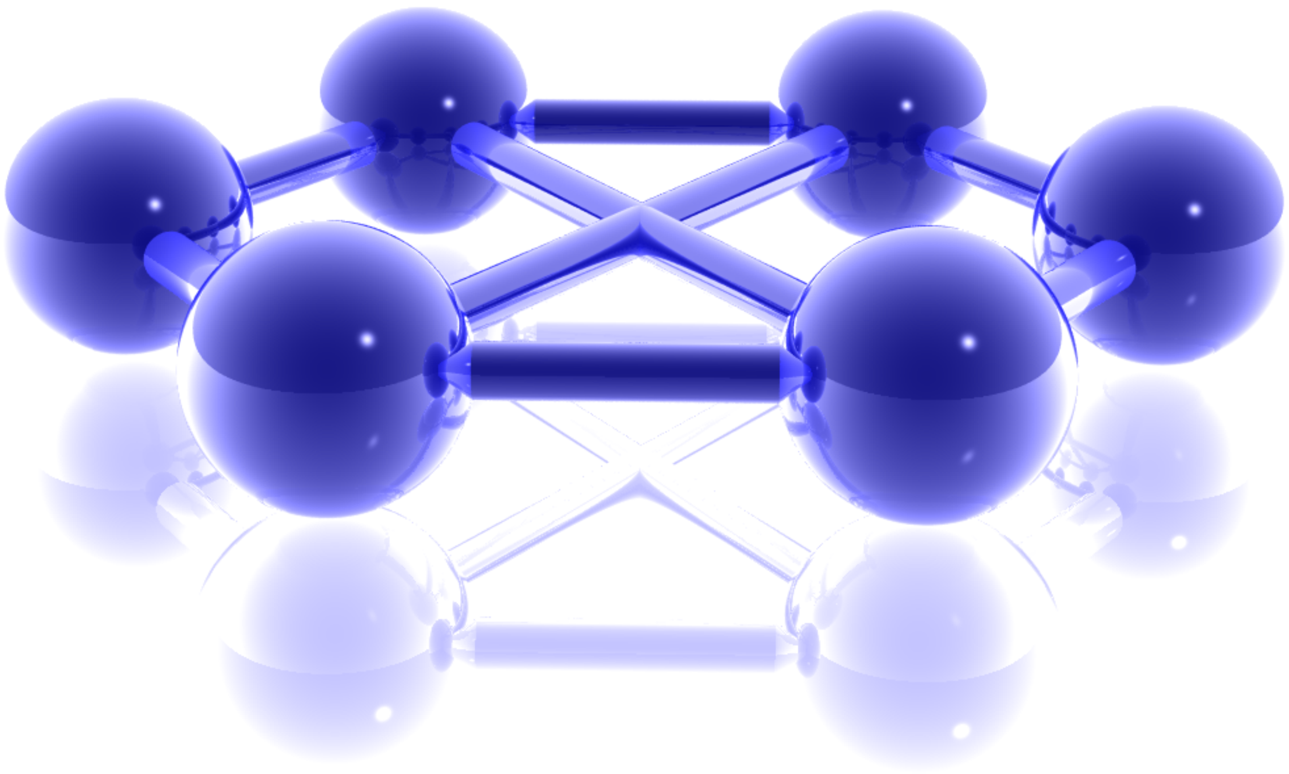
Predicting blood glucose levels of diabetics using artificial neural networks

Peter Kok

Research Assignment for the Master of Science program Computer Science,
Delft University of Technology, 2004



Foreword

This report is the result of a research assignment done as a part of the Master of Science program Computer Science at the Delft University of Technology. It represents approximately 420 hours of work. The assignment was supervised by prof.dr.ir. E.J.H. Kerckhoffs.

It is assumed that the reader has a basic knowledge of artificial neural networks, although this is not absolutely necessary. If you are not interested in the details of neural networks, chapters 1, 3.1-3.3, 4, 7 and 8 are recommended. Chapter 2 gives a detailed background of available research in diabetes management using various kinds of control systems. If you are only interested in the results, chapter 7 is also recommended besides chapters 1 and 8.

More information on this project can be found at: http://www.vexel.nl/igc

Contents

Summary

To prevent complications in diabetes, a tight control of the blood glucose level is needed. The level should ideally be kept between 4 and 8 mmol/l. In the most common treatment of type I diabetes, the diabetic measures the blood glucose level several times a day, based on which an optimal insulin dosage can be determined. However, there are a number of other factors that affect the blood glucose level, such as carbohydrate intake, level of exercise and level of stress. It is difficult to take all factors into account and predict if the resulting glucose level will be acceptable. The goal of this research assignment is to determine if it is possible to construct an artificial neural network that accurately predicts the blood glucose level, using a number of variables as inputs.

In literature, different kinds of systems can be distinguished. On one side there are model-based approaches that try to predict the glucose level by using a model of the glucose metabolism. On the other side are approaches based on reasoning. The strategy we implemented is a model-based approach, as we try to model the input-output behaviour of a diabetic (but we don't model any of the internal processes explicitly). The system is split up to make separate predictions for the morning, afternoon, evening and night. The inputs are the available variables at the start of the interval and the output (the prediction) is the resulting blood glucose level at the end of the interval. If a good predictor is obtained, it could be used to generate an advice for (for example) the insulin dosage by varying the input and evaluating the resulting glucose level. We only focus on making a prediction.

The artificial neural networks were trained with data from one patient, covering a period of 77 days. We then determined the optimal values for a number of parameters by comparing the results of identical training sessions while varying one parameter.

The effect of certain inputs on neural network performance during different parts of the day was studied. For example, during the night a neural network trained with long acting insulin as an input performed much better than one without. For the other intervals this effect was smaller. A number of input combinations were evaluated.

The effect of neural network architecture on performance varied. A network for the morning interval performed better having a small size, while the results on other intervals were better with larger networks. Single-hidden-layer and double-hidden-layer networks performed comparable.

Variation in the training parameters was also studied. Performance was good in all cases for a learning rate between 0.1 and 0.2 and a momentum between 0.1 and 0.7. On-line training seemed to give more predictable results than batch training.

The best neural networks made predictions with reasonable accuracies, which are comparable to some systems mentioned in literature. A lot more testing needs to be done to be able to say if the resulting system could be used in practice. There are a number of drawbacks to the neural-network approach. For example, the fact that a neural network is a black box prevents us from analysing and explaining its internal reasoning.

1. Introduction

Diabetes is a disease that is characterized by an elevated blood glucose level. This can be caused by a reduction of the production of insulin by the pancreas (type I diabetes) or by the insulin being less effective at moving glucose out of the blood stream and into cells that need it (type II diabetes). A blood glucose level that is elevated for a long period of time can result in vascular, neurological or metabolic complications, such as kidney failure, blindness and an increased chance of heart attacks. To prevent or postpone such complications, a strict control over the diabetic's blood glucose level is needed. Most type I patients are under conventional or intensified insulin therapy, which means that a number of subcutaneous insulin injections are administered each day. In another treatment, which is less often seen, insulin is administered using an external or implanted pump. Usually, a patient uses basal insulin that acts slowly, as well as normal insulin that acts fast and is used to cope with the glucose that needs to be processed during a meal. Patients often measure their blood glucose levels several times a day. These measurements enable them to take control actions, such as adjusting the insulin dosage or taking extra food. Ideally, blood glucose levels should always be kept between 4.0 and 8.0 mmol/l. Type II diabetes can be treated with medication, a diet, exercise or sometimes with insulin injections. Our focus will only be on diabetes treated with insulin injections, although some of the ideas may also be applied to the other treatments.

There are several factors that affect the blood glucose level, such as the amount of food, the insulin dosage, the level of exercise and the level of stress. In addition, there are a number of internal processes, such as absorption and production of glucose by the liver and renal excretion through urine. The large number of factors makes it difficult to predict how the glucose level will behave in, for example, the next few hours. It is desirable to be able to make an accurate prediction, so that control actions can be taken *before* the glucose level goes beyond the ideal bounds. The effects of the different factors are often not accurately known. They are dependent on again other factors, are different from person to person and vary over time. It can however be assumed that under the same circumstances, two similar situations will result in the same glucose level for the same person. If we can describe a 'situation' numerically, it is also reasonable to assume that we can interpolate between two situations that are different and in this way can evaluate a third that is located 'between' them. The problem is that here we have to deal with an unknown non-linear function that is multi-dimensional in its inputs, varies over time and is patient specific. In addition, we only have a limited number of known data points of this function, which, last but not least, contain a certain amount of noise. Artificial neural networks have certain properties that are ideal for approximating such a function.

The goal of this research assignment is to determine if it is possible to construct an artificial neural network that accurately predicts the blood glucose level. Or in other words: can we construct a neural network that simulates the input-output behaviour of a diabetic patient? To come to a solution, we will answer the following questions:

- What kind of research has already been done on how neural networks can be used to predict glucose levels?
- Which factors are most important in affecting the blood glucose level and how can these factors be expressed numerically?
- What is the best neural network architecture?
- What are the optimal learning parameters?
- How well does the resulting neural network perform and is this performance acceptable enough for the system to be used in practice?

In chapter 2 we give an overview of available research in the field of neural networks and diabetes management. Here we especially look at approaches that can help us to perform our own research. In chapter 3 we explain the way the research was conducted, where the data came from and what software was used to set up a neural network. Chapter 4 describes the selection of the inputs for the neural network. This comes down to identifying the most important factors, expressing them numerically and evaluating which combination works out best. In chapter 5, the selection of the optimal network architecture is discussed and in chapter 6, the optimal learning parameters are determined. Both problems are tackled by comparing the performance of neural networks that were trained with different parameters. In chapter 7 we discuss the performance of the obtained neural network(s) and try to determine if they could be used in real life. Here we also identify the problems and limitations of our approach. Finally, in chapter 8, we reach a conclusion and give a number of recommendations for future research. Appendix A contains more elaborate results than given in this document, mostly in the form of tables and plots. Appendix B explains a number of medical terms in Dutch.

2. Earlier research

In this chapter earlier research in the area of insulin dosage determination is discussed. Insulin dosage determination is closely related to blood glucose level prediction, as the latter is often a part of the process to come to an optimal insulin dosage. We will first give a short overview of the use of neural networks in diabetic treatment. Then we describe different approaches that are often used for managing insulin dosage. After that, we will take a closer look at how neural networks come into play. Finally, we list a few systems that have been implemented.

As we look at previous research, we try to keep in mind what we want to achieve with our study. Even if the research is only remotely related to ours, it is interesting to see how other researchers tackle specific problems. We are interested in the data that was used (how many people, over what period of time, etc.), in what data was selected as inputs and in the performance of the model. When neural networks are used, we also want to know what neural network architecture is used and what the learning parameters are.

2.1 Neural networks in diabetic treatment

Neural networks are widely used in the field of health care for a range of different purposes. Also in the field of diabetic treatment neural networks can be found, for example for diagnosing diabetes, selecting a proper treatment based on a number of variables or simply for supporting the decision making process of a physician. But neural networks are becoming more common in diabetic management as well. Insulin pumps for example can be equipped with an insulin dosage controller that is driven by a model based on a neural network. In these cases, a neural network is sometimes used to calculate the value of certain patient specific parameters. For glucose level prediction, neural networks are not often used. Some research in this field has been done however, as we will describe below.

Here we want to stress the difference between the insulin therapy and the insulin dosage. The insulin therapy consists of the number of injections, the time of injection, the type(s) of insulin and the insulin dosage. The first is often only a directive and patients can adjust the actual insulin dosage depending on the situation (current glucose level, anticipated level of exercise, anticipated food intake, etc.). This is what is meant by determining the insulin dosage. The determination of the optimal insulin therapy can in some ways be seen as a classification problem, which has also been tackled with neural networks [1, 2]. The determination of the optimal insulin therapy is not in the scope of our research however. It is assumed to be prescribed by a physician and we will not go into this any further.

2.2 Modelling versus reasoning

Lehmann and Deutsch describe two methods commonly used to aid the insulin dose determination: the compartment model and the algorithmic approach [3]. The first tries to model the internal biological processes that affect the glucose level. This way a prediction of the glucose level can be made. The compartment model is composed of a number of smaller models. Each sub-model models a specific aspect that affects the blood glucose level, such as glucose excretion through urine. The total model is then fitted to a specific patient by determining the value of a number of patient specific variables, such as different kinds of insulin sensitivities. The problems with compartment models named by Lehmann and Deutsch are:

- The fact that there is insufficient medical knowledge to specify a model that exactly describes the real system. For example: different types of food will act differently and the combination of different foods might be different from that. Such processes are simply not known.

- Methodological difficulties exist because of the complexity of the system. The sub-models may be specified quite accurately, but when combining them, the errors add up.
- Data that is collected by the patient to test the system is often not representative of a realistic situation. Often data is collected under unusual circumstances, for example where the patient is subjected to a diet.

Lehmann and Deutsch therefore conclude that compartment models are not ideal for managing insulin dosage; they still have use for research and education however. For managing the insulin dosage, the algorithmic approach might work out better. The algorithmic approach tries to model the way clinical specialists reason to recommend a proper insulin dosage. A number of rules are used to suggest stepwise changes to the insulin dosage when the blood glucose is at an undesired level for a specific number of times. In addition, rules to cope with short-term changes, such as a large meal, can be included. This approach can be directly mapped to the way insulin dosage is controlled clinically. If needed, a motivation supporting the suggested insulin dosage can be given.

Although compartment models and algorithmic approaches are not the only techniques, they are representatives of two classes of diabetes management systems. The first bases its reasoning on a model of the internal processes of a person and generates a prediction of the blood glucose level. The second bases its reasoning on the way physicians reason and is only used to give advice.

2.3 Diabetes management as a control system

Insulin therapy can be seen as a control system, where insulin is used to control the blood glucose level. For non-diabetics, the body continuously ‘monitors’ the blood glucose level and the pancreas releases insulin if needed. For diabetics, a limited number of measurements of the blood glucose level are used (among others) as feedback to determine the optimal insulin dosage. Controlling the insulin dosage can be done by the patient or by a physician. When we employ a computer system to take over control, a distinction between different approaches can be made based on the time scale that is used:

- If the goal is to create an artificial pancreas, the system will have to continuously measure the glucose level and taking action in the form of releasing insulin is possible at any point in time. This is a closed loop system. The system is in full control over the amount of insulin that is released. These kinds of systems are always based on some kind of model.
- If the goal is to create an insulin dosage advisor for a diabetic using injections, there is no continuous access to the body. Measurements are infrequent and the number of insulin injections that can be administered is limited. The system is not in full control over the insulin dosage; it only makes a suggestion. This can be done in a number of ways:
    - Using a model that calculates the effect of the insulin similar to the way this is done for artificial pancreata. Here, the model is an open loop system, because there is no feedback after the starting point. If we count the feedback at the end point, the system can be seen as a partially closed loop system.
    - Using a model that calculates the effect of the insulin in one step as if it is a controller of a closed loop control system itself (like in the artificial pancreas), but on a larger time scale. This is the kind of system we want to create.

- Using reasoning to suggest the insulin dosage, for example with an algorithmic approach.

An overview of different kinds of control systems is given by Carson and Deutsch [4].

The scheme where a patient receives a limited number of insulin injections a day is not the optimal one. This is because the insulin is administered in bursts instead of a continuous flow, which is the case for non-diabetics and for diabetics using insulin pumps. Fortunately, a number of insulin types have been developed that closely mimic the way the insulin is released after a meal for non-diabetic people. However, this still complicates the process of determining the optimal insulin dosage. We will give an overview of control systems for artificial pancreata as well as control systems for diabetics using injections.

With a combination of an insulin pump and a good controller it is possible to create artificial pancreata. Bellazzi et al [5] describe several strategies for designing a controller for such a system. It is in this area of diabetes management that neural networks are most often seen. Bellazzi et al name insulin, meals and physical exercise as the most important control variables. It is however problematic for patients to accurately quantify meals and physical exercise. For this reason, meals and physical exercise are often assumed to be fixed and only the insulin dosage is taken into account as a variable.

Trajanoski et al [6] describe a technique that is a combination of a neural network and a non-linear model predictive control technique. The resulting controller is then used for closed loop control with subcutaneous glucose measurement and continuous subcutaneous infusion of monomeric insulin analogues. This is in contrast to other artificial pancreata where insulin infusion and glucose measurement are done in a vein. The main problem with the investigated approach is that there are time delays concerning the insulin absorption and glucose measurement resulting from the fact that both are subcutaneous. A radial basis function neural network was used for identification of the input-output function that predicts the glucose level. The inputs here are the glucose levels and the insulin infusion rate, both over a specific period of time in the past. The output is the resulting glucose level. After adding noise to this function, the parameters of the radial basis function neural network were identified using a mathematical model that simulated the input-output behaviour of a diabetic patient. The resulting controller enabled a stable control of blood glucose levels only in the absence of sudden changes in the system such as meals. During meals, the controller had to be switched off.

Tresp et al [7] compare recurrent neural networks, time series convolution neural networks, linear models and non-linear compartment models to model the blood glucose metabolism. A linear error model is used to cope with uncertainty and missing blood glucose measurements. All models use a 15-minute time step. The recurrent neural network with the linear error model seemed to perform best. The inputs that were used for this model were:

- dosage of short acting insulin
- dosage of long acting insulin
- amount of food intake with 'fast' carbohydrates
- amount of food intake with 'medium' carbohydrates
- amount of food intake with 'slow' carbohydrates
- duration of regular exercise
- duration of intense exercise
- past glucose level estimates

The output was the resulting glucose level. The explained variance of the best model that was obtained is 45%, corresponding to 51 mg/dl (2.8 mmol/l). Tresp et al also state that blood glucose measurements seem to be highly stochastic and "that the standard deviation of the residual error which cannot be explained by the inputs is around 54 mg/dl" (3.0 mmol/l). These results are obtained when observing the blood glucose level of a diabetic on different days with identical diets, insulin injections and activities. Therefore, the model can be seen as an improvement.

A different approach is taken by Andreassen et al [8]. They describe a probabilistic network that models the carbohydrate metabolism in a way similar to a compartment model. By using this technique, a probability can be given for the results of each step in the model and for the predicted glucose level. The resulting model has an average prediction error of 3.3 mmol/l.

2.4 Neural networks and insulin injections

Research on the use of artificial neural networks in diabetes management of the most common treatment, with a number of insulin injections a day, is rare. One problem is that most times, glucose measurements are available only for a limited number of points during the day, often corresponding to the meals. Using a controller as it is used in an artificial pancreas is difficult, because this requires a much higher sampling rate. Bellazzi et al [5] state that at least eight measurements each day are needed to be able to reconstruct the glycaemic profile. This is one of the reasons that strategies for diabetes management in this type of therapy are always sub optimal solutions.

Other likely causes for the limited amount of research are problems with the available clinical data, such as the amount, completeness and consistency. These problems are named by Lakatos et al [1]. They describe a way in which an optimal insulin regimen could be suggested by a neural network approach. Data of the previous day, such as blood glucose measurements, carbohydrate intake and short and long acting insulin, is used as input and the recommended insulin regimen as output. The data that is available to train the neural network doesn't contain the recommended insulin regimen however, so first a model is constructed that can predict the glucose level. This model will then be used to recommend the insulin dosage that results in the optimal glucose level. A network was built with inputs for 4 blood glucose measurements during a day, 4 carbohydrate intakes, 2*4 insulin dosages, sex, age, weight and height. There were four outputs corresponding to the four resulting blood glucose levels during the day. To be able to take hypoglycaemic and hyperglycaemic events into account, a second network was constructed with a time resolution of one hour. A cubic spline interpolation technique was then used to provide a blood glucose profile during the day. The second network had a 100-50-24 architecture. The combination of the two neural networks can then be used to find and suggest the optimal insulin regimen. Unfortunately, the approach was not yet evaluated at the time of writing of their article. The parameters of the neural networks, such as the number of hidden layers, had not been optimised yet. There does exist some similar research however, which we will discuss now.

Andrianasy and Milgram [9] use a neural network to recommend a pump insulin dose. The way this is done might also be applied to regular insulin treatments, albeit with a few changes. The most important aspect that makes the method usable for regular insulin treatment is that the authors base their approach on *discontinuous* blood-glucose measurements, instead of a continuous stream of measurements, as is more common in case of a controller. A day is divided into 10 intervals at fixed times. For each interval a back propagation neural network is trained with a number of inputs, consisting mostly of previous glucose measurements and insulin dosages, including those of the previous intervals and the same interval on the previous day. Each network contains a layer with four hidden nodes. This way the system ultimately consists of ten 12-4-1 networks. Although several other factors that affect the glucose level, such as activity and meal content, are identified, these are not included in the neural network. The output corresponding to the inputs is the insulin dosage that has been prescribed by the medical staff based on the inputs. This way, the neural network is taught to model the insulin advisory strategy of the medical staff. The mean relative error of the resulting networks ranged from 0.06 to 0.15.

A different approach is taken by Liszka-Hackzell [10]. The principal component method and neural networks are combined to generate a model that is used to predict the glucose level. Factors that are identified to influence a patient's blood glucose are diet, insulin dose, glucose metabolism, insulin sensitivity, body mass index, level of activity and level of stress. Because the effect of some factors can be very different from person to person, the model is created for only one specific patient. It is stated that in the glucose metabolism, the value of a variable at a specific time depends on its value in the past, up to the correlation time. The information during this interval can be used to make a prediction. But because the process itself also changes in time, additional action is needed to create a reliable model. First, daily morning and evening measurements of a single patient for a period of about 2 years were studied for recurring patterns. To do this, a frequency analysis was done using a wavelet transform. Several long-term patterns could be recognized with 7, 31, 56, 83 and 111-143 day periods. Principle component analysis is then used to transform the data to a space that already incorporates the (for example) 7-day variation, in an attempt to smooth out the chaotic behaviour. The 6 most significant principal components are then considered in a 7-point (day) window, giving 42 inputs for the neural network. The network has 95 hidden nodes and 6 outputs. For 15 days after the last training data, the performance of the model is quite good with a correlation coefficient of 0.76 and an average difference of 0.33 mmol/l. After that performance deteriorates. Therefore, it is recommended to create a new model once a week. Another interesting result of this research is that morning and evening glucose values seem to be completely uncorrelated, so blood glucose levels may be assumed to be uncorrelated after a 12-hour period. This fact is also brought up by Tresp et al [7], who go even further. They state that research has shown that the blood glucose level stabilizes after one hour. In other words, the blood glucose level of more than one hour ago does not have any direct effect on the current glucose level.

2.5 Implemented approaches

Several systems have been designed that can suggest the optimal insulin dosage using a form of the approaches as described above. A summary of some of them is given by Bellazzi et al [5]. However, none of them make use of neural networks for glucose level prediction or insulin dose determination.

One of these systems is ‘The Diabetes Advisory System’ (DIAS). Using blood glucose measurements, food intake and past insulin injections an insulin dosage is suggested. This suggestion is based on a discrete-time finite-state stochastic model of the glucose-insulin system. The model has two unknown variables that are patient specific. The values of these parameters are determined by the system. A cost function is minimized to determine the optimal insulin dosage. Hyperglycaemic events have a high cost and hypoglycaemic events are even more costly. A double blind experiment with two groups of six patients gave only moderately good results. The performance was measured using the HbA1c value as an indicator.

Another example of a system that has been implemented is ‘The Automated Insulin Dosage Advisor’ (AIDA). This system is based on a combination of a rule-based system and a non-linear dynamic model of the glucose-insulin system to make predictions. This combination first constructs a typical daily profile of the patient. Then possible problems are detected by the rule-based part and a solution is suggested in the form of an insulin dosage. The system was tested with 30 patients for a period of 5-6 days. The root-mean-square error of the predictions was 34.5 mg/dl (1.9 mmol/l) and it is believed that the performance can be increased.

Yet another system is the ‘Telematic Management of Insulin Dependent Diabetes Mellitus’ (T-IDDM). This system consists of two units, a patient unit and a medical unit, which are connected through the Internet. The patient unit is used to record data and give suggestions. The medical unit contains a decision support system that generates the recommended therapy using:

- a sophisticated simulation tool
- a systematic way to deal with heuristics that are used to determine the therapy
- the ability to identify the context in which the decisions are to be taken

A rule-based reasoner and a case-based retrieval system are used to provide this functionality. The system was tested on 12 patients and physicians considered the performance acceptable.

POIRO is a system implemented on a handheld device. A description is given by Holman et al [11]. POIRO suggests an insulin dosage using algorithms that are modified for each patient. The patient can enter the glucose level, insulin injections, the size of a meal, the level of exercise and the state of health. Before a meal, the anticipated values of the size of meal and level of exercise need to be entered, as well as the glucose level (if known) and state of health. Time and severity of hypoglycaemic events can also be recorded. Based on the available data, an insulin dosage is suggested. The system considers short-term effects (such as a larger meal size than usual) as well as long-term effects (such as a continuously high glucose level) to come to an advice. The system was tested on six patients for a period of four weeks, two weeks of which the system was turned on. The average glucose level before meals decreased from 8.9 to 7.5 mmol/l.

2.6 Conclusion – earlier research

There has not been done much research on using neural networks to predict blood glucose levels and useful results have not been found in literature. However, there does exist research on other kinds of systems, which we will use to evaluate our own approach.

The approach we take is a model-based technique, although we don’t model any internal processes explicitly. The approach can be seen as a closed-loop control system with a large time step (a quarter of a day). This is different from controllers for artificial pancreata, which have *continuous* access to the body for measuring blood glucose and releasing insulin.

3. Research set-up

In this chapter we first give a description of the approach we are going to take and of how we think this could be used in diabetes management. Then we identify the parameters that need to be tuned to get an optimal neural network. After that, a global overview is given of the process from recording the raw data until the testing of the performance of the resulting neural network. Here we will also set some of the parameters that we will not be determining an optimal value for. The optimal values of the remaining parameters are determined in the next three chapters.

3.1 The approach

The approach we want to investigate is that of a back propagation neural network that can be used to predict the blood glucose level before the next meal based on a number of variables. This method is similar to some of those described in the previous chapter. The neural network is trained with data recorded by one patient, over a period of several months. The idea is that if a situation occurs that is similar to a previous one, it will have a similar outcome.

The resulting model could then be used to select the optimal values of the input variables, which is any combination of values that results in an acceptable glucose level (the output). This can be achieved by varying a number of variables that can be controlled by the patient, such as the amount of insulin injected or the carbohydrate intake, until an acceptable output glucose level is reached. If the system incorporates this functionality, an insulin dosage advisor would be the result. However, the focus of our research is only on making an accurate prediction of the glucose level.

We have to keep in mind that there are quite a few limitations to the neural network approach and the way we apply it. These issues will be discussed elaborately in chapter 7.

3.2 Overview of factors affecting performance

The difficulty of neural networks is that there are a lot of parameters that affect the performance of the model. Taking different values for one parameter might cause the optimal value of another parameter to change. When we change the value for the other parameter, the optimal value for the first may change yet again. For example, two neural networks with a different number of hidden nodes will very likely have different optimal values for the learning rate. The result is a difficult optimisation problem in which we would like to select the optimal values for all parameters in a systematic way. The high complexity of the problem limits us in our search for the neural network with the best parameters, because it is simply not feasible to try all possible combinations of parameter values. Instead we will fix a number of parameters in advance (with the necessary argumentation of course) and determine the optimal values of those that are left. When we determine the optimal input selection in chapter 4, we already seem to have found an optimal neural network concerning the architecture and learning parameters. In reality, a sub optimal set of parameters was used to determine the optimal input selection. A lot of tweaking, tuning and experimenting had already been done and the general direction of the optimal parameters was clear to some extent. It may be assumed that the optimal input selection will not be dramatically different for the optimal neural network. The same goes for chapter 5, where we use sub optimal values for the learning process when we try to determine the optimal network architecture. The values that are actually optimal are only determined in chapter 6 and we assume the optimal network architecture will not change dramatically for the new learning parameters. Note that ‘parameter’ doesn't need to have a numeric value; it can also indicate a choice of strategy, for example the learning strategy.

The factors that influence the performance of the neural network are among others:

- The data.
    - The fact if we use separate neural networks for each time of day (morning, afternoon, evening or night) or use one neural network for all times of day.
    - Choice of the inputs of the neural network.
        - What data is used as inputs.
        - How far in the past the inputs cover the selected data.
        - If certain data should be transformed or pre-processed.
    - The amount of data.
    - The amount of redundancy or clustered samples in the data.
    - Which fraction of the data is used for training, which for validation and which for testing.
    - The way the data is split up into train, validation and test data.
    - The way the data is normalized/scaled.
    - The complexity of the problem.
    - The quality of the data.
- The neural network architecture.
    - The number of layers the neural network has.
    - The number of nodes the neural network has in each layer.
- The learning parameters.
    - The activation function of the nodes.
    - The interval in which the weights are initialised.
    - The way the weights are initialised.
    - The learning strategy (standard back propagation, batch back propagation, using momentum or not, adaptive learning rate, etc.).
    - The parameters of the learning strategy itself.
        - Learning rate.
        - Momentum.
    - The order in which the samples are presented to the neural network.
    - The error criterion.
    - The stop criterion.

We will treat each of the factors in detail (not in order) to come to a neural network that performs as close to optimal as is feasible in the time available for the assignment. In literature one sometimes sees that a distinction is made between determining the optimal parameters so that a network performs well and determining the optimal parameters so that a network is trained fast. Our focus will solely be on creating a neural network that performs well.

3.3 Process overview

This section describes the general process from collecting the data until the comparing of different neural networks. This process, with the exception of collecting the data, was repeated over and over with different parameters.

We also give a description of the functionality of the software that was used. All supporting software was customly written. The main reason for this was that any desired functionality could easily be added. Another reason was that it might be desirable to use parts of the application for future research later on (e.g. for a handheld device), in which case they can easily be reused. A short search for available software was done, but this appeared not to be worth the effort. Either because the software always seemed to lack some functionality or because it was only available commercially. All applications were written in Java. This choice was made because of the reputation of Java to be platform independent and because it seemed natural to program a neural network in an object oriented way. A disadvantage is that programs written in Java are rather slow. Training one neural network typically took around 20 seconds on a 1 GHz Celeron processor.

3.3.1 The Data

For the experiment, data from one patient is used. This patient is the author himself. The patient led a normal life and was not subjected to any prescribed regimen like restricted exercise or fixed meal times or sizes. Blood glucose measurements were done at least four times a day, corresponding to the meals and bedtime. Any other blood glucose measurements during the interval will be ignored for now. For each day the patient filled in a table similar to the one in figure 3.1, which is a sample of a typical day.

9 apr 04								
Friday	**NT**	**BB**	**AB**	**BL**	**AL**	**BD**	**AD**	**BS**
Time		*8:27*		*12:25*		*17:29*		*23:42*
Glucose level		*7.8*		*6.6*		*7.4*		*8.3*
Short act. insulin		*7*		*7*		*9*		*0*
Long act. insulin		*22*		*0*		*0*		*3*
Carbohydrates		*81*	*0*	*93*	*16*	*92*	*30*	*8*
Exercise		*3*		*2*		*3*		*2*
Stress		*2*		*2*		*2*		*2*

Figure 3.1 A sample of a typical day in the raw data recorded by the patient.

A day consists of eight measurement points: night, breakfast, after breakfast, lunch, after lunch, dinner, after dinner and before sleeping. We will consider four intervals: morning, afternoon, evening and night. There are measurement points at the start of the interval, during the interval and at the end of the interval (e.g. breakfast, after breakfast and lunch for the morning interval). For each interval the following data is entered: time of glucose measurement, glucose measurement value, amount of short acting insulin, amount of long acting insulin, food intake, exercise and stress. Most values are only present at the start and the end of the interval. The patient recorded the glucose measurements in mmol/l and the insulin dosage in units. The short acting insulin that was used was NovoRapid; the long acting insulin was Lantus. The carbohydrate intake was an estimate in grams. Exercise is expressed on a scale from one to five, with one meaning doing nothing and five heavy exercise. A scale from one to five was also used for stress, with one meaning relaxing and five heavy stress. The unit of the variables is not really that important and can be any unit, as long as it is consistent within all of the data. The data will be scaled appropriately in the neural network.

It is assumed that the glucose level is measured before administering the insulin and before the meal. Short acting insulin during an interval is added up, just like long acting insulin. Food intake is expected to be expressed on a scale that can be added up. The amount of carbohydrates does not really meet this requirement because there are different kinds of carbohydrates (short and long acting carbohydrates). A better measure would have been the glycaemic load [12]. Exercise and stress are only considered at the start of the interval and are assumed to cover it entirely; values during the interval are ignored. We also assume that the intervals themselves are of reasonable length, so that food for example has been digested completely and the resulting glucose value is more or less constant.

The recorded data that was used covers a continuous period of 77 days: March 1 until May 16, 2004. We believe that 77 days of data is a reasonable choice for the period length the data should cover. A shorter period would result in even less data than we have now. But we also believe that a longer period would cover long-term changes in the glucose metabolism, which is also undesirable. These changes may be caused by factors that are not taken into account in the model. This might have a negative effect on the validity of the model, because the old data represents a situation that is likely not to be valid anymore.

An application was written to enter all data and save it to a file, which will be used to extract the train data from.

3.3.2 Train data extraction

We want to compare how well a neural network can predict the glucose value when it has been trained with different inputs. For example: we would like to examine if a neural network performs better if an input like stress is included in the train data. We will call a certain selection of inputs an input selection (i.s.). To be able to compare different input selections, we need an efficient way to quickly create such sets of train data.

An application was written to extract the desired data from the file with raw data into a file with train data. This file could then be read by the application that implemented the neural network. The train data extraction application was written in such a way that certain inputs could easily be added or removed. It was also possible to perform any operation on one variable or on a combination of variables.

It can be expected that some factors weigh heavier depending on the time of day. For example, insulin sensitivity changes during the day. To deal with such variations in our model, we have three options:

- Creating four separate neural networks, one for the morning, afternoon, evening and night each.
- Taking the time of day (morning, afternoon, evening or night) as one of the inputs and create one all-purpose neural network. This extra input would have to represent the time of day numerically, for example with the numbers 1 to 4.
- Assuming that variations during the day are negligible and create one neural network for all intervals.

The advantage of creating one large neural network as in the third option is that we have four times as much train data available than when we use separate neural networks, which may help to increase performance. The disadvantage is that we won't be able to model behaviour that depends on the time of day. The second option is a compromise between the first and the third. We will only consider the first option. (It would have been interesting to look at the other options as well, but this was not done due to time constraints.) The train data extraction phase consequently produced 4 sets of data for each input selection.

The train data was not filtered for redundancy or clustered training patterns.

Before training the neural network, the available data also had to be divided into a train data set, a validation data set and a test data set. Train data was used to train the neural network, the validation data to check the neural network for overfitting and the test data for testing. We come back to the details later. The data from the entire data set was randomly distributed over these three data sets. The train data set contained 40% of the data, the validation data set 30% and the test data the remaining 30%. These numbers were determined with some experimenting and they may not be optimal. Using distributions where one set was much smaller than another had a negative effect on performance.

3.3.4 Training the neural network

After extracting the train data, the neural network can be trained. This is done using an application that implemented the back propagation neural network. Special functionality was built in to run exactly identical training sessions while varying one variable. This allows us to obtain an optimal value for such a variable by comparing the performance of the resulting neural networks. The parameters that were actually used will be discussed throughout the rest of this document.

3.3.5 Comparing performance

To come to an optimal neural network, neural networks that were trained with different parameters will have to be compared. We do this by looking at the performance of the neural network. To get an unbiased measure of performance, the performance measure is calculated using the test data set. This set had not been used in the training process. We will express the performance of a neural network using a number of different methods. They are the performance plot, the root mean square error, the correlation coefficient, the slope coefficient and the performance coefficient. Each has its own useful properties, which we will describe now.

Occasionally, we will provide a plot of the observed versus the predicted glucose value, as recommended by Lehmann and Deutsch [3]. The performance plot gives a quick impression on how well the neural network has been trained. Figure 3.2 shows the performance plots of a neural network that has not been trained, one that did not train well and one that did train well, together with the corresponding performance measures. The performance plots were automatically generated by the application that trained the neural network.

The root-mean-square-error (RMSE) is given by

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - t_i)^2}$$

where y is the output of the neural network, t the target output (the measured value) and N the number of samples. The RMSE has the desirable properties that large errors are weighed proportionally heavier than small errors. This way, many small errors are preferred above a few large errors. Many small errors are preferable, because we know that even with an optimal model we will not be able to eliminate the small errors. Taking the square root enables us to express the error in the same unit as the error and thus gives us some insight into the magnitude of the error. We will always express the RMSE in mmol/l.

The correlation coefficient r^2 is given by

$$r^2 = \frac{ss_{ty}^{\ 2}}{ss_{tt} ss_{yy}}$$

where

$$ss_{tt} = \sum_{i=1}^{N} (t_i - \bar{t})^2$$

$$ss_{yy} = \sum_{i=1}^{N} (y_i - \bar{y})^2$$

$$ss_{ty} = \sum_{i=1}^{N} (t_i - \bar{t})(y_i - \bar{y})$$

where y is the output of the neural networks, t is the target output and N is the number of samples. The correlation coefficient is a measure of how well the predicted values match the measured values. A correlation coefficient that is smaller than 0.5 is considered to be weak and a correlation coefficient that is larger than 0.7 can be called strong.

A problem with the correlation coefficient can be seen when we draw a regression line into the performance plot. Ideally, the regression line would be equal to identity: y = t. But the correlation coefficient is equal to 1 for any straight line and not only for the line of identity. This can be seen by comparing the correlation coefficients in figure 3.2. The untrained network has a correlation coefficient that is higher than the badly trained neural network, even though the badly trained network performs better. To make sure that we don't end up with a model that has a high correlation coefficient but where the regression line doesn't fit the line of identity, we need another measure. For this purpose we calculate the regression line through the points of the performance plot:

$$y = a + sc \cdot t$$

where y is the output of the neural network, t the target output, 'a' a constant and sc the slope coefficient. We will use the value of the slope coefficient as the additional measure. Ideally, its value would be equal to 1. The slope coefficient can be estimated by calculating

$$sc = \frac{N \sum_{i=1}^{N} t_i y_i - \sum_{i=1}^{N} t_i \sum_{i=1}^{N} y_i}{N \sum_{i=1}^{N} t_i^2 - \left(\sum_{i=1}^{N} t_i \right)^2}$$

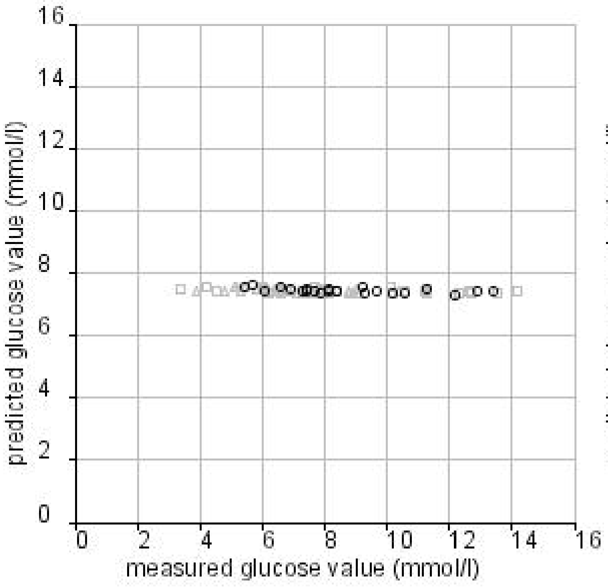
where y is the output of the neural network, t the target output and N the number of samples.

Last, we will calculate a rather unusual measure, which we will use to compare the performance of different neural networks. We will call it the performance coefficient and it is a combination of the correlation coefficient and the slope coefficient. It seemed natural to combine these measures, because both generally indicate a well trained neural network for values close to 1 and poorly trained neural networks for other values. Either measure on its own may sometimes be a bad indicator as we described above. Only if both measures are close to 1 we can say with certainty that the neural network performs well. The performance coefficient is given by:

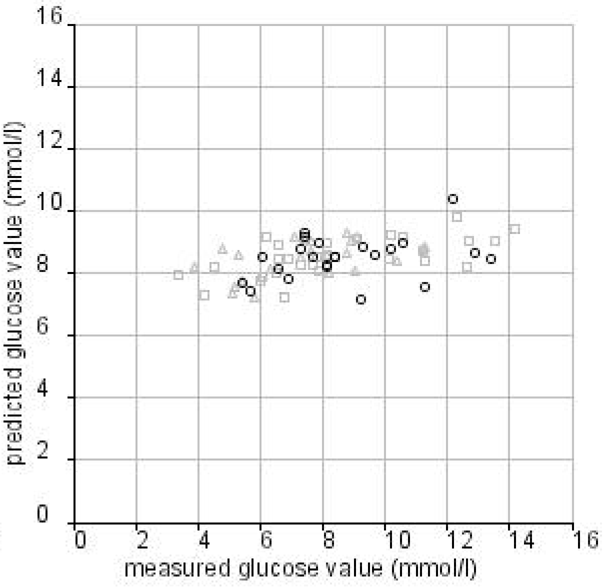
$$\begin{cases} pc = -\,cc \cdot sc & \text{if } cc < 0 \text{ and } sc < 0 \\ pc = cc \cdot sc & \text{otherwise} \end{cases}$$

where pc is the performance coefficient, cc is the correlation coefficient and sc is the slope coefficient. In our case it will be very rare that either the cc or the sc is negative. But in case this does happen we want to make sure the resulting pc is also negative. We also assume that both the cc and the sc will be between 0 and 1. The disadvantage of this measure is that it doesn't have a meaning, aside from the fact that the closer the value is to 1, the better the neural network performs. We will use it to compare performance between neural networks.

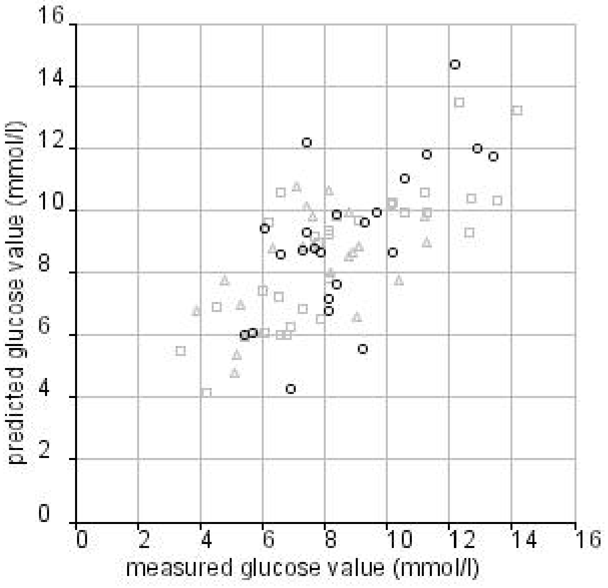
The application that trained the neural networks generated a table with the different performance measures automatically for each neural network.



0 epochs: rmse = 2.5390, cc = 0.2074, sc = -0.0133, pc = -0.0028



75 epochs: rmse = 2.0654, cc = 0.1055, sc = 0.1021, pc = 0.0108



1000 epochs: rmse = 1.9079, cc = 0.4517, sc = 0.7506, pc = 0.3390

Figure 3.2 Performance plots of neural networks that were (from left to right) not trained, badly trained and well trained. Square markers indicate train data, triangular markers are validation data and round markers represent test data. The test data is highlighted, because this is the data that is used to express the performance of the neural network. The performance on the train data and validation data are shown to be able to detect overfitting. The rmse, cc, sc and pc are given for the test data only.

The reason we use multiple methods to express the performance of the model is that each measure on its own is not enough. For example, if we look at the RMSE of the networks in figure 3.2, we see that there is not that much difference in the RMSE, while the performance increases dramatically from left to right. The network corresponding to the third plot has only a small decrease in RMSE compared to the second, while it is clearly much more preferable over the second. We are interested more in 'lining up' the performance plot with the line of identity than in getting a low RMSE. The increase in performance is clearer when we look at the correlation coefficient and the slope coefficient. The performance coefficient, a combination of the two, also shows the increase in performance.

The correlation coefficient, slope coefficient and performance coefficient do have a disadvantage. If all points are close together, their values can be considerably worse, while the RMSE is the same. This can be seen for example in the third plot of figure 3.2. The performance coefficient of the validation data is 0.1284, which is considerably worse than the performance coefficient of the test data (0.3390). The RMSE however, is 1.9143, which is roughly as good as the test data (1.9079). If we take a closer look at the performance plot, we can see that the cause lies in the way the data was distributed over the train, validation and test data. All triangles (corresponding to the validation data) are more or less located in the centre of the plot, causing the performance coefficient to be low, while the points seem to fit in nicely with the rest of the data. To reduce the chance that a certain set of data only covers a small 'area', we should have more data in each set. The problem is that we only have a very small set of data. We will discuss a different solution to this problem in section 3.3.6.

So although the RMSE does say something about the magnitude of the error, a small RMSE doesn't always mean that the performance is good. Similarly, a low performance coefficient doesn't always mean performance is bad. However, a large RMSE always seems to indicate a badly trained neural network and a high performance coefficient always indicates a well-trained neural network.

Last we note the usefulness of the performance plots, which provides a means of inspecting the performance of a neural network. For example, a phenomenon that occurred sometimes is shown in figure 3.3. Using the plot, the behaviour of the neural network can be interpreted as a case where a certain input has some effect on the predicted value and divides the predicted values into a number of 'categories'. These categories are created by the fact that the inputs are discrete, such as the insulin dosage in this case. For continuous inputs, the predicted values get spread out more evenly.

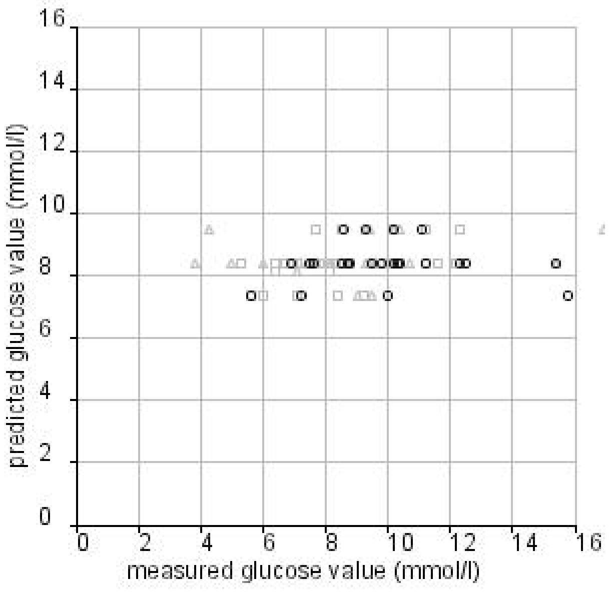


Figure 3.3 The relevance of a certain input can be derived from the information in the performance plot.

As stated above, we will use the performance coefficient for comparing performance of different neural networks. Only in chapter 7 we will discuss the other performance measures for the best neural networks and look closer at the performance plots. Then we will also compare the networks to other existing models using experiment results from literature.

3.3.6 Getting a reliable performance value

The performance of different neural networks can vary considerably. This is caused, among others, by two factors that are random. The first is the distribution of the data over the train data, validation data and test data. The distribution is random and as a consequence it cannot be said with certainty that all three sets of data cover the input space uniformly. A main factor in this problem is the small size of the data set. Situations can occur where all train data is located close together in the input space and test data is spread out. The opposite or a combination can also occur. These effects result in performance fluctuations, either because the neural network did not train well or because the test data happened to be badly (non-uniformly) chosen. The other random factor is the initialisation of the weights. The way the weights are initialised can have quite some effect on how well the resulting neural network will perform. This is caused by the fact that the weights could be initialised close to a local minimum on the error surface, in which case the network may be stuck there for the rest of the training session.

These are the reasons that training a neural network can be seen as a stochastic process that depends on the learning parameters as well as the initialisation of the neural network, including the distribution of the data [13]. To overcome this problem, a technique is used that is also applied by Vancoillie [14]. A number of holdout conditions is created. Each holdout condition consists of a random distribution of the data over the various sets and a random weight initialisation. The distribution and weight initialisation are kept constant within each holdout condition, but differ between holdout conditions. When we want to determine a reliable performance level of a neural network, the average performance of the neural networks over a number of different holdout conditions is taken. When we want to compare the performance of neural networks that were trained while varying a certain parameter (such as the learning rate) we do this by first training the neural network with one value of the parameter on a series of holdout conditions and then with the other value of the parameter on exactly the same series of holdout conditions.

The number of holdout conditions should be as high as possible to get an accurate average. Because of the limited amount of calculation time available, 10 or 20 holdouts were used. The actual number used will be specified with each problem.

3.4 Neural network

The most common form of back propagation neural network will be used. We don't give a description of how a neural network works; this is assumed to be known. We only mention the choice of the variables.

3.4.1 Activation function

The activation function that is used for each neuron is the following sigmoidal function:

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

where x is the net input of the neuron and the result is the output of the neuron.

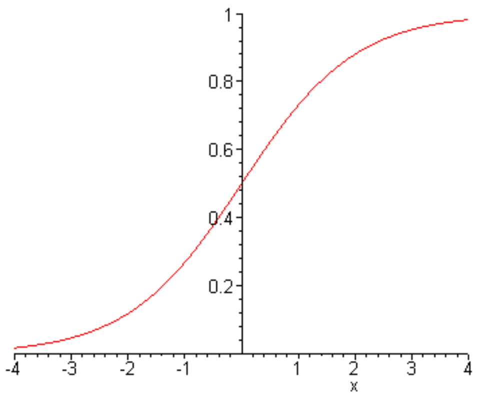


Figure 3.3 The activation function

Gradient-descent learning with a momentum term is applied to find the weights resulting in the smallest error. This results in the following weight update rule:

$$\Delta w_t = \eta \cdot \delta \cdot x + \alpha \cdot \Delta w_{t-1}$$

with Δw_t the weight change, η the learning rate, x the input corresponding to the weight to be updated, α the momentum and Δw_{t-1} the previous weight change. The optimal values of the learning rate and the momentum term will be determined in chapter 6. With the term 'learning rate' we indicate the learning rate of the output layer. For each preceding layer, the learning rate was set to three times that of the next layer. (This is a rule of thumb suggested in [13]). The meaning of δ differs between the output layer and the hidden layers:

For each neuron in the output layer:

$$\delta = y(1-y)(t-y)$$

with y the output of the neuron and t the target output.

And for each neuron in the hidden layer(s):

$$\delta = y(1-y)\sum w_i \delta_i$$

with y the output of the neuron, w the weights from the neuron to the next layer and δ the error of the corresponding neuron.

3.4.2 Scaling

The train data had to be scaled so that the neural network could process the data. A linear scaling to the interval [0.1, 0.9] was applied, with 0.1 corresponding to the minimum value and 0.9 to the maximum value.

3.4.3 Weight initialisation

The interval in which the weights were initialised did not seem to have much effect on the eventual performance of the neural network, although an initialisation with all the weights set to 0 sometimes resulted in a network that performed very bad. Therefore, a safe initialisation interval of [-0.5, 0.5] was chosen.

3.4.4 Stop criterion

There are two sides to performance:

- How well does the neural network model the data it has been trained with?
- How well does the neural network model data it has not been trained with?

These two sides are complimentary. If the network is trained too much, it memorizes the train data and doesn't perform well on other data (large variance or overfitting). It also starts modelling noise and thus doesn't generalize very well. On the other hand, training the neural network too short will result in a network that may not perform well overall (large bias). For this reason, a compromise needs to be found.

A solution can be found in using cross-validation as a stop criterion. A separate set of data (the validation data set) is used to monitor the performance. In the beginning of a training session, the error of the validation data set will go down just like the error of the train data set. At some point however, the error on the validation data will start to increase. At this point maximum generalization has been obtained.

We applied the cross-validation technique in the following way. First, the neural network was trained while the error of the network on the validation data was recorded. Training was stopped if the validation error was increasing over 500 consecutive epochs or if the number of epochs was equal to 20000. Then the epoch that corresponded to the smallest validation error was determined, the neural network was reset to exactly the same state as before training and training was redone until the epoch with the minimum validation error. In the rare case that the validation error was still decreasing at epoch 20000, the error on the third set (the test data) was often increasing, in which case stopping was desirable anyway.

4. Input selection

In this chapter we will evaluate the effect of different input selections. The selection of the inputs is one of the most important issues in creating a neural network. If there is no correlation between the inputs and the output, there is nothing for the neural network to learn. For this reason we don't want to select too few inputs. An input like the amount of injected short acting insulin will have a very small correlation to the resulting glucose value if it is used on its own. Adding the right inputs will result in a selection that has a higher correlation to the output and thus enables the neural network to learn something. But on the other hand we want to take care that we don't select too many inputs, because this could more easily lead to a neural network that doesn't generalize well and consequently leads to a decreased performance of the neural network on the test data. This would especially be the case with the small amount of available data. We first make a number of selections, after which we discuss the obtained results.

4.1 Selecting, combining and pre-processing data

If possible, variables from the raw data should be combined in a pre-processing phase. Take for example the level of exercise. There is evidence that heavy exercise over the past few days results in a lower overall glucose level, so we would like to include this as an input to the network. Selecting all exercise data over the past day would result in 4 inputs if we use four 'data points' for each day. Instead, it may be better to pre-process the 4 points into one input.

So during train data extraction, we will sometimes perform an operation on a variable. For example, we may want to take an average of exercise, which itself will be expressed on a scale of one to five. Strictly speaking, it is nonsense to take the average of a variable that is expressed on an arbitrary scale; it would not be possible to express the result on the same scale. For us this is not really a problem, because we are not interested in the value on the original scale. If we would like, we could perform any operation on the value, as long as it helps us to better train the neural network. Of course both the scale and the operation must be chosen so that the result is useful. For example: a neural network will have a hard time training if the scale of exercise alternates light with heavy exercise on increasing values.

It is very likely that each of the four networks corresponding to the different intervals have different optimal input selections. For example, it may be better to leave out exercise as an input for the night interval because of the lack of variation of this variable from night to night. This is why we will compare the performance of the neural networks on the different intervals separately.

4.2 The selections

Below, we describe which input combinations we chose to examine and give a short motivation for each of them. The output for each selection always is the resulting glucose level at the end of the period we are trying to make a prediction for.

1 - 4. Single inputs

First we tried several inputs like the glucose level (selection 1), amount of short acting insulin (selection 2), amount of carbohydrates (selection 3) and exercise (selection 4) separately. This is not expected to give good results, but it gives us an idea of how the performance can be increased by adding multiple relevant inputs later on.

5. Four main inputs, only covering current interval
This is the first elementary try with a combination of inputs. It covers the most important inputs that are believed to have a direct effect on the glucose level. Only the current interval is considered.
Inputs:

• Glucose level	at start of interval
• Short acting insulin	during interval
• Food intake	during interval
• Exercise	during interval

6. Four main inputs, only covering current interval, plus long acting insulin
The same inputs as with the previous selection, but now with the amount of long acting insulin over the past 24 hours as an additional input.
Inputs:

• Glucose level	at start of interval
• Short acting insulin	during interval
• Food intake	during interval
• Exercise	during interval
• Long acting insulin	during past 24 hours

7. Four main inputs, only covering current interval, plus stress
This selection includes the stress input, as it is one of the variables that is hardest to express numerically and the effects of stress are uncertain. Sometimes stress will increase the glucose level, but in other cases stress will decrease it.
Inputs:

• Glucose level	at start of interval
• Short acting insulin	during interval
• Food intake	during interval
• Exercise	during interval
• Stress	during interval

8. Four main inputs, only covering current interval, plus stress and long acting insulin
The same inputs as with the previous selection, but now with the amount of long acting insulin over the past 24 hours as an additional input.
Inputs:

• Glucose level	at start of interval
• Short acting insulin	during interval
• Food intake	during interval
• Exercise	during interval
• Stress	during interval
• Long acting insulin	during past 24 hours

9. Including data from previous day
This selection was inspired by the approach by Lakatos, et al and Andrianasy et al. Besides the usual inputs it includes the inputs and the result of the same interval on the previous day. It is believed that the situation and the result of the previous day can be used to say something about what will happen in the current situation. This way we try to eliminate certain variations in the glucose metabolism. If we know what happened the previous day, it is likely to happen

again today. In this case, it can be assumed that variations with a time span in the order of several days are smoothed.
Inputs:

• Glucose level	at start of interval
• Short acting insulin	during interval
• Food intake	during interval
• Exercise	during interval
• Glucose level	at start of interval on previous day
• Short acting insulin	during interval on previous day
• Food intake	during interval on previous day
• Exercise	during interval on previous day
• Resulting glucose	at end of interval on previous day

10. Including data from previous interval
This selection is similar to the previous one, but it contains the data of the previous interval instead of the data of the previous day. This can be seen as an attempt to smoothen the day-to-day variations.
Inputs:

• Glucose level	at start of interval
• Short acting insulin	during interval
• Food intake	during interval
• Exercise	during interval
• Glucose level	at start of previous interval
• Short acting insulin	during previous interval
• Food intake	during previous interval
• Exercise	during previous interval

(Note that we don't need to include the resulting glucose level at the end of the previous interval, since this is equal the glucose level at the start of the current interval.)

11. Four main inputs, stress, long acting insulin, exercise past day and interval length
We now focus on the current interval again. There are some variables we did not take into account yet, but which are known to affect the glucose level. These are the level of exercise during a certain period in the past and the length of the interval. A high level of exercise is known to have a delayed effect aside from the direct effect that is experienced during the interval. The effect can still be noticed a day after the exercise took place. Regular heavy exercise can have the same effect on the glucose metabolism as an increase in long acting insulin dosage would have. We will only look at the average exercise during the past day. The length of the interval can also have an effect on the resulting glucose level. For example, if the glucose level goes down during the interval, the next measurement will result in a lower glucose level if the measurement is done later. The interval length is measured in minutes.
Inputs:

• Glucose level	at start of interval
• Short acting insulin	during interval
• Food intake	during interval
• Exercise	during interval
• Stress	during interval
• Long acting insulin	during past 24 hours
• Exercise	average over past 24 hours
• Interval length	during interval

12. Four main inputs, stress, long acting insulin, pre-processed exercise past day and interval length

Last, we want to get an idea of how pre-processing an input in a different way can affect performance. We will do this by pre-processing the exercise over the past day differently than for i.s. 11. For i.s. 11, we simply added up the exercise over the past day. But we want to weigh heavy exercise relatively heavier than light exercise, because we believe that on the scale that was used, heavy exercise in one interval may have more effect than light exercise in two intervals. Therefore we square the values of the exercise before adding them up.

Inputs:

• Glucose level	at start of interval
• Short acting insulin	during interval
• Food intake	during interval
• Exercise	during interval
• Stress	during interval
• Long acting insulin	during past 24 hours
• Exercise	added up squared values during past 24 hours
• Interval length	during interval

Each input selection was trained on 20 holdout conditions. The neural networks were trained with the following (sub optimal) parameters: the architecture was a single-hidden-layer network with 10 hidden nodes. Batch training was used with a learning rate of 0.1 and a momentum of 0.1.

4.3 Results

The tables below show the performance of the neural networks after training with different input selections. Performance is calculated using the test data set only. We list the root mean square error (rmse), correlation coefficient (cc), slope coefficient (sc) and performance coefficient (pc). We look at both the average (avg) and the best (best) trained neural networks. The neural network that is 'best' is the one with the highest performance coefficient. This means that, for example, the RMSE of the best neural network may not be the lowest rmse overall. Each interval is treated separately. We will try to give a short explanation of the results.

More elaborate results that also include the number of train epochs needed and the performance on the train data and validation data are given in appendix A. Appendix A also contains the performance plot of the neural network with the highest performance coefficient for each input selection. Note that the neural network with the highest performance coefficient need not always be the neural network that performs best overall. The performance on the train data and validation data may very well be quite low.

morning								
input selection	avg rmse	rmse of best	avg cc	cc of best	avg sc	sc of best	avg pc	pc of best
1	2.9618	**2.8128**	0.0576	**0.0966**	-0.0011	**0.0036**	-0.0002	**0.0003**
2	2.9753	**2.9076**	0.0365	**0.1627**	-0.0021	**0.0051**	0.0000	**0.0008**
3	2.9553	**3.6566**	0.1189	**0.3541**	-0.0011	**0.0089**	0.0000	**0.0032**
4	2.7347	**1.8496**	0.1711	**0.3616**	0.1426	**0.2769**	0.0274	**0.1001**
5	2.7353	**2.8760**	0.1670	**0.3847**	0.1594	**0.3338**	0.0373	**0.1284**
6	2.7320	**2.2565**	0.1884	**0.4398**	0.2145	**0.4693**	0.0467	**0.2064**
7	2.6754	**2.3664**	0.2146	**0.4075**	0.2302	**0.5311**	0.0614	**0.2164**
8	2.6713	**2.1886**	0.2288	**0.4760**	0.2688	**0.5776**	0.0679	**0.2749**
9	2.7696	**3.1499**	0.1425	**0.3503**	0.1279	**0.1789**	0.0225	**0.0627**
10	2.8802	**1.8759**	0.1176	**0.4151**	0.0816	**0.2181**	0.0148	**0.0905**
11	2.3699	**1.8852**	0.3952	**0.5992**	0.4309	**0.5883**	0.1748	**0.3525**
12	2.3547	**1.8368**	0.4031	**0.6212**	0.4386	**0.5976**	0.1803	**0.3712**

Table 4.1 Performance of neural networks on different input selections for the ‘morning’ interval.

afternoon								
input selection	avg rmse	rmse of best	avg cc	cc of best	avg sc	sc of best	avg pc	pc of best
1	2.5178	**2.4613**	0.0606	**0.3691**	-0.0073	**0.0068**	-0.0002	**0.0025**
2	2.4980	**2.3555**	0.0401	**0.0696**	0.0046	**0.0902**	0.0004	**0.0063**
3	2.4666	**2.2142**	0.0786	**0.1605**	0.0372	**0.0987**	0.0036	**0.0158**
4	2.4034	**2.0588**	0.1916	**0.3331**	0.0724	**0.1804**	0.0159	**0.0601**
5	2.1359	**1.5739**	0.3528	**0.6348**	0.3894	**0.6446**	0.1575	**0.4092**
6	2.1620	**1.7889**	0.3293	**0.6121**	0.3470	**0.5681**	0.1277	**0.3477**
7	2.1628	**1.5180**	0.3459	**0.6470**	0.3962	**0.6608**	0.1572	**0.4275**
8	2.2031	**1.8237**	0.3132	**0.6473**	0.3569	**0.5068**	0.1287	**0.3280**
9	2.2806	**1.7606**	0.2779	**0.6388**	0.2640	**0.5734**	0.0872	**0.3663**
10	2.2966	**1.8571**	0.2795	**0.6113**	0.2663	**0.4939**	0.0991	**0.3019**
11	2.3231	**1.9693**	0.2638	**0.4651**	0.3148	**0.5385**	0.1074	**0.2505**
12	2.2977	**2.1229**	0.2685	**0.3804**	0.3235	**0.7306**	0.1129	**0.2779**

Table 4.2 Performance of neural networks on different i nput selections for the ‘afternoon’ interval.

evening								
input selection	avg rmse	rmse of best	avg cc	cc of best	avg sc	sc of best	avg pc	pc of best
1	2.4833	**2.0604**	0.0544	**0.0849**	0.0101	**0.1140**	0.0001	**0.0097**
2	2.5181	**2.3010**	0.0395	**0.0696**	-0.0107	**0.0042**	-0.0013	**0.0003**
3	2.4529	**2.0270**	0.0897	**0.1019**	0.0192	**0.0656**	0.0011	**0.0067**
4	2.4177	**2.7112**	0.1063	**0.1645**	0.0508	**0.1331**	0.0056	**0.0219**
5	2.4267	**2.3095**	0.1015	**0.2690**	0.0557	**0.1213**	0.0065	**0.0326**
6	2.4365	**2.5115**	0.0894	**0.1689**	0.0534	**0.1645**	0.0060	**0.0278**
7	2.4407	**2.3812**	0.0992	**0.2009**	0.0502	**0.2404**	0.0070	**0.0483**
8	2.4589	**2.4816**	0.0757	**0.2327**	0.0410	**0.0859**	0.0038	**0.0200**
9	2.5080	**2.4309**	0.0737	**0.3073**	0.0411	**0.0632**	0.0040	**0.0194**
10	2.4857	**2.2009**	0.0956	**0.3553**	0.0672	**0.1791**	0.0101	**0.0636**
11	2.4580	**1.9801**	0.0803	**0.1418**	0.0524	**0.1757**	0.0051	**0.0249**
12	2.4526	**1.9584**	0.0837	**0.1591**	0.0573	**0.1883**	0.0057	**0.0300**

Table 4.3 Performance of neural networks on different input selections for the ‘evening’ interval.

night								
input selection	avg rmse	rmse of best	avg cc	cc of best	avg sc	sc of best	avg pc	pc of best
1	2.6434	**2.3863**	0.0271	**0.1129**	0.0004	**0.0056**	0.0000	**0.0006**
2	2.6618	**2.4465**	0.0451	**0.1263**	0.0033	**0.0127**	0.0002	**0.0016**
3	2.6368	**1.9222**	0.1057	**0.1546**	0.0218	**0.0734**	0.0025	**0.0113**
4	2.6548	**2.7652**	0.0158	**0.0548**	-0.0010	**0.0042**	0.0000	**0.0002**
5	2.6582	**2.6982**	0.0630	**0.1757**	0.0167	**0.0631**	0.0013	**0.0111**
6	2.3533	**1.9872**	0.2728	**0.3348**	0.2383	**0.4270**	0.0692	**0.1429**
7	2.6441	**2.7055**	0.0698	**0.1645**	0.0171	**0.0604**	0.0016	**0.0099**
8	2.3438	**2.0185**	0.2697	**0.3568**	0.2443	**0.5231**	0.0730	**0.1866**
9	2.6261	**2.7113**	0.0876	**0.3022**	0.0673	**0.1482**	0.0093	**0.0448**
10	2.7285	**2.4611**	0.0383	**0.1430**	0.0166	**0.0831**	0.0011	**0.0119**
11	2.3515	**2.1070**	0.2617	**0.3344**	0.2367	**0.4614**	0.0699	**0.1543**
12	2.3380	**2.0564**	0.2720	**0.3734**	0.2489	**0.5095**	0.0766	**0.1903**

Table 4.4 Performance of neural networks on different input selections for the 'night' interval.

As expected, the neural networks for the first four input selections performed poorly with a pc mostly below 0.1 for all intervals. It seemed however that the amount of carbohydrates and especially the level of exercise as input resulted in considerably better results. This might indicate that these variables have a large contribution to the resulting glucose level. The performance plots (not shown) also showed poorly trained networks.

Combining the four main inputs (i.s. 5) increases the performance for most intervals, although some intervals seem to benefit more than others. The afternoon interval sees the most dramatic increase in performance over i.s. 4. The night interval still performs poorly.

Adding long acting insulin to the selection (i.s. 6) also seems to have mixed effects on performance. Both the night and the morning interval see a large improvement, where the afternoon and the evening interval actually give worse results. This might be explained by the fact that long acting insulin dosage has the greatest effect during the night and morning and less during the following afternoon and evening.

Input selection 7 is based on selection 5. These are the four main inputs and stress. The morning interval has a higher increase in performance over i.s. 5 than when the long acting insulin is used as an input. The afternoon sees a moderate increase in performance as well as the evening. The night interval doesn't seem affected at all, which is not so strange since the stress level had the same value for basically all nights.

When we use both long acting insulin and stress (i.s. 8), we see even more improvement on the morning interval. The afternoon interval gives worse results, as well as the evening. The night has a small increase in performance.

For input selection 9 we take the data of the previous day into account and for input selection 10 the data of the preceding interval. We only use the four main inputs, so we will compare the results to input selection 5. The morning interval sees a decrease in performance for both input selections compared to i.s. 5, although i.s. 10 seems to perform better. The afternoon also performs worse on both input selections, but here i.s. 9 gives the better results. The evening performs worse on i.s. 9, but i.s. 10 performs better. However, the results are still poor. The night sees a slight increase, but performance is poor here as well.

Input selection 11 considers the current interval again. It is based on i.s. 8 with the exercise during the past 24 hours and the interval length added as inputs. The morning interval shows a considerable increase in performance over i.s. 8. The afternoon however, shows a decrease in performance. Both the evening and the night improve a bit.

When we look at input selection 12, we see a small improvement for all intervals. The results suggest that this approach to calculating the exercise over the past day may be better, although the difference in performance is minimal. At least we can say that pre-processing inputs in a different way has some effect on performance.

Figure 4.1 shows for each interval the performance plot of the neural network that performed best. This way an impression of the performance for the best input selections can be obtained.

4.4 Conclusion – input selection

In this chapter we have made only a rough selection of possible input combinations. Including other inputs or combining and pre-processing them in some way can be expected to give better results. In the next chapters we will only use the input selection that performed best for each interval. These are:

• Morning:	input selection 12
• Afternoon:	input selection 7
• Evening:	input selection 10
• Night:	input selection 12

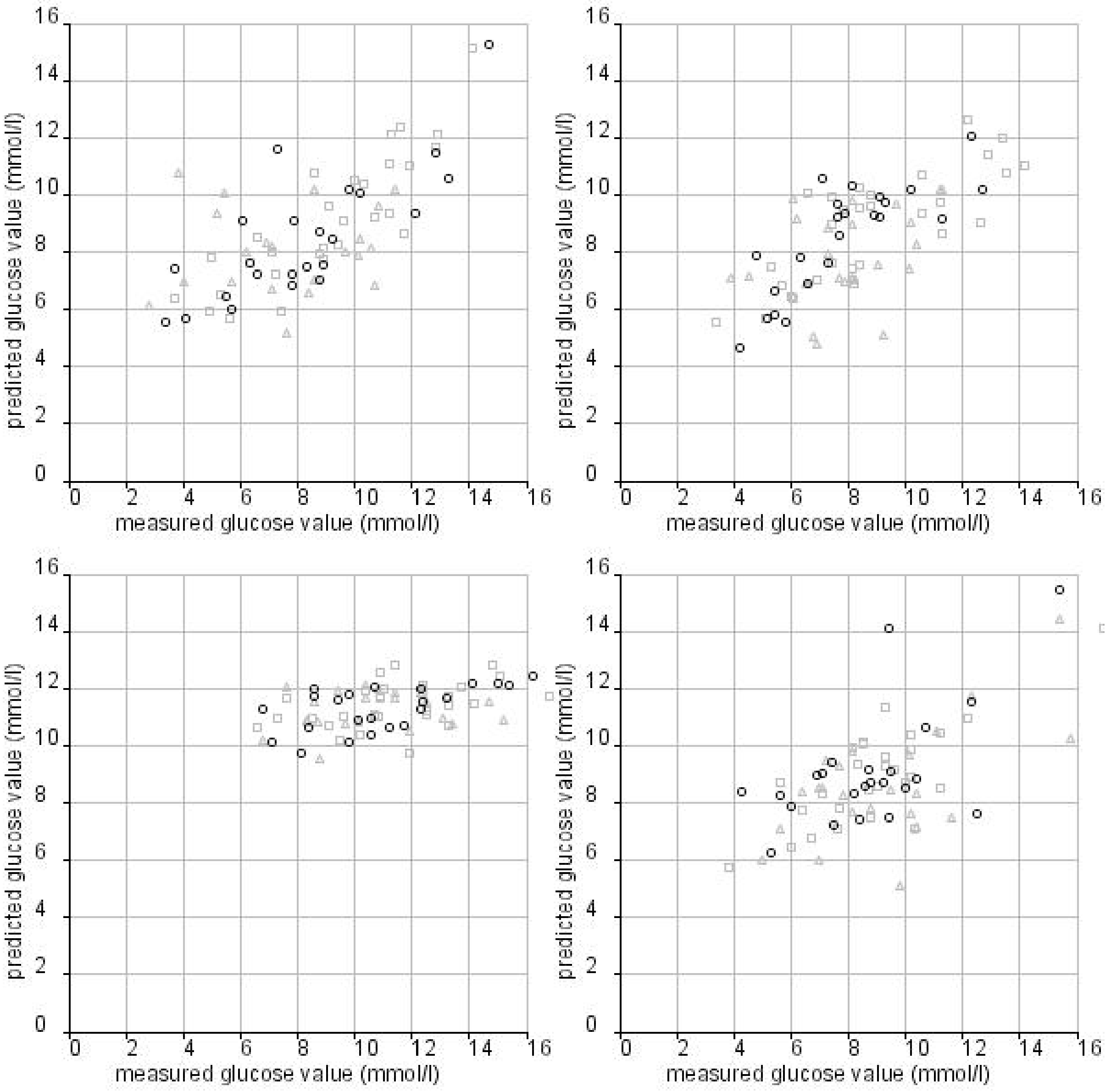


Figure 4.1 Performance plots of the neural networks with the highest performance coefficients for the best input selection. The input selections are (from left to right):

- Morning: 12
- Afternoon: 7
- Evening: 10
- Night: 12

5. Neural network architecture

In this chapter we describe how we determined the best neural network architecture. The network architecture is an important issue when creating a neural network, because it affects the ability to generalize. For a simple back propagation neural network, the choice of the architecture consists of the choice of the number of hidden layers and the number of nodes in each layer. If the total number of nodes is too small, the resulting neural network will not be able to accurately represent the train data and errors will be high. A large number of nodes could lead to overfitting. However, because our stop criterion (cross validation) will prevent overfitting, the latter might not be a problem. The only problem then will be that there are a number of redundant nodes.

5.1 Determining the number of neurons

There are numerous rules of thumb that give an indication of how large a neural network should be, most of them based on the size of the train data set or the number of input and output nodes. A list of such rules is given by Vancoillie [14]. Unfortunately, almost all rules recommend different sizes. Therefore we will determine the optimal network size in a different way.

5.1.1 One or two hidden layers

The universal approximation theorem states that a neural network with only one hidden layer can approximate any function with any desired accuracy, provided that it has enough neurons in the hidden layer and that the activation functions of the neurons are non-linear. However, for some functions the number of neurons needed in the hidden layer can be very large or even infinite. In these cases, a neural network with two hidden layers might work out better. Therefore we will examine neural networks with one hidden layer as well as two.

5.1.2 Layer size

There are a number of techniques that can be used to determine the optimal number of neurons in a neural network. Two of those techniques are growing methods and pruning methods. They respectively increase and decrease the number of nodes and connections dynamically during the learning process. But instead of using such dynamic approaches, we will statically train the neural networks, using a different number of neurons each time. The optimal architecture will be the smallest network that shows a considerable increase in performance over the smaller ones.

The architectures we will try are:

- One hidden layer with 1 to 20 neurons
- Two hidden layers, each with 1 to 10 neurons

Similar to the technique used to determine the best input selection, we test each configuration on a number of holdout conditions. In this case, each configuration is tested on 10 holdout conditions. For each interval, we use the input selection that performed best on that interval. Batch training was used with a learning rate of 0.1 and a momentum of 0.1.

5.2 Results

Because of the large amount of results, we will only visualize the average performance coefficient. The results are shown in figure 5.1 to 5.4. More results can be found in appendix A.

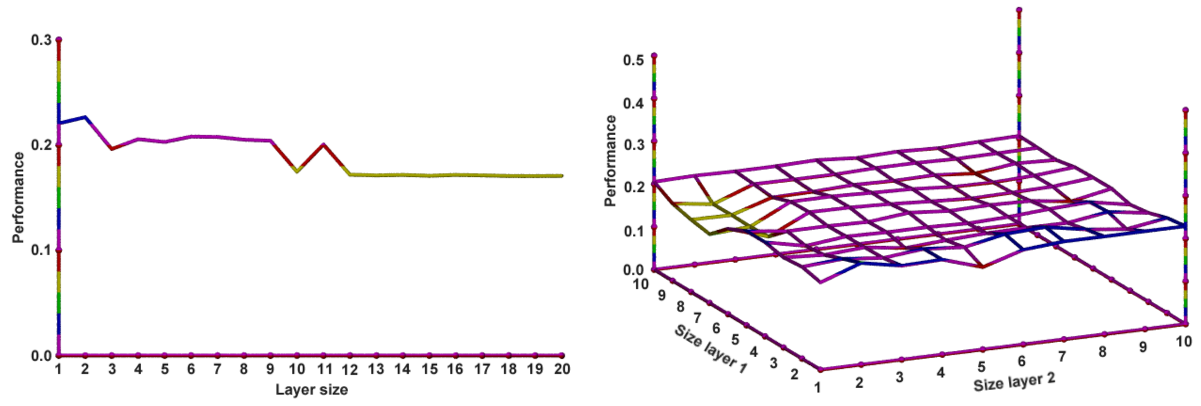


Figure 5.1 Performance coefficient for the morning interval for one -hidden-layer networks (left) and two-hidden-layer networks (right) with different layer sizes.

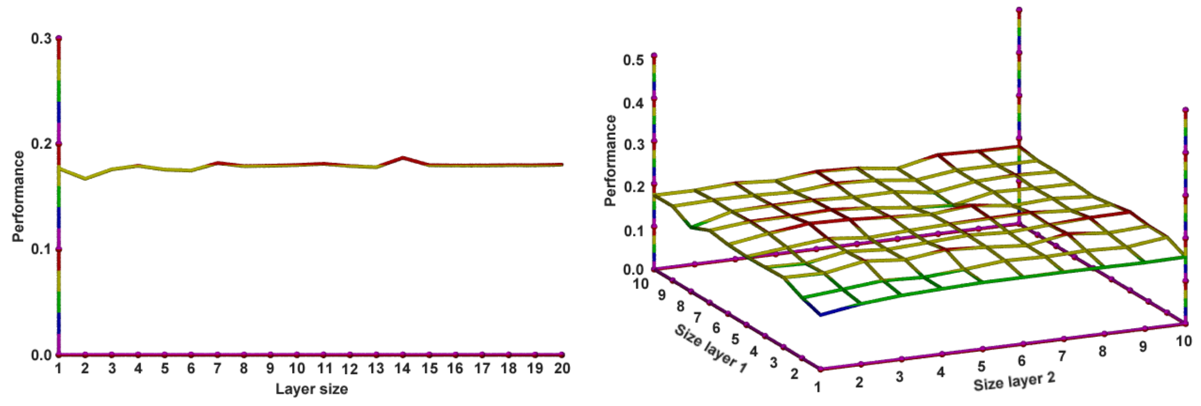


Figure 5.2 Performance coefficient for the afternoon interval for one -hidden-layer networks (left) and two-hidden-layer networks (right) with different layer sizes.

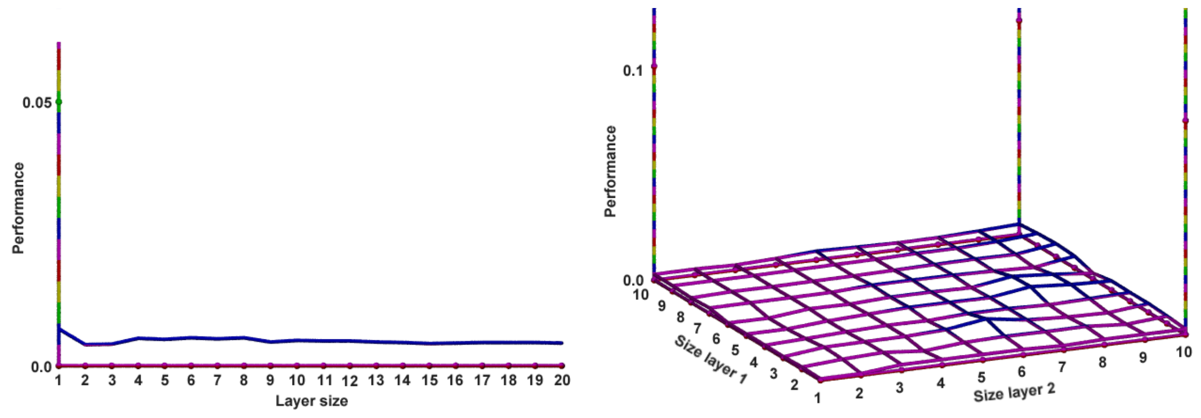


Figure 5.3 Performance coefficient for the evening interval for one -hidden-layer networks (left) and two-hidden-layer networks (right) with different layer sizes. Note that both i mages have been scaled with a factor 5.

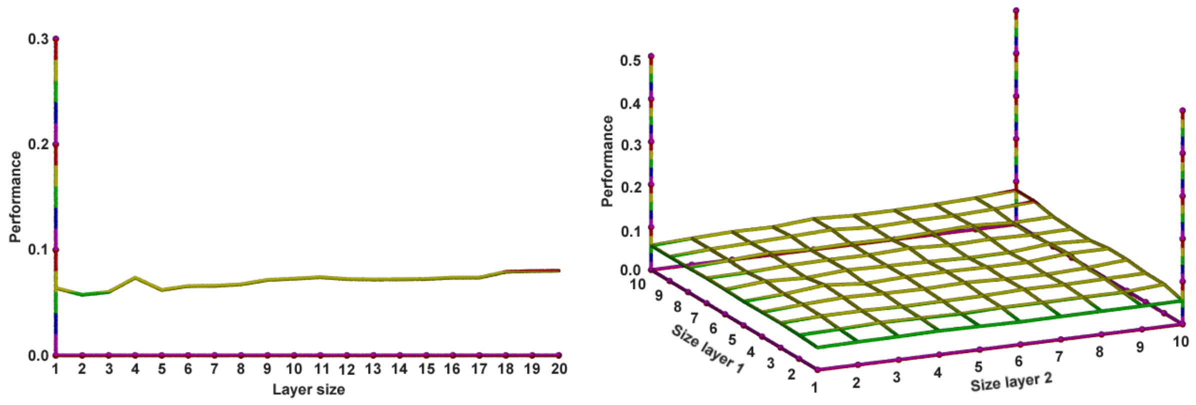


Figure 5.4 Performance coefficient for the night interval for one -hidden-layer networks (left) and two-hidden-layer networks (right) with different layer sizes.

We will first look at the results using a single hidden layer. For the morning interval, performance is decreasing, with a sudden decrease at 10 hidden units. A small hidden layer (2 nodes) actually seems to perform best. At first sight, the fact that the best neural network is actually one of the smallest seems to indicate that using a large number of neurons makes the neural network too complex for our problem. However, there are a number of other possible explanations. It could be that in reality the problem *is* complex, but that the level of noise is too high and the neural network cannot see past it. Another possibility might be that a larger neural network does have the *ability* to perform better than a small one, but that this occurs for other values of the learning parameters. Another explanation for the sudden decrease in performance is that one of the holdout conditions results in hitting a local minimum for a large number of nodes and pulls the average performance down. The networks for the afternoon interval behave more like we would expect, just like those for the night. Here the performance increases slowly for a larger number of hidden units. The networks for the evening interval all perform poorly, and it is hard to say which architecture performs better.

We now look at the situation with two hidden layers. For the morning interval, the performance decreases with increasing size of the first hidden layer, similar to using one hidden layer. However, increasing the size of the second layer seems to improve performance a bit. Also similar to the case with one hidden layer, the afternoon and night interval perform more as expected. When increasing one of the layer sizes, performance generally increases as well. This increase is not very predictable however. For some of the larger architectures performance is just as bad as with a smaller one. The evening interval also performs better for larger layer sizes, although it still performs poorly.

When we compare performance between one-hidden-layer to two-hidden-layer networks, we see that there is not much difference. That is, the worst and best performances are roughly the same.

5.3 Conclusion – network architecture

Generally, there seems to be a small improvement for large architectures, and very small networks tend to perform poorly. However, when we look at the morning interval, we see that the reverse can also be the case. The only hard conclusion we can draw here is that for the given parameters, performance changes only slightly for different architectures. There is not much difference between using one hidden layer and using two hidden layers. But because we want to use the smallest architecture that still gives good results, using a single layer is the preferred choice. For the next chapter we will use a single-hidden-layer network with 11 hidden nodes.

6. Neural network learning parameters

Determining the optimal architecture was the first step to come to an optimal solution. Now we have to make sure that we actually exploit the architecture to the fullest potential. To do this, the optimal training parameters have to be determined. In chapter 3 we have already set a number of training parameters, such as the activation function and the weight initialisation interval. In this chapter we will determine the optimal learning strategy, learning rate and momentum term.

The learning parameters can be used to speed up the training process, but they also affect performance. The latter is true because certain learning strategies can be used to prevent the training process from getting stuck in local minima on the error surface. We will only look at the effects on performance.

6.1 Learning strategy

Learning strategies are sets of rules on how to conduct learning. Among others, they specify when and how weight updates take place. We will look at two strategies that are commonly used. These are

- On-line training (or standard back propagation)
- Batch training

The difference between the two strategies is that on-line training updates the weights after each training sample that is presented, while batch learning first presents all samples before updating the weights.

6.2 Learning rate and momentum

A number of algorithms exist that dynamically adapt the learning rate. One such algorithm starts the training process with a large learning rate and slowly decreases it. Other approaches use a large learning rate when the error is decreasing and a small learning rate when the error increases. Changing the learning rate dynamically can lead to faster convergence. Using a dynamic learning rate also avoids local minima, increasing the chance that the neural network reaches the global minimum. The disadvantage of these methods is that a number of new parameters are being introduced for which an optimal value needs to be determined. Even if they may improve performance, this often doesn't weigh against the trouble of optimising another set of parameters. Therefore we will use a static learning rate.

However, we will use one improvement of the standard learning strategies: the momentum. A momentum term can be added to help the training process overcome local minima. We will try different values for the momentum term, including 0, which is of course equal to training without a momentum term.

The combinations of learning rates and momentum terms we will try are:

- Learning rate: 0.1 - 1.0, with a step size of 0.1
- Momentum: 0.0 - 0.9, with a step size of 0.1

Each combination of learning parameters is tested on 10 holdout conditions. As in the previous chapter, we use the best input selections for each interval. We use one architecture size for all networks, namely the single-hidden-layer with 11 nodes.

6.3 Results

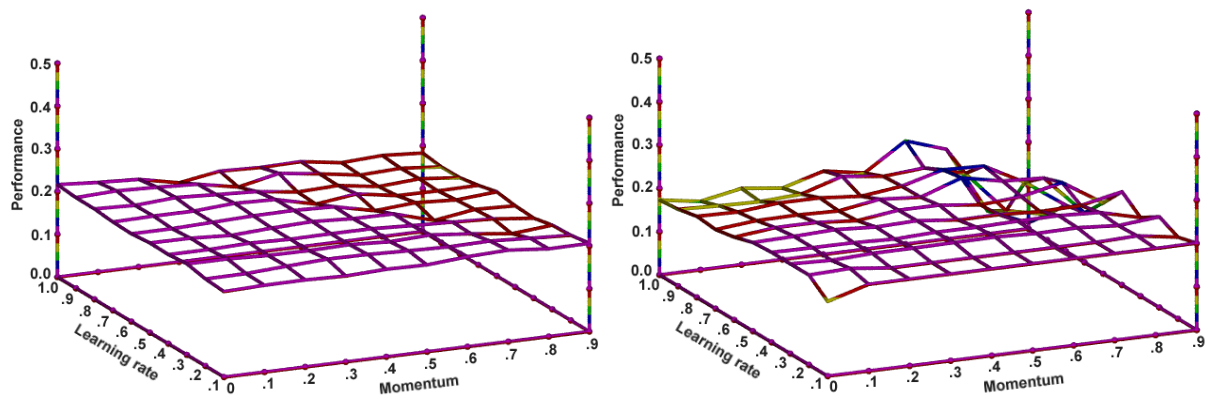


Figure 6.1 Performance coefficient for the morning interval for on -line (left) and batch (right) learning with different values for the learning rate and the mo mentum term.

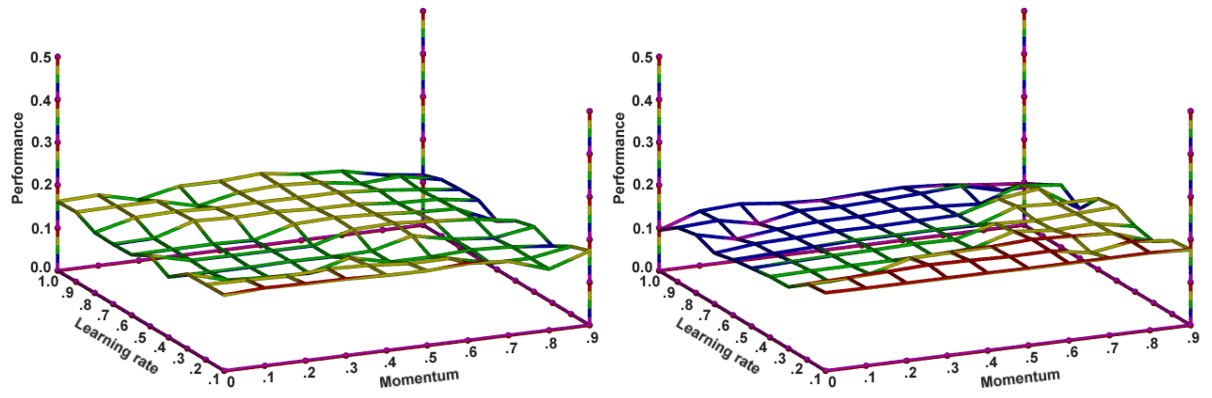


Figure 6.2 Performance coefficient for the afternoon interval for on -line (left) and batch (right) learning with different values for the learning rate and the momentum term.

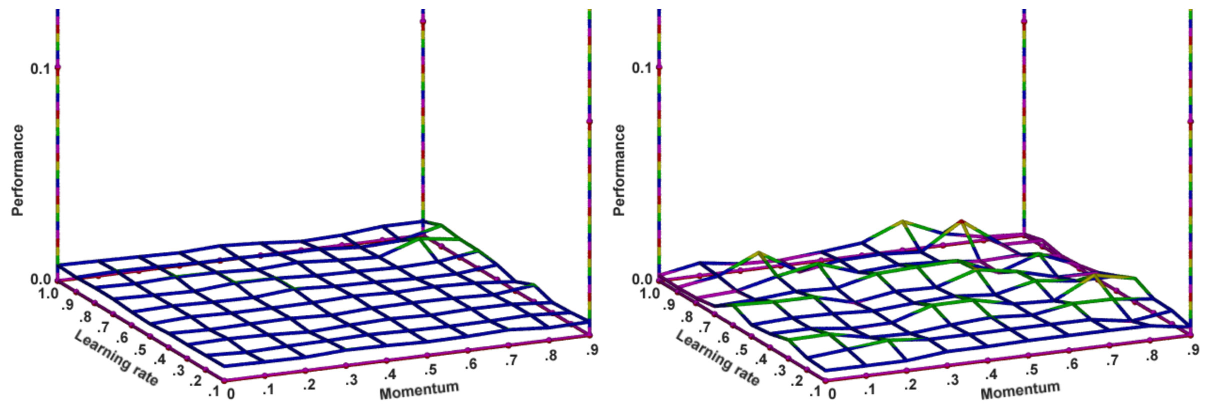
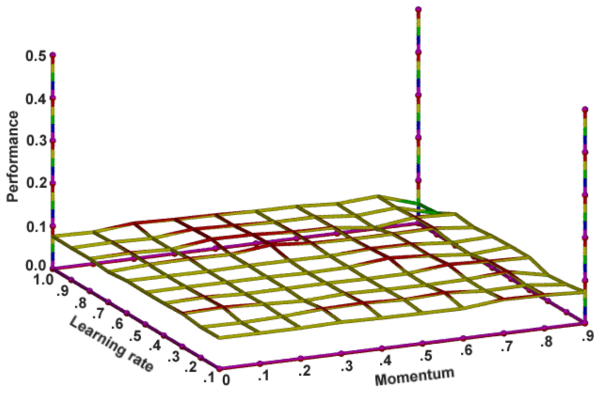


Figure 6.3 Performance coefficient for the evening int erval for on-line (left) and batch (right) learning with different values for the learning rate and the momentum term. Both graphs have been scaled five times.



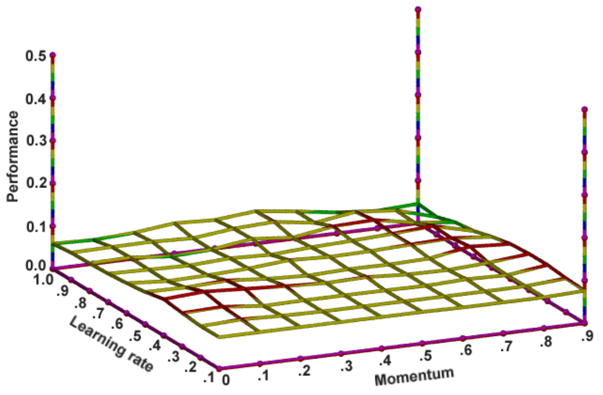


Figure 6.4 Performance coefficient for the night interval for on-line (left) and batch (right) learning with different values for the learning rate and the momentum term.

Figure 6.1 to 6.4 show the average performance coefficient for the networks trained with the different learning parameters.

The first thing we notice for the morning interval is that batch learning gives results that are rather unpredictable compared to the on-line learning strategy. For low values of either the learning rate or momentum, both learning strategies behave comparable. As both values get higher, performance decreases a bit for the on-line learning strategy. Note that this effect seems similar to the effect that occurred when we determined the optimal architecture for the morning interval. It may also have the same cause. For the batch learning strategy we see better as well as worse results for larger values of the parameters. When both learning rate and momentum are above 0.8, a dramatic decrease in performance occurs. A learning rate of 0.1 and momentum of 0 also results in a slightly worse performance.

The afternoon interval gives very diverse results. The results of on-line learning suggest to either use a low or a high learning rate with an intermediate momentum, where a low learning rate performs a bit better. The results of batch learning suggest to only use a small learning rate. Similar to the morning interval, results seem to be more stable and better for on-line learning when we compare the strategies. We also note that using a high value for both learning rate and momentum results in poor performance for both strategies.

On-line learning gives stable results for the evening interval, where performance seems to increase for larger values of the learning rate and momentum. Batch learning performs better than on-line learning in some areas. The last is rather unpredictable however, except for the fact that performance is very poor when both learning rate and momentum are high.

Results are more or less stable for both learning strategies for the night interval. Here we also note the typical decrease in performance for large values of both learning rate and momentum.

6.4 Conclusion – learning parameters

In general, results of on-line learning are more stable and predictable than those of batch learning. Batch learning sometimes performs better for certain values of the learning rate and momentum, but it is hard to tell for which values this happens and in some cases performance drops with only small changes of the training parameters.

For the cases we examined above, the safest choice of parameters seems to be 0.1-0.2 for the learning rate and 0.1-0.7 for the momentum term. For these values, results are good (but not necessarily best) in all cases. This goes for both learning strategies. Note that in literature a momentum term between 0.5 and 0.9 is recommended [13]. It may be the case that the error surface of the cases presented here doesn’t have any deep local minima. To improve performance on future cases that do have deep local minima, it may be best to use a momentum between 0.5 and 0.7.

Another conclusion that can be drawn with certainty is that batch learning gives poor results when both learning rate and momentum are greater than 0.8. For on-line learning this seems to be the case less often.

7. Discussion

In the previous chapters we have determined which configurations work best compared to other configurations, but we have not yet looked at how well such a configuration actually performs and if it performs good enough to be used in practice. This will be discussed first. Here we also try to compare the obtained model with the results of other systems reported in literature. After that we discuss the validity of the obtained model. Finally, we consider the validity of the optimal parameter values we determined.

7.1 Performance

It is difficult to find a good measure for the performance of a decision support system, which according to Tresp et al [7] is one of the causes that such systems have not gained acceptance widely. There are two major problems that are hinted at by Lehmann and Deutsch [3]. The first problem is that there is no standard for expressing the performance of insulin dosage advisors and glucose level prediction systems, which makes it very difficult to compare different systems. The ways researchers use to express the performance include among others: for glucose level prediction systems:

- the mean error
- the mean relative error
- the mean square error and root-mean-square error
- the percentage of cases that gave good results

for insulin dose advisors:

- the HbA1c value of the patient compared before and after the use of the system
- the fact if physicians found the performance to be acceptable or not

The other problem is that there is no publicly available data set that can be used to test the performance of models that predict the glucose level. The performance is often measured on a limited number of patients for a limited period of time. The fact that these tests are done with different groups of patients also limits the possibilities to compare results from different approaches.

7.1.1 Results

Table 7.1 shows the approximate average and best RMSE that were achieved using the best parameter combinations. For the performance plots corresponding to the best parameters we refer to figure 4.1. The results in figure 4.1 were actually for sub-optimal parameters, but the results for the optimal parameters are very similar.

	average RMSE (mmol/l)	best RMSE (mmol/l)
morning	~2.3	~1.8
afternoon	~2.1	~1.8
evening	~2.4	~2.1
night	~2.3	~2.2

Table 7.1 Root-mean-square error for neural networks trained with the optimal parameters

Looking at table 7.1 we first notice that the evening interval has a reasonably low error, despite it performing poorly when we consider the performance coefficient. This is at least partly caused by the effect we discussed in section 3.3: the data is all close together (see figure 4.1), resulting in a lower performance coefficient.

There were only a few cases found in literature that could be used to compare performance to. The AIDA system has a root-mean-square error of 1.9 mmol/l [3, 5]. Lehmann and Deutsch [3] also report the existence of systems with root-mean-square errors of 2.8-3.5 mmol/l, but they consider these results to be "even poorer predictive accuracies". Considering the results, we could say that our system performs quite reasonably compared to AIDA. However, care has to be taken with comparing the RMSE. We stated before that a low RMSE does not imply good performance. The second plot in figure 2.1 for example shows a neural network with an RMSE of 2.0, while performance appears to be poor when looking at the plot. But the fact that our final RMSE is reasonably low *does* say something about the quality of the model, because we *know* that our model performs reasonably by looking at the performance plots (those of figure 4.1). So what we want (and what we seem to have) is a good performance plot *and* a low RMSE. We don't need to know what AIDA's performance plot looks like, because, bluntly said, it could only be similar to or worse than ours (if it was better, it would also have a lower RMSE). Once again, knowing that we have a good performance plot *and* an RMSE close to that of a system like AIDA allows us to conclude that our system performs quite good.

It is difficult to say if performance is acceptable enough for the system to be used in practice. At the least, more research needs to be done to determine how the system performs using different data sets and other patients.

7.1.2 The neural network as a data interpolator

The approach we have described here is not based on an explicit model of the glucose metabolism. It can be seen as a data interpolator that is created from the viewpoint of the data. We assumed that two similar situations would result in a similar glucose level. However, this is not always the case. There are two problems at work here.

One of the problems is the quality of the data. We can safely assume that there is at least some noise in the data. For example: the amount of food intake is only a rough estimation and insulin dosages may also contain an error caused by 'the leaking of insulin' after injection. A patient cannot be expected to always specify an accurate value. A neural network should be ideal to generalize and see past some of the noise. This also means that even if we include all factors affecting blood glucose into the model, we will never be able to create a neural network that exactly predicts the resulting blood glucose level, because there will always be noise.

The other problem is the lack of relevant inputs. It is impossible to take every factor that affects the glucose level into account, either because the factor is not known or because it is difficult or impossible to quantify.

Even though these problems seem to limit the possibility to come to a good glucose level predictor, it may still be an improvement from the patient's viewpoint. If a patient determines his or her insulin dosage without the help of a computer, this can be done by looking at previous situations along with the resulting glucose level. By interpolating or looking at similar cases, the patient determines what is going to happen in the current situation. Although the two problems we just mentioned limit the neural network in its ability to make a prediction, they will limit the ability of a human being to make a prediction even more. For people it will be even harder to see past the noise. And taking all important factors into account requires interpolating in a multidimensional input space, which becomes undoable for a large number of inputs. Therefore a patient usually looks at only a few factors. We believe that at least at these two points, artificial neural networks will outperform people.

7.2 Validity of the model

Of course, the model is valid only for the person that provided the train data. But even for this one person, the model will not be valid forever. Slow changes or long term periodical effects, like changes in weight, life-style and habits will all have effect on the validity of the model, simply because these factors were not included in the train data for the model. It is expected that the model validity will decrease considerably after a period of time. This is similar to what is stated by Liszka-Hackzell [10], where the obtained model only performed well for 15 days after the date corresponding to the last train data. Consequently, we could say that our model needs to be retrained every now and then to include the latest data. We have not determined how long our model stays valid.

It is important to note that a large part of the glucose metabolism cannot be modelled by the approach we take. Suppose for example that a patient usually has a sudden decrease in glucose level after a meal, after which it returns to a normal level. If the insulin dosage is increased, this could lead to a glucose level that is too low just after a meal, even though it might return to a more normal value after a period of time. Because our model takes ‘time steps’ the size of the interval, this hypoglycaemic event cannot be detected. Even though the event might be prevented by decreasing the amount of injected short acting insulin and raising the dosage of basal insulin, such advice cannot be given by our system, because it will not be trained for this. The model assumes that the therapy as prescribed by a physician has been optimised in such a way that these events do not occur and that the behaviour of the blood glucose level during an interval is as close to linear as possible.

7.2.1 Performance fluctuations

We have seen that there were quite some differences in performance from holdout condition to holdout condition. To make certain that the resulting glucose prediction system uses a neural network that has been trained well, additional action is required. Poorly trained neural networks should be rejected. We now list some options that enable us to do this. Multiple options can be combined.

One option is to check the performance of the neural network by looking at the performance scatter plot. The disadvantages of this option are that this procedure is not easily automated and a person needs to understand the meaning of the scatter plots.

Another way is to have the system check automatically if the neural network has a high correlation coefficient and a low RMSE. Here the problem is where to draw the line between neural networks that perform well and those that perform poorly.

We can also assume that most neural networks train well and use a committee of neural networks created on different holdout conditions. The output of the committee would then be the average of the outputs of the neural networks. If the errors have a zero mean and are uncorrelated, the error of the committee is 1/(number of networks) times the mean error of the neural networks [13, 14]. Usually this does not hold however and a smaller increase in performance is obtained. A number of other methods for creating ‘ensembles’ of neural networks are listed by Vancoillie [14]. Various other parameters, such as the network architecture and learning parameters, could also be varied. Performance could be increased further by giving the neural networks that perform well more weight in the committee.

7.2.2 Black box

One of the biggest disadvantages of the model we have obtained is that it is a black box. We cannot say anything of the reasoning behind a certain prediction. It could very well be that the predicted glucose level behaves chaotic when changing one input slightly. Another scenario is the one where the good results of a neural network are coincidence, because there appeared to be a correlation in the data that was actually there by chance. The fact that a neural network is a black box is one of the drawbacks that prevent such systems from gaining acceptance, especially in health care.

However, there do exist so called rule extraction techniques, which try to capture the knowledge contained in a neural network into rules in the form of decision trees. Vancoillie gives an overview of such methods [14]. This way, the behaviour of the neural network can be interpreted and acceptance may be easier. The disadvantage is that the extracted rules may not be valid for all neural networks that are trained with diabetes data. The neural network may be able to 'recognize' different rules for different patients.

The reverse side to the black box problem is that we cannot insert rules. We know that certain rules are valid, such as 'eating more will result in a higher glucose level', but it is difficult to build in such rules in addition to the rules the network learns during training. The ability of the neural network to learn these rules totally depends on how well the rules are represented by the data.

7.2.3 Boundary cases

Another issue is that of boundary cases. Neural networks do not always extrapolate very well. They only perform well on data that lies between the data it has seen when it was trained. If a prediction is made for a situation where one of the inputs lies outside the interval of values it was trained with, the prediction is likely to be inaccurate. Even near the 'edge' performance may be bad. The same problem arises in the case of outliers. A single outlier is likely to have little influence on the training of the neural network, so predictions made for situations that lie between the outlier and the rest of the train data may also not be reliable. If the neural network is to be used in practice, a mechanism to warn for these kinds of situations should be implemented in case they arise.

Directly relating to this problem is the problem that arises when the system is to be used as an insulin dosage advisor. In the ideal case, the use of such a system causes the blood glucose level to remain normal, i.e. between 4.0 and 8.0 mmol/l. This, however, also narrows the interval for which data is available for a future neural network, resulting in an increase of the chance that the system cannot accurately predict most cases, because they will be boundary cases.

7.3 Validity of the optimal parameters

We have determined the parameters that can best be used to create an optimal neural network for the given data set. These parameters might not be optimal when they are used to create a new model using different data. To determine if they are also optimal in general, the optimal parameters would need to be determined for different sets of data and for other patients. The optimal values that were found in this study will at least indicate the direction in which they may be found for future cases. They may also be different from person to person. Ideally, the search for the optimal parameters should be done automatically each time new data is used, so that an optimal neural network can be constructed for each case. The biggest problem preventing this is that there is no systematic way to do this and the amount of calculation time that would be required is very large. Here, using a committee of neural networks trained with different parameters (as described above) could also be a solution.

We also want to stress again that the optimal parameters we determined are likely to be only sub-optimal ones, because when the optimal value for one parameter is determined, the optimal value for another parameter might have changed. To be sure that the parameter values are really optimal, the research would have to be iterated until the optimal values do not change anymore. This on itself could be seen as a learning process with its inherent problems, such as the question if the parameter values are only ‘in a local minimum’ or not. This may be an interesting subject for future research.

8. Conclusions and recommendations

8.1 Conclusions

Artificial neural networks can be used to predict blood glucose levels of a diabetic with reasonable accuracy. The best neural networks have prediction accuracies with a root-mean-square error in the range of 1.8-2.2 mmol/l. Typical results are presented in figure 8.1, which shows the predicted blood glucose level plotted against the blood glucose level that was actually measured.

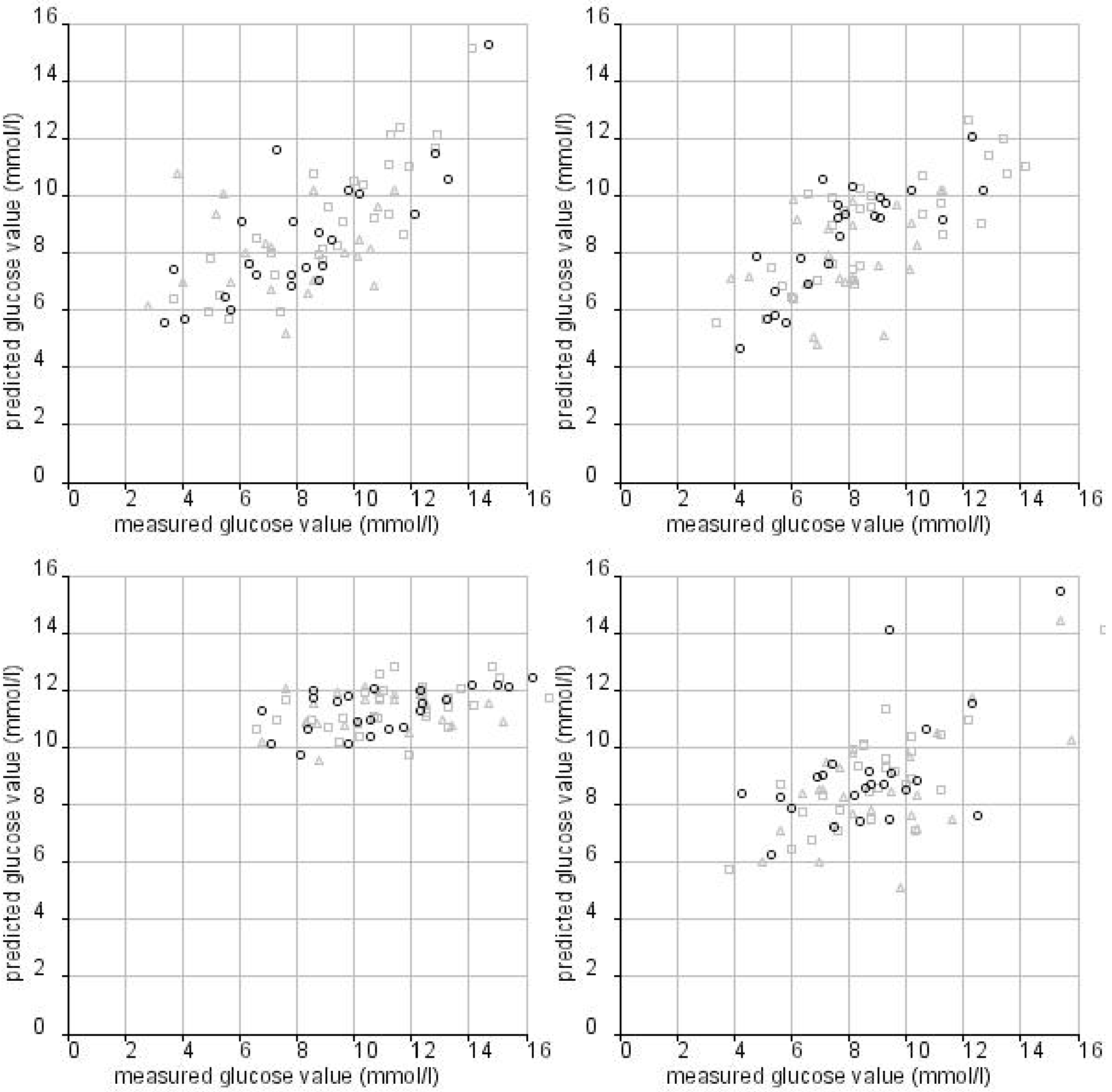


Figure 8.1 Predicted blood glucose levels against measured blood glucose levels for (from left to right) the morning, afternoon, evening and night. Square markers indicate train data, triangular markers are validation data and round markers represent test data.

Earlier research on using neural networks to predict glucose levels is rare. However, there are a few other systems that we can use for comparison. A comparison of the system to AIDA ('The Automated Insulin Dosage Advisor'), which has a root-mean-square error of 1.9 mmol/l, seems to suggest that the neural network approach performs quite well.

The factors that affect the glucose level vary during the day. That is, neural networks for different parts of the day performed best when taking different factors into account depending on the time of day. For example, taking the amount of long acting insulin into account resulted in a dramatic increase of performance of the neural network for the night interval, while this was not the case for the afternoon interval. The best input selections for each part of the day were:

- Morning:
  Glucose level, amount of short acting insulin, food intake, exercise and stress during the interval, long acting insulin over the past 24 hours, addition of squared exercise for past 24 hours and the interval length.
- Afternoon:
  Glucose level, amount of short acting insulin, food intake, exercise and stress, all during the interval.
- Evening:
  Glucose level, amount of short acting insulin, food intake and exercise during the interval and glucose level, amount of short acting insulin, food intake and exercise during the previous interval.
- Night:
  Glucose level, amount of short acting insulin, food intake, exercise and stress during the interval, long acting insulin over the past 24 hours, addition of squared exercise for the past 24 hours and the interval length.

Note that these results are the best of only a limited selection of possible combinations and that other input selections can be expected to give even better results.

There was no particular neural network architecture that performed better on all intervals. But we can say that:

- For the morning interval small architectures performed best, such as a single-hidden-layer with 2 nodes and a double-hidden-layer with 1 or 2 nodes in the first layer. For the other intervals a steady but slow increase in performance could be seen for larger architectures. The largest were 20 nodes for the single-hidden-layer networks and 10 nodes in each layer for the double-hidden-layer networks.
- Because single- and double-hidden-layer networks performed equally well, using a single-hidden-layer network is recommended.

Concerning the learning parameters, we can say the following:

- Overall, the on-line training strategy performed similar to the batch training strategy. However, on-line training gave more stable results, where batch training was less predictable. Therefore we recommend using on-line training.
- A learning rate of 0.1 to 0.2 in combination with a momentum term of 0.1 to 0.7 performed well for all cases. Although there were peaks in performance for other values, there were no trends. To cope with local minima in future cases, a momentum term of 0.5 to 0.7 is recommended.

At this point it is not possible to say if the resulting neural networks (as in figure 8.1) perform well enough to be used in practice. Further testing on other data sets and other patients is required.

It needs to be remembered that our approach is very simplistic. We made glucose level predictions that are of a reasonable quality, without making any assumptions about the underlying glucose metabolism, but based on only 2.5 months of data from one patient who was not restricted in life style in any way. Of course there are a number of limitations to our approach, such as the fact that a neural network only does what it is good at. For example, the neural network will not be able to predict hypoglycaemic events *during* the interval, but only at the end.

8.2 Recommendations for future research

We believe that there is room for improvement on a number of points. We now give an overview of options that may be explored in future research.

- The system needs to be tested on other data sets and with other patients.
- Other ways to express certain variables need to be evaluated. Exercise for example was expressed on a scale from one to five. It may have been better to use the duration of both light and heavy exercise as a measure. Similarly, for food intake a measure like the glycaemic load [12] could have been used instead of the amount of carbohydrates.
- More research needs to be done on which inputs to select and on how to combine and pre-process them. For example, does performance increase when the level of exercise during the past two days is included instead of only the past 24 hours? Even simple operations, such as a quadratic scaling instead of a linear one could have a positive effect on performance.
- It could be investigated if filtering the data for redundancy, clustered samples and outliers improves performance.
- A committee of multiple neural networks that were trained with different parameter values may improve performance of the model.
- A more sophisticated method to determine the optimal parameter values is desirable. In this study we determined the optimal parameters more or less by hand. Such a method could itself be similar to a learning process.
- Another way to improve performance may be to look at the error criterion. As it is now, the neural network is trained to minimize the output error. Training on other criteria, such as maximizing the performance coefficient, may give better results.
- It may be desirable to make a model of the blood glucose profile to cope with hypoglycaemic and hyperglycaemic events during the interval. This may be done in a way similar to that described by Lakatos et al [1].

References

[1] **Lakatos, G.; Carson, E.R.; Benyo, Z.**, 'Artificial neural network approach to diabetic management'. *Engineering in Medicine and Biology Society*, 1992. Vol.14. *Proceedings of the Annual International Conference of the IEEE*, Volume: 3 , October 29- November 1, 1992 p.1010 - 1011

[2] **Ambrosiadou, B. V.; Gogou, G.; Maglaveras, N.; Pappas, C.**, 'Decision support for insulin regime prescription based on a neural-network approach'. *Medical Informatics*. Jan.-March 1996; 21(1): 23-34

[3] **Lehmann, E. D.; Deutsch, T.**, 'Compartmental models for glycaemic prediction and decision support in clinical diabetes care: promise and reality'. *Computer Methods and Programs in Biomedicine*. May 1998; 56(2): 193-204

[4] **Carson E.R.; Deutsch T.**, 'A spectrum of approaches for controlling diabetes'. *IEEE Control Systems Mag* 12(12): 25-31, Dec. 1992.

[5] **Bellazzi, R.; Nucci, G.; Cobelli, C.**, 'The subcutaneous route to insulin dependent diabetes therapy'. *IEEE Engineering in Medicine and Biology Magazine*. Jan.-Feb. 2001; 20(1): 54-64

[6] **Trajanoski, Z.; Regittnig, W.; Wach, P.**, 'Simulation studies on neural predictive control of glucose using the subcutaneous route'. *Computer Methods and Programs in Biomedicine*. May 1998; 56(2): 133-9

[7] **Tresp, V.; Briegel, T.; Moody, J.**, 'Neural-network models for the blood glucose metabolism of a diabetic'. *IEEE Transactions on Neural-Networks*. Sept. 1999; 10(5): 1204-13

[8] **Andreassen, S.; Benn, J. J.; Hovorka, R.; Olesen, K. G.; Carson, E. R**, 'A probabilistic approach to glucose prediction and insulin dose adjustment: description of metabolic model and pilot evaluation study'. *Computer Methods and Programs in Biomedicine*. Jan. 1994; 41(3-4): 153-65

[9] **Andrianasy, F.; Milgram, M.**, 'Applying neural networks to adjust insulin pump doses'. *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop Cat. No.97TH8330*. 1997, p.182-8.

[10] **Liszka-Hackzell, J. J.**, 'Prediction of blood glucose levels in diabetic patients using a hybrid AI technique'. *Computers and Biomedical Research*. April 1999; 32(2): 132-44

[11] **Holman, R. R.; Smale, A. D.; Pemberton, E.; Riefflin, A.; Nealon, J. L**, 'Randomized controlled pilot trial of a hand-held patient-oriented, insulin regimen optimizer'. *Medical Informatics*. Oct.-Dec. 1996; 21(4): 317-26

[12] The glycemic index. http://www.glycemic index.com, The University of Sydney, viewed: August 2004.

[13] **Principe, José C.; Euliano, Neil R.; Lefebvre, W. Curt**, 'Neural and Adaptive Systems: Fundamentals through Simulations'. USA: John Wiley & Sons, Inc., 2000.

[14] **Vancoillie, Friedl**, 'Design and application of artificial neural networks for digital image classification of tropical savanna vegetation'. Universiteit Gent, 2002 -2003.