Master Thesis

Modeling the Development of Obesity in Populations

Maximilian Bügler

Master Thesis 12-17

Thesis submitted in partial fulfillment
of the requirements for the degree of Master of Science
of Operations Research at the Department of Knowledge
Engineering of the Maastricht University

Thesis Committee:

Dr. F. Thuijsman, Dr. K. Staňková, MSc. E. Aller

Maastricht University
Faculty of Humanities and Sciences
Department of Knowledge Engineering
Master Operations Research

July 5, 2012

Contents

Abstract

Due to the increased consumption of food in the western society, obesity and overweight are developing into global problems. These diseases can be the result of numerous causes. Primary issues are of course nutrition and physical activity. After analyzing data, that was provided by the Department of Human Biology at Maastricht University, we developed a model to estimate the effects of preventive and reactive psychological treatments on populations. The model allows us to compare costs of preventive treatments, taking into account that less reactive treatments might be required. This data is particularly interesting to health insurance companies, considering to cover preventive treatments. Furthermore it offers new ways to estimate the effects of new treatments.

Chapter 1

Introduction

Due to the increased consumption of food in the western society, obesity and overweight are developing into global problems. Those diseases can be the result of numerous causes. A primary issue is of course nutrition. But not only the amount of food that is consumed is relevant, but also the composition of the food. For instance food with high sugar content causes the insulin levels to rise. The blood sugar and insulin levels are influencing how nutrients are processed. The large amounts of sugar, other short carbohydrates, and fat in modern food are contributing to development of overweight. Especially in low income areas, people are often less educated about those effects and therefore consume an unhealthy combinations of food[1]. Another factor is the amount of physical activity. Due to the decreasing amount of physical work in our society, while more time is spent at work, ab increasing number of people suffers from obesity[2]. The lack of time also contributes to the large amounts of processed and fast food that are currently consumed.
In order to deal with this problem, different kinds of treatments have been developed. The common kinds of treatments include surgical approaches like gastric banding and liposuction. On the other hand, there are non invasive treatments, namely psychological treatments that can be reactive or preventive.

As an initial source for our model we have been provided with a data set for a psychological treatment for obesity patients by the Department of Human Biology at Maastricht University. The data set contains data recorded at the patient's first visit to the hospital and data measured in half year intervals during the treatment. The treatment consists of an advisory session on nutrition and sports twice a year, where also the measurements are taken.

There are indications that the application of preventive treatments helps keeping the population more healthy and reduces overall costs. In order to investigate the efficiency of those treatments altogether, we developed an interactive agent based model, that instead of averaging over the whole population, models every single individual. The model allows us to simulate populations with respect to different criteria. For instance, we can add different types of treatments that are defined by a number of parameters and estimate the effect and costs of this treatment on a specific population.

Chapter 2

Data Analysis

In order to estimate parameters, required to build the model, we used different methods for extracting information from provided data. This chapter describes these methods and the results on a data set provided by the Department of Human Biology at Maastricht University.

2.1 Provided Data

The provided data consists of a list of 700 patients quantified by a number of parameters. Those parameters were split into 3 categories. The first category is the static patient data, like gender and age, while the second category is the set of parameters, that we want to predict. These consist of weight, BMI, and other obesity related body measures over the time of treatment. The third category consists of the main input parameters, the blood values, and the measured obesity related values at the first entry of the patient. The full list of parameters is provided in Appendix A. For each of the obesity related parameters, the rate of change during the treatment was extracted to be used as a measure of success. We refer to these as our output parameters. Furthermore, the initial measurements of the obesity related parameters were also used as input parameters.
Due to the fact that the classification methods investigated may be subject to over-fitting the data, we have randomly split our data set into two equally sized subsets of size 350. Only one of the subsets was used as input for the creation of the classifier, while the other one was used to verify the result, while ruling out over-fitting effects.

2.2 Bayesian Belief Networks

A Bayesian Belief Network is an information structure that can be used to reason in an uncertain domain. It is implemented as a directed acyclic graph with a probability table for each node. The nodes in the graph represent parameters, and the edges between the nodes represent the relationships among the parameters[3]. In order to create such a network, a node is created for each parameter, followed by creating edges between the nodes according to the structural algorithm

used. We investigated three such algorithms for the accuracy of the resulting networks.

2.2.1 Creating Structures

The most obvious way to create a Bayesian network is to select the node we like to predict and connect any other node to it. This approach is called Naive Bayes. In our case a more elaborate way to create the networks is desirable, since we want to be able to represent more complex relationships that might exist in our data. The following approaches achieved remarkable improvement over naive Bayes in terms of classification accuracy.

Conditional- And Mutual-Information

Conditional and mutual information is the information shared between a set of nodes. In a Bayesian network, if two nodes are dependent on each other, they share information. This information is called mutual information. This mutual information can be used to measure how close two nodes are related. The mutual information of two nodes, X_i and X_j is defined as

$$\mathbf{I}(X_i, X_j) = \sum_{x_i \in V(X_i)} \sum_{x_j \in V(X_j)} \mathbf{P}(v(X_i) = x_i \wedge v(X_j) = x_j) \log \frac{\mathbf{P}(v(X_i) = x_i \wedge v(X_j) = x_j)}{\mathbf{P}(v(X_i) = x_i)\mathbf{P}(v(X_j) = x_j)} \tag{2.1}$$

where $V(X_i)$ is the set of all possible values for the set of nodes X_i and $v(X_i)$ returns the current values of the the nodes, hence $\mathbf{P}(v(X_i) = x_i)$ returns the probability of the set of nodes X_i having values x_i.
The conditional (mutual) information is the mutual information between two nodes X_i and X_j, assuming the value for another node C is known. It is defined as

$$\begin{aligned}\mathbf{I}(X_i, X_j|C) = \sum_{x_i \in V(X_i)} \sum_{x_j \in V(X_j)} \sum_{c \in V(C)} & \mathbf{P}(v(X_i) = x_i \wedge v(X_j) = x_j \wedge v(C) = c). \\ & \log \frac{\mathbf{P}(v(X_i) = x_i \wedge v(X_j) = x_j|v(C) = c)}{\mathbf{P}(v(X_i) = x_i|v(C) = c)\mathbf{P}(v(X_j) = x_j|v(C) = c)}\end{aligned} \tag{2.2}$$

When $\mathbf{I}(X_i, X_j)$ is smaller than a certain threshold ϵ, we say the nodes X_i and X_j are marginally independent given C. [4]

We have visualized the results of both formulas applied to our data in Figures 2.1 and 2.2. The displayed values have been normalized to a diagonal of ones. Furthermore for conditional information, every other node was tested as a conditional node C and the maximum is displayed. Since both formulas are symmetric with respect to X_i and X_j, so $\mathbf{I}(X_i, X_j) = \mathbf{I}(X_j, X_i)$ and $\mathbf{I}(X_i, X_j|C) = \mathbf{I}(X_j, X_i|C)$, both figures are symmetric with respect to the diagonal. In both figures we can see a cluster of dark spots around the diagonal on the outer edge of the output parameters. Those clusters correspond to the four blood pressure values, which are obviously

strongly related. Furthermore the figures illustrate that the algorithm did not find other significant relationships that would link input and output data.

The Conditional Information-Based Search Algorithm

The CI-based (Conditional Information) search algorithm calculates the optimal network structure for a given set of nodes[4] by making more advanced use of the mutual and conditional information formulas.
The algorithm works in 3 steps: drafting, thickening, and thinning. In the first phase, the algorithm computes mutual information of each pair of nodes as a measure of relatedness, and creates a draft based on this information. This draft is then a basic network, in which each arc is undirected. In the second phase, the algorithm adds edges to the current graph when the pairs of nodes cannot be separated based on a group of tests based on the conditional information formula. In the third and final phase of the algorithm, each edge is examined using another group of tests based on conditional information and will be removed if the two nodes of the edge are conditionally independent. [4] A more detailed description of this algorithm can be found in Appendix B.1.

The K2 Algorithm

The K2 algorithm heuristically searches for the most probable network structure given a set of cases. It uses node ordering and an upper bound for the number of parents a node can have. The algorithm results in a list of parents for each node in the network, that maximizes the following function for each node.

$$f(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} \alpha_{ijk}! \tag{2.3}$$

where i is the node and π_i is the set of parents of the node. A detailed description can be found in Appendix B.2.

The Tree Augmented Naive Bayesian Algorithm

The TAN algorithm (Tree Augmented Naive Bayesian) was first described by Friedman, Geiger and Goldszmidt in 1997. It creates structure as follows: First the conditional information between any two nodes is computed (see Equation 2.2). The resulting weights are attached to the respective arcs between the nodes. A maximum weight spanning tree is created by applying Kruskal's algorithm[5]. Now the resulting undirected graph is transformed to a directed one by introducing a root node and setting the direction of all edges to be outward from it. The node corresponding to the desired output parameter is selected as the root. A detailed description of the algorithm can be found in Appendix B.3

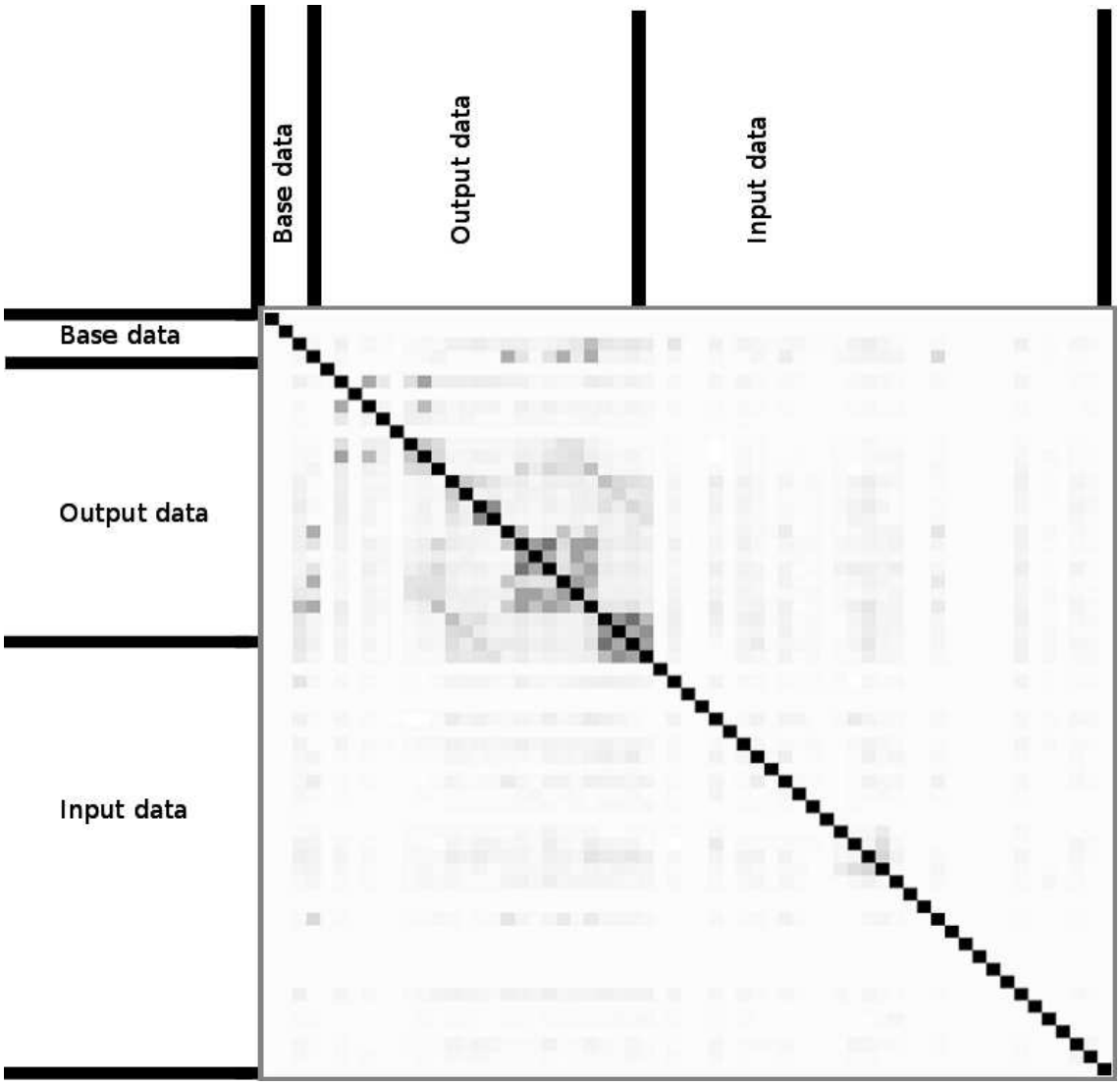

Figure 2.1: Mutual information incidence matrix. Dark pixels indicate high mutual information.

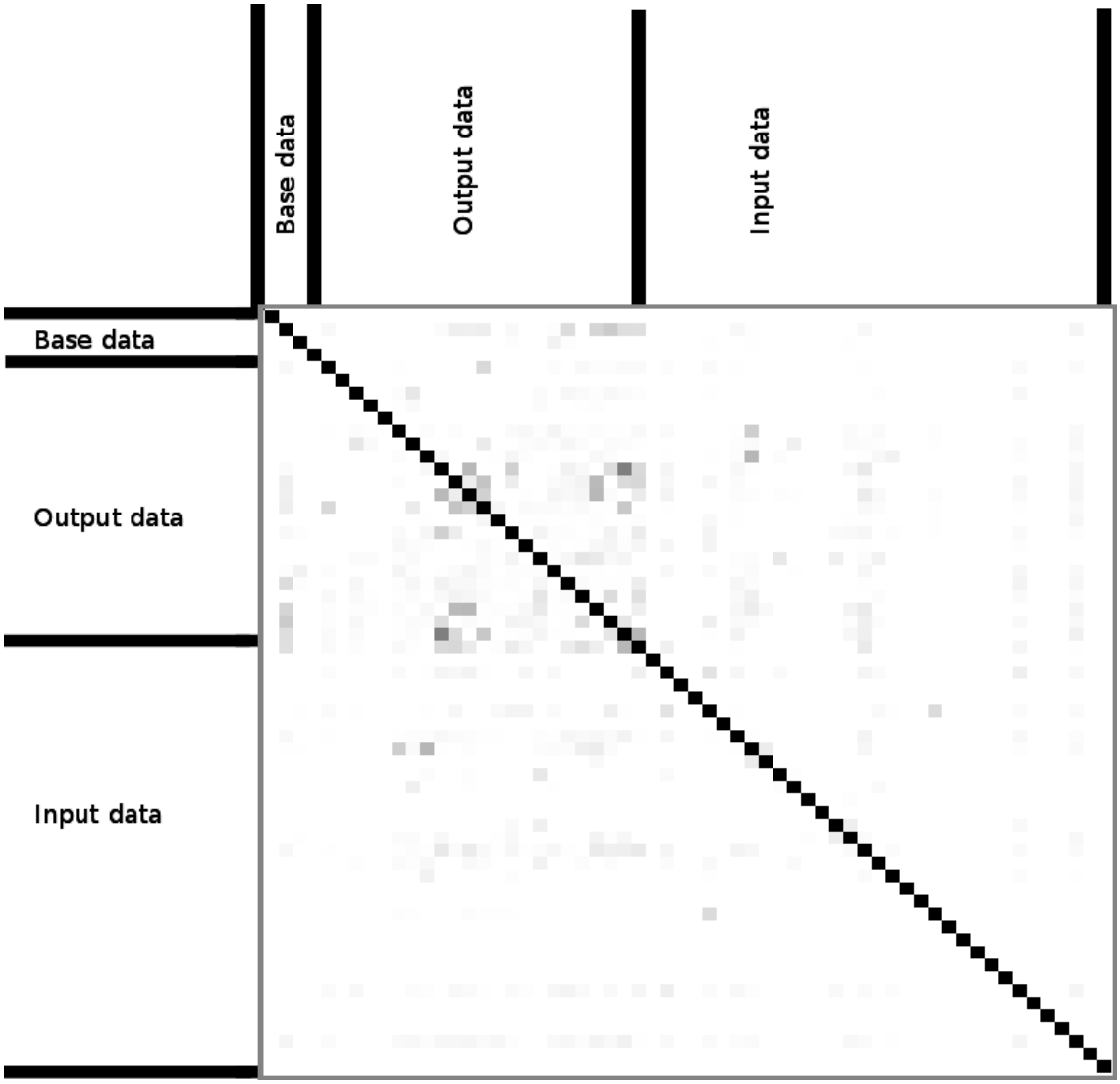

Figure 2.2: Conditional information incidence matrix. Dark pixels indicate high conditional information.

2.2.2 Comparison

In order to compare the described algorithms we downloaded 3 data sets, namely "Thyroid", "Diabetes" and "Post-Operative". The following tables show the error rates of the mentioned algorithms for the 3 tested data files with different characteristics. The Thyroid file is very complex (10.000 patients with 26 parameters and 4 classes), while the diabetes file is comparably simple (750 patients, 9 parameters and 2 classes). The post-operative file is very small and simple (90 patients, 9 parameters and 3 classes).

	1^{st} guess rate	2^{nd} guess rate	3^{rd} guess rate	1^{st} α error	1^{st} β error
TAN	0.22	0.09	0.05	0.16	0.04
K2	0.26	0.24	0.23	0.26	0.01
CI	0.26	0.21	0.16	0.26	0

Table 2.1: Algorithm comparison for data set 'Thyroid'

	1^{st} guess rate	1^{st} α error	1^{st} β error
TAN	0.17	0.11	0.14
K2	0.32	0.32	0
CI	0.22	0.21	0.02

Table 2.2: Algorithm comparison for data set 'Diabetes'

	1^{st} guess rate	2^{nd} guess rate	1^{st} α error	1^{st} β error
TAN	0.38	0.04	0.26	0.15
K2	0.25	0.04	0.11	0.2
CI	0.14	0.04	0	0.2

Table 2.3: Algorithm comparison for data set 'Post Operative'

The tables show 1^{st}, 2^{nd} and 3^{rd} guess rates depending on how many classes were present in our data set. The 1^{st} guess rate column shows the returned probability that the actual class is returned as the most likely class by the network, while the 2^{nd} guess rate shows the probability the right class is the second most likely class returned by the network. α error is the rate of false positive answers, while β error is the rate of false negative answers.

Table 2.1 shows that the TAN algorithm performs best for the big and complex thyroid file. Followed by the CI algorithm. Table 2.2 shows TAN performing the best for the diabetes file, while K2 shows very bad results here. For the very small post-operative file, shown in Table 2.3, the TAN algorithm performs the worst, while the CI algorithm returns very good results. Further experiments showed that the TAN algorithm needs a certain number of patients to work properly. That number seems not to depend on the complexity of the file. The cross-over point between the CI and TAN algorithms showed to be approximately at the data set size of 400 patients.

2.2.3 K-Cross Fold Validation

To furthermore improve the results of the algorithms, validation algorithms like K-Cross fold validation can be used. K-Cross fold validation splits the data set into K equal subsets and creates a network structure for each of these subsets. Each of the resulting networks is then assessed using the remaining $K-1$ subsets. The networks are eventually merged into one network using the scores as weights. As shown in Table 2.4, 10-Cross fold validation reduced error rates by about 1 percent.

	Thyroid	Diabetes	Post Operative
CI	0.26	0.22	0.14
K2	0.26	0.32	0.25
TAN	0.22	0.17	0.38
10-Cross CI	0.25	0.2	0.13
10-Cross K2	0.25	0.32	0.24
10-Cross TAN	0.21	0.17	0.37

Table 2.4: Error rates of 10-Cross fold validation

2.2.4 Creating Distributions

Since each node must contain distributions for all possible setups of its parents these distributions need to be calculated in order to be able to use the network. The distributions can be calculated from the data files. The needed probabilities and conditional probabilities are calculated through iteration of the data files.

2.2.5 Inference Algorithms

In order to make use of a Bayesian network structure an inference algorithm is needed. An inference algorithm calculates a probability distribution for desired nodes, given a set of known values. The three most important ones are described in this section.

Inference By Enumeration

Once a Bayesian network is created it provides a complete description of the domain. In order to answer a query through the network it is required to eliminate all variables that have unknown values (free variables). From the definition of conditional probability we can derive a rule to achieve that[6].

$$\mathbf{P}(v(X_i) = x_i | v(X_j) = x_j) = \frac{\mathbf{P}(v(X_i) = x_i \wedge v(X_j) = x_j)}{\mathbf{P}(v(X_j) = x_j)} \tag{2.4}$$

where X_i is the queried variable and X_j is the set of evidence variables with values x_j.
Introducing a normalization factor α saves us time for additional iteration of the data for the denominator.

$$\mathbf{P}(v(X_i) = x_i | v(X_j) = x_j) = \alpha \mathbf{P}(v(X_i) = x_i \wedge v(X_j) = x_j) \tag{2.5}$$

where

$$\alpha = \frac{1}{\sum_{x_i \in V(X_i)} \mathbf{P}(v(X_i) = x_i \wedge v(X_j) = x_j)} \tag{2.6}$$

Taking all free variables X_k into account, we have to extend the equation to sum over the possible values of those.

$$\mathbf{P}(v(X_i) = x_i | v(X_j) = x_j) = \alpha \sum_{x_k \in V(X_k)} \mathbf{P}(v(X_i) = x_i \wedge v(X_j) = x_j \wedge v(X_k) = x_k) \tag{2.7}$$

this will iterate over all possible values for all free variables, producing a probability distribution for the desired parameter. Since we have to sum over every free variable in the network, this procedure has to be considered NP-hard in most networks. The runtime complexity for a network with n free binary variables is $O(2^n)$[6]. In realistic cases with a high number of parameters this approach is not feasible. Therefore other approaches to solving this problem would be desirable.

Inference By Variable Elimination

When using the enumeration algorithm, many values are calculated multiple times. To avoid that, the variable elimination algorithm stores values calculated for each node into a vector (also called bucket) and looks them up on demand instead of calculating again. This is realized by ordering all nodes in the network into a list, such that each node is preceded by all its parent nodes. That way the variable elimination algorithm achieves much higher performance than enumeration in practice, while still staying exact. Though in the worst case, it will still take as long as the initial approach[6].

Inference By Simulation

Unlike the other two inference algorithms, the Markov Chain Monte Carlo algorithm (MCMC) estimates probabilities by sampling from the distributions of the free variables using the Gibbs-sampler. Sampling means taking a random value from a distribution according to its probabilities. Since a variable is considered independent of all other variables given its parents, children and children's parents (Markov blanket) the Gibbs-sampler uses the following formula to calculate probabilities for sampling. [6].

$$\mathbf{P}(v(X_i) = x_i' | \mathbf{mb}(X_i)) = \alpha \mathbf{P}(v(X_i) = x_i' | \mathbf{parents}(X_i)) \prod_{Y_j \in \mathbf{children}(X_i)} \mathbf{P}(v(Y_j) = y_j | \mathbf{parents}(Y_j)) \tag{2.8}$$

where $\mathbf{mb}(X_i)$ is the Markov blanket of node X_i, $\mathbf{parents}(X_i)$ is the set of parents of node X_i, and $\mathbf{children}(X_i)$ is the set of children of node X_i.
After a certain number of samplings over all free variables, the number of occurrences of the questioned state is counted and divided by the number of samplings. The more samplings are executed the more precise the result will be.

Comparison

While enumeration and variable elimination are both exact and so return the same results, the MCMC algorithm's precision can be influenced by the number of samplings executed. Furthermore Variable elimination was tested to be slow compared to to the MCMC algorithm, given few evidence in the network. Table 2.5 shows the MCMC algorithm in comparison to the exact result when 10 and 100 samplings are performed.

	Thyroid	Diabetes	Post Operative
VE	0.22	0.17	0.14
MCMC 10 samplings	0.22	0.19	0.23
MCMC 100 samplings	0.22	0.17	0.14

Table 2.5: Error rates inference algorithms

2.2.6 Scoring

In order to determine the quality of a network a scoring algorithm is needed. The easiest way to do this is to take a set s of patients which has not been used during creation of the network and for each patient in s set the non class nodes in the network respectively and run an inference algorithm on the class variable. By counting the number of right results an error rate can be determined. Another common way of measuring the error is the Mean Squared Error (MSE) which takes the probability the network returns for the result into account. So it measures the average probability of values for the class node that do not match the inputted patients data.

2.2.7 Results

Applying the described methods to the provided data we obtained several Bayesian networks, that were then scored. Table 2.6 shows the minimal error rates of different networks that were created.

	2 classes	3 classes
CI	0.47	0.62
K2	0.49	0.65
TAN	0.48	0.63

Table 2.6: Bayesian network error rates on provided data

For each value ten networks were created from a training set of 350 patients, and the one with the lowest error was selected. Furthermore, for easier comparison, we used the same number of samples for each class. The first column is obtained by splitting the training data into two classes, those who did lower their BMI and those who did not. For the second column the data is split into three classes, adding a class for a change in BMI of less than 1.
When using 2 classes the error can be observed to be close to 0.50, hence close to a random classifier. The same can be observed in the 3 class case. This indicates that the networks did not succeed in finding any significant relationship allowing to generally predict the outcome of the treatment in advance.

2.3 Artificial Neural Networks

An artificial neural network is a mathematical model resembling the structure of biological neural networks [7]. It consists of a number n of input neurons and a number m of output neurons. In case of a linear network the input and output layers are completely connected. That means any input neuron may influence any output neuron. This allows representation of linear relationships. Such a network is usually represented by a matrix w and a vector b, where w represents the strengths of the connections between the input and output layer and therefore has dimension $m \times n$. Since multiplying the weight matrix by a zero input vector would always yield zero, which might not be desired, we also need the bias vector b of length m.

$$w = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \tag{2.9}$$

The output of a linear network is calculated using the following equation.

$$a = wp + b \tag{2.10}$$

where a is the activation or output of the network and p is the input pattern.
In order to train the network for some problem we require a rule to update w and b. This is performed by iterating through a number of training examples and comparing the result with the desired output, then update w and b accordingly. For linear networks the perceptron learning rule can be applied.

$$e = t - a \tag{2.11}$$

where e is the error, and t is the desired output.

We can update the weight matrix w and bias vector b as follows:

$$w_{new} = w_{old} + ep \tag{2.12}$$

$$b_{new} = b_{old} + e \tag{2.13}$$

Additionally a neural network can be used as a classifier by using one binary output for each class using the **hardlim** function.

$$\textbf{hardlim}(x) = \begin{cases} 1, & \text{if } n > 0 \\ 0, & \text{if } n \leq 0 \end{cases} \tag{2.14}$$

Applied to Equation 2.10 this results in:

$$a = \textbf{hardlim}(wp + b) \tag{2.15}$$

Since we did not observe any linear relationships using the mutual and conditional information measures, we are mostly interested in non linear networks.

A non linear neural network additionally adds a number of hidden layers between the input and output layer through which the information is fed forward, usually using the **logsig** function.

$$\textbf{logsig}(x) = \frac{1}{1 + e^{-x}} \tag{2.16}$$

The information is fed forward through the layers using the following equation:

$$a_{i+1} = \textbf{logsig}(w_i a_i + b_i) \tag{2.17}$$

where a_i is the activation at the inputs of layer i, and w_i and b_i are the weights and bias of layer i. Furthermore $a_1 = p$. The activation vector of the last layer is the output of the network. In order to train a non linear network different algorithms can be used. In our experiments the back propagation algorithm was used which is described in [6].

2.3.1 Results

As explained in Section 2.2.7 we have split our training data in either two or three classes and processed the same number of samples for each class. Additionally we try to continuously predict the actual BMI reduction. Table 2.7 shows the performance of different neural networks given the three different tasks.

Hidden layers	2 classes	3 classes	Continuous
25	0.49	0.66	21.23
50	0.49	0.65	20.95
100	0.49	0.65	20.91
25,25	0.48	0.64	21.12
50,50	0.47	0.63	21.11
100,100	0.48	0.63	19.98
100,100,100	0.48	0.62	19.99

Table 2.7: Neural network error rates on provided data

Each network was trained using a training set of 350 patients. The training was continued until the change of the error was below 1% for a period of 1000 iterations over the training set. The first column specifies the structure of the hidden layers in comma separated form, where 25,25 means we have two hidden layers containing 25 neurons each. The second and third columns show the error rates when using the network as a classifier, while the last column uses the network as a regressor and shows the minimum mean squared error of the predicted BMI change.
In the two and three classes columns we can again observe a performance close to an random classifier. The continuous case the deviations are very large, so the network does not manage to predict the change of BMI in advance.

Chapter 3

Population Model

While the analysis of the provided data did only give us little insight on when the treatment works, we were still able to obtain a good understanding about the factors that play a role in the development of obesity and overweight. Based on this knowledge we were able to develop a population model that will eventually allow us to simulate the complex events that influence how persons interact, eat and grow weight. The model will simulate a large number of individuals based on different sub-models. A population may refer to a large geographic area, like The Netherlands, or just a smaller group with selected properties.
In order to be able to build a population model to determine effects of treatments on a larger scale, we required data about different treatments depending on quantifiable parameters. This will allow us to build a model (Development model) for the development of obesity given different treatments. Furthermore data on how obesity initially develops under different circumstances, can be used to build a model (Obesity model), taking external factors into account. This can include social factors, like peer pressure, environmental influences, like local food quality, and individual parameters, like income. Another factor would be the likelihood of obese people to actively search for help and so actually be able to benefit from treatment (Treatment model). This can at a later stage also be used to for instance analyze the effects of marketing campaigns (Marketing model) on the population. Additionally we can investigate ways to optimize the application of treatments and estimate the effects of this optimization. This means finding factors that help to select the optimal treatment for a certain patient. Finally we can have a look at financial aspects by analyzing data about the costs of different treatments (Financial model). The relations between the different sub-models are illustrated in Figure 3.1.

Compared to common simulation models, like replicator dynamics, we are modeling our population as a group of individual agents making individual decisions [8]. This means we do not average over our population, but stick with the full range of possibilities. This allows to study vastly more complex scenarios and gives opportunity to introduce local effects.

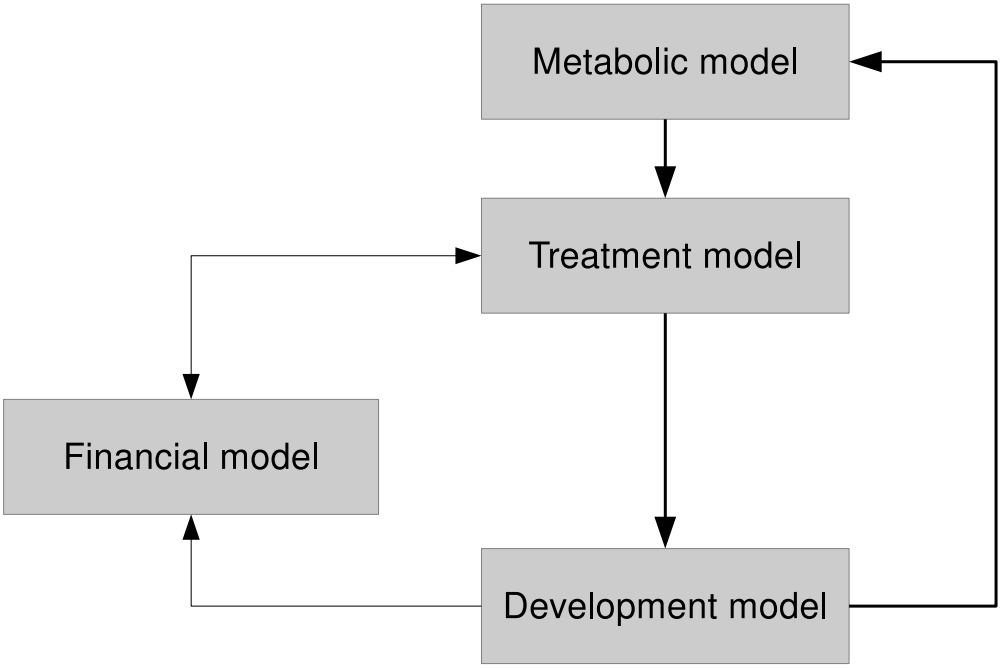

Figure 3.1: Structure of the obesity population model.

3.1 Implementation

In order to implement the model as a computer program, we need to quantify all relevant parameters. Furthermore we need to define the relations between the different parameters and sub-models in terms of equations.

3.1.1 Parameters

Each individual is modeled by a number of parameters defining the behavior and dynamics of a person.

- Age a
- Gender
- Height l
- Weight m
- Physical activity p
- BMR (Basal Metabolic Rate) r
- DCI (Recommended daily calorie intake) d
- BMI (Body mass index) b
- Satisfaction s
- Discipline f
- Caloric intake c
- Treatment probability
- Costs k
- Extreme physical activity calorie burn rate p_r
- Seasonal caloric intake c_s
- Satisfaction [Change rate / Noise / Decay]
- Discipline [Change rate / Noise / Decay]
- Caloric intake [Range / Noise]

3.1.2 Obesity Measures

The development of body weight is calculated using different formulas. Since the state representation gives complete information of how the body will develop, this part of the model is implemented as a Markov chain.

Body Mass Index

The body mass index (BMI) as defined by Adolphe Quetelet [9] is used to calculate satisfaction and the probability of treatment.

$$b = \frac{m}{l^2} \tag{3.1}$$

where m is the weight in kg and l is the height in meters of the person.

Basal Metabolic Rate

The basal metabolic rate (BMR) as defined by the Harris-Benedict equations [10].
The formula for men is

$$r_m = 66.5 + (13.75m) + (500.3l) - (6.755a) \tag{3.2}$$

and the formula for women is

$$r_f = 655.1 + (9.563m) + (185.0l) - (4.676a) \tag{3.3}$$

where m is the weight in kg and l is the height in meters and a is the age in years of the person.

Recommended Daily Caloric Intake

The recommended daily caloric intake (DCI) as included in the Harris-Benedict equations is defined by Table 3.1. We can reduce this table to an equation of p and r.

$$d = (0.7p + 1.2)r \tag{3.4}$$

where p is the physical activity parameter in the range $[0, 1]$ and r is the BMR.

Satisfaction

In order to estimate the satisfaction of the person with her/his weight and BMI the following formulas are used:

$$\bar{b} = \frac{|b - b_a|}{b_r} \tag{3.5}$$

Physical activity	DCI
Little to no exercise	$d = 1.2r$
Light exercise (1-3 days per week)	$d = 1.375r$
Moderate exercise (3-5 days per week)	$d = 1.55r$
Heavy exercise (6-7 days per week)	$d = 1.725r$
Very heavy exercise (twice per day, extra heavy workouts)	$d = 1.9r$

Table 3.1: Daily Caloric Intake according to Harris Benedict

where b_a is the average normal BMI of 21.75 and b_r is an upper limit for the deviation from the norm. Currently defined as 18.25.

Since the satisfaction value s will only change slowly over time we use the satisfaction change rate α to define how quickly it may change.

$$s_{t+1} = (1 - \alpha)s_t + \alpha(\frac{1}{e^{4-16\bar{b}} + 1} + \epsilon) \tag{3.6}$$

where s_{t+1} and s_t are the satisfaction values at times $t + 1$ and t and ϵ is the noise term. The values of s are bounded to the range $[0, 1]$.

Seasonal Function

Humans behave differently in different times of the year. For instance in the western culture more calories are consumed during the Christmas time. This variation is often the cause for people gaining weight. In order to include this factor into the model the "seasonal function" was included. This function allows us to model different seasonal behaviors in different regions (See Figure 3.2). It is defined as follows:

$$\begin{aligned} f_{seasonal}(t) = -1.25x10^{-16}t^{20} + 7.36x10^{-15}t^{19} - 1.79x10^{-13}t^{18} + 2.31x10^{-12}t^{17}. \\ -1.66x10^{-11}t^{16} + 6.37x10^{-11}t^{15} - 1.01x10^{-10}t^{14} \end{aligned} \tag{3.7}$$

where t is the month of the year.

Caloric Intake

The actual caloric intake of a person is calculated using the following formula:

$$b_s = \text{signum}\,|b - b_a| \tag{3.8}$$

$$c = d + b_s c_r f(1 - s) + (1 - f) f_{seasonal} c_s + \zeta \tag{3.9}$$

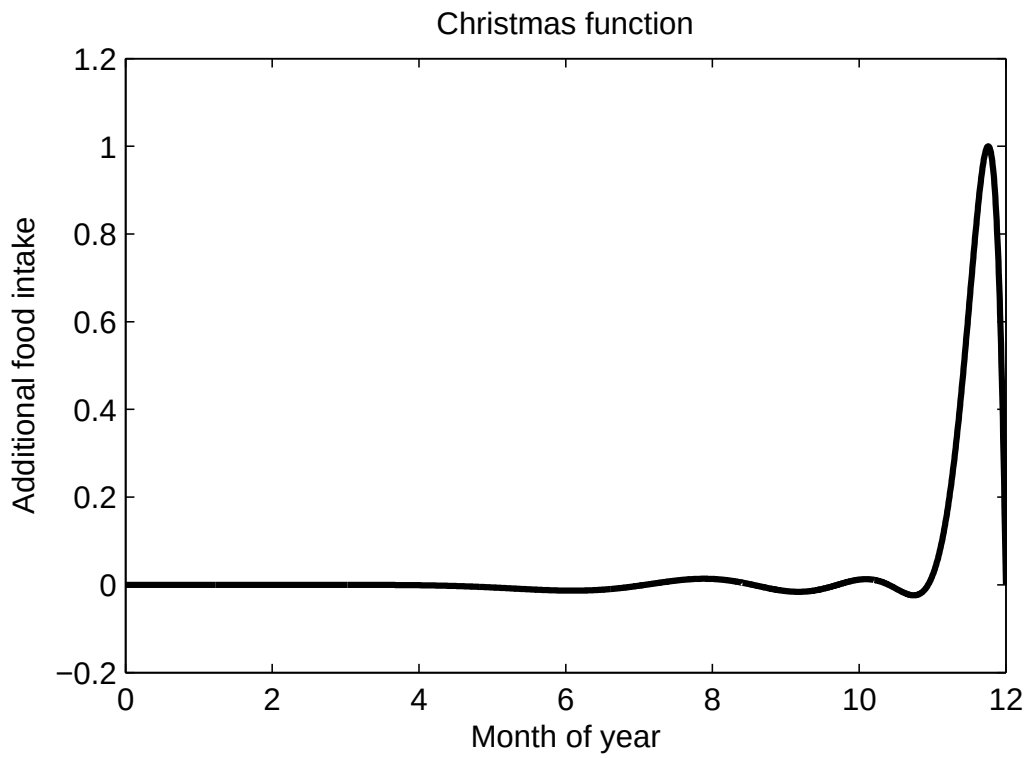

Figure 3.2: The seasonal function for Christmas.

where c_r is the amount of calories, a person can physically deviate from the DCI, f is the discipline value in the range $[0, 1]$, $f_{seasonal}$ is the seasonal function, c_s is the seasonal caloric intake range and ζ is the noise term. The values of c are bounded to the range $[0, 1]$.

Weight Change

The weight change calculation is based on the assumption that a kilo of fat contains about 3500kcal of energy ρ. This gives the following formulas:

$$\Delta m = (c - d - pp_r)/\rho \tag{3.10}$$

$$m_{t+1} = m_t + \Delta m \tag{3.11}$$

where m_{t+1} and m_t are the body weights at times $t + 1$ and t, p_r determines the maximum additional calories that can be burned per day by extreme physical activity.

The values for m are lower bounded to the mass of bones and organs, approximated to be 25kg per meter of height.

3.1.3 Treatments

In order to do comparisons of treatments three hypothetical treatments A, B, and C were designed and examined. Treatment A is a 6 month preventive treatment for overweight people with a BMI above 25, that is supposed to act preventive to Obesity. Treatment B is an 18 month treatment for obese people with a BMI above 30. Treatment C is an additional treatment to follow up either one of the others.
Each treatment's effect is defined using a number of parameters:

- Activity increase g [Mean $\hat{g}$ / StDev $\bar{g}$ / Decay $\dot{g}$]
- Discipline increase h [Mean $\hat{h}$ / StDev $\bar{h}$ / Decay $\dot{h}$]
- Daily cost i
- Duration of treatment j

The increase values g and h are only calculated when the treatment starts. This is where Gaussian noise is applied using the normal distribution $N(\mu, \Theta)$, with mean μ and standard deviation Θ.

$$g = N(\hat{g}, \bar{g}) \tag{3.12}$$

$$h = N(\hat{h}, \bar{h}) \tag{3.13}$$

The development of physical activity p and discipline f over the time of treatment is implemented using the calculated increase values and the decay values $\dot{g}$ and $\dot{h}$.

$$\Delta p = \frac{g}{j} \tag{3.14}$$

$$\hat{g}_{t+1} = g_t \dot{g} \tag{3.15}$$

$$p_{t+1} = p + \Delta p \tag{3.16}$$

$$\Delta f = \frac{h}{j} \tag{3.17}$$

$$\hat{h}_{t+1} = h_t \dot{h} \tag{3.18}$$

$$f_{t+1} = f + \Delta f \tag{3.19}$$

3.2 Simulation Results

When running the simulation on an average adult population it stabilizes at an average caloric intake of around 2000 calories with an average BMI of around 22. The simulation output of an average person is illustrated in Figure 3.3. Since we initialize each persons parameters independently, each configuration needs to first converge to its stable configuration. This could be observed to always happen within about the first 2 years of simulation. After this phase we can see the person to maintain stable weight, while the seasonal function causes a slight weight increase each year, which depletes over the remaining year. According to current publications the actual average caloric intake in Europe currently exceeds 3000 calories per day[11]. We optimized the parameters of the model to match this value. It turned out that there does not seem to be a stable configuration matching a caloric intake that high. In our model any population with a caloric intake this high would result in a very obese population on the long run. After only 1000 days of simulation using a constant caloric intake of 3000 calories per day the population averages with a BMI of above 50 and an average weight of more than 130 kg (see Figure 3.4). This shows that there is an urgent need to tackle nutrition problems in order to prevent the western societies from becoming very unhealthy.

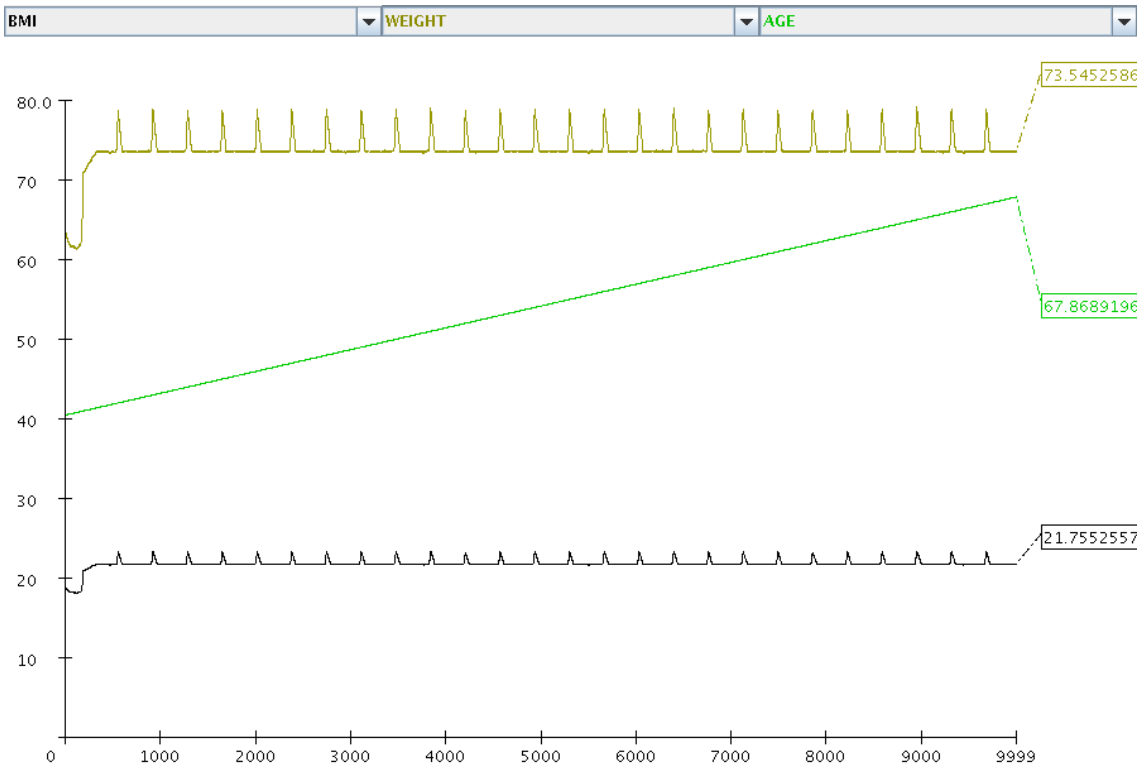

Figure 3.3: Simulation output of an average person.

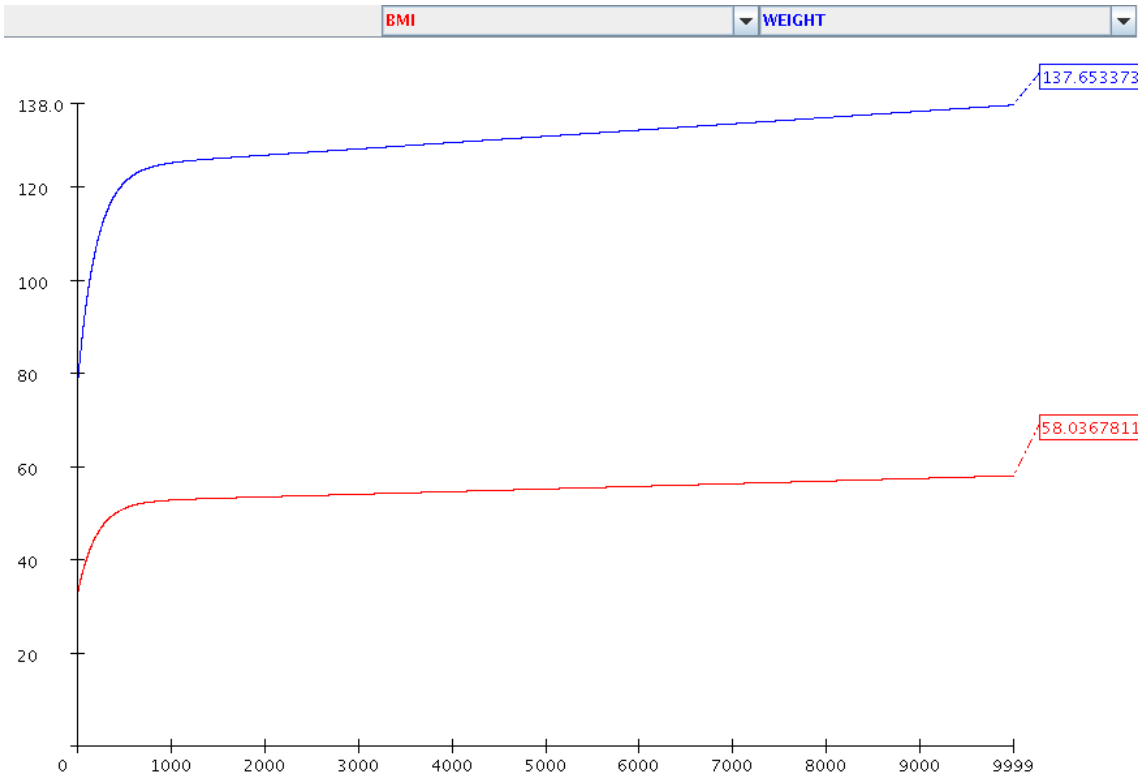

Figure 3.4: Simulation output on an average person in a population on 3000kcal nutrition.

3.3 Application of Treatments

In order to test the effects of the different treatments on our population we defined simple rules on when to start application. For the treatment A, that is a preventive treatment for overweight people, we used the BMI threshold of 25, that defines overweight. So everyone who exceeds that threshold will receive treatment. Similarly treatment B is applied when the BMI exceeds 30. Treatment C is only applied after treatment A or B, if the weight of the person has not yet normalized.
To be able to show the effect of preventive treatments we assigned different daily costs to our treatments. Treatments A, B, and C have daily costs of 6, 7 and 5 Euro respectively. When running the model on large populations with different properties, we can consistently show that applying preventive treatment at this cost ratio reduces the average BMI and the average costs. This is visualized in Figure 3.6. The first bar in each category results from a run without any treatments applied. The second run only applies the reactive treatment B. The third run also applies the preventive treatment A, while the fourth run additionally applies the follow up treatment C. All runs were executed on the same population containing 1000 persons. It can be observed that the application of the reactive treatment reduces the average BMI of the population. The introduction of the preventive treatment A reduces the BMI further, while creating less costs. Furthermore applying the long term follow up treatment C results in another slight reduction of the BMI, while creating additional costs. This indicates that the introduction of the preventive treatment reduces the overall costs in this setting.

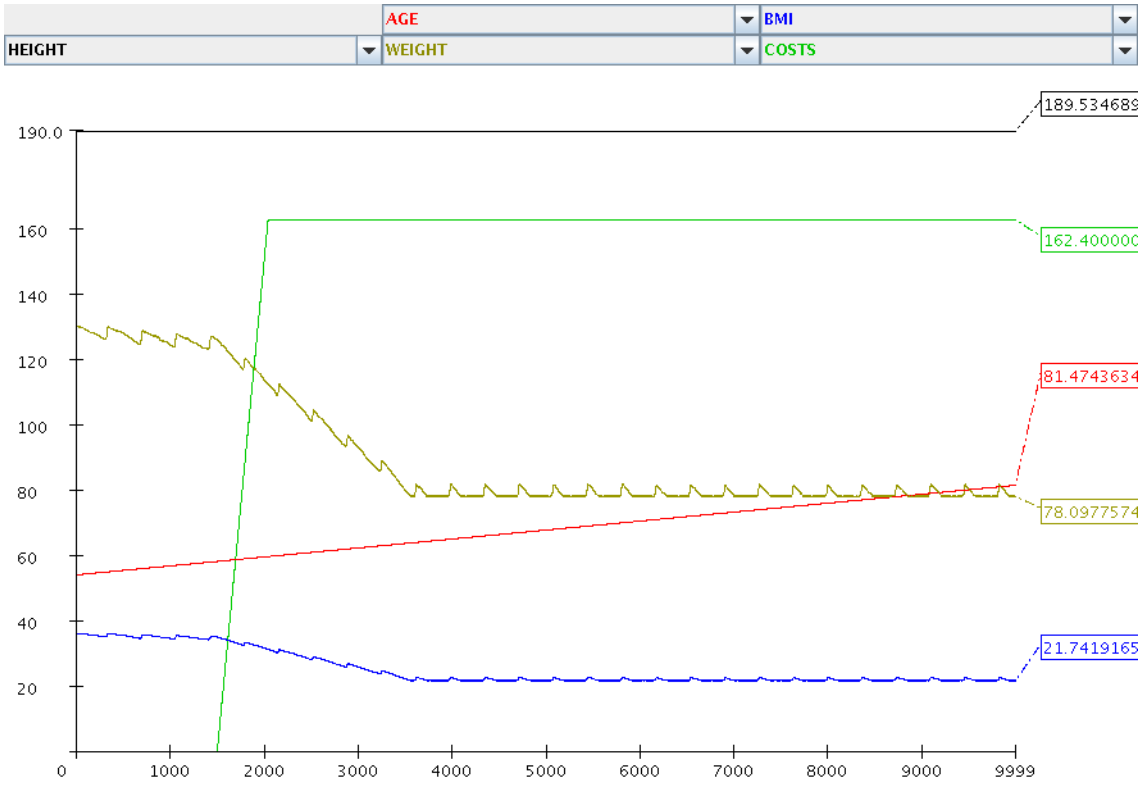

Figure 3.5: Application of a reactive treatment on an obese person.

Figure 3.6: Comparison of Treatment applications.

Chapter 4

Conclusions

The discussed methods and the described model allow us to simulate large populations, based on data measured from real populations. The Bayesian network approach allows us to find relationships in data, giving us insight about important factors. Although the approach did not succeed in finding such relationships in the data provided by the Department of Human Biology, the approach generally showed to be capable to extract this information. Furthermore the research about the provided parameters gave us a lot of insight into the topic of obesity and allowed us to obtain the required tools to design a population model. The model consists of a number of intuitive sub-models that together form a working simulation model. While the model still needs to be extended to cover more factors, we were already able to extract useful information about the development of obesity and the application of simple treatments. We will be able to use the completed model to simulate a high number of scenarios and optimize the application of treatment and marketing under different aspects, including finance and population health. The results can be very useful for pharmaceutical and health insurance companies, in order to evaluate the efficiency of certain treatments or whether those treatments should be covered by health insurance from a financial perspective.

Chapter 5

Future Work

The current implementation of our model does not yet include social interaction, epitaxy, genetic effects, marketing influences and most importantly long term effects of obesity and their respective costs. This will for instance contain heart disease, diabetes and depression. Additionally the metabolic model will be worked out in more detail, taking the composition of the calories, insulin, blood sugar, and other factors that might influence the metabolism, into account. In order to estimate parameter values for the simulation, we need to obtain additional data about populations, diseases and treatments. Once the model is complete we will be able to optimize strategies for the application of different specific treatments with respect to a variety of criteria. These criteria will include factors like population health and minimizing costs. Furthermore, we will be able to estimate the cost efficiency of the application of preventive treatments, or even their necessity given the alarming caloric intake statistics discussed in Section 3.2. After completion of the model we expect the sub-models to interact as illustrated in Figure 5.1.

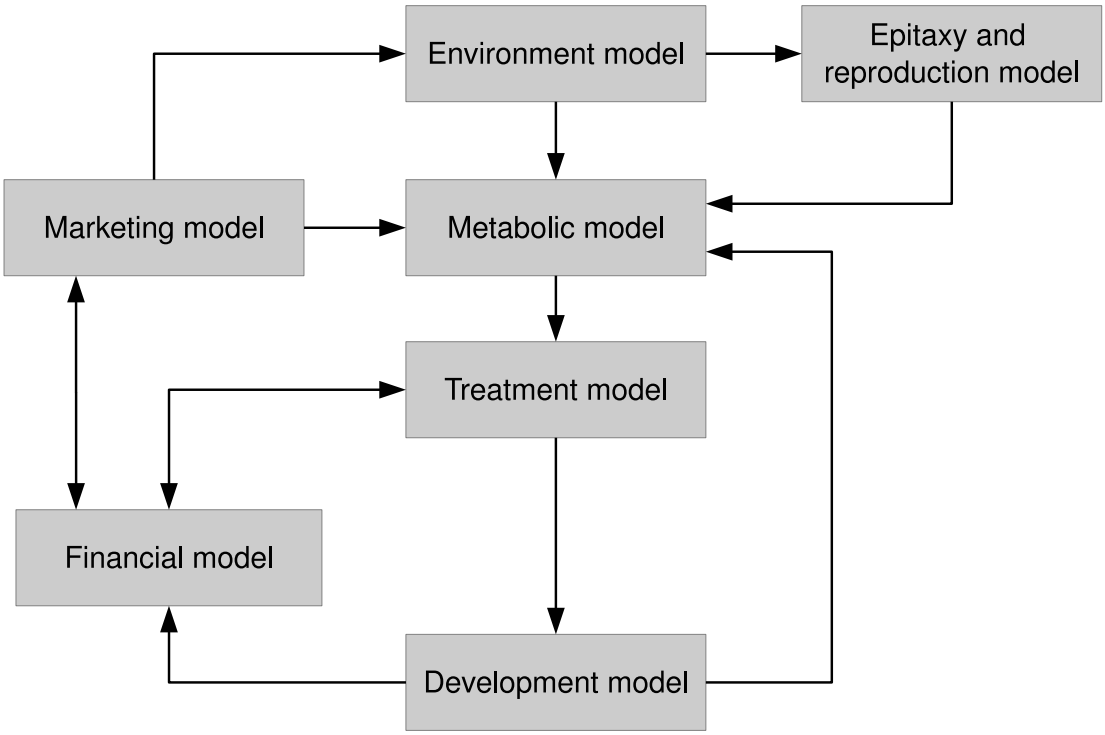

Figure 5.1: Future model layout.

Bibliography

[1] Quetelet A. The average man and indices of obesity. *Nephrology Dialysis Transplantation*, 23(1):47–51, 2008.

[2] Popkin BM and Gordon-Larsen P. The nutrition transition: worldwide obesity dynamics and their determinants. *International Journal of Obesity*, 28:2–9, 2004.

[3] Cooper GF and Herskovits E. A bayesian method for the induction of probabilistic networks from data. 1993.

[4] Hussein I. An individual-based evolutionary dynamics model for networked social behaviors. In *Proceedings of the American Control Conference, St. Louis 2009*.

[5] Cheng J, Bell D, and Liu W. Learning bayesian networks from data: An efficient approach based on information theory. 1997.

[6] Kruskal J. On the shortest spanning subtree and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48, 1956.

[7] Schmidhuber J and Traill TB. The changing structure of diets in the european union in relation to healthy eating guidelines. *Public Health Nutrition*, 9(5):584–595, 2005.

[8] Fogelholm M and K. Kukkonen-Harjula. Does physical activity prevent weight gain a systematic review. *Obesity Reviews*, 1(2):95–111, 2000.

[9] Braspenning PJ, Thuijsman F, and Weijters AJMM. *Artificial Neural Networks*. Springer, 1995.

[10] Neapolitan RE. *Learning Bayesian Networks*. Prentice Hall, Upper Saddle River, NJ, 2004.

[11] AM Roza and HM Shizgal. The harris benedict equation reevaluated: resting energy requirements and the body cell mass. *Am J Clin Nutr*, 40(1):168–82, 1984.

[12] Russel S and Norvig P. *Artificial Intelligence - A Mordern Approach*. Pearson Education International, second edition, 2003.

Appendix A

Full list of provided parameters

Base parameters:

- Age
- Gender

Obesity related parameters:

- Height
- Weight
- BMI
- Abdominal girth
- Fat percentage
- Fat mass
- Visceral fat
- Blood pressure systole left arm
- Blood pressure diastole left arm
- Blood pressure systole right arm
- Blood pressure diastole right arm

Blood values:

- NA
- K

- UREUM
- KREA
- MDRD
- CA
- BILT
- AF
- GGT
- ASAT
- ALAT
- ALB
- CRP
- CHOL
- HDL
- LDL
- TRIG
- CHOHDL
- GLUC
- TPSA
- BSE
- HB
- HT
- MCV
- MCH
- MCHC
- ERY
- LEUK

- TROM
- HBA1C%
- HBA1C
- FT4
- TSH

Appendix B

Pseudo Code

This section provides pseudo code for most of the mentioned algorithms.

B.1 CI-Based Search

The CI-based algorithm is composed out of three phases, drafting, thickening and thinning. The algorithm and the explanation are all taken from [4], and displayed here as a quick reference. In the first phase a draft of the network is created using mutual information of each pair of nodes. In the second phase, the algorithm adds arcs when the pairs of nodes are not conditionally independent on a certain condition set. In the third phase, each arc of the network is examined using conditional independence tests and will be removed if the two nodes of the arc are conditionally independent.

Phase 1;

1. Initiate a graph $G(V, E)$ where V=all the attributes of a data set, E=$\{\}$. Initiate an empty list L.

2. For each pair of nodes (v_i, v_j) where $v_i, v_j \in V$ and $i \neq j$, compute mutual information $I(v_i, v_j)$. For all the pairs of nodes that have mutual information greater than a certain small value ϵ, sort them based on their mutual information values and put these pairs of nodes into list L from large to small. Create a pointer p that points to the first pair of nodes in L.

3. Get the first two pairs of nodes of list L and remove them from it. Add the corresponding arcs to E. Move the pointer p to the next pair of nodes. (In this algorithm, the directions of the arcs are decided by the node ordering.)

4. Get the pair of nodes from L pointed by the pointer p. If there is no open path between the two nodes, add the corresponding arc to E and remove this pair of nodes from L. (A open path is an adjacency path where all the nodes are active.)

5. Move the pointer p to the next pair of nodes and go back to step 4 unless p is pointing to the end of L.

Phase 2;

1. Move the pointer p to the first pair of nodes in L.
2. Get the pair of nodes $(node1, node2)$ from L at the position of the pointer p. Call Procedure *find_cut_set(current graph,node1, node2)* to find a cut-set that can d-separate $node1$ and $node2$ in the current graph. Use a conditional independence test to see if $node1$ and $node2$ are conditionally independent given the cut-set. If so, go to next step; otherwise, connect the pair of nodes by adding a corresponding arc to E.
3. Move pointer p to the next pair of nodes and go back to step 1 unless p is pointing to the end of L. Otherwise the procedure is finished.

Phase 3;

1. For each $arc(node1, node2)$ in E, if there are other paths besides this arc between the two nodes, remove this arc from E temporarily and call Procedure $find\_cut\_set(currentgraph, node1, node$ to find a cut-set that can d-seperate $node1$ and $node2$ in the current graph. Use a conditional independence test to see if $node1$ and $node2$ are conditionally independent given the cut-set. If so, remove the arc permanently; otherwise add this arc back to E.

B.2 K2 algorithm

procedure K2;
{Input: A set of n nodes, an ordering on the nodes, an upper bound *u* on the
number of parents a node can have, and a database *D* containing *m* cases.}
{Output: For each node, a printout of the parents of the node.}
for *i*:= 1 to n do
$\pi_i := \emptyset$;
P_{old} :=f (i, π_i); {This function is computed using the equation below.}
OKToProceed := **true**;
While OKToProceed and $\mid \pi_i \mid <$ *u* do
let *z* be the node in Pred(x_i) - π_i that maximizes f (i, $\pi_i \cup \{z\}$);
P_{new} :=f (i, $\pi_i \cup \{z\}$);
if $P_{new} > P_{old}$ **then**
P_{old} := P_{new} ;
π_i := $\pi_i \cup$ {z};
else OKToProceed := **false**;
end {**while**};
write ("Node: ", x_i , " Parent of x_i : ", π_i);
end {**for**};
end {**K2**};

$$f(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} \alpha_{ijk}!$$

where:

π_i : set of parents of node x_i

q_i = $|\phi_i|$

ϕ_i : list of all possible instantiations of the parents of x_i then ϕ_i is the Cartesian product $\{v_1^{p_1}, ..., v_{r_{p_1}}^{p_1}\}x...x\{v_1^{p_s}, ..., v_{r_{ps}}^{p_s}\}$ of all the possible values of attributes p_1 through p_s.

r_i = $|V_i|$

V_i : list of all possible values of the attribute x_i

α_{ijk} : number of cases (i.e. instances) in D in which the attribute x_i is instantiated with its k^{th} value, and the parents of x_i in π_i are instantiated with the j^{th} instantiation in ϕ_i.

N_{ij} = $\sum_{k=1}^{r_i} \alpha_{ijk}$. That is, the number of instances in the database in which the parents of x_i in π_i are instantiated with the j^th instantiation in ϕ_i.

The pseudo code was taken from [12].

B.3 TAN algorithm

procedure TAN;
{Input: A set of n nodes, a class node and a database *D* containing *m* cases.}
{Output: A list of edges between pairs of nodes.}
for *i*:= 1 to n do
    for *j*:= 1 to n do
        Let weight$(i, j) = I(i, j)$
    end {**for**};
end {**for**};
Use Kruskal's algorithm to create a maximum weight spanning tree.
Select a root node and set the direction of all edges to be outwards from that. Add an edge from the class node to each of the n nodes.
end {**TAN**};

$$\mathbf{I}(X_i, X_j) = \sum_{X_i, X_j} \mathbf{P}(x_i, x_j) \log \frac{\mathbf{P}(x_i, x_j)}{\mathbf{P}(x_i)\mathbf{P}(x_j)}$$