Freie Universität Berlin
Department for Informatics and Mathematics
Dahlem Center for Machine Learning and Robotics

Interpretable and predictive embeddings from EHR data using canonical polyadic tensor decomposition

A thesis presented for the degree of Master of Science
by

Dorian Wachsmann

Enrollment number: 4756468
dorian.wachsmann@fu-berlin.de

First Supervisor:	Prof. Dr. Tim Landgraf
Second Supervisor:	Prof. Dr. Dr. Raúl Rojas
Advisor:	Benjamin Wild

Berlin, June 24, 2021

Abstract

A large amount of available electronic health records nowadays provide a great opportunity for modern medicine to disentangle hidden connections between diseases, as well as to make better predictions of disease development and potential medication. However, due to the high-dimensional and sparse nature of the data, it appears challenging to use the records to their full extent.
In this work, I propose a tensor decomposition method on the UK Biobank linked electronic health records. I show that the resulting embeddings are interpretable on the patient as well as the disease level. Additionally, the vectorized patient topic associations add value to common hazard models in the context of disease prediction.
I find that the accuracy of the predictions thereby is strongly connected to the quality of the decomposition. As the data increases above certain levels, the quality suffers and therefore the significance of the prediction change compared to the baseline.

Selbstständigkeitserklärung

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

I hereby declare that I have developed and written the enclosed Master thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Master thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

[illegible]

Berlin, June 24, 2021 Dorian Wachsmann

Acknowledgements

Ich möchte mich herzlich beim Berlin Institut of Health bedanken. Sie haben alle Rechenkapazitäten zur Verfügung gestellt, die ich für meine Versuche benötigt habe. Im Besonderen möchte ich Thore Buergel und Jakob Steinfeldt danken, die mir während der ganzen Zeit der Arbeit für Fragen zur Verfügung standen und die außerdem einen Großteil der Datenvorbereitung übernommen haben.
Außerdem möchte ich Benjamin Wild meinen Dank aussprechen, der immer ein offenes Ohr hatte und dessen Anregungen mir für meine Arbeit wesentlich geholfen haben.

This research has been conducted using data from UK Biobank, a major biomedical database.

Contents

List of Figures

List of Tables

Abbreviations

EHR	Electronic health records
ALS	Alternating Least squares
CPD	Canonical Polyadic decomposition
CoxPH	Cox Proportional Hazard Model
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
KRP	Khatri-Rao Product
MTTKRP	Matricized Khatri-Rao Product
DAG	Directed Acyclic Graph
SGD	Stochastic Gradient Descent
BCD	Block Coordinate Descent
ICD	International Statistical Classification of Diseases and Related Health Problems
CORE	Clinical Observations Recordings and Encoding
UMLS	Unified Medical Language System
CVD	Cardiovascular disease
MACE	Major adverse cardiovascular events

SCORE	Systematic COronary Risk Evaluation
t-SNE	t-Distributed Stochastic Neighbor Embedding

1 Introduction

With the exponential growth of health information over the last few decades, we see an increasing interest in precision medicine [1]. However, despite or because of its popularity in the scientific and political world, the term lacks a precise definition. Roughly speaking, precision medicine aims to maximize the quality of health care on the individual level by focusing on the unique, time-evolving health status of the patient. It can be seen as a process including three steps: first, pre-processing of data and variable selection to identify hidden structures. Secondly, variables from the prior step are used for diagnostic and predictive tasks, e.g. disease development. Finally, model development to predict treatment responses. [2] It is easy to see that each step comes with various computational and conceptual challenges.
Additionally, systems that aim to operate in a medical environment must come with a mechanism that allows to understand and verify the result. This potentially opposes Deep learning approaches as these models are often treated as black-box functions, even though there are attempts to establish the notion of trust in this context. [3]
In this thesis, I try to answer the question if one can find an embedding of the high-dimensional, sparse Electronic health records (EHR) data that helps understand disease risk development over time on an individual level. I present a tensor decomposition that, besides its relative ease of computation, comes with high guarantees in interpretability as well as a natural clustering. I show that the vectorized disease embedding of an individual gradually increases the prediction accuracy on a common hazard model compared to a baseline.
The thesis is organized as follows. I end the introduction part with an overview of the main contributions of this work. In chapter 2 I lay the knowledge foundation by presenting key aspects of Canonical Polyadic decomposition (CPD) and Cox

Proportional Hazard Model (CoxPH). I follow up with related work in the context of the analysis of EHR and large-scale tensor decomposition.
The Chapter 3 is threefold. In section 3.1, I introduce the UK-Biobank data [1]. I describe the tensor creation process and the embedding of health diagnosis into broader concepts. Tensor decomposition is relatively straightforward to compute if the tensor is small. In the case of large-scale and sparse data, it becomes increasingly hard to find a good approximation of the tensor. I describe memory usage and time issues in detail in the section 3.2. This leads to two different approaches I explain in section 3.3.
In the chapter 4, I compare the two models and evaluate the results. The analysis is divided into two parts. On the one hand, I cluster patients and diseases and interpret correlations between topics. On the other hand, I ask if the embedding helps to improve the quality of time-to-event predictions with common (semi)-parametric models like the CoxPH and a clinical outcome measure like Major adverse cardiovascular events (MACE).
I finish with limitations and possible improvements of the proposed model in the chapter 5.

1.1 Contribution

For an overview of the general workflow, I refer to the figure 1.1. It shows a summary of data extraction, preparation, method application, and, in the bottom part, the analysis of the results. I want to highlight three main contributions of this thesis.

1. Description and implementation of two different ways to compute CPD. Comparison with respect to the reconstruction accuracy. I find that the Alternating Least Squares algorithm performs better compared to a stochastic minimization approach.

[1] `https://www.ukbiobank.ac.uk`

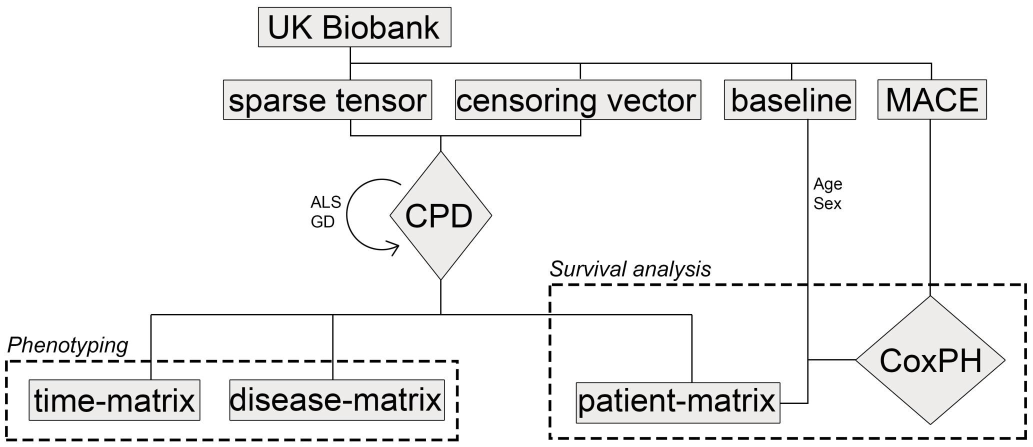

Figure 1.1: High-level perspective on the general workflow from top to bottom. Data is visualized with grey boxes, functions are diamond shapes. The application area is shown with dotted lines.

2. I find that the *factor matrices* hold valuable information for a better understanding of the high dimensional tensor. Patients can be clustered into groups and the associated diseases may offer clinical relevance.

3. The ability to predict diseases remains unclear. I find that the tensor decomposition approach might either be not powerful enough to use the data to its full extends or might not approximate the data accurately enough.

2 Related work

After shortly introducing key notation, I review relevant background knowledge to be able to follow through the thesis. I will cover CPD as well as CoxPH. CPD is a quite complex topic and I do not attempt to cover all aspects of it. For a comprehensive introduction, I guide the interested reader to [4] where the authors provide a survey with an overview of different tensor decomposition methods, their computation and possible applications.
I follow up with the state of the art of decomposition methods that have been applied to EHR data, as well as an overview of existing approaches to decompose large and sparse tensors.

2.1 Background

The notation I use is similar to what is proposed in [4] and should be in line with what is considered standard in the context of tensor decomposition. Therefore, I will only re-iterate over the most important definitions:

Operator	Operation
$\boldsymbol{\mathcal{X}_n}$	Mode n-Matricization
$\odot$	Khatri-Rao product
$\circ$	Outer product
$\times$	Hadarmard product
$\boldsymbol{X}^T$	Transpose matrix
$\boldsymbol{X}^{-1}$	Inverse matrix

Table 2.1: Comprehensive list of operations

I use low letters boldface $\boldsymbol{a}$ for a vector where the ith entry is denoted as a_i. Analogues, the entry (i, j) of a matrix with capital boldface letters $\boldsymbol{A}$ is denoted as $A_{i,j}$. I use the Euler script letter for a third order tensor $\boldsymbol{\mathcal{X}}$ with entry (i, j, k) written as $\mathcal{X}_{i,j,k}$.
To address a sub-array, e.g. column j of a matrix, $\boldsymbol{A}$ I write $a_{:,j}$.
A tensor can be seen as a stack of matrices. A single matrix is then called a *Slice* and is defined by fixing all but two indices, e.g. to address a frontal slice k of a tensor $\boldsymbol{\mathcal{X}}$ I write $\boldsymbol{X}_{:,:,k}$. For horizontal and lateral slides, this can be done analogously.
For the n-th element in a sequence, A I write $A^{(n)}$.
All relevant and not commonly used operators are summarized in table 2.1. I assume the operations are known, and I will not explain them further.

2.1.1 Canonical polyadic decomposition

CPD was independently invented by several research groups throughout the 20th century [5] [6] and found its application in a wide range of research areas nowadays, e.g. anomaly detection, clustering, and trend detection to name only a few.
Generally speaking, CPD is a higher-order generalization of Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) from matrix decomposition. Here, I only consider the case of a three-way tensor, but in general CPD is easily adaptable to higher-order tensors as well.
In CPD, a tensor $\boldsymbol{\mathcal{X}} \in \mathcal{R}^{I \times J \times K}$ is decomposed into a sum of R components. A visualization of it is shown in figure 2.1. Mathematically, it is the sum over the outer products of the rank-one components:

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r = [[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]] \tag{2.1}$$

$\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are called *factor-matrices.* They refer to the combination of the corresponding rank-one components respectively.
R is called the rank of the tensor, written $rank(\boldsymbol{\mathcal{X}})$, if it is the smallest number of rank-one components to exactly reconstruct $\boldsymbol{\mathcal{X}}$. Different to the matrix-rank,

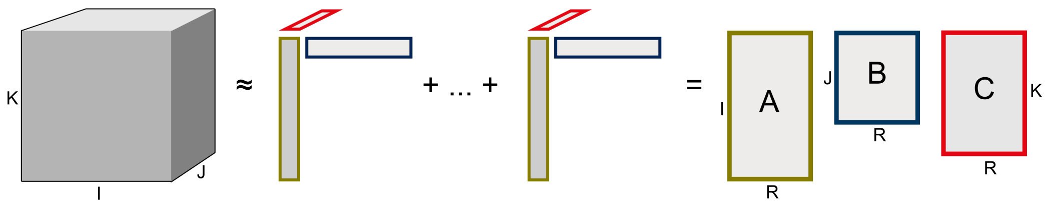

Figure 2.1: Three-dimensional tensor $\boldsymbol{\mathcal{X}}$ decomposed into a sum of R rank-one tensors and ordered into factor matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$.

no direct algorithm to determine the tensor rank exists. It is even shown that the problem is NP-hard [7]. Therefore, CPD usually operates with an approximation of R.
An interesting property of CPD is the uniqueness property. It holds under much weaker conditions than for standard matrix decomposition.

Computing the CPD is essentially a minimization problem to find the best rank R approximation of $\boldsymbol{\mathcal{X}}$:

$$\underset{A,B,C}{min} ||\boldsymbol{\mathcal{X}} - [[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]]|| \tag{2.2}$$

$||.||$ can be an arbitrary norm. The most commonly used norm is the Frobenius norm $||.||_F$. The optimization problem is well studied in literature and different approaches exist. The most popular approach is Alternating Least squares (ALS). The idea is to divide the non-convex minimization problem into a set of convex problems and solve them in an iterative fashion. In the three-dimensional case, each iteration consists of three steps. Each step optimizes one factor-matrix, while the other two remain fixed. Assuming $\boldsymbol{B}$ and $\boldsymbol{C}$ are fixed, the minimization problem reduces to a linear least-squares problem for $\boldsymbol{A}$ and can be rewritten as:

$$\boldsymbol{A} = \boldsymbol{X}_{(1)}[(\boldsymbol{C} \odot \boldsymbol{B})^T]^{-1}$$

Thanks to the properties of the Kronecker product [8] we can rewrite the update step:

$$\boldsymbol{A} = \boldsymbol{X}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})(\boldsymbol{C}^T\boldsymbol{C} \times \boldsymbol{B}^T\boldsymbol{B})^{-1} \tag{2.3}$$

$\boldsymbol{B}$ and $\boldsymbol{C}$ are calculated accordingly.

Intuitively, one can see that a naive implementation of the update rule 2.3 will run into problems, as soon as the data becomes significantly large because of a computational bottleneck in calculating the dense matrix products. Furthermore, it is known that ALS is not guaranteed to converge even to a stationary point and can be very slow. Nevertheless, it is still considered the go-to approach in the context of CPD, as it has been shown that other optimization techniques do not produce better results [9].

One interesting extension to CPD is to apply a non-negativity constraint to the factor matrices when dealing with non-negative data, as it is the case here. Nonnegativity provides an inherent clustering property that results in more interpretable *factor matrices.*
Different solutions have been developed to compute non-negative CPD. In the simplest case, one could apply non-negativity constraints on the *factor matrices* by mapping the negative values onto the positive space. However, this might significantly affect the reconstruction error and hinders the algorithm to converge properly. Welling et al. [10] propose a multiplicative update rule. To update *factor matrix* $\boldsymbol{A}$, the update step looks as follows:

$$\boldsymbol{A} = \boldsymbol{A} \times \frac{\boldsymbol{X}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})}{\boldsymbol{A}(\boldsymbol{C}^T\boldsymbol{C} \times \boldsymbol{B}^T\boldsymbol{B})^{-1}} \tag{2.4}$$

Again, the denominator can be calculated quite efficiently because $(\boldsymbol{C}^T\boldsymbol{C} \times \boldsymbol{B}^T\boldsymbol{B})^{-1}$ reduces to a $R \times R$ matrix. There is no guarantee that a model converges. Instead, the only guarantee giving is that the error does not increase. In section 4 I compare the standard ALS with its non-negative variant.

2.1.2 Cox Proportional Hazard model

The Cox Proportional Hazard Model (CoxPH) was first introduced by Cox in 1972 [11] and is a regression model often used in medical research. It is still today one of the most used methods when dealing with survival analysis data. Its core idea is to regress multiple covariates against another variable, usually a specific event like death or infection, and to inspect how the covariates influence the likelihood of the event happening at one moment in time. As it is a hazard function, it returns the probability of an event happening at a time t under the assumption that the event has not happened before. CoxPH consists of two terms, a so-called *baseline hazard* that is only dependent on time and a partial hazard, a linear function of all covariates that is time-invariant. Thus, for individuals with n covariates x, the hazard function is written as:

$$h(t|x) = h_0(t) \exp\left(\sum_{i=1}^{n} b_i \times x_i\right) \tag{2.5}$$

The coefficient vector $\boldsymbol{b}$ measures the impact of the covariates $\boldsymbol{x}$. The quantity $\exp(b_i)$ is called *hazard ratio*. A hazard ratio above one (or equivalently b_i above zero) indicates a positive association of the corresponding covariate with the event probability and vice versa.
To evaluate the prediction performance of CoxPH, one possible measurement is the *concordance index*. Generally speaking, it is a generalization of the area under the ROC Curve (AUC) and measures how discriminant the model is. Discrimination is the ability to reliably predict, between two individuals, who has a shorter survival time. In other words, it is the model's ability to rank survival times based on the individual risk scores.[1] Mathematically it is expressed as:

$$c_{index} = \frac{\sum_{i,j} 1_{T_j < T_i} \cdot 1_{\eta_j > \eta_i} \cdot \delta_j}{\sum_{i,j} 1_{T_j < T_i} \cdot \delta_j} \tag{2.6}$$

with:

- η_i risk score of patient i
- $1_{T_j<T_i} = 1$ if $T_j < T_i$, else 0
- $1_{\eta_j>\eta_i} = 1$ if $\eta_j > \eta_i$, else 0

It follows that, as for AUC, the $c\_index = 0.5$ is the expected result for random predictions, $c\_index = 1$ means perfect concordance and $c\_index = 0$ means perfect discordance.

[1]https://square.github.io/pysurvival/metrics/c_index.html

2.2 State of the Art

The sparse tensor factorization model proposed by J. Henderson et al., 2017 [12], operates on Poisson-distributed data like count-Electronic health records (EHR) data with axis *patients* $\times$ *diagnosis* $\times$ *medications*. Their main goal is to derive distinct and diverse phenotypes through the natural clustering of non-negative Canonical Polyadic decomposition (CPD). They optimize an objective function f with hand-crafted regularization terms in order to promote sparsity and orthogonality in the factor matrices. While they focus on finding clinically relevant phenotypes, they ignore the temporal aspect of health records or, in different words, *when* a disease occurs.
This is a problem field tackled by Zhao, Juan, et al. in 2019 [13] with their proposition of a tensor factorization to detect time-evolving phenotypic topics. In contrast to the first paper, they narrow down the scope to Cardiovascular disease (CVD) patients and aim to find sub-phenotypes of the disease and to use the patient matrix to apply survival analysis. They consider around 12,000 CVD patients and collect 1068 distinct PheCodes out of their EHR data ten years prior to the baseline date. They apply non-negative CPD and find statistically significant correlations between single topics and the 10-year CVD risk calculated with conventional methods like atherosclerotic cardiovascular disease risk score (ASCVD).
On a different note, Kim, Yejin, et al., 2017 [14], proposed a novel supervised non-negative CPD method with a similar goal of distinct and discriminative phenotypes. With the addition of a supervised term, they encourage phenotypes that are more distinct with respect to the mortality of the patients.
The problem of decomposing large tensors is not unknown, and models have been proposed for different use cases. Tamara Kolda and David Hong [15] developed a stochastic framework for a generalized version of CPD that allows for different loss functions. In each iteration, they sample elements from the tensor, calculate the element-wise gradient and construct a sparse tensor Y of the size of the tensor. They claim that their method is efficient, because they explicitly construct a sparse Matricized Khatri-Rao Product (MTTKRP) tensor kernel and use it to form the gradient.

A randomized block sampling method is presented by Vervliet and De Lathauwer [16]. Instead of sampling elements, they sample indices from each mode of the tensor. They construct a sub-tensor and optimize it using alternating least squares as well as the Gauss-Newton method. Afterward, only the corresponding variables are updated.
I studied their approaches and used ideas from both of them when I implemented my solution. I will refer to their paper wherever appropriate. To my best knowledge, at the point of writing, no work exists where CPD is applied to large scale, sparse EHR-data, in particular the UK-Biobank data.

3 Methods

In the first part of this section, I introduce the UK Biobank database. I describe the pre-processing, tensor creation process and motivate the reason behind the need to censor the data.
Afterwards, I discuss issues with memory usage as the tensor exceeds critical sizes and it becomes complicated to perform computational steps of CPD.
Finally, I present two decomposition models that take into account discussed issues.

3.1 Data

UK Biobank[1] is a long-term and large-scale biomedical database that started in 2006 and is still ongoing. The resource contains a wide amount of data from genetic, lifestyle, and general health information with half a million UK participants. Between 2006 and 2010 people between the ages of 49-69 years underwent a detailed checkup that contained questions and measurements to provide a baseline for further analysis.
A great advantage, compared to other databases, is linked health-related records. These include data from national death, cancer registries, and hospital inpatient records. The records are coded in International Statistical Classification of Diseases and Related Health Problems (ICD) 9 and 10. Relevant linkage and further details are provided in the table 3.1. Most recently, linkage to COVID-19 related records was added as well. I refer to their data showcase for more information [2].

[1] `https://www.ukbiobank.ac.uk`
[2] `https://biobank.ndph.ox.ac.uk/showcase/label.cgi?id=100091`

In this thesis, I use data from health-related records. Additionally, I use disease outcome measures like Major adverse cardiovascular events (MACE) and Systematic COronary Risk Evaluation (SCORE) to perform survival analysis on the events.

3.1.1 Preprocessing health records

As most of the pre-processing has been done by the Berlin Institute of Health, I will use *we* instead of *I*.
Health records feature the whole bandwidth of available ICD codes. In order to construct a tensor that can be used for further computational tasks, there is a need to group synonymous terms and related diseases into broader concepts.
The Unified Medical Language System (UMLS) can be seen as a thesaurus that brings together many health and biomedical vocabularies and standards. The Clinical Observations Recordings and Encoding (CORE) Problem List is a UMLS project with the goal to define a subset of UMLS that is best for encoding clinical information at a summary level.[3] The CORE list contains 6776 UMLS concepts with corresponding SNOMED CT codes. [17]. SNOMED CT is the most used clinical healthcare terminology. It is systematically organized[4]. Each concept has an identification code, a fully specified name, and a formal definition. Furthermore, all concepts are connected through relationships ordered in a Directed Acyclic Graph (DAG), creating a semantic net.
Starting from the SNOMED CT concepts in CORE list, we used the relationships defined in the DAG to find related concepts. We mapped the SNOMED CT codes to ICD-10 to link them with the UK Biobank health records. To give an idea of the final mapping, the table 3.2 lists the corresponding ICD codes for the disorder of the cardiovascular system. We did not use all 6776 concepts from CORE, but limited ourselves to 3521 because some concepts appeared to be too sparse to provide clinical meaning.
The figure 3.1 illustrates a single patient slice of the tensor. The diagnosis embedding

[3]https://www.nlm.nih.gov/research/umls/Snomed/core_subset.html
[4]https://www.snomed.org/snomed-ct/five-step-briefing

Data	Details
Death	ICD-10 coded death registry. Information on date, age, cause of death
Hospital admission	ICD-9 and ICD-10. Information on operations, diagnosis, care
Primary Care	Primary care records, information on diagnosis, medication, referrals

Table 3.1: Biobank health-outcome linkage

axis refers to the CORE list as described above. We set the time axis to 101 discrete time steps. Independent of the patient's birth time, the beginning of the observation starts at time 0. In other words, the time axis is independent of the actual time. This allows comparison of participants throughout time and makes them independent of their age at recruitment.

3.1.2 Censoring

Censoring is necessary when information on a time to outcome event is not available for all study participants. A participant is said to be censored when information on time to event is not available. [18] Through the alignment in time, we make sure the observation period starts at point zero for every patient. The observation time varies from patient to patient as they are of different ages and might drop out because of death. This motivates the need to accurately censor the data. It is necessary to distinguish between two cases. Firstly, the observation period might have ended. In this case, I speak of missing data because future diseases are not captured in the tensor.

Secondly, the patient died. Consequently, the patient can not develop any more diseases. It is some sort of philosophical question whether or not the data should be censored here. From a practical machine learning standpoint, it might be helpful to censor the data because a good model might correctly predict diseases in the future that could have happened if the patient would not have died earlier. To predict that, the model would need to precisely predict any cause of death. As neither data nor model is sophisticated enough for such predictions, I took a practical position and

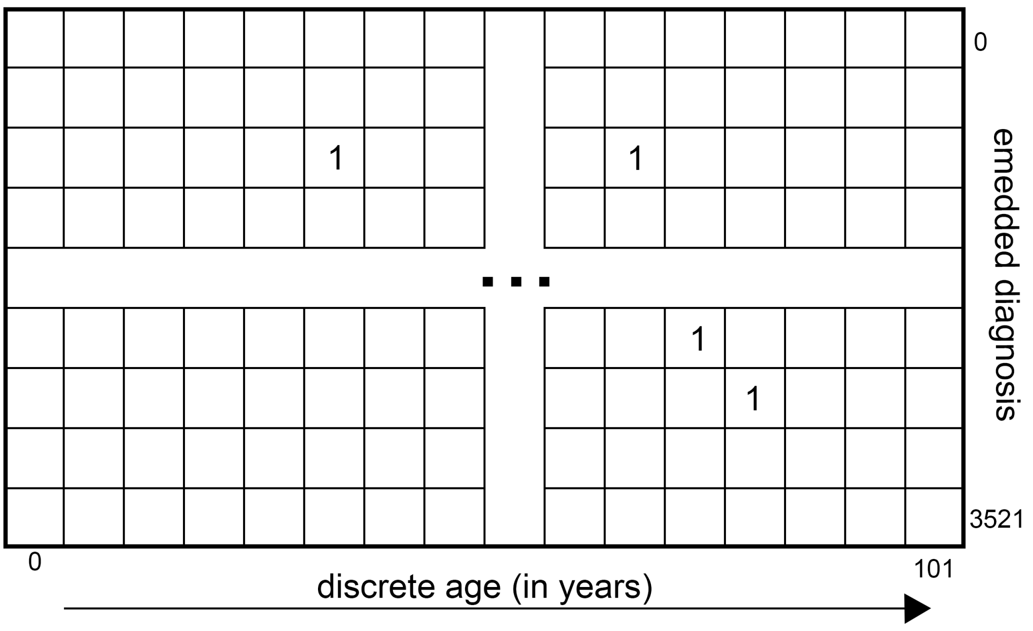

Figure 3.1: One participant's diagnosis matrix. The diagnosis is based on the CORE problem subset of the UMLS. Every diagnosis embedding encodes numerous ICD codes.

Disorder of cardiovascular system (1090)
A01, A18, A32, A36, A38, A39, A50, A52, A54
B05, B06, B26, B27, B33, B37, B57, B58
C38, C45
D15, D18, D69, D86
E08, E09, E10, E11, E13, E75, E88
G40, G43, G44, G45, G46, G83, G90
H21, H31, H34, H35, H44, H93
I01, I02, I05, I06, I07, I08, I09, I10, I11, I13, I15, I16, I20, I21, I22, I23, I24, I25, I26, I27, I28, I30, I31, I32, I33, I34, I35, I36, I37, I38, I39, I40, I41, I42, I43, I44, I45, I46, I47, I48, I49, I50, I51, I52, I60, I63, I65, I66, I67, I68, I70, I71, I72, I73, I74, I75, I76, I77, I78, I79, I80, I81, I82, I83, I85, I86, I87, I95, I97, I99
J10, J11, J84
K22, K25, K28, K29, K31, K55, K63, K64, K75, K76
L41, L81, L95
M05, M30, M31, M32, M35, M47
N02, N03, N05, N07, N26, N48, N50, N52
O03, O04, O07, O08, O10, O11, O13, O14, O15, O16, O22, O26, O29, O43, O69, O74, O86, O87, O88, O89, O90, O99
P12, P25, P28, P29, P50, P51, P91
Q20, Q21, Q22, Q23, Q24, Q25, Q26, Q27, Q28, Q87, Q93
R03, R23, R57
S06, S09, S14, S15, S24, S25, S26, S35, S45, S55, S65, S75, S85, S95
T67, T75, T79, T80, T81, T82, T86

Table 3.2: Through the SNOMED CT knowledge graph we collected numerous ICD codes related to the concepts defined in CORE. As an example, the table shows the ICD codes related to Disorder of cardiovascular system

decided to censor the data also in a case of death. I call this process death censoring.

Additional censoring is necessary for time-to-event predictions dependent on the period of measurement. As an example, MACE is a composite endpoint used in cardiovascular research. For a given observation time, each patient has an associated observation outcome. Time is defined as the years till the event happened or till the observation ended without the event happening. In the latter case, we need to censor the tensor accordingly to the observation time.

3.2 Memory usage

Memory usage might cause issues in the optimization algorithm as the tensor becomes larger. To better understand why and where a bottleneck occurs, I inspect the size of $\boldsymbol{\mathcal{X}}^{I \times J \times K}$ in more detail. It is of shape $466428 \times 101 \times 3521$ with 90236502 non-zero elements. This is equal to a density of 0.0005. The memory usage would extend $1.TiB$ if $\boldsymbol{\mathcal{X}}$ is dense, but can be reduced to $2.4GiB$ if it is saved in the sparse COO-format. Therefore, I only use sparse data types, even though it limits the use of common libraries. For example, some basic operations like index assignments do not work on COO.
The goal is to compute factor matrices $\boldsymbol{A}^{I \times R}$,$\boldsymbol{B}^{J \times R}$ and $\boldsymbol{C}^{K \times R}$, where R is a predefined rank of the desired decomposition. Even though I operate on a sparse tensor $\boldsymbol{\mathcal{X}}$, the resulting factor matrices are typical of dense structure. The reconstruction of $\boldsymbol{\mathcal{X}}$ becomes challenging, as the product from the dense factor matrices usually appears to be dense as well. In this case, $\hat{\boldsymbol{\mathcal{X}}}$ would exceed a size of $1.TiB$.
I want to give one important example of memory issues while running the minimization operation. As explained, during training each *factor matrix* is updated while the other two remain fixed. Each update step, given in equation 2.3, involves the calculation of the Khatri-Rao Product (KRP) of the two fixed matrices and afterward the multiplication with the matrix $\boldsymbol{X}_{(N)}$, the matriciation of the tensor $\boldsymbol{\mathcal{X}}$. An example with real data shows how devastating the memory explosion is. The update step for the matrix $\boldsymbol{B}$ is given as follows:

$$\boldsymbol{B} = \boldsymbol{X}_{(2)}(\boldsymbol{A} \odot \boldsymbol{C})(\boldsymbol{A}^T\boldsymbol{A} \times \boldsymbol{C}^T\boldsymbol{C})^{-1} \tag{3.1}$$

The second part of the right side of the equation is the pseudo-inverse of a $R \times R$ matrix. As R is typically much smaller than the dimensions of the tensor, it can be calculated efficiently. The first part is called the Matricized Khatri-Rao Product (MTTKRP). The size of matrix $\boldsymbol{X}_{(2)}$ is $[101 \times (466428 \times 3521)]$ and the KRP $(\boldsymbol{A} \odot \boldsymbol{C})$ is of size $[(466428 \times 3521) \times R]$. This results in a multiplication of two billion size matrices. They are significantly larger than the original tensor because of their dense structure. Even on high-end machines this it is not feasible to calculate such a product. The problem is not unknown and is referred to as the

Intermediate data explosion.
Three strategies are possible to deal with this problem. Firstly, do not calculate the MTTKRP at all. This could be done by changing the optimization strategy. As discussed already, this might lead to a worse reconstruction.
Secondly, calculate the KRP in a more efficient way regarding memory. As a trade of it might lead to increased time consumption.
Lastly, sub-sample the problem, solve it in lower-dimensional space, and reconstruct it. This might affect the quality of decomposition or even not converge at all.
I implemented solutions for all three approaches, but only succeeded with the first two approaches. I refer to the GitHub repository to review the code.[5]

3.3 Models

I describe two different algorithms to calculate the Canonical Polyadic decomposition (CPD). Both are implemented in Python. In addition to standard libraries, I use TensorLy [19]. TensorLy is a high-level API for tensor operations. It offers a range of different tensor decompositions as well as useful tensor-related calculations like an efficient implementation of the MTTKRP. It also offers a flexible backend system that makes it usable together with PyTorch or NumPy.
TensorLy offers some functionality for sparse data types. At the point of writing, they were only supported for the NumPy backend. With NumPy backend, TensorLy has a sparse implementation of the parafac decomposition utilizing Alternating Least squares (ALS) algorithm[6]. Unfortunately, the implementation is not suitable for very large tensors with the need for masking data. TensorLy handles masking in such a way that in each iteration, the tensor is multiplied with the mask to zero out all missing data points. Because a zero does not indicate missing data but instead is information, they add the reconstructed data from the *factor matrices*. Hence, the masked tensor is given as:

$$\boldsymbol{\mathcal{X}}_{masked} = \boldsymbol{\mathcal{M}} \times \boldsymbol{\mathcal{X}} + (1 - \boldsymbol{\mathcal{M}}) \times \boldsymbol{\mathcal{X}}_{rec} \tag{3.2}$$

[5]https://github.com/thbuerg/medTNMF/tree/master/medtnmf/decomposition
[6]http://tensorly.org/stable/modules/generated/tensorly.decomposition.CP.html#tensorly.decomposition.CP

where $\boldsymbol{\mathcal{M}}$ is the mask tensor and $\boldsymbol{\mathcal{X}}_{rec} = [[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]]$ is the reconstructed tensor. I discussed in the section 3.1 why the tensor reconstruction would lead to a dense object that does not fit into memory. Therefore, it is not possible to use the provided implementation.

3.3.1 Sophisticated Alternating Least squares

This method is an implementation of the ALS algorithm and uses a memory-efficient way to calculate the MTTKRP. For the most part, it is based on the implementation provided by TensorLy. For completion, I quickly want to present their implemented solution before I speak about what I added and changed to make it usable for my case.
Their core implementation of the MTTKRP uses the fact that the multiplication of the matricized tensor with the KRP is equivalent to the multiplication of the Kronecker product of each factor for each rank, as introduced in [20]. To understand this better, I provide an example. Let us consider the first step in one ALS iteration, in which we solve for factor matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \boldsymbol{X}_{(1)}(\boldsymbol{B} \odot \boldsymbol{C}) \tag{3.3}$$

We rewrite the solution in such a way that every column of $\boldsymbol{A}$ is calculated independently. The cost is equal to the computation of the product of the sparse tensor with $2 \times R$ vectors.

$$\boldsymbol{a_r} = \boldsymbol{\mathcal{X}} \bar{\times}_1 \boldsymbol{b}_r \bar{\times}_2 \boldsymbol{c}_r \text{ for } r = 1, 2, 3, ..., R \tag{3.4}$$

With $\boldsymbol{a}_r$,$\boldsymbol{b}_r$ and $\boldsymbol{c}_r$ are the rth-column of the *factor matrices* respectively and the multiplication is alongside the respective axis of the tensor.
For the masking process, I need to vary from the TensorLy implementation. I came up with two ideas on how two implement a mask without reconstructing the whole tensor. The easiest and efficient method is mean masking. The idea is to replace the to-be-masked entries with the mean value over the tensor. It is efficient because it needs to be done only one time at the beginning of training. On the downside, it

has a non-negligible effect on the training as values have been changed to some other value and therefore potentially affect the quality of reconstruction.
A more accurate solution requires calculating the expected values of the respective entries of the tensor in each iteration of training. During my experiments, I did not succeed to find an implementation that archives that in an acceptable time. Therefore, I only used mean masking in the experiments.

Loss and prediction error

It is desirable to inspect the reconstruction error during training to verify the training process. Additionally, it is helpful to compare different training sets with respect to the error. The main question is how to calculate the error without having to reconstruct the tensor. Fortunately, it is possible to re-use the intermediate MTTKRP calculation. Let $\boldsymbol{\mathcal{X}}_{rec}$ be the reconstructed tensor. Re-formulating the loss function as follows:

$$||(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{X}}_{rec})||_F = ||\boldsymbol{\mathcal{X}}||^2 + ||\boldsymbol{\mathcal{X}}_{rec}||^2 - 2 < \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}}_{\boldsymbol{rec}} >$$

The right side of the equation consists of three terms. The first one is the norm of the tensor. It does not change and only needs to be computed one time before training. The second term is the norm of the reconstructed tensor. Without reconstructing, the Frobenius norm is calculated directly from the factor matrices:

$$||\boldsymbol{\mathcal{X}}_{rec}||_F = \sqrt{\sum \boldsymbol{A}^T\boldsymbol{A} \times \boldsymbol{B}^T\boldsymbol{B} \times \boldsymbol{C}^T\boldsymbol{C}}$$

The third term is the inner product of the tensors. It is efficiently calculated with the use of the MTTKRP from the third step of the optimization iteration.
I introduce a *prediction loss* or validation loss. The idea is to mask a set of entries in the tensor to see how well the reconstructed tensor *predicts* these values never seen in the reconstruction process. Comparable to a validation set in the training of neural networks, it acts as a verification that the model indeed learns to correctly predict a future event. I randomly choose a fixed number of indexes on the patient axis before training. For each sampled index, I use the masking method described

prior to mask all values from *age at recruitment* on. I calculate the prediction loss of the sub-sampled tensor with the validation indexes only. It can be done as long as the number of indexes does not exceed a critical level. I found that 1000 appears to be an upper bound before memory or time usage becomes an issue.

3.3.2 Stochastic minimization using Adam

As exhaustively discussed in the earlier section, ALS is not perfect and faces many problems in real-life applications. It motivates the need to come up with a different method.

Randomization is a well-known tool to speed up algorithms that deal with large data. One arguably most successful method in recent machine learning history is Stochastic Gradient Descent (SGD). In short, Gradient descent minimizes an objective function $f(\theta)$ by updating the corresponding parameters θ of the model in the opposite direction of the gradient of the function. In a high dimensional optimization problem, SGD is used to estimate the gradient from a single random sample point. This is extended by using mini-batches instead of single data points to avoid a high variance in the loss.

In CPD, θ is equal to the *factor matrices*. The objective function f is given as

$$\underset{\theta}{min} f = ||\boldsymbol{\mathcal{X}} - rec(\theta)||_F \tag{3.5}$$

with $rec(\theta) = \sum_{r=1}^{R} a_r \circ b_r \circ c_r$. The function f can easily be expanded with arbitrary regularization terms to promote sparsity and or orthogonality between the factors. To promote non-negativity, it is possible to project the gradient onto the positive space. For simplification, I did not add any terms here. Also, my experiments have shown that the results appear to not change significantly using regularization terms. I refer to the paper [12] where the authors derive a minimization function with angular regularization and a $l2$-regularization term to promote sparsity.

The main idea for the stochastic minimization of CPD is to update only a subset of θ using a (pseudo)-random sample of the tensor. It is possible thanks to the local

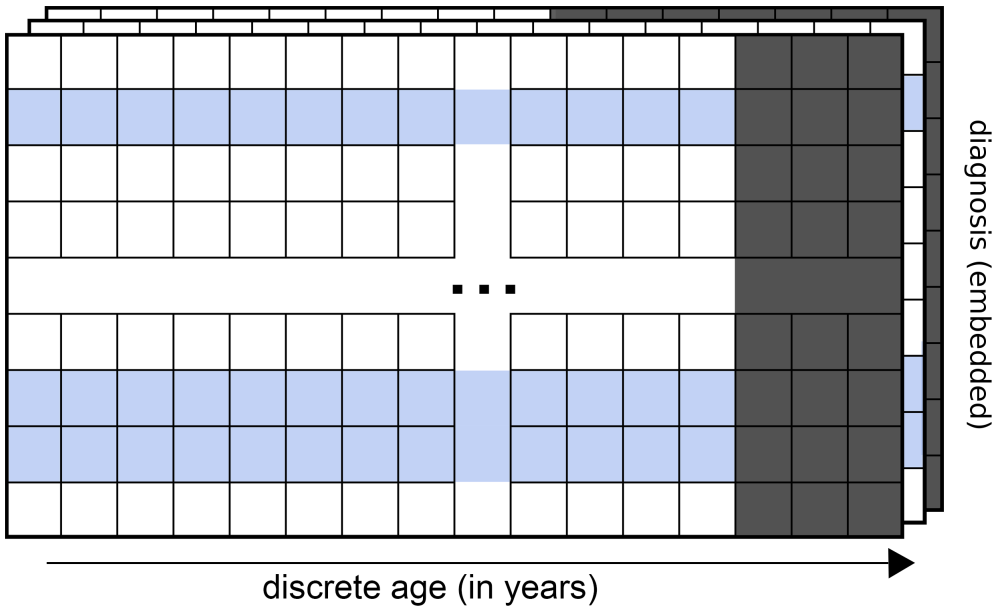

Figure 3.2: This is an example of how I sample a sub-tensor from the original one. In a pseudo-random fashion, I choose a number of participant matrices, here three. For each one, I choose the same amount of diagnosis, indicated with the blue colour. Removing the white space, I am left with a much smaller tensor. I do not sample on the time axis. The grey area indicates death censoring. In this example, the second participant has data for all time steps and no death-censoring is applied.

property of the CPD. Each entry in a third-order tensor with rank R only affects $3 \times R$ variables in the factor-matrices. The property derives from the reconstruction equation 2.1. Therefore, it is possible to sample a block from the tensor and optimize only the corresponding variables.

The method is comparable to a Block Coordinate Descent (BCD) optimization approach. In BCD, the minimization of a multivariate function is simplified by solving it only for one (or a set) of variables at the time, while the other remaining fixed. [21]

Sampling

To get an idea of how the sampling process looks like, I provide the figure 3.2 that illustrates a simplified version of it. I experimented with different sampling strategies and I want to describe the one that worked best for me. I found that it needs a different sampling strategy on the patient axis compared to the diagnosis axis.
I initialize the factor matrices with values close to zero. Slices of the tensor with mostly zero entries are therefore relatively good approximated *by design* and do not need as many optimization iterations as the entries that are further away from the initialized values.
Sampling on the patient axis is done in a pseudo-random way. In the first round, I randomly sample a fixed amount and keep track of the already sampled indexes. In the second step, I draw only from those not already drawn. After all the indexes have been sampled, I call it an *epoch*.
I decided to do some sort of *biased sampling* on the diagnosis axis to acknowledge the fact that some diseases appear more often and thus are more *important*. I assign a mass to each index that captures its importance through the marginal sum over the axis:

$$m_c(k) = \sum_{i=1}^{I} \sum_{j=1}^{J} \boldsymbol{\mathcal{X}}(i,j,k) \text{ for } k = 1, ..., K \tag{3.6}$$

The higher the value for entry k of the axis, the higher the likelihood to be sampled. Having defined m_c the probability for k is then given by:

$$p_C(k) = \frac{m_c(k)}{\sum\limits_{k=1}^{K} m_c(k)} \tag{3.7}$$

For the optimization process, I decided to use the AdamW optimizer. Adam is an extension to stochastic gradient descent and was introduced by Diederik Kingma and Jimmy Ba in 2014 [22]. The name is derived from *adaptive moment estimation*. The key difference to classical stochastic gradient descent is its adaptive learning rate using the first and second moments of the gradient. The authors later proposed

AdamW as an improved version of Adam. They show experimentally that AdamW yields better training loss and that models generalize better [23].

In comparison to ALS, censoring is not a problem. The sub-sampled tensor is significantly smaller and a reconstruction is possible without memory issues. The loss is calculated using only not-censored entries. Therefore, the censored entries in the reconstructed tensor are real predictions, as they did not influence the error at all.

4 Results

In this chapter, I provide the results of my experiments with the two models I presented in 3.3. To guide my experiments, I came up with seven questions I want to answer. They are divided into three blocks. In the first block, I compare the two models with respect to their losses on different hyperparameters:

1. How does the size of the tensor influence the reconstruction and prediction error if the rank is fixed?
2. How does the rank influence the reconstruction error and prediction error if the size of the tensor is fixed?
3. How well does a model with non-negativity constraint performs against a model without any constraints?

For further analysis, I only focus on the better performing model. Also, I want to find out if I can use non-negativity constraints, as they help with the interpretability of the data.
I analyse the *factor matrices* in detail. I ask the following questions:

4. How *similar* are patients that are assigned to a topic through their maximum topic value? Is there a meaningful clustering or correlation in the context of death events?
5. How *related or similar* are the diagnosis embeddings with the highest values for each topic?

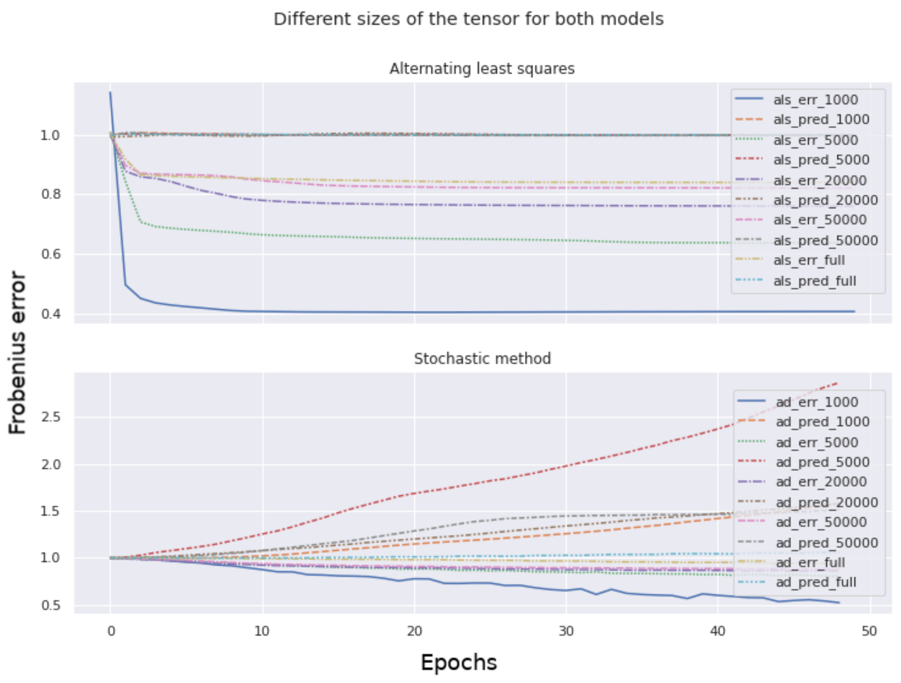

Figure 4.1: Train and prediction error from the training of the two models. I compare different sizes of the tensor with a fixed rank of 20. The training error for ALS converges to 0.4 for the smallest tensor size of 1000 and 0.8 for the full-size tensor. The stochastic method is a little worse with 0.5 and 0.9 respectively. The larger the tensor, the worse the reconstruction error.
The prediction error stagnates around 1.0 for ALS. It increases using the stochastic method.

6. How does the relationship of the time-matrix and diagnosis-matrix help to define distinct topics?

In the last part of my experiments, I go one step further and inspect the possibility of using the patient-matrix on a survival analysis task. I ask myself:

7. How useful are the individual patient vectors when applied to a survival analysis task compared to a simple baseline?

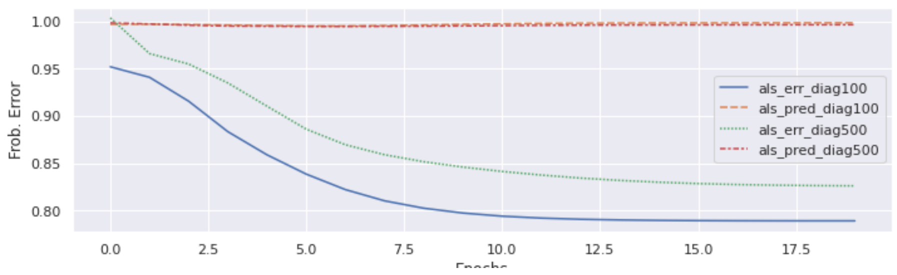

Figure 4.2: Train and prediction error from the training of non-negative variant of ALS. I reduced the amount of diagnosis to the 100 and 500 most common, while I kept the full size on the other axis. Fewer diagnosis lead to a lower error of approximately 0.8. the prediction error does not change for any of the two sizes.

4.1 Method comparison

To address the first question, I varied the size of the tensor on the patient-axis as well as the diagnosis-axis. First, I compared five different tensor sizes of 1000, 5000, 20000, 50000 and the full size of 466428 on the patient axis while I kept the other axis the same. Secondly, I compared tensor sizes of 100, 500 and the full size of 3521 on the diagnosis axis while I kept the other axis the same. For the disease subset, I decided to choose the most prevalent diseases as they contain more information. For all tests, I set the rank to 20. I ran 50 iterations of ALS and 20000 iterations of the stochastic method with a sample size of 250. The Frobenius error is normalized with the respective tensor norm. The error of the stochastic method is divided by the norm of the sub-sampled tensor. The error convergence is visualized in the figure 4.2.

Both methods show the correlation of tensor size and error convergence level. The larger the tensor, the higher the final error. The error of ALS converges after around 10 to 20 iterations dependent on the size while the stochastic methods keep slowly decreasing. For the full tensor, the error of both methods converge at around 0.9, ALS being slightly lower.

For ALS, the prediction error does not change throughout the training. Interestingly, the behaviour is different for stochastic optimization. The figure 4.3 shows the error convergence for the first 2000 iterations captured every 10 iterations.

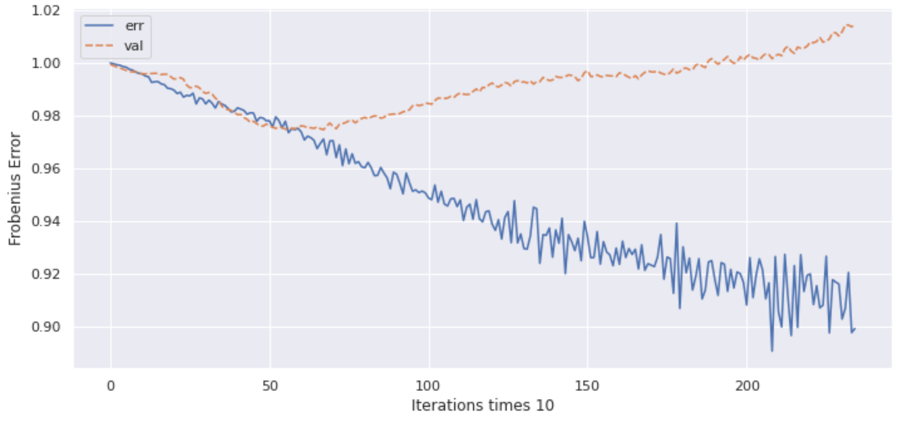

Figure 4.3: The error of the stochastic method. After the initial drop of the prediction error, it starts to diverge and quickly overfit.

One can see an initial decrease of the validation error that begins to increase after around 50 iterations. It suggests that the model begins to overfit and eventually performs even worse than random initialization.

Regarding the second question, I compared different rank sizes 5, 10, 20 and 50 using ALS on the full tensor. The results are visualized in the figure 4.4. The error stagnates after around 5 epochs for all sizes. It makes sense that the larger the R the lower the reconstruction error because the *factor matrices* become more expressive with higher ranks. Unfortunately, the time consumption scales linearly with the size of the rank. Therefore, it was unfeasible to try much larger rank sizes for comparison.
The prediction error does stagnate around 1 even with varying rank size. This might be due to the way I apply the masking on the tensor as described in 3.3.

In another experiment, I compared two ALS variants, one without any constraints, one with non-negativity constraints. In the figure 4.5, I show the error convergence. I ran the decomposition with 20 iterations on the full tensor and set the rank to 20. The time consumption for both methods was about equal. The reconstruction error

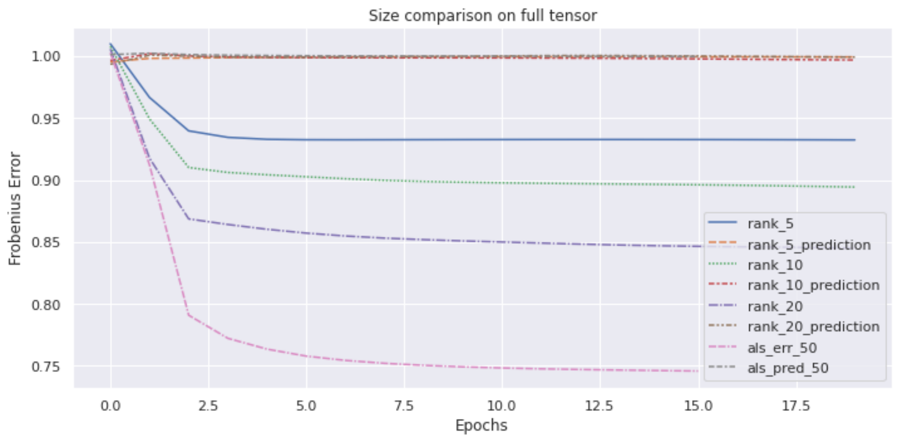

Figure 4.4: Error convergence over 20 epochs using ALS algorithm on the full-size tensor. Larger ranks yield a lower error. Prediction error does not change.

stagnates around 0.85 for both methods. The prediction error stagnates around 0.98 for the non-negative version. The constraint appears to not influence the quality of the reconstruction too much.
Overall, comparing both methods, ALS seems to lead to more promising results. Especially the early overfitting of the stochastic method questions whether this method is appropriate for further analysis, while the stagnation of the prediction loss of ALS might result out of the way the masking was done. Moreover, non-negativity constraints do not penalize the reconstruction too much while providing inherent clustering properties on the *factor matrices*. Consequently, I came to the conclusion that for further analysis I will focus on that method.

4.2 Interpretation of the factor matrices

In this section, I address questions 4 to 6. As the analysis of the convergence suggests, I use the non-negative variant of ALS to compute the *factor matrices*. To maintain high interpretability, I chose $R = 20$. Even though higher ranks help in terms of reconstruction, they potentially jeopardize meaningful clustering.

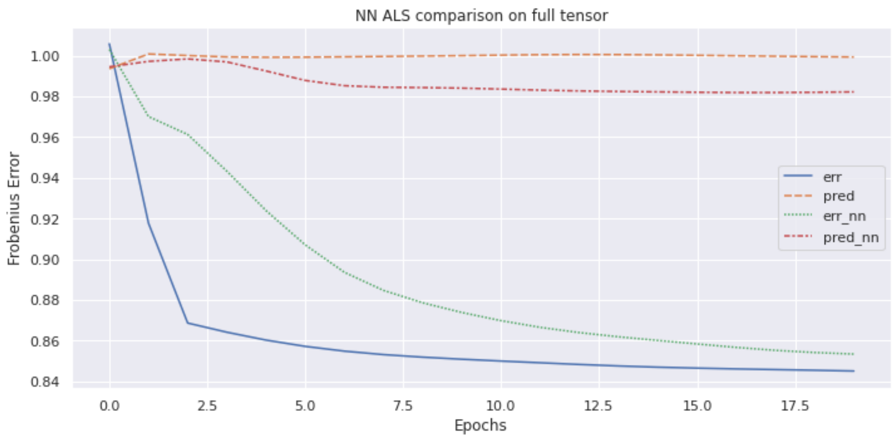

Figure 4.5: Loss convergence over 20 iterations of ALS without constraints versus ALS with non-negativity constraint. The reconstruction error approaches the same level, while the prediction error of the non-negative variant appears to be slightly better.

I define a *topic* as the respective rank-one component. In the first step, I assigned each patient to one of the 20 topics using the highest value respectively. This makes sense as it can be seen as the *association* one diagnosis or patient has to the corresponding topic. To verify it, I plotted a two-dimensional map using t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization of the patient matrix in the figure 4.6. t-SNE converts similarities between data points to joint probabilities and minimizes the Kullback-Leibler divergence between the joint probabilities of embedding and data.[1]. The plot shows that the topics form distinct groups with only partial overlaps.

The patient's distribution across all topics is visualized in 4.7. As one can see, the vast majority of the patients are assigned to topic 13, followed by topic 6. The other topics have 25000 or fewer patients assigned. To observe if the topic clustering reveals hidden information, I utilized additional covariates from the data. The second histogram in the figure shows the distribution of people who reportedly died to any cause (on a log scale). Unfortunately, they appear to be equally distributed relative to the absolute number per topic. No topic shows to have a higher likelihood of early death. The third histogram shows the distribution

[1]https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

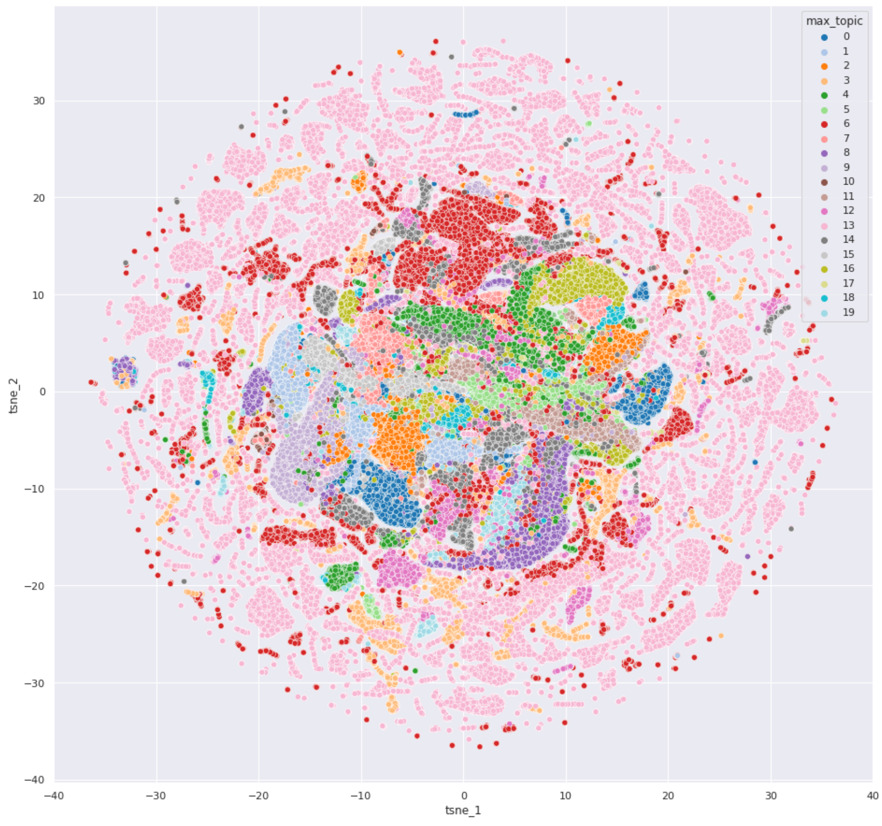

Figure 4.6: T-SNE visualization of the topic clusters. Each point represents a single patient. The hue represents the topic with the maximum score.

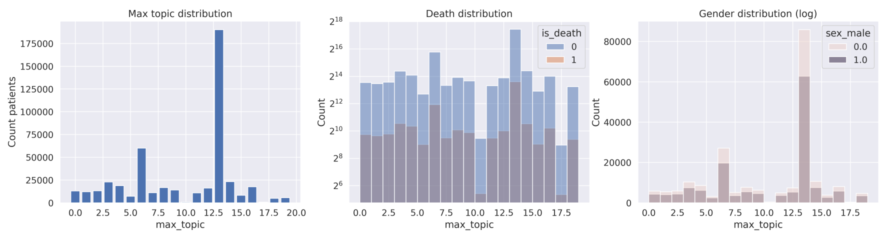

Figure 4.7: Distribution of all patients according to their association through highest value over 20 topics.

of gender. It is of interest if the decomposition implicitly recognized the gender, but again, it is equally distributed.

In the second step of the analysis, I draw my attention to the disease matrix. As I did with the patients, I associate a disease to a topic using its highest value. Tables 4.1 ff. show the diseases for the four topics with the largest amount of patients. As described in 3.1, each disease embeds a number of different codes. These codes are classified by letters ranging from A-Z. I counted the classes to get an idea of how similar the disease embeddings are. From a medical perspective, it could be interesting to examine the clinical relevance of single topics. Topic 13 is enriched for diseases of the kidney. Referring back to the first part of the analysis, topic 13 has the most assigned patients. Literature verifies [24] that kidney disease is one of the leading public health problems worldwide. Nearly one-third of the population is affected. As it is such a large problem, it makes sense that it often appears in the data.

Topic 3 is enriched for neoplastic diseases. Abdominal mass and mass of body structure seem to be correlated with it. It is beyond the scope of this thesis to look deeper into disease correlations, but judging of the corresponding International Statistical Classification of Diseases and Related Health Problems (ICD) codes they seem to be of a similar kind.

Interestingly, topic 6 and topic 14 show exact same diseases. The Pearson correlation coefficient between the two topics on the patient matrix shows a positive association with 0.24, much higher than with most other topics. It follows that patients who are assigned to one topic are likely to have a high value in the other topic as well.

Table 4.1: **Topic 13**

Disease	ICD codes	Strength
disorder of kidney and/or ureter	('N', 24), ('E', 9)	37.97
kidney disease	('N', 23), ('E', 9)	37.91
renal impairment	('N', 4), ('O', 4)	32.18
renal failure syndrome	('N', 4), ('O', 3)	27.94

Table 4.2: **Topic 6**

Disease	ICD codes	Strength
disorder of cardiovascular system	('I', 70), ('O', 22)	47.08
disorder of soft tissue	('L', 68), ('S', 56)	41.45
traumatic and/or non-traumatic injury	('S', 99), ('T', 58)	32.92
disorder of blood vessel	('I', 30), ('S', 13)	30.52

Table 4.3: **Topic 14**

Disease	ICD codes	Strength
disorder of cardiovascular system	('I', 70), ('O', 22)	44.22
disorder of soft tissue	('L', 68), ('S', 56)	35.46
traumatic and/or non-traumatic injury	('S', 99), ('T', 58)	30.84
disorder of blood vessel	('I', 30), ('S', 13)	28.04

Table 4.4: **Topic 3**

Disease	ICD codes	Strength
mass of body structure	('C', 91), ('D', 52)	50.46
neoplastic disease	('C', 91), ('D', 49)	47.99
malignant neoplastic disease	('C', 90), ('D', 5)	47.24
abdominal mass	('C', 38), ('D', 30)	40.56

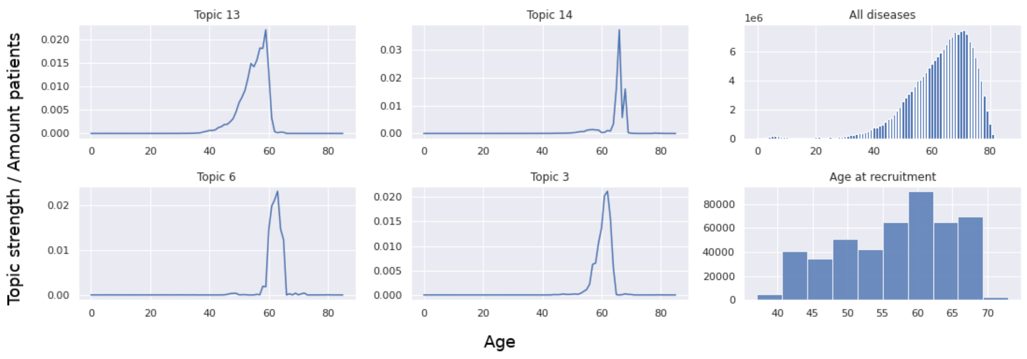

Figure 4.8: Strength of the topics compared to the disease distribution over time. All topics show a spike around the age 60 and sharply decrease after age 70.

To further inspect how the two topics differ, I analysed the time matrix. I visualized the four topics in the figure 4.8 and compared them to the disease distribution over time and the age of the recruitment for all patients.

As most disease events lie between age 60 to 80, it is observable that the topics show a spike somewhere in between with a sharp decrease after. None of the topics show any strength before age 40 and later than age 80. Topic 14 and 6 share the same diseases, but their time-matrices are shifted by a few years. The accumulation of disease events might not purely result from the fact that older people are more likely to develop diseases. As considerable more patients are recruited at age 60 to 70, and they underwent a detailed checkup on recruitment, it is likely that more diseases have been reported at these times.

To sum it up, even though I was not able to identify similarities (gender and death) of patients associated with a shared topic, I was able to show that the diagnosis embeddings indeed share similarities. I also showed that the time matrix precisely identifies the ages of high risk. It is not the goal of this thesis to verify the medical relevance of single topics. Instead, they serve as a verification that the decomposition indeed captures some dependencies and offers some ability to interpret each topic. This might come to use if it is possible to use the topics of the patient embeddings in a predictive task.

4.3 The predictive power of patient embeddings

In section 2.1 I introduced to the Cox Proportional Hazard Model (CoxPH). It utilizes a vector $\boldsymbol{x}$ of covariates to predict the hazard of an event. Let $\boldsymbol{x}_p$ be the vectorized topic associations for patient p. Let Major adverse cardiovascular events (MACE) be the hazard to predict. As a baseline, I use two covariates that both have a high influence on MACE, sex and age. For the experiments, I used CoxPHFitter implementation from the *lifelines* [25] package in python. Two performance measurements are of particular relevance. On the one hand side, the *concordance index* evaluates the accuracy of the predictions and on the other hand, the coefficients of the individual covariates review the influence on the prediction.
First, I provide baseline measurements using only the two covariates. I ran the training process on a train set and evaluated the accuracy on a test set with a split of the data 1 to 5. The evaluation looks as follows: The *c_index* is 0.69. The coefficient for *age* is 0.58 with its 95% confidence interval $0.57 - 0.60$. Gender appears to have a higher coefficient of 0.81 and a slightly broader CI, with a range of $0.78 - 0.83$. Already, the two covariates significantly increase the prediction accuracy.
In a second step, I added the patient matrix as additional covariates to the model. First, I tested it on a subset with 5000 patients. I calculated 20 topics for the patients using standard non-negative Alternating Least squares (ALS) with a final reconstruction error around 0.6, as seen in the figure 4.2. When I ran the CoxPH model, I experienced convergence issues due to high collinearity between single topics. To avoid the error, I ran Principal Component Analysis (PCA) beforehand to reduce topics from 20 to 5. I split the 5000 patients into a train and test set 1 to 5. The *c_index* on the test set is 0.75 and the five PCA coefficients and their CIs are visualized in the figure 4.9. The two baseline covariates appear to have the strongest positive correlation, followed by topic 3. The 95% CI of the other three topics crosses 0.0. Compared to the baseline, the *c_index* improves +0.6. It verifies that the embedding indeed capture some useful information for the hazard prediction.
Next, I ran the model on the full patient matrix.In comparison to the smaller tensor, I did not experience any collinearity issues. Therefore, no additional PCA

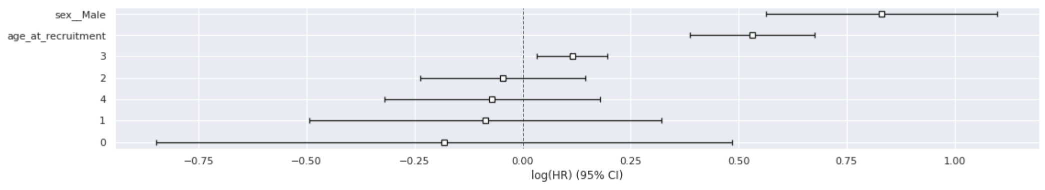

Figure 4.9: The figures show the CI 95% for 5 topics and the two baseline covariates. 20 topics were calculated on a subset of the data with 5000 patients and afterwards reduced using PCA.

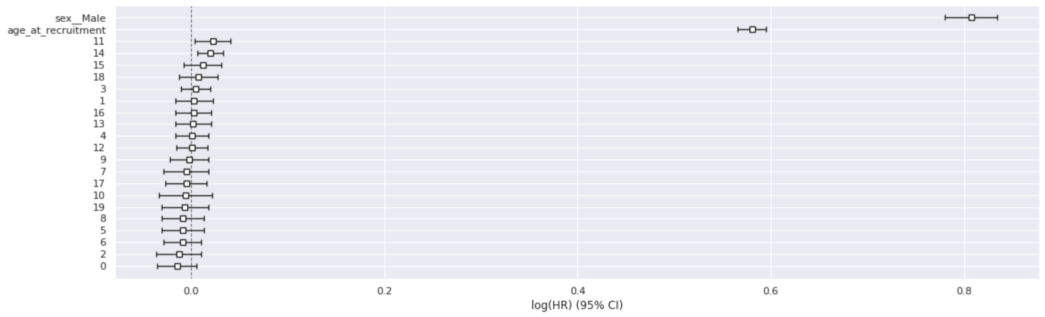

Figure 4.10: The figures show the CI 95% for 20 topics and the two baseline covariates. Topics were calculated on the full dataset using non-negative ALS.

was needed. Unfortunately, the *c_index* only marginally increases to 0.7. The coefficients are visualized in the figure 4.10. They show that the two baseline covariates have the most influence, while the topics have few to no impact on the prediction. Notable is the strong decrease in prediction accuracy compared to the subset. It barely adds any value to the baseline.

In table 4.5, I present the *c_index* for five different events and compare it against the two models on different tensor sizes. Systematic COronary Risk Evaluation (SCORE) operates with a hard endpoint of cardiovascular-related death (cvd_death). *Death* in comparison collects all different causes of death. *Breast cancer* and *Dementia* are two major population-wide diseases that are highly associated with gender and age. The table shows that adding covariates improves the *c_index* on all endpoints. With the reduced amount of patients of 5000 and

Method	Data	MACE	SCORE	Death	Breast cancer	Dementia
Baseline	Full	0.68	0.74	0.69	0.73	0.76
NN-ALS	5k pat	0.71	0.86	0.78	0.75	0.82
NN-ALS	20k pat	**0.73**	0.76	0.71	0.74	0.78
NN-ALS	100 diag	0.68	0.74	0.69	0.74	0.76
NN-ALS	500 diag	0.68	0.74	0.69	0.74	0.76
NN-ALS	Full	0.69	0.75	0.69	0.74	0.76
ALS	5k pat	0.72	0.86	**0.79**	**0.79**	0.79
ALS	20k pat	0.72	0.79	0.71	0.74	0.78
ALS	Full	0.68	0.75	0.69	0.74	0.76
SGD	5k pat	0.70	**0.87**	0.77	0.74	**0.83**
SGD	20k pat	0.71	0.78	0.71	0.72	0.79
SGD	Full	0.68	0.75	0.69	0.73	0.76

Table 4.5: Different endpoints compared against the baseline of the two covariates *age* and *gender* with respect to the concordance index. 20 additional covariates have been calculated with standard ALS, non-negative ALS and Stochastic Gradient Descent (SGD) on a subset of 5000 and 20.000 randomly chosen patients compared to the full data set. Additionally, I compared against a reduced number of diagnosis using the 100 and 500 most common ones.

20000 all three methods show to improve the prediction accuracy significantly. In comparison, reduced amount of diagnosis to the most common, 100 respectively 500, do not improve the prediction. With full data, MACE and SCORE improve slightly, with no improvement on the other endpoints.

To sum it up, the embeddings indeed code valuable information that can be used to increase the prediction accuracy of disease events. Throughout all results, the significance of the impact dwindles as the data increases in size to a point where the embeddings barely add any value at all. I find a negative association of prediction improvement and reconstruction error of Canonical Polyadic decomposition (CPD). As it becomes more complicated to find accurate embeddings, they lose in expressiveness and become less distinct. They capture only broad concepts, what leads to topics with high correlation. Naturally, these topics perform worse in the prediction tasks. As CPD deals with many convergences as well as performance issues, one can ask whether there is a better method to find embeddings that represent the data more precisely.

5 Discussion

In the last chapter of this work, I point out limitations and possible improvements to the proposed method based on the given results. I finish with a conclusion and a short outlook.
As the comparison of the models in the table 4.5 has shown, the addition of disease history embeddings as covariates in the context of survival analysis indeed benefits the prediction accuracy if and only if the embeddings are *good enough*. The goodness of the embeddings is linked to the reconstruction error of the tensor and it remains an open question what level of error is good enough. Nonetheless, it proves that the disease history of an individual does help to predict risk of future diseases.
None of the experiments have shown any significant change of the validation error. Using the stochastic method, the error even increased. I could not find a correlation between *c_index* and validation error. I can see two potential reasons for it. First, the prediction tasks is too complicated. A precise prediction of the exact year of a disease occurrence is possible too hard, based on solely the disease history. To weaken the task and improve the prediction error, a time warping algorithm could be added. In short, time warping measures similarities between two temporal sequences, even if one is faster than the other one [26]. With such algorithm, the approximate time prediction of a disease can be enough to lower the overall error. As a second reason, the mean-masking for ALS might have influenced the factor matrices too much. If the impact of the mean entries in the tensor is too strong, then the factor matrices will represent the mean value instead of generalizing and actually making a prediction. In this case, it would be necessary to find a more efficient way to correctly mask values. I find both reasons likely. The second reason additionally explains why the stochastic methods performs worse than ALS as it

uses a different masking method. In the case of the stochastic method, masking is done directly on the loss function. Therefore, the masked values do not influence the error at all and the entries in the reconstructed tensors are *real* predictions. With the first reason in mind it makes sense that the predictions are mostly wrong and keep getting worse as the model diverges from mean predictions.

Besides the prediction and masking issues, the thesis has several limitations and offers a range of possible improvements and additions. Generally speaking, the thesis serves as an entry point or proof of work to examine the potential of the UK Biobank database. I follow a broad approach with no specific disease class or application field in mind. As a result, it appears challenging to extract valuable information for medical use cases. As seen in [13], narrowing down the target group to tackle a specific disease class might reveal new insight to the task of sub-phenotyping. Additionally, fewer participants result in a smaller dataset and better reconstruction accuracy. With fewer patients, data regarding treatment methods and medications might be added to get an even clearer picture of disease classes and treatment methods.

The UK Biobank has a lot more potential regarding the availability of data. For example, I aimed to measure similarities of patients that are assigned to the same topic by its highest value. Due to time restrictions I limit myself to use the gender and the time of death as additional covariates. However, the database offers the unique possibility to use each individual's genotype. Studies have shown that phenotypic correlations are often assumed to reflect genotypic correlations in evolutionary biology [27]. Unveiling relationships and phenotypic correlations might be an appealing area of application for further studies.

Finally, a more robust and scalable decomposition method most likely would lead to better results. So far, in this thesis I have worked on a single machine. However, literature shows [28] that CPD can be drastically scaled in a distributed environment. Collaborating machines can share the computational cost to provide a unique solution to the intermediate data explosion.

5.1 Conclusion

In this work I presented two models, an improved ALS and a stochastic method, to fit EHR data from the Biobank UK dataset. The main goal of the two algorithms was to provide a low dimensional representation of the disease history of the participants on an individual level with high guarantees in interpretability. A second goal was to improve a survival analysis task. Despite its mediocre results, I show the potential of the method and its success on smaller datasets. I discussed the limitations of the work and hinted on possible improvements of the models. Further research might help to unveil better ways to keep improving the fast evolving field of precision medicine and survival analysis

Bibliography

[1] R. Hodson, "Precision medicine," *Nature*, vol. 537, no. 7619, pp. S49–S49, 2016.

[2] I. R. Koenig, O. Fuchs, G. Hansen, E. von Mutius, and M. V. Kopp, "What is precision medicine?," *European respiratory journal*, vol. 50, no. 4, 2017.

[3] S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. Srivastava, A. Preece, S. Julier, R. M. Rao, *et al.*, "Interpretability of deep learning models: a survey of results," in *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI)*, pp. 1–6, IEEE, 2017.

[4] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[5] R. A. Harshman *et al.*, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[6] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[7] J. Håstad, "Tensor rank is np-complete," in *International Colloquium on Automata, Languages, and Programming*, pp. 451–460, Springer, 1989.

[8] C. F. Van Loan, “The ubiquitous kronecker product,” *Journal of computational and applied mathematics*, vol. 123, no. 1-2, pp. 85–100, 2000.

[9] N. K. M. Faber, R. Bro, and P. K. Hopke, “Recent developments in candecomp/parafac algorithms: a critical review,” *Chemometrics and Intelligent Laboratory Systems*, vol. 65, no. 1, pp. 119–137, 2003.

[10] M. Welling and M. Weber, “Positive tensor factorization,” *Pattern Recognition Letters*, vol. 22, no. 12, pp. 1255–1261, 2001.

[11] D. R. Cox, “Regression models and life-tables,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–202, 1972.

[12] J. Henderson, J. C. Ho, A. N. Kho, J. C. Denny, B. A. Malin, J. Sun, and J. Ghosh, “Granite: Diversified, sparse tensor factorization for electronic health record-based phenotyping,” in *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 214–223, 2017.

[13] J. Zhao, Y. Zhang, D. J. Schlueter, P. Wu, V. E. Kerchberger, S. T. Rosenbloom, Q. S. Wells, Q. Feng, J. C. Denny, and W.-Q. Wei, “Detecting time-evolving phenotypic topics via tensor factorization on electronic health records: Cardiovascular disease case study,” *Journal of biomedical informatics*, vol. 98, p. 103270, 2019.

[14] Y. Kim, R. El-Kareh, J. Sun, H. Yu, and X. Jiang, “Discriminative and distinct phenotyping by constrained tensor factorization,” *Scientific reports*, vol. 7, no. 1, pp. 1–12, 2017.

[15] T. G. Kolda and D. Hong, “Stochastic gradients for large-scale tensor decomposition,” *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 4, pp. 1066–1095, 2020.

[16] N. Vervliet and L. De Lathauwer, “A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 284–295, 2015.

[17] K. W. Fung, C. McDonald, and S. Srinivasan, “The umls-core project: a study of the problem list terminologies used in large healthcare institutions,” *Journal of the American Medical Informatics Association*, vol. 17, no. 6, pp. 675–680, 2010.

[18] S. Prinja, N. Gupta, and R. Verma, “Censoring in clinical trials: review of survival analysis techniques,” *Indian journal of community medicine: official publication of Indian Association of Preventive & Social Medicine*, vol. 35, no. 2, p. 217, 2010.

[19] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, “Tensorly: Tensor learning in python,” *Journal of Machine Learning Research (JMLR)*, vol. 20, no. 26, 2019.

[20] B. W. Bader and T. G. Kolda, “Efficient matlab computations with sparse and factored tensors,” *SIAM Journal on Scientific Computing*, vol. 30, no. 1, pp. 205–231, 2008.

[21] S. J. Wright, “Coordinate descent algorithms,” *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[22] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[23] I. Loshchilov and F. Hutter, “Fixing weight decay regularization in adam,” 2018.

[24] J.-C. Lv and L.-X. Zhang, “Prevalence and disease burden of chronic kidney disease,” *Renal Fibrosis: Mechanisms and Therapies*, pp. 3–15, 2019.

[25] C. Davidson-Pilon, J. Kalderstam, N. Jacobson, S. Reed, B. Kuhn, P. Zivich, M. Williamson, AbdealiJK, D. Datta, A. Fiore-Gartland, A. Parij, D. WIlson, Gabriel, L. Moneda, A. Moncada-Torres, K. Stark, H. Gadgil, Jona, K. Singaravelan, L. Besson, M. S. Peña, S. Anton, A. Klintberg, GrowthJeff, J. Noorbakhsh, M. Begun, R. Kumar, S. Hussey, S. Seabold, and D. Golland, “Camdavidsonpilon/lifelines: v0.25.11,” Apr. 2021.

[26] P. Senin, “Dynamic time warping algorithm review,” *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, vol. 855, no. 1-23, p. 40, 2008.

[27] S. M. Sodini, K. E. Kemper, N. R. Wray, and M. Trzaskowski, “Comparison of genotypic and phenotypic correlations: Cheverud’s conjecture in humans,” *Genetics*, vol. 209, no. 3, pp. 941–948, 2018.

[28] A. L. De Almeida and A. Y. Kibangou, “Distributed large-scale tensor decomposition,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 26–30, IEEE, 2014.