908
Mean serum creatinine	93.800	211.529

TABLE 7.3: Differences in parameters of the predicted sequence 'types'.

From Table 7.3 can be observed that most sequences belong to 'type' 1 and that these sequences mainly concern patients with higher body weight (80 kg) than the patients of sequence 'type' 2. Since men are on average heavier (82 kg) than women (71 kg), we are not surprised to see a difference in the percentage being men too (70% versus 40%). Finally, a difference in mean serum creatinine of the two sequence 'types' is found too. The mean serum creatinine of sequences of 'type' 2 are approximately twice as high as of sequences of 'type' 1: 93.8 versus 211.5 $\mu mol/L$.

8 Discussion

In this research, an LSTM model is used to predict vancomycin serum concentrations in the first 24 hours after a vancomycin drug administration. For this purpose, two datasets were created: one historical dataset with the measurements until the drug administration, and a future dataset with records of the observed serum concentrations. For the historical dataset, it was decided to fill in the missing values with the median value, but it would probably be better to impute these values by values provided by domain experts because these values were assumed to be normal. In other words, if they were assumed to be different, they would be measured. Therefore, it might not be a good idea to use the measurements for imputing missing values.

The final model scores only slightly better than predicting the mean value (the benchmark). However, this was not found significant, for which multiple reasons can be given.

First of all, there where only a few observations available in the data. Which resulted in zero (78%) or only one (21%) observation for most sequences. Therefore, it is not strange that the model predictions show an (almost) straight line.

Secondly, most observed vancomycin serum concentrations were measured right before the next dose was given. This means that the observations are biased to the expectations of the intensivists because they decide on the daily dose of the medicine and the time interval between administrations [4]. When the intensivist expects a slow elimination of the drug, the next administrated dose (and possible observation) will be later than when the intensivist expects a normal drug elimination.

Thirdly, the long run analysis for the number of generations was only performed on a single run. It would be better to perform multiple runs because of the stochastic nature of the Evolutionary Algorithm. In addition, the grid search was only performed for a limited number of options, on a little amount of data and for only a few (5) iterations. Therefore, the best parameters may not be found. However, due to the limited amount of time for this research, it was not possible to expand the long run analysis and the grid search.

The model in this research was trained by an Evolutionary Algorithm, which resulted in applying the model on each sequence for each individual in each generation. Since the individuals in a generation are independent, it would be possible to run the computations of each individual in parallel. However, this was not implemented in the used package. Therefore, it might be a good idea to use a different implementation of the Evolutionary Algorithm, in order to speed up the training phase.

8.1 Future Work

In this research, a dependency between consecutive sequences was assumed. However, it would be interesting to investigate the effect of resetting the state after each sequence of a patient. If there is no significant difference in the prediction error, there is no specific advantage of using an LSTM above the usage of other Machine Learning algorithms.

Future work could be done by applying the model on a dataset with more observations to investigate if the model is able to detect the sequential pattern of the pharmacokinetics of vancomycin.

Finally, it would be useful to compare the final results of the LSTM model with results from NONMEM®, because that program is the current standard for pharmacokinetics.

9 Conclusion

The LSTM model proposed in this research was not able to predict the vancomycin serum concentrations in critically ill patients more accurately than the benchmark, which predicted always the mean serum concentration. The most important reason for the low performance suggested in this research is the small number of observations per sequence (interval between two doses). Therefore, the model was not able to learn how the vancomycin drug was eliminated by the body over time.

Bibliography

[1] Avent ML, Vaska VL, Rogers BA, Cheng AC, van Hal SJ, Holmes NE, et al. Vancomycin therapeutics and monitoring: a contemporary approach. Internal Medicine Journal. 2013 February;43(2):110–119.

[2] Marsot A, Boulamery A, Bruguerolle B, Simon N. Vancomycin: A Review of Population Pharmacokinetic Analyses. Clinical Pharmacokinetics. 2012 January;51(1):1–13.

[3] Tolle KM, Chen H, Chow HH. Estimating drug/plasma concentration levels by applying neural networks to pharmacokinetic data sets. Decision Support Systems. 2000 December;30(2):139–151.

[4] Hu PJ, Cheng T, Wei C, Yu C, Chan ALF, Wang H. Managing Clinical Use of High-Alert Drugs: A Supervised Learning Approach to Pharmacokinetic Data Analysis. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans. 2007 July;37(April):481–492.

[5] Risi S, Togelius J. Neuroevolution in Games: State of the Art and Open Challenges. arXiv csNE. 2015 October;Available from: https://arxiv.org/pdf/1410.7326v3.pdf.

[6] Salimans T, Ho J, Chen X, Sidor S, Sutskever I. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. arXiv statML. 2017 September;Available from: https://arxiv.org/pdf/1703.03864.pdf.

[7] Birkett DJ. Therapeutic drug monitoring. Australian Prescriber. 1997 January;20(1):9–11.

[8] Kang JS, Lee MH. Overview of Therapeutic Drug Monitoring. The Korean Journal of Internal Medicine. 2009 March;24(1):1–10.

[9] Rybak M, Lomaestro B, Rotschafer JC, Moellering R, Craig W, Billeter M, et al. In: Therapeutic monitoring of vancomycin in adult patients: a consensus review of the American Society of Health-System Pharmacists, the Infectious Diseases Society of America, and the Society of Infectious Diseases Pharmacists.. vol. 66; 2009. p. 82–98.

[10] Vancomycin; 2018. [Online; accessed 31-Aug-2018]. Available from: https://labtestsonline.org/tests/vancomycin.

[11] Burton ME, Shaw LM, Schentag JJ, Evans WE. In: Applied Pharmacokinetics & Pharmacodynamics: Principles of Therapeutic Drug Monitoring. Lippincott Williams & Wilkins; 1992. p. 330–331.

[12] Nankivell BJ. Creatinine clearance and the assessment of renal function. Australian Prescriber. 2001 January;24:15–17.

[13] National Kidney Foundation. Creatinine: What is it?; 2017. [Online; accessed 24-June-2018]. Available from: https://www.kidney.org/atoz/content/what-creatinine.

[14] Boeckmann AJ, Sheiner LB, Beal SL. In: NONMEM Users Guide - Part V. Introductory Guide; 1994. p. 1–7. Available from: https://nonmem.iconplc.com/nonmem720/guides/v.pdf.

[15] Miller R. Population Pharmacokinetics. In: Principles of Clinical Pharmacology. 3rd ed.; 2012. p. 139–149.

[16] Nonlinear Mixed-Effects Modeling;. [Online; accessed 15-February-2019]. Available from: https://nl.mathworks.com/help/simbio/ug/what-is-nonlinear-mixed-effects-modeling.html.

[17] Roberts JA, Taccone FS, Udy AA, Vincent JL, Jacobs F, Lipman J. Vancomycin Dosing in Critically Ill Patients: Robust Methods for Improved Continuous-Infusion Regimens. Antimicrobial agents and chemotherapy. 2011 June;55(6):2704–2709.

[18] Choi E, Schuetz A, Stewart WF, Sun J. Using recurrent neural network models for early detection of heart failure onset. Journal of the American Medical Informatics Association. 2017 March;24(2):361–370.

[19] Lipton ZC, Kale DC, Elkan C, Wetzel R. Learning to Diagnose with LSTM Recurrent Neural Networks. arXiv ICLR 2016. 2017 March;Available from: https://arxiv.org/pdf/1511.03677.pdf.

[20] Chow HH, Tolle KM, Roe DJ, Elsberry V, Chen H. Application of neural networks to population pharmacokinetic data analysis. Journal of Pharmaceutical Sciences. 1997 July;86(7):840–845.

[21] Brzaković B, Vučićevic K, Kovačević SV, Miljković B, Prostan M, Ž Martinović, et al. Application of counter-propagation artificial neural networks in prediction of topiramate concentration in patients with epilepsy. Journal of pharmacy & pharmaceutical sciences. 2015;18(5):856–862. Available from: https://journals.library.ualberta.ca/jpps/index.php/JPPS/issue/view/1729.

[22] Delicourt A, Bussières JF, Lebel D. Pediatric pharmacokinetics of vancomycin: a Canadian perspective. The Canadian journal of hospital pharmacy. 2011;64(2):156–157.

[23] Beierle I, Meibohm B, Derendorf H. Gender differences in pharmacokinetics and pharmacodynamics. International journal of clinical pharmacology and therapeutics. 1999 12;37:529–47.

[24] Rybak MJ. The Pharmacokinetic and Pharmacodynamic Properties of Vancomycin. Clinical Infectious Diseases. 2006 01;42:S35–S39. Available from: https://dx.doi.org/10.1086/491712.

[25] Viswanath V. Deep Learning - ANN, RNN, LSTM networks; 2017. [Online; accessed 28-June-2018]. Available from: https://vishnuviswanath.com/ann_rnn_lstm.html.

[26] Graves A. Supervised Sequence Labelling with Recurrent Neural Networks. vol. 385 of Studies in Computational Intelligence; 2012. p. 16–45.

[27] Karn U. A Quick Introduction to Neural Networks; 2016. [Online; accessed 28-June-2018]. Available from: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/.

[28] Olah C. Understanding LSTM Networks; 2015. [Online; accessed 27-June-2018]. Available from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[29] Medsker LR, Jain LC. Recurrent neural networks. Design and Applications. 2001;5:12–13. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.375.5562&rep=rep1&type=pdf.

[30] Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press; 2016. http://www.deeplearningbook.org.

[31] Yan S. Understanding LSTM and its diagrams; 2016. [Online; accessed 31-Aug-2018]. Available from: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714.

[32] Greff K, Srivastava RK, Koutnìk J, Steunebrink BR, Schmidhuber J. LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks and Learning Systems. 2017 October;28(10):2222–2232. Available from: http://www.jmlr.org/papers/volume3/gers02a/gers02a.pdf.

[33] Graves A. Generating Sequences With Recurrent Neural Networks. arXiv csNE. 2014 June;Available from: https://arxiv.org/pdf/1308.0850.pdf.

[34] Zocca V, Spacagna G, Slater D, Roelants P. In: Python Deep Learning. Packt Publishing Ltd; 2017. p. 175–177.

[35] Floreano D, Dürr P, Mattiussi C. Neuroevolution: from architectures to learning. In: Evolutionary Intelligence. vol. 1. Springer; 2008. p. 47–62.

[36] Bäck T, Foussette C, Krause P. Contemporary Evolution Strategies. Natural Computing Series. Springer; 2013.

[37] Deb K. Multi-objective optimization using evolutionary algorithms: An introduction. KanGAL Report. 2011;(2011003).

[38] Ehrgott M. Multicriteria optimization. vol. 491. Springer Science & Business Media; 2005.

[39] Miettinen K. Nonlinear multiobjective optimization. vol. 12. Springer Science & Business Media; 2012.

[40] Srinivas N, Deb K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary computation. 1994;2(3):221–248.

[41] Deb K, Agrawal S, Pratap A, Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: International conference on parallel problem solving from nature. Springer; 2000. p. 849–858.

[42] Salimi R, Bazrkar N, Nemati M. Task scheduling for computational grids using NSGA II with fuzzy variance based crossover. Advances in Computing. 2013;3(2):22–29.

[43] Configuring the Non-Dominated Sorting Genetic Algorithm (NSGA-II) Technique;. [Online; accessed 25-February-2019]. Available from: https://abaqus-docs.mit.edu/2017/English/IhrComponentMap/ihr-t-Optimization-NSGAIIConfig.htm.

[44] Bhuvaneswari M, Shanthi M. Design of Operational Amplifier. In: Application of Evolutionary Algorithms for Multi-objective Optimization in VLSI and Embedded Systems. Springer; 2015. p. 47–67.

[45] Bhuvaneswari M, Subashini G. Scheduling in Heterogeneous Distributed Systems. In: Application of Evolutionary Algorithms for Multi-objective Optimization in VLSI and Embedded Systems. Springer; 2015. p. 47–67.

[46] Perez RE, Jansen PW, Martins JRRA. pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization. Structures and Multidisciplinary Optimization. 2012;45(1):101–118.

[47] LeCun YA, Bottou L, Orr GB, Müller KR. Efficient backprop. In: Neural networks: Tricks of the trade. Springer; 2012. p. 9–48.

[48] Biscani F, Izzo D, Jakob W, Märtens M, Mereta A, Kaldemeyer C, et al.. esa/pagmo2: pagmo 2.10; 2019. Available from: `https://doi.org/10.5281/zenodo.2529931`.