UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Egert Georg Teesaar

Clustering Methods for Interpreting Medical Data

Master's Thesis (30 ECTS)

Supervisor: Sven Laur, PhD

Tartu 2020

Clustering Methods for Interpreting Medical Data

Abstract:

The medical bills can be analyzed to identify disease trajectories. By applying machine learning methods it is possible to find answers to questions, like which diagnoses occur together and from what these conditions arise.

This study uses various clustering methods, like Bernoulli mixture models and autoencoders compression with K-means, to divide patient into groups based on the diagnoses they have received. The results of the models are visualized on the heatmaps showing how likely it is to encounter specific diagnoses in those groups.

Also a guided hidden Markov model was used to form a lifelong disease path from the short segments of the different patients' treatment. This provides a way to observe how certain conditions arise in different ages and allows to track the disease development over time. It found similar results, what had been previously reported in medical studies, like development of J35 from H65.

The models interpretability was also improved by using support vector machines as a feature selection method for I11. This way it was possible to get rid of all the diagnoses, which had no connection to I11 and only keep those contributing to the development of the disease. Result on the processed data also agreed with the medical findings, like I50 development from I11.

Keywords: disease trajectory, medical bills, clustering, visualisation, interpretation, diagnose ranking, unsupervised learning, Bernoulli mixture models, hidden Markov models, K-means, autoencoder, support vector machines

CERCS: P170, Computer Science

Klasterdusmeetodid Meditsiiniandmete Interpreteerimiseks

Lühikokkuvõte:

Haiguste trajektoore saab uurida kasutades raviarveid. Kasutades masinõppe meetodeid on võimalik kindlaks teha, millised diagnoosid esinevad koos ning millest teatud seisundid on põhjustatud.

Antud lõputöö kasutas erinevaid klasteranalüüsi meetodeid, nagu näiteks Bernoulli segumudeleid ning automaatkodeerijate abil andemete kokku surumist koos K-keskmiste algoritmiga, et gruppeerida patsiendid neile väljakirjutatud diagnooside põhjal. Mudelite töö tulemused olid kujutatud soojuskaartide abil, mis näitasid, kui tõenäoline on kohata teatud diagnoose leitud gruppides.

Varjatud Markovi mudelite abil konstrueeriti ka elupikkune diagnoosikaart kasutades lühikesi lõike erinevate patsientide raviteekonnast. Selle abil on võimalik kindlaks teha erinevaid vanusegruppe puudutavaid seisundeid ning vaadelda nende arenemist läbi aja. Mudel kinnitas meditsiinilistes uuringutes kajastatud tulemusi, nagu näiteks J35 tekke

H65-st.

Mudelitöö paremaks lugemiseks rakendati ka tugivektormasinaid(TVM) andmete selekteerimisel. Vaatluse alla võeti diagnoos I11 ning TVK abil tehti kindlaks, mis diagnoosid mängivad haiguse arengus olulist rolli ning millised mitte. Töödeldud andmete põhjal tehtud uuringud kinnitasid samuti eelnevalt meditsiinivaldkonnas kindlaks tehtud, nagu I50 teke I11 poolt.

Võtmesõnad: haiguste trajektoorid, raviandmed, klasteranalüüs, raviandmete visualiseerimine, Bernoulli segumudelid, varjatud Markovi mudelid, K-keskmiste algoritm, automaatkodeerija, tugivektormasinad

CERCS: P170, Informaatika

Contents

1 Introduction

The goal of this project is to introduce methods, which help to study different disease trajectories from medical bills and represent them in a simple and human readable way. Disease trajectories can be defined as certain paths, which patients follow during recovery from specific diseases. Overall it focuses on clustering patients into different groups based on their current and previous condition. It looks for usually co-occuring diagnoses and find out what caused them.

Given it is a Master's thesis of Computer Science, the project focuses only on data driven approach with little to none domain knowledge available. This means that every conclusion the project arrives on, can be purely deducted from the present data. Furthermore, since this work mainly tries to describe a process, there is not such a thing as the ground truth at our disposal, which means mostly unsupervised learning algorithms are used. This includes clustering algorithms which let us represent patient condition with hidden states at various time steps.

Previously the treatment bills have been analyzed by M. Lippus in 2017 [8], who mainly used hidden Markov models. He focused on the diagnoses C50 and J44 and tried to predict how many people are likely to fall ill based on their condition. He looked at the specific services provided to a patient during treatment. This approach of focusing on a very detailed information had a somewhat negative effect, since the results weren't easily readable. Therefore this thesis tries to improve on the interpretability and focus more on the bigger picture.

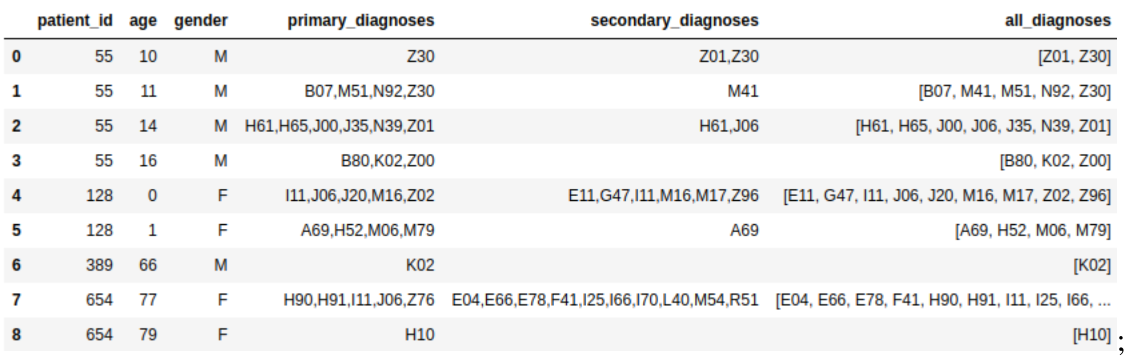
The section 2.1.2 gives a theoretical overview of the Clustering methods used in these thesis. It is followed by a section 3, which tells about how the models are and how the they are interpreted. Overall, there are three types of approaches which are looked at. The simplest is using Bernoulli mixture models to divide patients into groups based on just the diagnoses they had received in one year period. The second method is accompanying Bernoulli mixture models with hidden Markov models to also take advantage of the patients past conditions. However, both of these two models assume the diagnoses to be independent from each other, which is not necessarily a case. Therefore, the last approach tries to counter this issue by compressing the diagnose vector into the smallest representation possible and then applying a K-means clustering algorithm on the compressed data.

The last two sections 4 and 5 focus on making the output of the models more specific and interpretable. The first of them introduces age related guiding signals to hidden Markov models and puts together a lifelong diagnoses path from just a short sequences. The second one looks only at the patients diagnosed with hypertensive heart disease (I11) and applies support vector machines to compose a diagnose ranking, which would help to identify the most relevant diagnoses in progression of this disease and get rid of rid of all the non-important others.

1.1 Dataset overview

The data in hand consists of the medical bills of patients from all ages, genders and backgrounds. The data is gathered from 2010 to 2018. It consists of primary and secondary diagnosis vectors. For every secondary and primary diagnoses vector pair, there is specified a patient's ID, gender and age. However, the data is not always continuous, because some patients might have not payed a visit to a doctor at some ages. There are also different amounts of data available for different patients.

An example of the used dataset is shown in figure 1. Each row shows a diagnoses info a patient has gotten at a certain age. Since primary and secondary diagnoses aren't well differentiated, they are therefore concatenated together to form a column 'all diagnoses' and the values displayed there are hereafter referred to as *diagnoses vectors*.



	patient_id	age	gender	primary_diagnoses	secondary_diagnoses	all_diagnoses
0	55	10	M	Z30	Z01,Z30	[Z01, Z30]
1	55	11	M	B07,M51,N92,Z30	M41	[B07, M41, M51, N92, Z30]
2	55	14	M	H61,H65,J00,J35,N39,Z01	H61,J06	[H61, H65, J00, J06, J35, N39, Z01]
3	55	16	M	B80,K02,Z00		[B80, K02, Z00]
4	128	0	F	I11,J06,J20,M16,Z02	E11,G47,I11,M16,M17,Z96	[E11, G47, I11, J06, J20, M16, M17, Z02, Z96]
5	128	1	F	A69,H52,M06,M79	A69	[A69, H52, M06, M79]
6	389	66	M	K02		[K02]
7	654	77	F	H90,H91,I11,J06,Z76	E04,E66,E78,F41,I25,I66,I70,L40,M54,R51	[E04, E66, E78, F41, H90, H91, I11, I25, I66, ...
8	654	79	F	H10		[H10]

;

Figure 1. Structure of the dataset

Values in the table are random, but show the overall characteristics of the dataset.

2 Various clustering methods

In the following section the theoretical background for the main clustering methods is given. It starts off with introduction to Bernoulli mixture models, which in the upcoming section are used to cluster single diagnoses vectors. It is mentioned, how the parameter values for these type of models are learned with the expectation–maximization algorithm. It moves on to the hidden Markov models, which unlike Bernoulli mixtures also have an time dimension and can access the previous state. This proves useful when dealing with sequential real world data. The section is concluded with a brief overview of an autoencoder, which is a type of a neural network. This can be used to compress the data together and lose the correlation between different features, which comes in handy with the diagnoses vectors, where some sets of diagnoses are not independent, although Bernoulli mixture and hidden Markov models both assume them to be.

2.1 Mixture Models

2.1.1 Bernoulli distribution

The following is the adaption from [3, p. 440-441, 445–447].

A random variable X, which can only take values $\{0, 1\}$, is said to follow the *Bernoulli* distribution, if it equals to 1 with the probability of μ and 0 with the probability of $(1 - \mu)$. The resulting probability mass function is therefore

$$p(x|\mu) = \prod_{i=1}^{D} \mu^{x_i}(1 - \mu_i)^{(1-x_i)}$$

, where D represents the number of dimensions of random variable X. It can be seen that the values across dimensions have to be independent from each other.

2.1.2 Bernoulli mixture models

Mixture of Bernoulli distributions with K clusters is represented as:

$$p(x|\mu, \pi) = \prod_{k=1}^{K} \pi_k p(\boldsymbol{x}|\boldsymbol{\mu}_k).$$

Each cluster is weighted by $\boldsymbol{\mu}$, which is vector of length K and sums up to 1. Probability for $\boldsymbol{x}$ to belong to a cluster k before weighting is equal to:

$$p(\boldsymbol{x}|\boldsymbol{\mu}_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i}(1 - \mu_{ki})^{(1-x_i)},$$

where μ_{ki}, also referred to as *emissionvector* shows the probability for X to be one in dimension i, when X is from the kth cluster.

The likelihood of a dataset $X = x_1, ...x_m$ can be calculated with multiplying the probabilities for individual samples giving the formula

$$\ln p(\boldsymbol{X}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_n|\boldsymbol{\mu}_k)\}.$$

2.1.3 Expectation–maximization algorithm

The parameters $\boldsymbol{\mu}$, $\boldsymbol{\pi}$ can be learned by maximizing the log likelihood function and this can be done with the EM algorithm, which is great at dealing with models having latent variables.

Given the log likelihood function

$$\ln p(\boldsymbol{X}|\boldsymbol{\theta}) = \ln\{\sum_{\boldsymbol{Z}} p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}\},$$

where from the newly introduced symbols $\boldsymbol{Z}$ stands for the latent variables of the model and $\boldsymbol{\theta}$ stands for all of the models parameters. As it can be seen, the complication arises from the fact that the summation over the latent variables happens inside of the logarithm, therefore the logarithm can't act directly on the joint distribution. This wouldn't be a problem, if the corresponding latent value for each observation would be known, reducing the likelihood function to the form $\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})$.

In reality the latent values aren't known and this gives rise to the EM-algorithm, were in E step the expected value of Z under the posterior distribution $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}$ is evaluated, followed by the M-step, where this expectation is maximized. Therefore with some parameter values for $\boldsymbol{\theta}$ the posterior distribution $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}^{old})$ is found and the parameter $\boldsymbol{\theta}$ are updated with the formula

$$\boldsymbol{\theta}^{new} = \underset{\theta}{\operatorname{argmax}} \boldsymbol{\Gamma}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}),$$

where

$$\boldsymbol{\Gamma}(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}^{old}) \ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta}).$$

It can be seen that in this formula the log appears inside the sum and acts directly on the join distribution, making the M-step tractable. Each repetition of these two steps results in the increase of the log likelihood.

To summarize the whole algorithm:

1. Initialise parameters.

2. E-step Evaluate $p(Z|X, \theta^{old})$
3. M-step Evaluate θ^{new}
4. Check if log-likelihood of training data is converged or maximum number of iterations is reached. If not then replace θ^{old} with θ^{new} and repeat steps 2-4.

.

EM for Bernoulli mixture models. Coming back to Bernoulli mixture models,

$$p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\mu}) = \prod_{k=1}^{K} p(\boldsymbol{x}|\boldsymbol{\mu}_k)^{z_k},$$

where $\boldsymbol{z} = (z_1, .., z_k)$ is a latent variable with one component equal to 1 and other equal to 0.

Prior distribution for a latent variable is

$$p(\boldsymbol{z}|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{z_k}.$$

Complete-data log likelihood function with latent variables becomes

$$\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\{\ln \pi_k + \sum_{i=1}^{D}[x_{ni}\ln \mu_{ki} + (1 - x_{ni})\ln(1 - \mu_{ki})]\}.$$

Expectation of the complete-data log likelihood function, where $\gamma(z_{nk})$ is posterior probability of component k given x_n is therefore expressed as:

$$\mathbb{E}[\ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\mu}, \boldsymbol{\pi})] = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk})\{\ln \pi_k + \sum_{i=1}^{D}[x_{ni}\ln \mu_{ki} + (1 - x_{ni})\ln(1 - \mu_{ki})]\}.$$

In E-step the posterior probabilities are evaluated by firstly applying the mixture weight μ_k on a probability for an input vector given a μ_k and then normalizing the result over values from all the clusters

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = \frac{\pi_k p(\boldsymbol{x}_n|\mu_k)}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}_n|\boldsymbol{\mu}_j)}.$$

The expected number of samples to belong to a cluster k is equal to summation of the posterior probabilities of all the samples for a given cluster:

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

The following formula calculates the mean of the input vector for a cluster k:

$$\bar{\boldsymbol{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n.$$

In M-step the emissionvectors are updated by setting them equal to means of input vectors:

$$\mu_k = \bar{\boldsymbol{x}}_k.$$

New weight for the mixture k equal to the expected number of samples belonging to this cluster divided by all the samples count.

$$\pi_k = \frac{N_k}{N}.$$

2.2 Hidden Markov Models

The following is the adaption from [2, p 7-10].

Joint distribution of a collection of random variables $\{O_1, ..., O_T, Q_1, ..., Q_T\}$, where O_t stands for an observed variable at time t and $Q_t \in \{1, ...N\}$ for a hidden discrete variable or a *state* at time t. The HMM assumes, firstly, that the value of a hidden variable at time t only depends on the the hidden variable at time $t-1$ represented as a formula :

$$P(Q|Q_{t-1}O_{t-1}, ..., Q_1, O_1) = P(Q_t|Q_{t-1}).$$

Secondly, it assumes that the observation at time t, depends only on the hidden variable at time t and is represented by the formula:

$$P(O_t|Q_tO_t, ..., Q_1, O_1) = P(O_t|Q_t).$$

Thirdly, it assumes that $P(Q_t|Q_{t-1})$ is independent of time t, which let's to define a transition matrix $A = a_{i,j} = p(Q_t = j|Q_{t-1} = i)$, which in other words shows the probability for a move from state i to a j. The special case is only when $t = 1$, when initial state is defined as $\pi_i = p(Q_1 = i)$.

The emissionmatrix is defined as $B = \{b_j(.)\}$, where $b_j(o_t) = p(O_t = o_t|Q_t = j)$ gives the probability for an observation o_t while being in a state j. It was also one of the parameters in a mixture model. Therefore, HMM is fully defined when values for $\lambda = \{\pi, A, B\}$ are known.[2, p 7-8]

The three problems for a HMM are:

1. Find $p(O|\lambda)$ given $O = (o_1, ..., O_T)$
2. Find $Q = (q_1, ..., q_T)$, given λ and $O = (o_1, ..., O_T)$
3. Find $\lambda * = \text{argmax}_\lambda\, p(O|\lambda)$, given $O = (o_1, ..., O_T)$

2.2.1 Forward-backward algorithm

In the upcoming section, the forward-backward algorithm is presented with HMM which output follows the *multinomial* distribution.

Multinomial distribution. Let $X \in \{v_1, ..., v_L\}$ be a categorical random variable with L states, where

$$p(X = v_j) = \theta_j.$$

Then the multinomial distribution is given by a formula:

$$p(X|\theta) = Multi(X|\theta) = \prod_{i=1}^{L} \theta_i^{\delta_{X,v_i}},$$

where δ_{X,v_i} is equal to one, when $X = v_i$ and 0 otherwise [9].

The emissionprobability for state j seeing an observation with value v_k

$$b_j(o_t) = p(O_t = v_k | q_t = j).$$

Forward procedure. The probability of seeing a particular observation sequence from time 1 to time t and ending up in a state i given λ

$$\alpha_i(t) = p(O_1 = o_1, .., O_t = o_t, Q_t = i)|\lambda).$$

1. $\alpha_i(1) = \pi b_i(o_i)$
2. $\alpha_j(t+1) = [\sum_{i=1}^{N} \alpha_i(t) a_{ij}] b_j(o_{t+1})$
3. $p(O|\lambda) = \sum_{i=1}^{N} \alpha_i(T)$

Backward procedure. The probability of seeing a particular observation sequence from time t to time T starting from the state i given λ.

$$\beta_i(t) = p(O_{t+1} = o_{t+1}, .., O_T = o_T, Q_t = i)|Q_t = i, \lambda)$$

1. $\beta_i(1) = 1$
2. $\beta_j(t+1) = [\sum_{i=1}^{N} a_{ij} b_j(o_{t+1}) \beta_j(t+1)$

3. $p(O|\beta) = \sum_{i=1}^{N} \beta_i(1)\lambda_i b_i(o_1)$

When combining the backward and forward procedures, it is possible to calculate the posterior probability for cluster i at time t given an observation sequence and λ

$$\gamma_i(t) = p(Q_t = i|O,\lambda) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_i(t)\beta_i(t)}.$$

It is also possible to calculate the probability for transitioning from state i to state j at time t when seeing an observation O given λ

$$\xi_{ij}(t) = \frac{p(Q_t = i, Q_{t+1} = j, O|\lambda)}{p(O|\lambda)} = \frac{\gamma_i(t)a_{ij}b_j(o_{t+1}\beta_j(t+1)}{\beta_i(t)}.$$

Summing the posterior probability values of state i over all timesteps, it is possible to get, how much time it was spent in state i

$$\sum_{t=1}^{T} \gamma_i(t).$$

Summing the transitions from state i to j over all timesteps, it is possible to get, how much transitions were made from i to j

$$\sum_{t=1}^{T-1} \xi_{ij}(t).$$

EM for HMM. In M-step the parameter values are updated as follows. Starting probability for state i is set to be the posterior probability at timestep 1.

$$\bar{\pi}_i = \gamma_i(1)$$

Transitionmatrix field corresponding to transition from state i to j is updated to transitions from state i to j divided by how much time was spent in i

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}.$$

The probabilities for seeing particular observations in state i is updated by the number of times this observation value was seen while being in state i divided by the total time it was spent in state i.

$$\bar{b}_i(k) = \frac{\sum_{t=1}^{T} \delta_{o_t,v_k}\gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}.$$

2.3 Bernoulli Mixtures with Hidden Markov Models (BMMHMM)

When adapting hidden Markov models protocol to Bernoulli mixtures, it is only needed to change the formulas for emissionprobabilities.

When given an observation sequence $\boldsymbol{O} = (\boldsymbol{o}_1, ..., \boldsymbol{o}_T)$, where $\boldsymbol{o_j} = \{o_1, ..., o_D\}$, consists of D dimensional binary vectors, the emissionprobability for state j is calculated with the formula, where μ_j shows the means vector for state j

$$b_j(\boldsymbol{o}_t) = P(\boldsymbol{o}_t | q_t = j)) = \prod_{i=1}^{D} \mu^{\boldsymbol{o}_{ti}} (1 - \mu_i)^{(1 - \boldsymbol{o}_{ti})}.$$

The means are updated with the formula [11]:

$$\bar{\boldsymbol{\mu}}_j = \frac{\sum_{t=1}^{T} \gamma_j(t) \boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_j(t)}.$$

Everything else stays the same.

2.4 Guided Hidden Markov Models

Let's now define observation sequence to consist of two separate tracks $\boldsymbol{O} = (\boldsymbol{G}, \boldsymbol{H})$, where $\boldsymbol{G} = (g_1, .., g_T)$ and $\boldsymbol{H} = (h_1, .., h_T)$. This means $\boldsymbol{o}_t = (g_t, h_t)$. When the tracks are independent, then emissionprobabilty for a state j is expressed as:

$$b_j(\boldsymbol{o}_t) = p(G_t = g_t, H_t = h_t | q_t = j) = p_G(G_t = g_t | q_t = j) p_H(H_t = h_t | q_t = j) = b_{G,j} b_{H,j}.$$

In M-step the emissionmatrixes are updated separately: $\bar{b}_j = (\bar{b}_{G,j}, \bar{b}_{H,j})$

The defined structure essentially stands for two separate HMMs sharing the same starting probability π and transitionmatrix A, but two different emissionmatrixes B_G and B_H. It gets its name from the fact that one track can be used to define an HMM, which serves as a guide to another. Whenever the guide HMM observes a critical event in his track, he signals the other when $b(\boldsymbol{o})$ is calculated.

2.5 Neural Networks

The following is the adaption from [10]

Neural networks provide a way to learn complex non-linear functions from the data. Its lowest building blocks are hidden units called neurons.Output of a neuron, which is referred to as a neural activation, can be an input for another neuron. This means neurons form layers and stacking layers on top of each other form a network. The number of neurons in a layer define its dimensionality. Layer is said to be dense, if its every unit receives an input from its previous layer's every neuron.

The first layer of a network is called an input layer and the last layer is called an output layer, layers between these are hidden layers.The most common architecture consists of an input layer, one or more dense layers and an output layer. They are called feed-forward networks because the activations of one layer can be calculated and passed on in a feed-forward fashion. This means that there are no cycles in a network.

Weights for each neuron are learned by comparing the output of the last layer to the target value using some loss function. After this the weights are updated by the *back-propagation*, which modifies the weights of each neuron, so that the loss function value would decrease. This assumes that the target value is known and available, otherwise it would have nothing to compare the output against.

Autoencoders are a type of the neural networks, which deal with unsupervised learning process. They set the target value, what the network is trying to predict, equal to the input value. An example of an autoencoder can be seen in figure 2. When setting restrictions on the hidden layers, interesting structures about a data can be found. One option is to limit the number of hidden units in the hidden layers. When the dimensionality of the middle layer is smaller than the input and output layer, the data is passed through a bottleneck and the network is forced to learn a compressed version of the original data. This task would be very complicated with random data, but if the features of the input are correlated to each other, the network can make use of them.

2.5.1 K-Means

The following is the adaption from [3, p 424-425].

Let's say there is a dataset $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ with N samples, each with D dimensions and that there are K different clusters and μ_k is a vector showing centres for a cluster k. Let's also assume that inter-point distances are small for the samples of the same cluster and large for samples between different clusters. Then let's define an objective function

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n_k} ||\boldsymbol{x}_n - \boldsymbol{\mu_k}||^2,$$

where $r_{nk} \in \{0, 1\}$, depending whether sample n is assigned to cluster k or not. Overall goal of the algorithm is to cluster the dataset into K clusters by finding appropriate values for r_{nk} and find a value for each u_k to find a minimum the the objective function. This can be by an iterative process called EM algorithm described in previous section. Steps of the process:

1. Some initial values are set for μ.

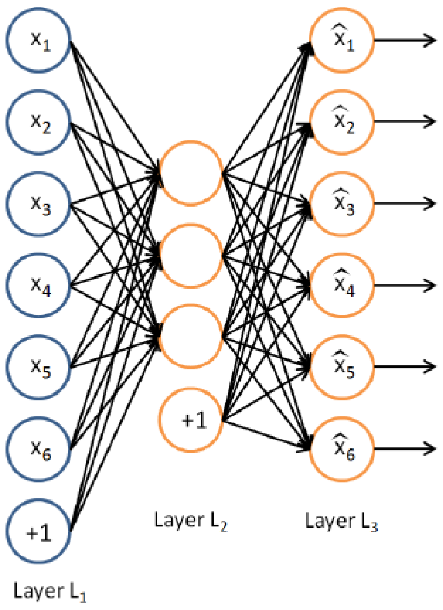


Figure 2. Example of an autoencoder

Picture taken from [10]. Layer L_1 serves as an input, layer L_2 as a compression and a layer L_3 as an output layer.

2. In the first step, values for r_{nk} are found by minimizing the objective function, while keeping the μ fixed. In other words, assigning a sample into the closest cluster,

$$r_{nk} = \begin{cases} 1 & \text{if } k = argmin_j ||\boldsymbol{x}_n - \boldsymbol{\mu}_j||^2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

   Committing a sample fully into just one of the clusters is called hard-clustering.

3. This is followed by a M step where updated values for μ are found by keeping the r_{nk} fixed. In other words, finding the mean of all data points which were signed to a particular cluster

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}.$$

4. These 2 previous steps are done until convergence or a maximum number of iterations is reached. Since in every iteration the objective function value is reduced, the algorithm converges into local or global minimum.

3 Results

3.1 Bernoulli Mixture Models

Motivation. The overall goal was to find answers to two questions. What diagnoses appear together and which diagnoses are followed by which diagnoses? The simplest approach which could be taken to answer the first of them, was to use BMMs to find clusters of similar diagnoses from patients' diagnoses vectors. This method assumed that the individual diagnoses are independent from each other and a patient's previous condition doesn't play any role in his/her current state.

3.1.1 Training process

Model was trained on the diagnoses vectors from 40k randomly selected patients from all ages and genders. One diagnoses vector formed one sample of the training set. Over 1500 different diagnoses were encountered in the dataset. Each patient could provide 1 to 8-9 diagnoses vectors, because of the data was gathered from 2010 to 2018. The chosen number of mixtures was set to be 5 to maintain the interpretability of the results.

Transitionmatrix, which isn't a parameter of a BMM, was calculated using the posterior probabilities for the diagnoses vectors from the patients having more than one year of data available and tracking the movement between two consecutive pairs of probability vectors.

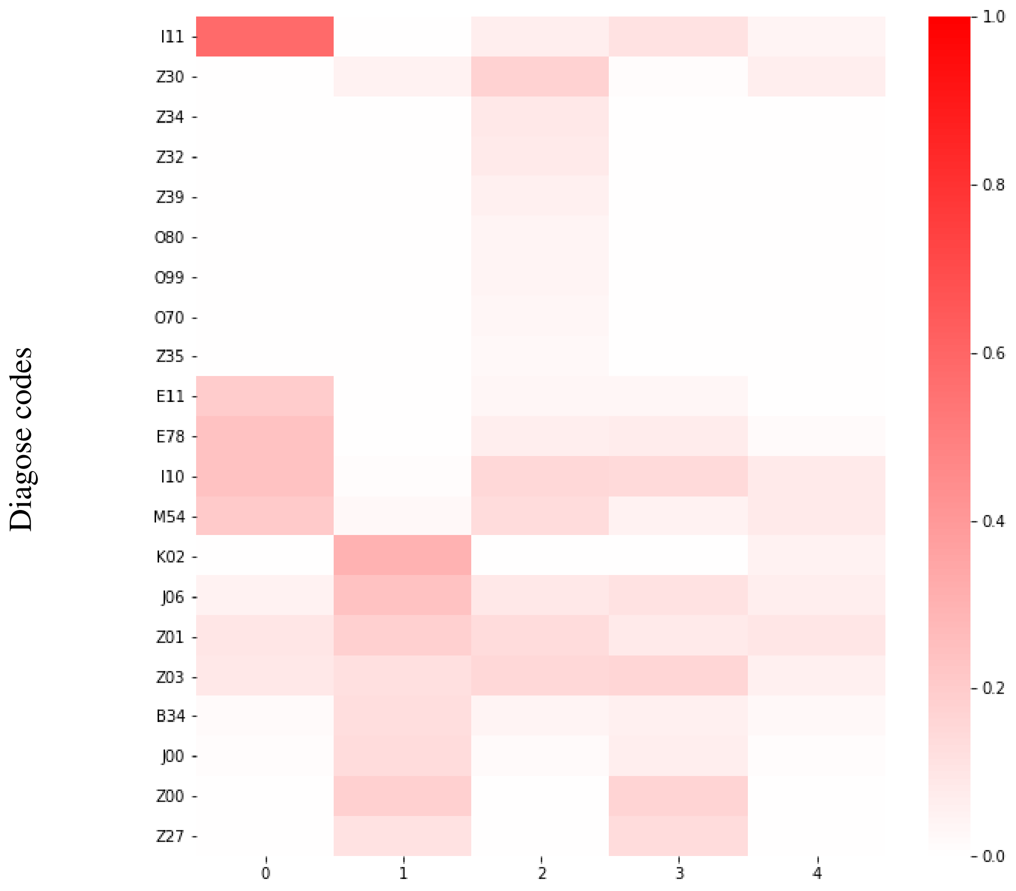
3.1.2 Model interpretation

Emissionmatrix. The emissionmatrix of the trained model, which shows how likely it is to encounter particular diagnoses in each of the states, was visualized using a heatmap. Since there were over 1500 possible diagnosis, it was not feasible to plot the probabilities all at once. Therefore only the diagnoses, which had a probability greater than 0.2 in any of the 5 states, were displayed.

The results can be seen in figure 3. States 0 and 1 are showing the most clear signals, other 3 are not so clear. State 1 shows high involvement of diagnoses from K02 to Z27. State 0 shows high probability for I11 accompanied by E11, E78, I10 and M54. States 2,3,4 show somewhat middle ground for the mixtures encountered in the states 0 and 1.

Transitionmatrix. The transitionmatrix, which shows the patients' movement between the states, was visualised using a graph. Nodes of the graph represent the states and edges show the movement from one state to a another. A self-loop signals that a patient stayed in the same state as he/she was in the previous time.

The results can be seen in 4. Patients in states 0 and 1 tend to stay in their states a bit more than patients from other states respectively. Most traffic seems to be to the state 4.



States

Figure 3. Emissionmatrix of the BMM

Heatmap of the filtered emissionmatrix, where diagnoses are from top to bottom and states are from left to right. From the full emissionmatrix only the diagnoses, which had a probability greater than 0.2 in any of the 5 states, were displayed. Dark red shows high chance to encounter a certain diagnose at a particular state, while light red or white shows low chance. For example K02 is present in over 40% patients being in state 0, but it isn't a common diagnose for state 1,2 and 4 patients.

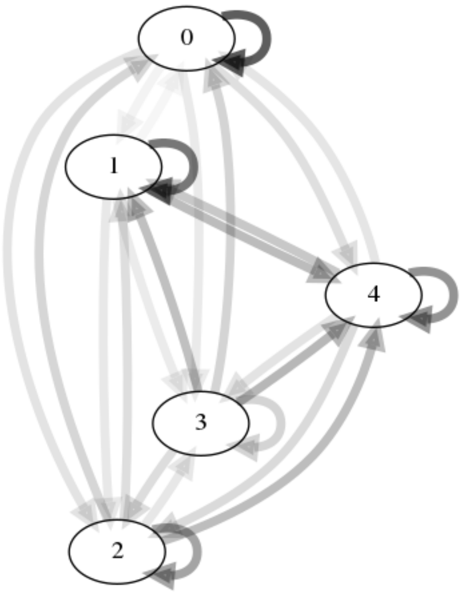


Figure 4. Movement between the states of the BMM

Graph displaying the probabilities for a patient to move from one state to another. Non transparent lines show probabilities close to 1, while light grey lines show low probabilities.

3.2 Bernoulli Mixtures with Hidden Markov Models (BMMHMM)

In the previous method, the assumption was made that a patient's previous condition doesn't play any role in his/her current and future states. It is generally known that this is rarely the case, maybe only with some unexpected misfortunes, like the car accidents and the bone fractions caused by falling out of carelessness. A 2019 medical guideline [5, p 266] reported that patients with diabetes have two fold-excess risk of a heart disease, ischaemic stroke, and vascular deaths. Therefore taking a patient's previous condition into account is essential. This problem could be countered with HMMs, since they can access a patient's previous state.

3.2.1 Training process

The model was trained on the same dataset as was the BMM. Instead of just the single vectors, which the BMM took as input, the BMMHMM was provided a sequences of diagnoses vectors. One sample from the training data consisted of all the diagnoses vectors about a particular patient. The number of mixtures was kept the same as with the BMM for comparison reasons.

3.2.2 Model interpretation

Emissionmatrix. The visualisation of the emissionmatrix can be seen in a heatmap in figure 5. Although BMMHMM results are showing a bit stronger colors, they are pretty similar to BMM' findings.

States 0,1 are almost identical to the states 0, 1 of BMM, only that there seems to be a stronger signal for active diagnoses in state 1, meaning that BMMHMM was more concrete when clustering patients with these diagnoses.

Smaller differences arise in states 2 and 3. In state 2, there is lightly lit block going from Z30 to Z35 and in state 3, it gives the strongest probabilities for I10, M54 accompanied by Z01 and Z03. In BMM plot however, these 2 states seem to be both represented by state 2. Keeping this in mind, BMMHMMs states seem to be a slightly more clearly defined and therefore probably makes better job at clustering people based on their condition.

Transitionmatrix. The movement between the states can be seen in figure 6. The differences to the BMM are much more evident. BMMHMM shows very strong colors for self loops, meaning that patients clustered to some state are very likely to stay in that state, which was to be expected since BMMHMM had an access to the whole patient's previous condition. This hints that BMMHMM's emissionmatrix gives more accurate prediction about the diagnoses a patient might receive in the future.

3.3 Autoencoder with K-means

Motivation. Although BMMHMM could look into past states, it still assumed individual diagnoses to be independent from each other like the BMM. This assumption, however, doesn't hold as well in real world. For example, high blood pressure was deemed to be a common feature with type 1 and type 2 diabetes according to medical guidelines of 2018 [12, p 3082]. This problem could be countered by compressing the dimensionality of the input vector to the lowest representation possible and afterwards cluster the data on the compressed representation. This could be done with autoencoder followed by K-means clustering. Although this approach doesn't look into patients' past, it helps with the conditional independence issue between the diagnoses.

3.3.1 Training process

First step was to train an autoencoder to get the compressed representation of a diagnoses vector. Autoencoder with just one hidden layer was chosen. The next step was determining the sufficient size/dimension of the compression layer. This was done by continuously reducing the size of the layer and tracking the recreation loss and when it went over some threshold, the process was stopped and the previous layer size was fixed.

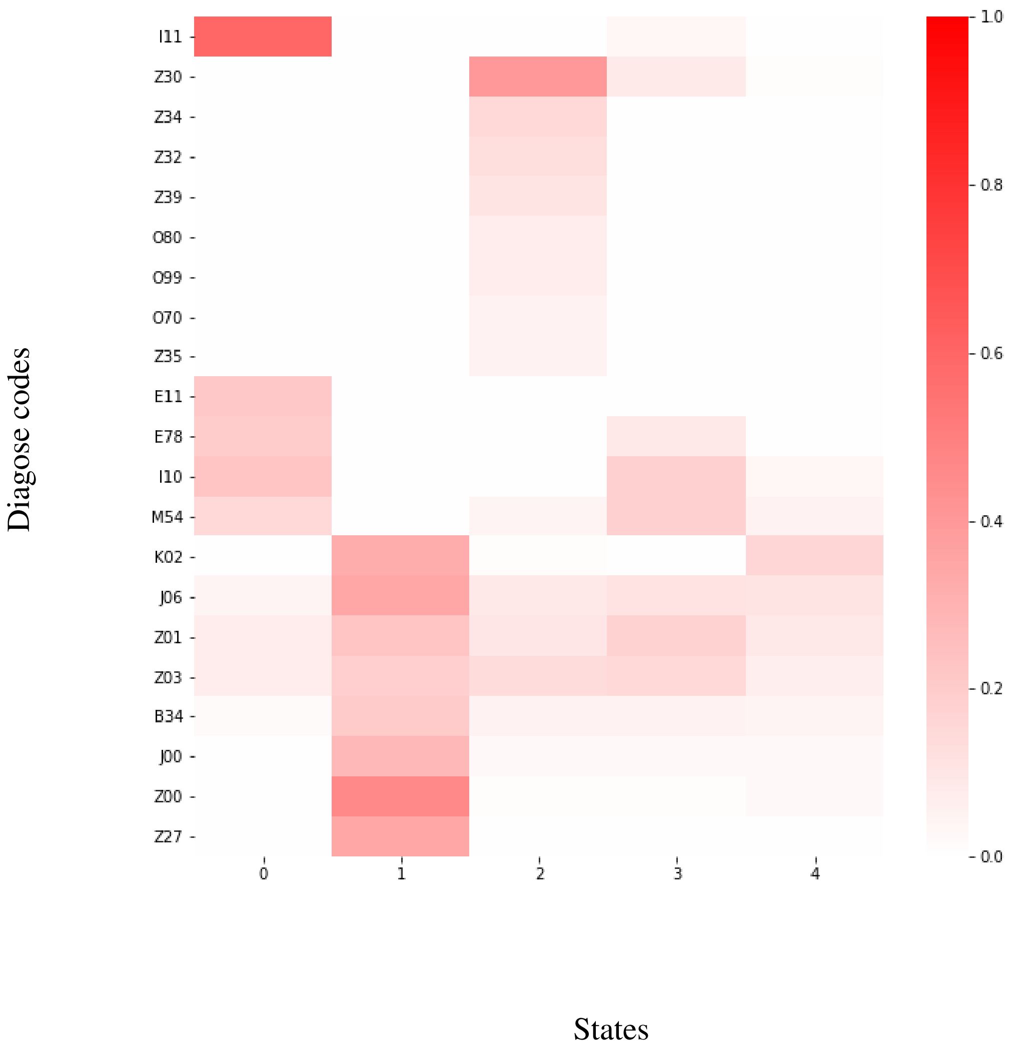
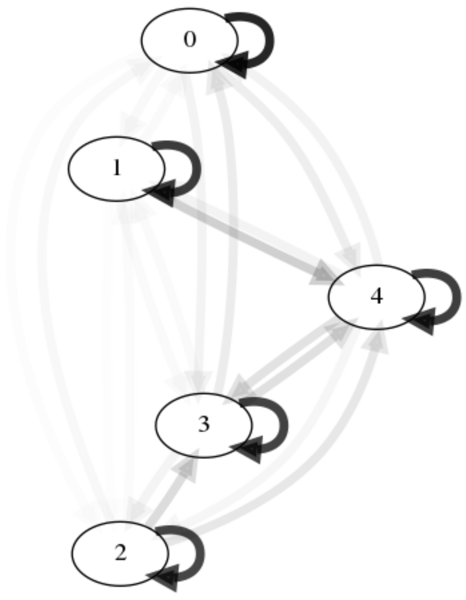


Figure 5. Emissionmatrix of the BMMHMM

Results for this process can be seen in figure 7. They show that with the compression layer size of 400, the autoencoder gets wrong about 2 diagnosis from the complete diagnosis vector. Since going from size 1600 to 400 were sufficient enough, adding more hidden layers was deemed unnecessary.

After completion of the previous steps it was possible to cluster the data on the output of the compression layer with K-means algorithm. States number was kept the same as with BMM and BMMHMM.



;

Figure 6. Movement between the states of the BMMHMM

Finally the predictions of the K-means model were used to deduce the emissionmatrix and transitionmatrix. Transitionmatrix was calculated with the similar process as was done with the BMM previously, only that instead of evaluating the posterior probabilities for each diagnoses vector, K-means assigned a compressed version of the diagnoses vector to a specific cluster.

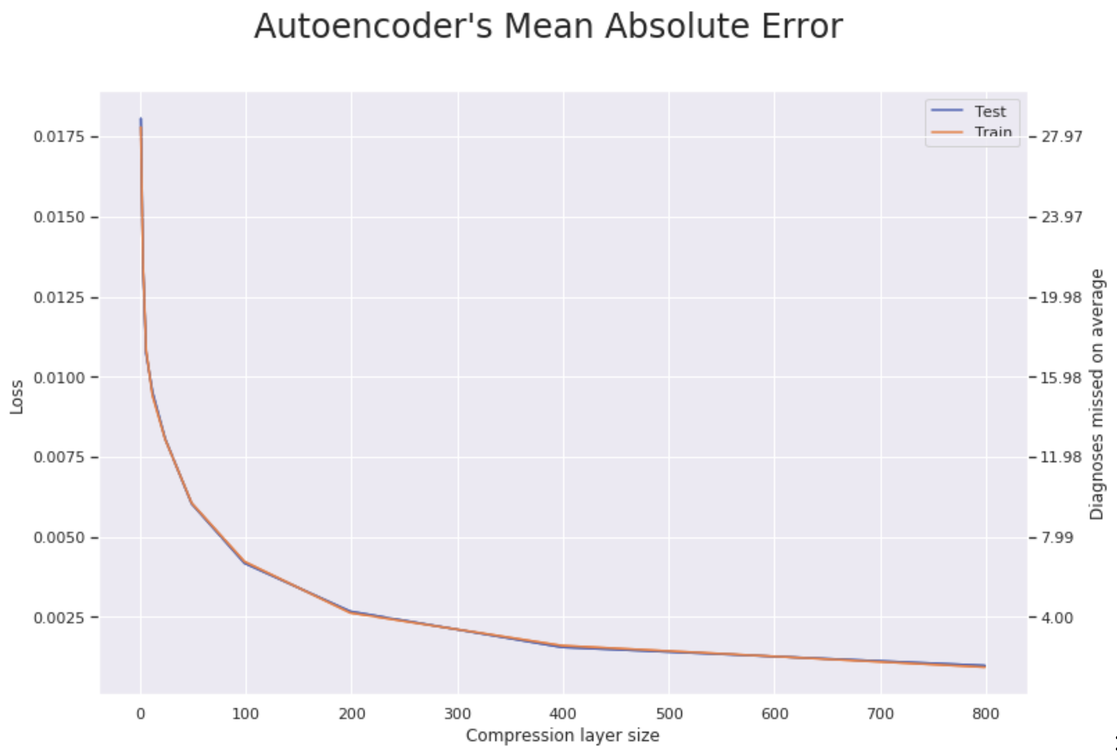
The emissionmatrix was put together by looking at diagnoses vectors and K-means assignments. Its value for a field corresponding to a diagnose d and state j equaled to an amount of how much the diagnose d had been seen in diagnoses vectors, which compressed representations K-means had assigned to the cluster j.

3.3.2 Model interpretation

Emissionmatrix. The visualisation of the emissionmatrix can be seen in figure 8. The results are pretty similar to results of BMM and BMMHMM, although the current model seems to show the most concrete clusters of diagnoses color-wise when compared to the two previous models. Contrasting colors could be also caused by the soft- vs hard clustering differences.

State 1 is identical to BMMHMM' and BMM' state 0, with the exception that K02 is more highlighted. State 2 is the same with previous models' state 1, with more emphasis put on I11. State 4 is lightly lit as were the state 4s in previous models.

The main difference comes in with state 0, where diagnosis Z34 to Z35. In BMMHMM these were associated with state 2, but the signals there weren't as clear as they are in



;

Figure 7. Plot for determining the compression layer size

The left y-axis shows the mean absolute error of the autoencoder, while the right y-axis tells how many diagnoses the model got wrong on average with the recreated input.

this model and in BMM this state was practically missing.

Overall there seems to be some improvement definition-wise and most states are showing a specific hint for the patient condition.

Transitionmatrix. Patients movement of the current model across states, which can be seen in figure 9 seems more similar to the ones of BMM than BMMHMM. This was to be expected because K-means as BMM doesn't have access to patients previous conditions. Similarly to BMM most movement is in the direction of state 4, with the exception that the lines are a bit darker, meaning that patients tend to stay in the other states for only a short time.

3.4 Summary of the models and discussion

The comparison for all previous can be seen in figure 10, where all of their states are put together. The models seem to agree on a correlation of diagnoses I11, E11, E78, I10 and

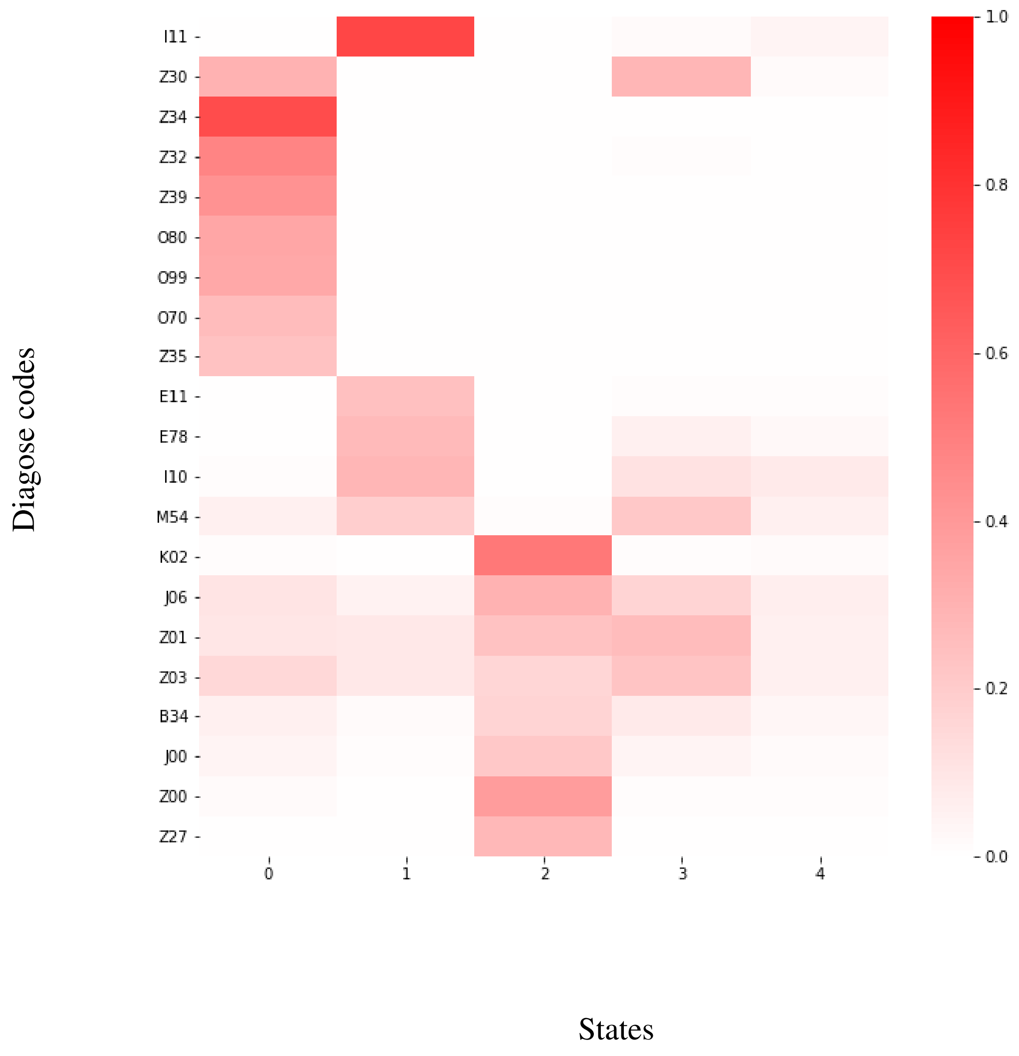


Figure 8. Emissionmatrix of the K-means with autoencoder compression

M54. Also they all provided a state, where on the heatmap diagnoses from K02 to Z27 were represented.

The main difference game with the diagnoses from Z34 to Z35 for which K-means defined a very clear state, but BMM and BMMHMM, on the other hand, reported only a slight activation.

Lightly colored state 4s make sense when looking at the patients' movement between states. In BMM and K-means models the state 4 is the most popular destination and

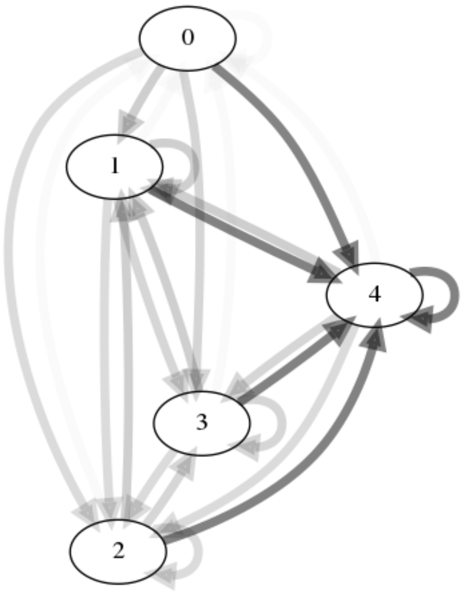


Figure 9. Movement between the states of the Autoencoder + K-means

logically with greater accumulation of patients it is harder to find the common ground diagnose-wise. Overall movement, however, was most consistent in the BMMHMM, because of highly active self-loops.

If there is a need to measure, how good is the model quantitatively, one way to do it would be by calculating the probabilities for an unseen diagnose vector to belong to a particular state, then weighting and summing together the model's emissionvectors based on these probabilities and lastly comparing this summation to the original diagnoses vector. Secondly, for HMM it is also possible to give a partial sequence of observations and let him predict posterior probabilities for the upcoming state. Then the probable diagnoses vector can be composed based on these and it can be compared with the actual diagnoses vector.

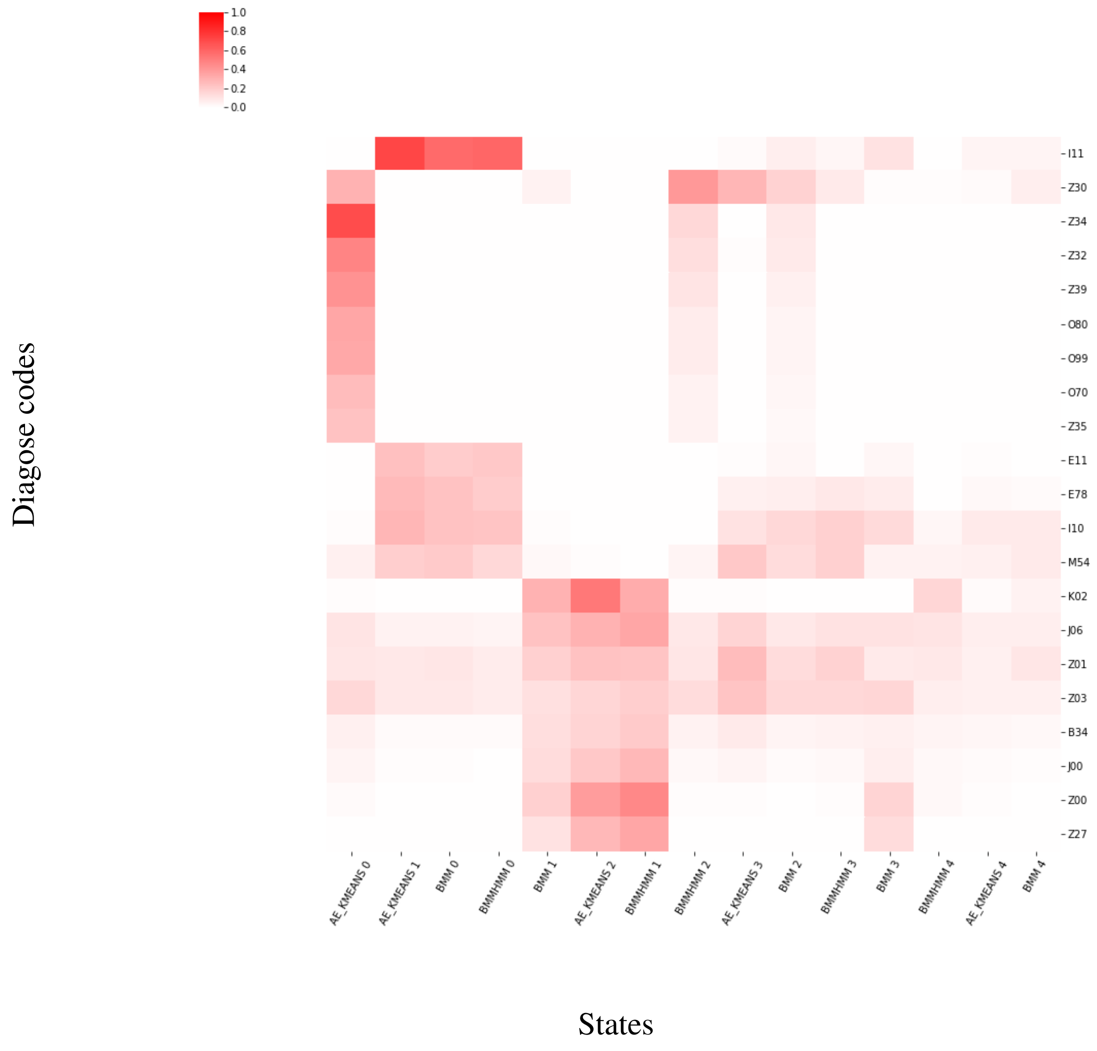


Figure 10. Comparison of the emissionmatrixes of BMM, BMMHMM and Autoencoder + K-means

Heatmap comparing the emissionmatrixes from all three models. Rows and columns are clustered by hierarchical linkage clustering is done by seaborn clustermap (viide).

4 HMM with guiding signals

4.1 Motivation

Previously, with just 5 states for a model, very little could be read from the results. The second question of which diagnosis are followed by which diagnoses was left unanswered, because it was hard to see the causal effect for the diagnoses since there were only 5 states total and patients could move freely between all states. Therefore adding a concrete timeline would definitely help with the clarity and interpretability of the results. This could be done by introducing age groups and using the guiding signals with HMM to better track patients movements.

4.2 Results

4.2.1 Training process

Patients' ages were divided into age groups. A study [7] published in 2013 found the meaningful age groups to be: 0-2, 3-5, 6-13, 14-18, 19-33, 34-48, 49-64, 65-78 and 79-98, using K-means algorithm. A guided HMM combining tracks from two separate HMM models were defined: one focusing on the diagnoses vectors (DiagnoseHMM) and the other one on patient's age (AgeHMM). Each age group was given 5 states to work with, with the total of 45 states over all groups. Transitionmatrix was initialised in the way that a patient can either stay in the same age group or move higher, but not backwards. The probabilities for the legal moves were allowed to change during training. AgeHMM emissionmatrix was with the shape of (45,9): 45 states and 9 possible age groups. It was initialised in a way that all 5 states dedicated to some age group had a value 1 in the emissionvector fields for ages in that respective group and 0 for ages from the other age groups. In other words, AgeHMM only expected to see one age group ages in a particular states. This emissionmatrix was not updated during training.

4.2.2 Model interpretation

Results can be seen in figure 11. It can be read from the plot at what point in life some specific conditions arise and in which diagnose combinations they come from. For example, there are signs of I11, which is a type of heart failure, already in ages of 34 to 48. From there on it gets more defined as the time goes by and becomes almost inevitable in the later stages of life. In the second state of the age group 49-64, the I11 is accompanied by E11, E78 and somewhat with M54, I50 and I25.

The other good example of a state is the last state in age group 19-33, where maternity related diagnoses Z334 to O99 are active. Some confusion might arouse around Z39, the "Encounter for care and examination of mother immediately after delivery", which is slightly lit also in the first state of age group 0-2, but there are indeed some newborn patients in the database with such diagnose. This confirms the problem, that there are entries in the data about some patients, which don't actually concern the patient at all.

If looking at the progression of some specific disease over multiple age groups, one example is that in the 3rd state of the age groups 0-2 the active diagnose H65 is accompanied by a raised J35 activity in the 3rd state of the age group 3-5. This finding has also been reported before [1].

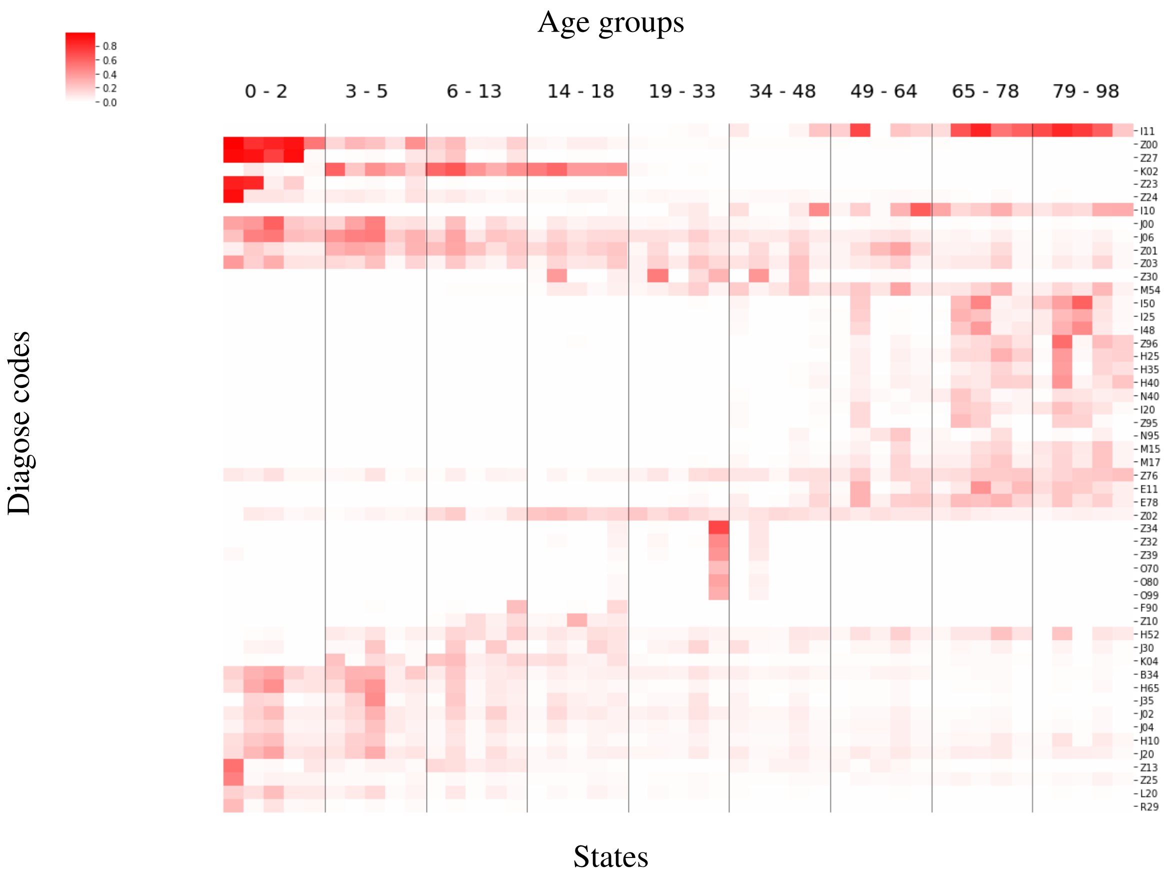


Figure 11. BMMHMM with age related guiding signals

Heatmap divided into sections based on the age groups shown above. In each age group there are 5 states. Only the diagnoses exceeding the probability threshold of 0.2 in any of the states across all age groups are being displayed.

5 Feature selection with Linear Support Vector Machines

5.1 Motivation and background

Motivation. On previously presented results many highly active diagnoses were just a general visits to a doctor or simple look-ups, which didn't tell anything specific. Also patients from all backgrounds and conditions were being analyzed all at once, which made the results mixed and hard to orient around. Therefore it would be wise to look closely at some specific disease group patients. Anchoring around a concrete diagnose it is possible to filter out the general diagnoses with feature selection method such as SVM and only keep in the most relevant diagnoses.

Feature ranking with Linear Support Vector Machines. Support vector machines are used mainly in binary classification tasks in machine learning. They try to separate two classes with a hyperplane, which gives the biggest margin between two classes[4, p 54].

They try to optimize the function

$$\min \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\epsilon(\boldsymbol{w}, b, \boldsymbol{x}_i, y_i)$$

, where ϵ is a loss function, for example a L2 loss given with the formula

$$\epsilon(\boldsymbol{w}, b, \boldsymbol{x}_i, y_i) = \max(1 - y_i(\boldsymbol{w}^T\Phi(\boldsymbol{x}_i + b), 0)^2$$

.

The function, on which SVM later makes its decision on, is expressed as follows:

$$f(\boldsymbol{x}) = sgn(\boldsymbol{w}^T\Phi(x) + b)$$

Kernel function K used during the training phase of the SVM is defined as:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T\Phi(\boldsymbol{x}_i)$$

.

For linear kernel $\Phi(\boldsymbol{x}) = \boldsymbol{x}$. This let's to rewrite the decision function:

$$f(\boldsymbol{x}) = sgn(\boldsymbol{w}^T\boldsymbol{x} + b)$$

. It can be seen that coefficient vector field w_i is only applied to input vector field x_i. This means that after training, the model's coefficient vector $\boldsymbol{w} \in R^n$ shows how relevant each feature in $\boldsymbol{x}$ is. Larger values have a bigger effect on the previously defined decision function. Therefore different features can be ranked according to the absolute value of the coefficient vector.[4, p 54-55].

5.2 Results

5.2.1 Ranking diagnoses

Since I11 struck out from the previous graphics, it was chosen as the target diagnose to be looked at more closely. Due to I11 concerning mostly the older patients, people younger than 34 weren't taken into consideration to prevent skewed results caused by unbalanced data. There was also an upper limit set for age and it was 79. As it came out in Figure 11 almost everyone have it in the very end and it is probably caused just by an old age, therefore it is pointless to include them.

A SVM was trained on patients, from whom 33% had received the diagnose at some point in their treatment and the remaining 67% had not. One training sample consisted of all the diagnoses a patient had been diagnosed with before being diagnosed with I11, if that ever happened. The downside of this approach was that since the original dataset wasn't final, then some patients, who were considered as pure from I11 by the model, might still have had received it in the future, making them the data mislabeled.

After training all encountered diagnoses were ranked according to the coefficient vector of the trained Linear SVM. Thing to note here is that coefficient vector was sorted by its absolute values, since having some diagnoses might rule out getting diagnosed with I11 and they would have appeared in the negative values of the coefficient vector. Results can be seen in figure 12.

The majority of the diagnoses revolve around 0, meaning they don't have much say in the prediction SVM has to make and having them or not and therefore doesn't make much of a difference in development of I11. However, there are approximately 200 diagnoses in the negative values and another 200 in the positive values of y-axis with larger absolute coefficient values and these are the ones, who might make the good predictors.

5.2.2 Determining the number of the relevant diagnoses

The final count of how many diagnoses were really relevant in predicting I11 was determined by the following process:

1. Set n to be the number of all encountered diagnoses
2. Take n diagnoses from the ranked coefficient vector, starting from the high end.
3. Filter out the diagnoses from the original training set, which didn't appear in the diagnoses determined in the last point.
4. Train an SVM on the updated training set and evaluate its performance.
5. If the metrics don't reflect a sudden drop, reduce n by 50 and repeat the steps 2-4, otherwise return the diagnoses found in step 2.

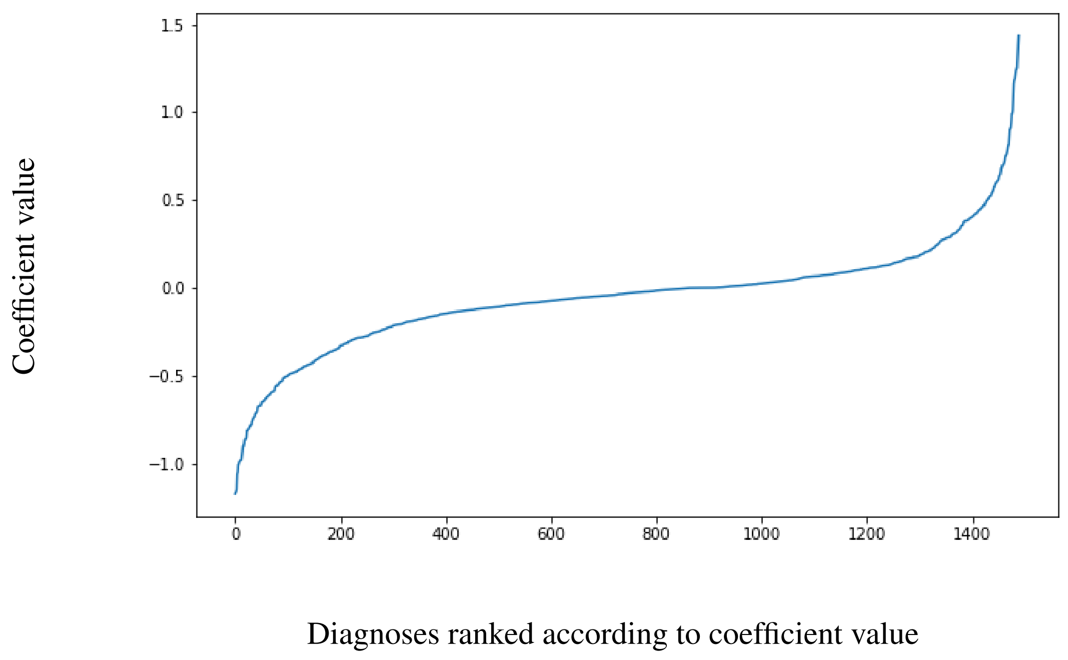


Figure 12. Sorted coefficient vector of the SVM

Sorted coefficient vector of the SVM, showing the magnitudes of the contribution, which different diagnoses involvement have in the prediction.

These n diagnoses, which were determined by the process, are essential in predicting whether or not a patient is to be diagnosed with the target disease, since they caused a drop in performance. Results of the described process are presented in figure 13.

The test set accuracy with all diagnoses included was 0.758 with the balance of 66.5% negative and 33.5% positive class patients. When the prediction is made on less than 500 top diagnoses, the model starts to only predict the opposite class (patient not sick). It is around the same number when model accuracy starts to decrease towards random.

5.2.3 Making use of the feature selection

A random sample of 40k patients who at some point had been diagnosed with I11 was selected from ages 34 to 79. From their yearly diagnose vectors only the top 500 diagnoses from the ranked coefficient vector were kept in based on the results of the last subsection. All three models and a guided HMM from section 2 and 4 were trained on filtered dataset.

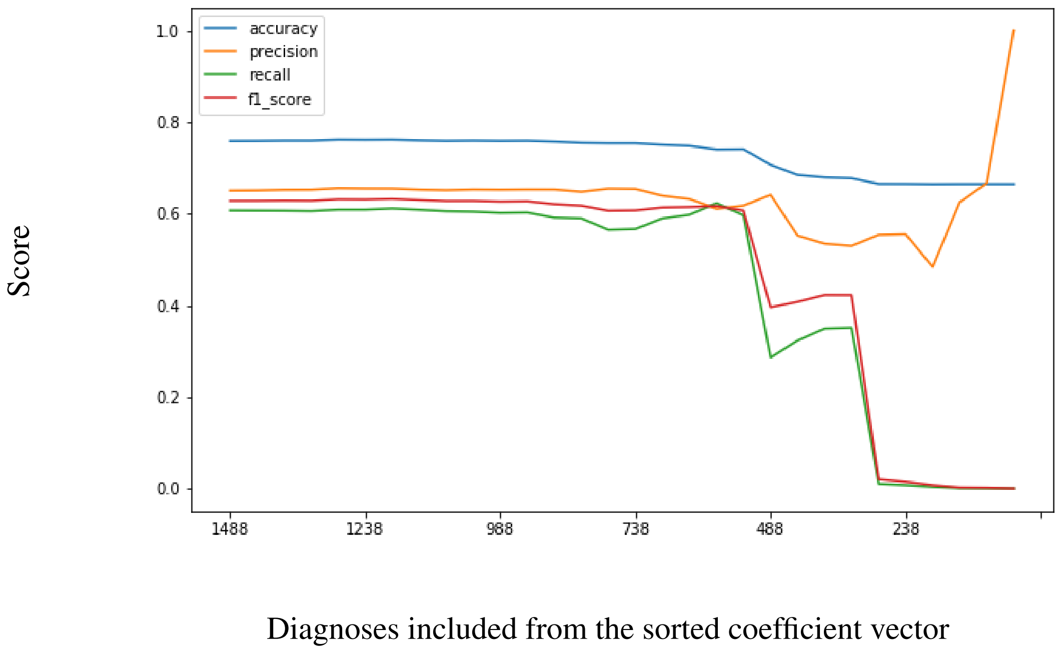


Figure 13. SVM performance evaluation with different number of diagnoses included from the sorted coefficient vector

Plot showing the performance scores of the SVM, when it was trained with different number of diagnoses included from the sorted coefficient vector, based on the absolute values. For example the decrease in the y-axis value of the red line around the value 488 in the x-axis shows that according to f1-score the SVM took a sudden drop in performance when only the patient history concerning the TOP488 diagnoses from the coefficient vector were presented to the model.

Comparison of different models after feature selection. The results for the three models can be seen in figure 14.

BMM states aren't that concrete as are K-means and BMMHMM states. There seems to be some trade-off between I10-I11. K-means has defined 1 and BMMHMM 2 states, where I10 activation is higher than target diagnose I11. In the remaining states for these models, the other have little to none I10 involvement next to I11, like state 1 for BMMHMM and state 2 for K-means. Also BMMHMM and K-means state 0s agree somewhat on Z95 and I50 importance.

BMMHMM with guiding signals after SVM feature selection. The results of the guided HMM's work on SVM reduced feature set can be seen in figure 15.

The second state of ages 65-78 seems to arise from the 4th states of ages 34-48 and

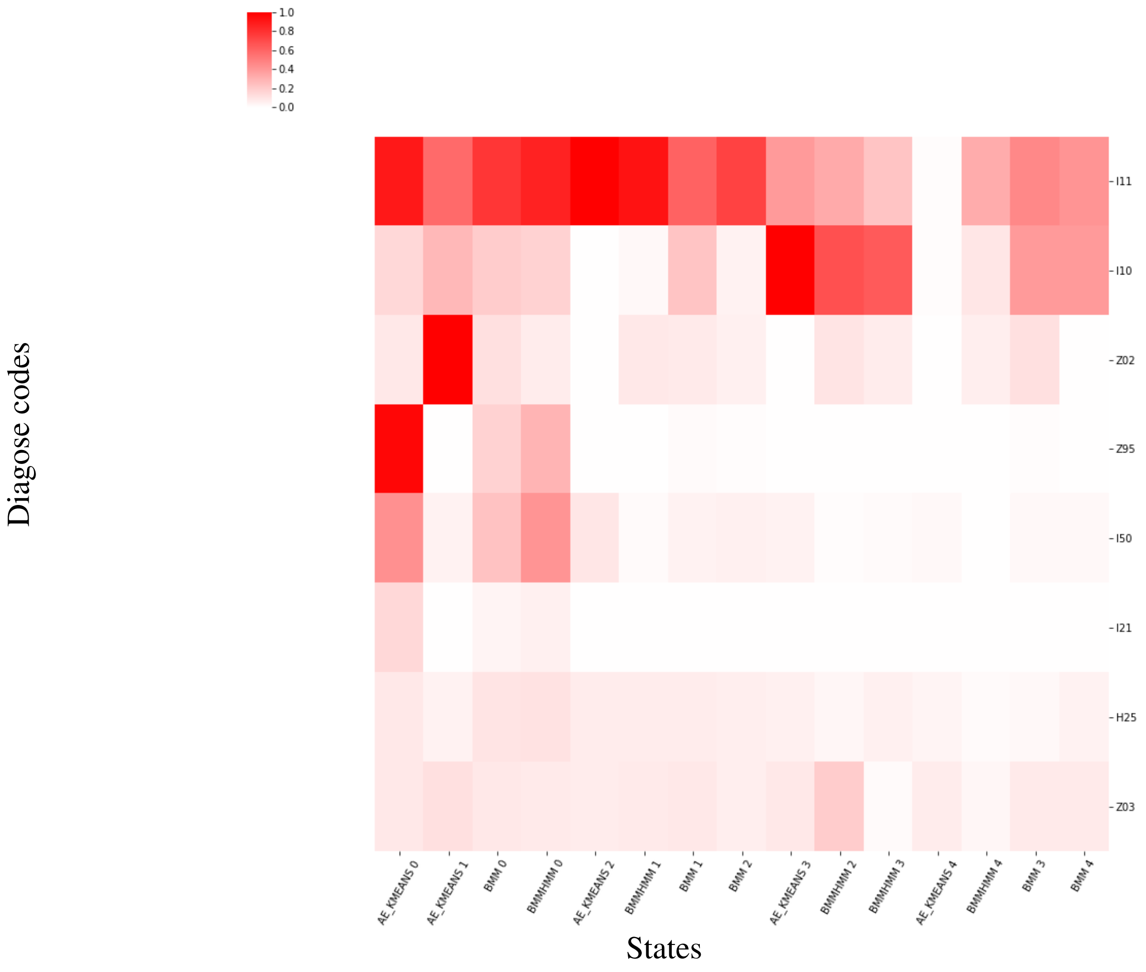


Figure 14. Comparison of the BMM, BMMHMM and Autoencoder + K-means after SVM feature selection

49-64 with the update of stronger signals from I50 and Z95 and with some newly added N18 activity. The progression from hypertensive heart disease (I11) to a heart failure (I50) is quite logical and has been reported before [6].

Like in figure Figure 14, there also is a trade-off between I10 and I11. It is not exactly clear what happened to the last state of the 34-48 group, but it probably was distributed over the states of the next group.

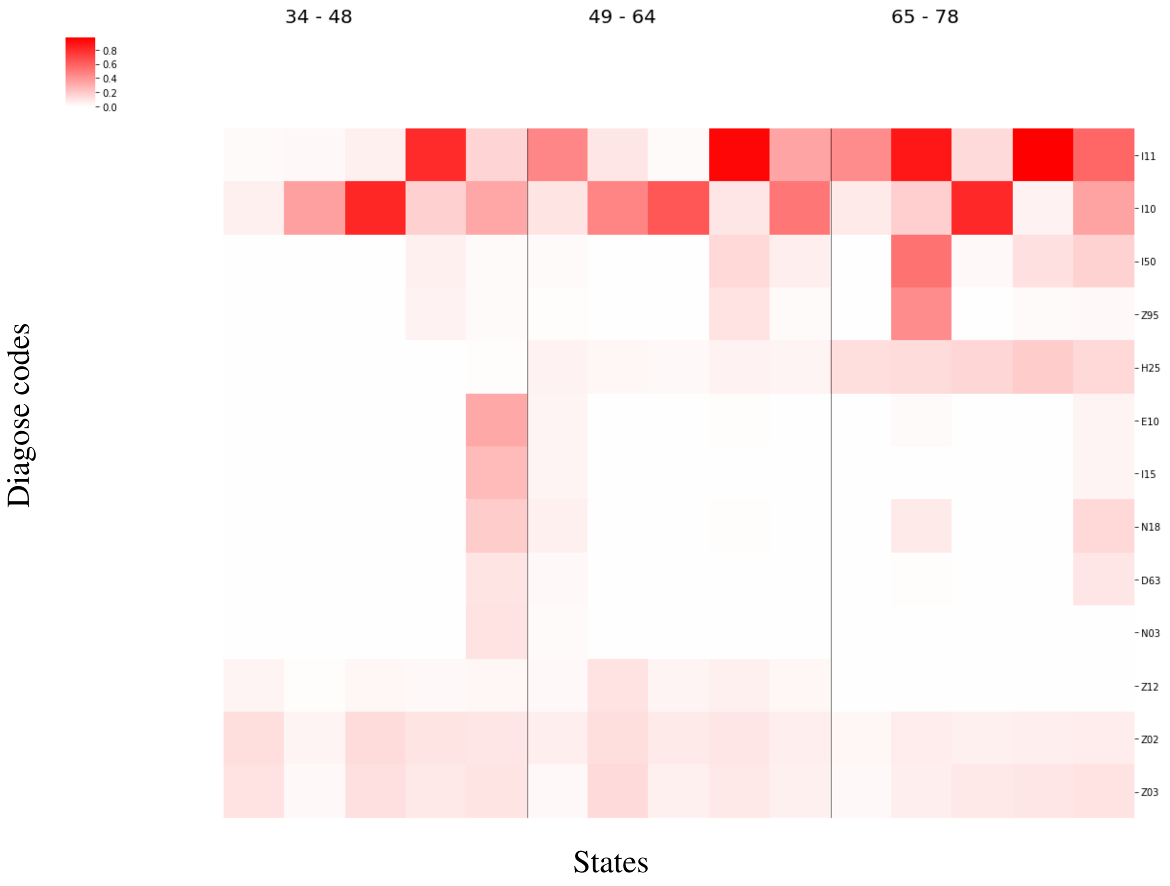


Figure 15. BMMHMM with age related guiding signals after SVM feature selection

6 Discussion and future work

The thesis focused on clustering patients into different groups by their current and previous condition. It looked for co-occuring diagnoses and tried to identify which conditions arise from what. Clustering was done by using BMM, BMMHMM and K-means with autoencoder compression. Also an HMM with guiding signals was used to put together a lifelong diagnoses path from just the short sequences of diagnoses vectors from different aged patients. In the final part, the thesis looked more concretely at a specific diagnose I11 and ranked diagnoses with SVM to identify only relevant diagnoses in development of I11.

All three models did find similar results. When they were given 5 states to work with, they agreed on multiple of them, like I11 is accompanied by E11, E78, I10, and M54. K-means seemed to form the most concrete clusters, followed closely by BMMHMM. BMMHMM, which was suppose to fix the issue of not taking a patient previous condition into account, did indeed provide the most consistent results when looking at the predictions for all diagnose vectors from the same patients. Guided HMM provided a nice overview what might happen in various stages of life and allowed to somewhat analyze the diseases development over time. Some of its findings are also confirmed before, like development of J35 from H65.

Looking specifically at patients with an I11 diagnose, the other diagnose in addition to I11, that stood out, was I10. The models reported some trade-offs between these two diagnoses by defining states, where one of them dominated over the other. Guided HMM on these patients also agreed with the trade-off and also found I11 to be somewhat responsible for I50 development.

There are also a lot of work for the future. Since HMM took a patient's previous condition into account, but also assumed the features to be independent from each other, while K-means with autoencoder did the exact opposite, it would be good to take the positives from both of these approaches and try an autoencoder with GMMHMM. Secondly, there are some developments, that could be done to an age guided HMM. Although it provided a nice overview of the diseases, the progression of the diseases was hard to track since the states across the age groups weren't well connected. Therefore some methods, which connected the states better, would benefit the interpretability of the results.

For more concrete results there are also other ways that could be tried. For example patients from different genders could be looked at independently, also their treatment cost could be taken into consideration. Furthermore, more focus should be put into feature selection to filter out diagnoses that aren't relevant to a disease or a condition but to a treatment, like most of the Z** diagnoses. Finally, it would be good to identify the right amount of states needed for the model to work with to avoid duplicate states or the situation when two actual clusters are merged to one because of the deficiency of the available clusters.

References

[1] David F Austin. “Adenoidectomy for secretory otitis media”. In: *Archives of Otolaryngology–Head & Neck Surgery* 115.8 (1989), pp. 936–939.

[2] Jeff A Bilmes et al. “A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models”. In: *International Computer Science Institute* 4.510 (1998), p. 126.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Yin-Wen Chang and Chih-Jen Lin. “Feature ranking using linear SVM”. In: *Causation and Prediction Challenge*. 2008, pp. 53–64.

[5] Francesco Cosentino et al. “2019 ESC guidelines on diabetes, pre-diabetes, and cardiovascular diseases developed in collaboration with the EASD: the task force for diabetes, pre-diabetes, and cardiovascular diseases of the European Society of cardiology (ESC) and the European association for the study of diabetes (EASD)”. In: *European heart journal* 41.2 (2020), pp. 255–323.

[6] Prakash C Deedwania. “The progression from hypertension to heart failure”. In: *American journal of hypertension* 10.S7 (1997), 280S–288S.

[7] Nophar Geifman, Raphael Cohen, and Eitan Rubin. “Redefining meaningful age groups in the context of disease”. In: *Age* 35.6 (2013), pp. 2357–2366.

[8] Markus Lippus. “Predicting Illness and Type of Treatment from Digital Health Records”. In: ().

[9] Kevin P Murphy. “Binomial and multinomial distributions”. In: *University of British Columbia, Tech. Rep* (2006).

[10] Andrew Ng et al. “Sparse autoencoder”. In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[11] Adrián Giménez Pastor. “Bernoulli HMMs for Handwritten Text Recognition”. PhD thesis. Universitat Politècnica de València, 2014.

[12] Bryan Williams et al. “2018 ESC/ESH Guidelines for the management of arterial hypertension: The Task Force for the management of arterial hypertension of the European Society of Cardiology (ESC) and the European Society of Hypertension (ESH)”. In: *European heart journal* 39.33 (2018), pp. 3021–3104.

II. Licence