UNIVERSITY OF TARTU
FACULTY OF SCIENCE AND TECHNOLOGY
Institute of Computer Science
Computer Science Curriculum

Lauri Tammeveski

Learning DNA mutational signatures using neural networks

Master's Thesis (30 ECTS)

Supervisors:
Raul Vicente Zafra, PhD
Leopold Parts, PhD
Tambet Matiisen, MSc
Ardi Tampuu, MSc

Tartu, 2016

Learning DNA mutational signatures using neural networks

All cancers are caused by mutations in the cells of an organism. It is found that these mutations result from the combination of specific mutational signatures, which often have known underlying processes. That is why learning these signatures is important — it can give better information about the mechanisms of cancers and also be helpful for cancer prevention and therapy. The aim of this thesis is to test and compare different methodology to improve the discovery of mutational signatures. In particular, we compared three new methods of neural networks (NN), rectified factor networks (RFN) and topic modelling to the currently used non-negative matrix factorization (NMF). We experimented with the methods on three organic and three synthetic data sets by measuring reconstruction error, sparsity and time taken and compared them with NMF. The results show that NMF produces the smallest error on easier data sets, but error of RFN is comparably good also and on all other data sets produces the best result. NN performs equally well with RFN on more difficult data sets and overall produces the sparsest results. The advantage of NMF is the stability functionality that determines very well the correct number of signatures. Future work will be needed to add this capability to RFN and NN methods which would enable their practical use for the problem of finding mutational signatures.

Keywords: mutational signatures, neural networks, non-negative matrix factorization, rectified factor networks, topic modelling

CERCS: B110 Bioinformatics, medical informatics, biomathematics, biometrics

DNA mutatsiooniliste signatuuride õppimine tehisnärvivõrkude abil

Kõik pahaloomulised vähkkasvajad on põhjustatud organismi rakkudes toimuvate mutatsioonide poolt. On leitud, et need mutatsioonid on moodustatud spetsiifiliste mustrite ehk signatuuride kombinatsioonist, mille aluseks olevad protsessid on tihti teada. Seetõttu on nende signatuuride õppimine andmetest väga tähtis — see võib anda paremat informatsiooni vähkkasvajate mehhanismide kohta ja olla abiks vähi ennetamisel ja teraapial. Antud töö eesmärk on testida ja võrrelda erinevaid metoodikaid, et parandada mutatsiooniliste signatuuride leidmist. Me võrdlesime kolme uut meetodit — tehisnärvivõrgud (NN), mittenegatiivsed faktorvõrgud (RFN) ja teemade modelleerimine — praegu kasutatava mittenegatiivse maatriksi faktoriseerimisega (NMF). Me eksperimenteerisime meetoditega kolmel orgaanilisel ja kolmel sünteetilisel andmestikul ning mõõtsime rekonstrueerimise viga, tulemuse hõredust ja arvutusteks kulunud aega. Tulemused näitavad, et NMF annab väikseima veaga tulemuse kergematel andmestikel, kuid ka RFN-i tulemus on ligilähedane ning kõikidel teistel andmestikel saavutab see parema tulemuse. NN esineb sama hästi kui RFN keerulisematel andmestikel ning lisaks saavutab üleüldiselt kõige hõredamad tulemused. NMF-i eeliseks on stabiilsuse arvutamise funktsionaalsus, mis väga täpselt suudab määrata õige signatuuride arvu. Tulevikus tuleb teha edasist arendustööd, et sarnane võimekus ka RFN ja NN meetoditele lisada, mille järel oleks võimalik nende praktiline kasutamine mutatsiooniliste signatuuride õppimisel.

Võtmesõnad: mutatsioonilised signatuurid, tehisnärvivõrgud, mittenegatiivne maatriksi faktoriseerimine, mittenegatiivsed faktorvõrgud, teemade modelleerimine

CERCS: B110 Bioinformaatika, meditsiiniinformaatika

Contents

Introduction

All cancers are caused by mutations in the cells of an organism. The mutations may be the result of a defective DNA repair, exposures to mutagens, modification of DNA by enzymes, etc [1]. A mutational signature is a combination of mutation types, caused by different mutational processes [1]. Studies have shown that some mutation types occur more often in specific cancers [2]. For example, substitutions where cytosine mutates to adenine and guanine to thymine occurs very often in smoking-associated lung cancer and CC:GG to TT:AA i.e. cytosine to thymine and guanine to adenine double nucleotide substitutions caused by UV light are typical in skin cancers [2]. Finding these signatures from genome mutation data is very important for finding underlying mechanisms of cancers as well as prevention and therapy [1]. This thesis focuses on improving the methodology to discover mutational signatures.

Problem

Biochemical processes that take place in cells can often be reasonably thought to be acting independently [3]. Thus, we assume that mutations in the genome are the sum of all mutagenic process activities across cells. The data we have are mutation counts of different mutation types over samples. A possible model for this data is a decomposition to a matrix multiplication of non-negative values, $Y = W \times X$, where Y is the complete data, W is the collection of the different signatures, and X are the strengths determining their activity. Usually, we are presented with observations Y, and care about X (e.g. given a lung cancer genome, how large is the effect of smoking) or W (given data from many cancers, what does a mutational signature for defective DNA mismatch repair look like). When the initial matrix

Y has the dimensions of number of mutation types F times the number of samples S, after the decomposition, we get a matrix W of dimensions F times the number of signatures P and a matrix X with dimensions P times S. This usually makes the interpretation of the data better, since instead of having one large matrix, we have two smaller ones, which are both related to the number of signatures.

Since we only observe Y and neither W nor X are known, the task is solved as an inference problem that deduces underlying distributions from the available data [4].

Motivation

In recent years, many approaches have used general decomposition methods (principal components analysis, factor analysis, non-negative matrix factorization, independent component analysis) to solve the problem that rely heavily on normal distributions or strong assumptions about distributions [2, 5, 6, 7, 8]. These assumptions make the models inappropriate to apply to data that are not ideal but realistic — sparse, categorical, heavy tailed, etc.

The main goal of this thesis is to extend the approach based on factor analysis and to apply neural networks, a more flexible, non-linear model class, both expected to give more accurate results for the decomposition. Our goal is to reduce reconstruction error while producing sparse signatures. These improvements both alone and combined can make the interpretation of the decomposition and the signatures a lot better and more realistic.

Contributions

In this thesis, we apply three new methods to organic genome mutation data and to synthetic data sets: rectified factor networks, latent Dirichlet allocation and neural

networks (autoencoders). The results are compared with each other and with validated method [2] that is based on non-negative matrix factorization. Before that, we also have to modify algorithms to match our non-negativity constraints. Our hypothesis is that the decomposition $Y = W \times X$ using rectified factor networks and autoencoders is sparser, while having similar or smaller reconstruction error, compared to previously used techniques. In addition, latent Dirichlet allocation that comes from an alternative theoretical background of topic modelling, will also be able to achieve results as good as using non-negative matrix factorization. When comparing rectified factor networks and autoencoders, we expect the former to give even better results than latter and if that is the case, should be favored in the future over autoencoders [9].

In addition we will experiment with hierarchical latent Dirichlet allocation that organizes signatures into a tree structure where the most general signatures are near he root and more specific ones near the leaves [10].

Outline

Background includes a detailed description of the problem and gives an insight into the biological background of the task.

Materials and methods explains and describes the methods that are used to make the learning of mutational signatures more effective and parameters needed for each of the methods. Also, the data sets that were used and synthetic data that was generated for further analysis are described.

Results gives a detailed explanation of all the different results obtained using the methods given in the previous chapter.

Discussion includes a brief summary of the main results and brings out the limitations that prevented us from having better results. We also discuss what should be the future directions to further improve the learning of mutational signatures and what are the questions that are still left to answer.

Conclusion gives the final notes to take away from this thesis and the most important results achieved.

References lists articles and websites, where extra information is available for used methods and the problem of finding mutational signatures.

Appendix provides tables of more detailed results, gives the location to the code used to achieve the results and a licence that allows reproduction of this thesis and making it public.

1 Background

In this chapter, the problem of learning mutational signatures of human cancer is described and an overview of the gene biology and aspects that should be considered regarding this problem are presented.

1.1 Biology of DNA mutations

In addition to being a complex computational problem, deciphering mutational signatures also requires knowledge of what goes on in the cells of an organism.

According to the book by Robbins, in the nucleus of each human body cell are DNA molecules in the form of chromosomes. Each chromosome consists of four different nucleotides - Adenine, Cytosine, Guanine and Thymine. In bioinformatics, these are usually represented by their first letters, so substrings of DNA are usually just repetitions of four letters in some order.[11]

More background is explained by Branderberg et al. Since DNA is a double helix, there is a strand and its complementary strand and a basepair is formed between the nucleotides that are in the same location. The four nucleotides actually form two pairs. Adenine and Thymine are complementary to each other and the same goes to Cytosine and Guanine. Also, there is a strict rule, that the basepair has to consist of nucleotides that are complementary to each other, so for example when A is on one strand, T is on the other and when G is on one strand, C has to be one the other, at the same location.[12]

1.1.1 Mutational signatures

When a mutation occurs in a cancer genome, one of the nucleotides is swapped by another, e.g., T is substituted by A [3]. There are also other mutations, in addition to substitutions, like insertions and deletions, but these are not considered in this thesis [1].

The mutations may be the result of a defective DNA repair or different mutational processes, like mutagen exposures (UV, other radiations, smoking), enzymatic modification of DNA, etc. [1]. It is believed that majority of mutations are "passengers" that have no negative effect and only a minority are "drivers" that cause cancers [2]. So actually most of the mutations are harmless.

When a mutation occurs, the nucleotide in the complimentary strand is also changed. So when in principle, four nucleotides that each could be mutated to the other three nucleotides, means 12 possible mutations, then actually only six are considered: C > A (read C to A), C > G, C > T, T > A, T > C and T > G, because the other six happen on the complementary strand [2]. To get more information out of the mutations, bases immediately after and before the mutation are also considered and since there are 4 possible nucleotides before and after, there is in total 6 * 4 * 4 = 96 different mutation types. This is written as TCG > TTG, where the center nucleotide is mutated [2]. For example, this kind of surrounding is useful for distinguishing mutational signatures that cause the same substitutions but in different sequence contexts.

A mutational signature is a certain combination of mutation types, caused by different mutational processes, and then divided by the total number of mutations caused by this signature such that in the end proportional contributions of each mutation type is considered and these sum to one [2]. So it could be said, signatures are normalized. All in all, signature is characterized by which mutation types out of the 96 are prevalent and how much they contribute.

Since the number of signatures that could be found is very large, there is also a list of validated signatures that is used to compare to the ones discovered

using the decomposition. For the moment, 30 signatures are validated and a table indicating in which cancer types they are found is provided in ref. [13]. There are two signatures that should be present in every cancer sample, there are some that appear in several different types of cancer, and others are limited to very few cancers [13].

1.2 Learning mutational signatures

Our task is to find mutational signatures that make up the mutations in cancer genomes. According to an earlier paper published in this field by Alexandrov et al., this is a problem of finding the decomposition of the given mutation data matrix into two matrices [2]. The situation is the one as in Figure 1.1. There, F stands for mutation types or features, S are the samples or cancer genomes and P stands for patterns or signatures. Since the result is two smaller matrices, they are easier to interpret.[2]

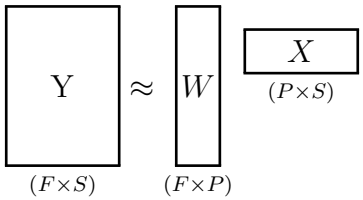

Figure 1.1 – Mutation data decomposition, where original matrix is decomposed into two representing patterns W and strengths X.

According to ref. [2], this problem of decomposition resembles very much to the cocktail party problem, where lots of people are at the party and speaking simultaneously. There are also several microphones at the party at different locations which capture the mixture of sounds and the task is to separate the different conversations. Thanks to the fact that microphones are at different locations, they record conversations with different volumes and this helps to distinguish them. This is very similar to our problem, where the recordings are the cancer genome samples, microphones are the cancers and conversations are the mutational signatures.

The main constraint that has to be considered and achieved during decomposition is that all the resulting matrices have to be non-negative. That is because they represent counts of mutations and there cannot be negative counts. In addition to having these constraints, it would be better if they were also sparse (meaning more zeros). That would mean patterns are easier to interpret and differentiate between each other.

2 Materials and methods

Machine learning methods are divided into two main groups — supervised and unsupervised methods. To use supervised methods, data has to be labeled and then the best prediction from data points to labels is found. In our case, there is no labeling, so unsupervised methods have to be used that try to find patterns and structure in the data [10]. This is still a very broad category where lots of different methods are used for solving numerous problems. We need decomposition methods that would produce precise results while also respecting our constraints and preferably give sparse output.

This chapter gives a description of the original method used in the article by Alexandrov et al. [2] regarding learning somatic signatures and all the methods that we tried, to improve the results. The measure of performance for all the methods is Frobenius norm, which characterizes the reconstruction error. The formula is $\sqrt{\mathrm{Tr}(A^\top \times A)}$, where A is the difference matrix between the original data matrix and the reconstructed matrix after applying methods and the smaller the result, the better. In addition, an overview of the data sets that contain counts of somatic mutations and synthetic data sets for testing out methods with known ground truth is present in this chapter.

2.1 Data

Catalogs of mutations from tens of thousands of cancer genomes are generated by many sequencing projects and this number is growing rapidly thanks to advancing technologies [2]. This is very important, since for more stable and precise signatures, more samples are needed [2].

Data sets of mutations represent the counts of mutations in a sample divided among features [2]. In this thesis, the number of features is 96, representing 6 substitution types and its 16 possible different surrounding nucleotide pairs. But it is possible to have many other categories, like larger surroundings, insertions and deletions in addition to substitutions, strand bias, etc. [2].

2.1.1 Mutation counts data set

The data sets we used, were provided in the Matlab code by Alexandrov et al [14]. There were both exome and whole genome data sets. Exomes are parts of genes that are converted from DNA to RNA and for that reason are more important part of the genome [15]. They constitute $\sim 1\%$ of the whole genome, so that is why the counts are also much lower [15]. For this reason, we use whole genome data sets. The same was done by Alexandrov et al. in ref. [2] where a simulated data set of 100 whole genomes and an organic 21 breast cancer genome data set were used, both consisting of whole genome substitutions. In the article, they also stated that it is better to use fewer samples with many mutations than more samples with just a few mutations as is the case with exome data sets.

Their Matlab code had the data set of 21 breast cancer genomes, data set of 119 breast cancer genomes and data sets of acute myeloid leukemia (AML), chronic lymphocytic leukemia (CLL), liver cancer, lung adeno, lymphoma B-cell, medulloblastoma, pancreas cancer and pilocytic astrocytoma [2].

For the validation of their method, we used the same data set of 21 breast cancers. For comparison with other methods, we also use the data set of 119 breast cancer samples, since we know from [2] what the signatures of breast cancer should look like, but having 98 additional samples makes the decomposition more accurate. To see how the methods are suited for larger matrices, we also created a data set containing all the aforementioned cancer types for a total of 506 samples with 96 substitution types. So we have one small (21 samples, 183 916 mutations), one average sized (119 samples, 647 692 mutations) and one large (506 samples, 3 375 007 mutations) matrix to test the methods.

To get an intuition how the data sets are formed, we also tried to make a new data set using catalogue of somatic mutations in cancer (COSMIC) data of mutations that consisted of over three and a half million mutations [16]. We expected to get a bigger resulting data set, but mutations were too scattered over lots of samples and only consisted of exomes, so the mutations per sample rate was not good enough. But we become acquainted with how the counts are formed: if strand bias is not considered, reverse complement of the nucleotide triplet on the complementary strand has to be taken, when the mutation occurs on that strand. If the mutation is on the main strand, then counts can be added as regular [13].

2.1.2 Synthetic data set

The problem with using organic data is that the ground truth is not known and we can not be sure that the signatures that are extracted are actually correct. To solve this problem we also created synthetic data.

Figure 2.1 – 5 synthetic signatures on top of each other along the y-axis with their true distributions over 96 mutation types on the x-axis.

We used 500 samples and 2000 mutations per sample to replicate quite closely how the largest original data set looked like. We created distributions of 5 signa-

tures and 96 mutations that can be seen in Figure 2.1 and from normal distribution with mean 0 and standard deviation 100 created the proportions of signatures for each sample. With these parameters, there is quite a lot of variance between samples that should make the decomposition more difficult. To see how well the methods are resistant to noise, we created three data sets. For the first, we sampled all 2000 mutations by first sampling the signature and then the mutation according to created distributions. For the second one we sampled 200 mutations per genome sample and then multiplied each value by 10 to get 2000 mutations per sample and for the last one only 20 samples and multiplied by 100. Since in the last one, at least 76 mutation types are zeros, for each sample, algorithms have to gather lots of information across samples to distinguish the real patterns and the result depends a lot on the number of samples available.

Table 2.1 – Error difference between ground truth and sampled data.

Nr of sampled mutations	2000	200	20
Frobenius error	0.005	0.017	0.046

In Table 2.1 are the error values which indicate how much noise is in the data. These are calculated by taking the Frobenius norm from the difference of ground truth W matrix and the matrix that is generated by sampling certain amount of times from signature distributions. As can be seen, there is always around 3 fold increase in the error. We will see, how these noise levels affect the result of the decomposition.

2.2 Algorithms

In this section, description of the method used by Alexandrov et al. [2] and the new methods of rectified factor networks, topic modelling and artificial neural networks are presented. How these perform will be presented in the next chapter - Results.

2.2.1 Non-negative matrix factorization (original method)

The first well known article about deciphering signatures of mutational processes in human cancer was published by Alexandrov et al. in 2013 [2]. They also decomposed the matrix of mutation types F as rows and samples S as columns into two matrices. One representing patterns, with also F for rows and mutational signatures P for columns, and the other, strengths, representing the proportions of each signature for each sample, with dimensions P x S [2]. We will use their method for validation and comparison with new methods.

The algorithm created by Alexandrov et al. is in large part based on non-negative matrix factorization (NMF), but has some additional pre- and postprocessing. Overall, the algorithm consists of 6 steps [2]:

1. **Dimension reduction**, where mutation types that together account for $\leq 1\%$ of mutations across genomes are removed.
2. **Bootstrap**, using Monte Carlo bootstrap resampling to avoid overfitting the data.
3. **NMF** using the multiplicative update algorithm by Lee et al. [17] for the decomposition.
4. **Iterate**, where steps 2 and 3 are iterated until convergence criteria is met.
5. **Cluster**, where the signatures resulting from NMF of each iteration are clustered into N clusters based on cosine similarity and also standard deviations are found.
6. **Evaluate** computes reproducibility of the averaged signatures by comparing the tightness and separation of clusters based on silhouette width method in the range [-1,1]. Each cluster is averaged into one signature. Also Frobenius reconstruction error is found.

According to the article [2], extra steps make the result the average of several runs, so the signatures could be more generalized also to other data. In addition, the standard deviation error bars of each mutation type, give an intuition on the variance of the feature.

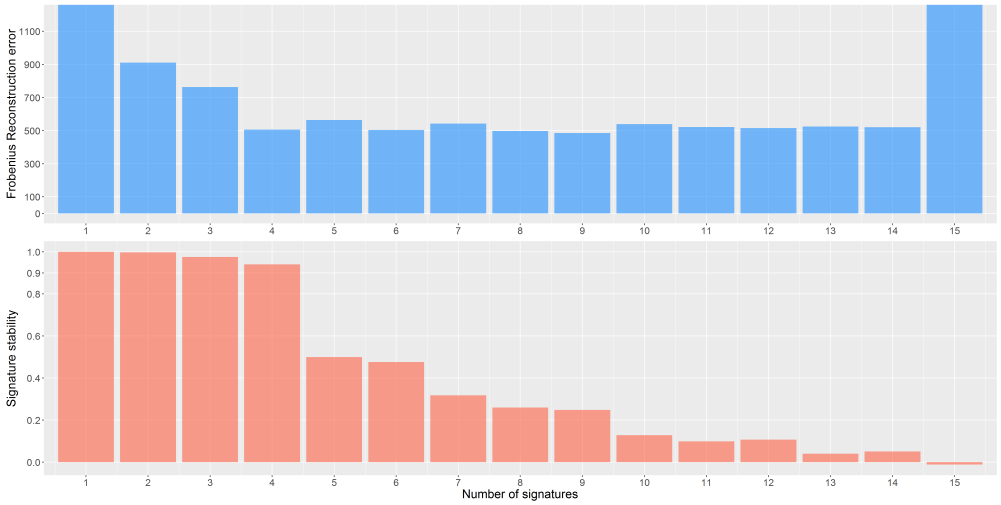

Figure 2.2 – Finding number of signatures: upper plot — Frobenius error, bottom — Signature stability from 1 to 15 signatures.

In their article [2], two data sets were used. The larger one of 100 samples was simulated and a smaller with 21 samples was collected using genome sequencing technologies. Since their code came with other data sets, we did not use the synthetically generated matrix and used the data set of 21 breast cancer genomes for direct comparison with the article and data sets with sizes of 119 samples and 506 samples (which was the maximum) to see how their method works with larger matrices. The article included their implementation that we used for validation [14].

In addition to the convergence parameter that we left as the default, the only parameter needed to set for the decomposition was the number of signatures N. For identifying it, they ran the algorithm on different N from 1 to lowest dimension of the data matrix and compared signature stability and Frobenius reconstruction error to see which gave the most stable signatures (at least over 0.9) while maintaining low error [2]. In Figure 2.2 is shown the result of using NMF for 21 sample

data set for number of signatures ranging from 1 to 15. The optimal number of signatures is 4, since after that the stability of signatures decreases drastically. This had to be done for all of the data sets. The algorithm is developed well, but newer methods are also available for the decomposition and four of them are presented in sections to come.

2.2.2 Rectified Factor Networks

Rectified factor networks (RFN) apply factor analysis to the data in the shape of a network, so they resemble a bit to neural networks. It is the newest method of the ones tried out in this thesis and according to the authors, should give sparser representations of the input, while still having similar reconstruction error as older methods and explain well the covariance structure of the data [9].

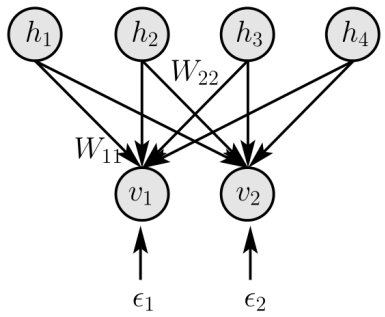

Figure 2.3 – Factor analysis model: data v connected with hidden factors h by weights W and noise ϵ [9].

In Figure 2.3 is a factor analysis model. RFN consists of single or stacked models like these. We only concentrated on the single layer model. The network reconstructs the data using formula $v = W \times h + \epsilon$, where h are hidden factors and W are weights [9].

While the formula might suggest that it is similar to non-negative matrix factorization, the actual algorithm is significantly more complex. It uses posterior regularization method, which similarly to expectation-maximization alternates between E-step and M-step to achieve desired results [9]. In the case of RFN, the

desired constraints are non-negativity and sparsity for posterior of the hidden variables [9].

As the name suggests, rectifying is used to achieve the main goals - sparsity and non-negativity of the hidden factors. Rectifying means applying rectified linear units (ReLU) to hidden variables and are defined as $x = \max(0, x)$ [9]. As can be seen, they enforce both non-negativity and sparsity. To achieve even more sparsity, some positive cutoff value should be subtracted from all the values so that the values smaller than the cutoff would now be negative and substituted with zero [9].

In the paper by Clevert et al. [9] non-negativity and sparseness constraints were only applied to the hidden factors. Since our biological problem decomposes data matrix into two smaller matrices that represent distributions of mutations and proportions of signatures, they can be interpreted only when both are non-negative. That is why we had to modify the algorithm to suit that constraint. In the original algorithm, W is initialized using the normal distribution, where values can be both negative and positive. To have the smallest difference with the article, but still be applicable for us, we took absolute values of the weights to start off. After that, we had to apply ReLUs for W also in the right step of the algorithm. These two combined gave satisfactory results for us.

A large part of the RFN article by Clevert et al. [9] concentrated on proving the convergence and correctness properties of their algorithm. After our modifications, the properties do not always seem to hold, but by doing restarts and taking the best reconstructions obtained, the results are similar and comparable to those obtained with other methods, so our modifications seem to be justified. The comparisons can be seen in chapter 3 - Results.

The paper by Clevert et al. [9] also presents a table comparing RFN and other unsupervised methods for data representation. The other methods compared were RFN without normalization, denoising autoencoders with ReLUs, restricted Boltzmann machines, different kinds of factor analysis, independent component analysis and principal components analysis. They created 9 synthetic data sets where bi-

clusters were planted and the methods had to discover that structure. Sparseness of the components, input reconstruction error and covariance reconstruction error was compared. According to their tests, RFN (specially the normalized version) gave the best sparsity while maintaining similar reconstruction error with the second best. The second best after both versions of RFN was autoencoder and after that the results got worse. For more exact results, look up ref. [9]. It is not sure how well these kind of data sets compare with real life data sets, but these results were good enough to experiment with the method.

In the original method my Alexandrov et al. [2] data sets were trained using resampling several times and then clustering was used to find most similar signatures of each training. From this formed the notion of signature stability — are all the signatures in one cluster really the same or is there actually variance between each training. We used resampling with all of our methods, but this kind of stability approach did not work with our methods, so we took the metric of cosine similarity which was also the underlying foundation when calculating the stability and used this for determining what is the right number of signatures [2]. It has the following formula [2]:

$$cosineSimilarity(X, Y) = \frac{\sum_{k=1}^{n} X_k Y_k}{\sqrt{\sum_{k=1}^{n} X_k^2}\sqrt{\sum_{k=1}^{n} Y_k^2}}$$

where X and Y are two signatures and result ranges in $[0, 1]$ for non-negative values. Since we prefer sparse signatures, cosine similarity might result in false positive result, since if one mutation is zero, it also cancels out the other one. That is why we also evaluate using Pearson correlation coefficient [18]:

$$cor(X, Y) = \frac{\sum_{k=1}^{n}(X_k - \overline{X})(Y_k - \overline{Y})}{\sqrt{\sum_{k=1}^{n}(X_k - \overline{X})^2}\sqrt{\sum_{k=1}^{n}(Y_k - \overline{Y})^2}}$$

which is just $cosineSimilarity(X - \overline{X}, Y - \overline{Y})$, where the means of X and Y are subtracted, solving the issue with sparsity. These measures are also both invariant to scaling, so independent of whether normalized or unnormalized values are used, the result is the same, which is an important feature. This leaves the last question of what should be the threshold where we consider two signatures similar. We set it to be 0.7, but also verified visually, because often when having more signatures than optimal, there were not similar signatures, but they were just vague, in the sense that each signature had a small amount of almost all mutations with some fluctuations and there was not a distinctive shape for a signature, like the ones officially confirmed by COSMIC [13].

We used the implementation provided with the article [19]. It also had some hyperparameters to tune. The most important were learning rates and number of iterations. Letting it run for too many iterations made it overfit and also took more time and also setting a too low learning rate made reaching the minima time consuming. Some probing had to be done to find the best combination, which for us seemed to be 100 iterations and 0.1 learning rate for the initial model and 0.01 for consecutive sets, because then smaller steps should be made. We also analyzed other parameters, but changing those from the defaults did not improve the result. Complete list of settings and code used for producing the results are available on GitHub [20].

2.2.3 Topic modelling

Topic models are algorithms that find main themes in collections of documents and then can organize collections based on the found themes. These are probabilistic algorithms that use statistical models for analyzing the words of the texts and based on that try to find the main topics. In addition to just being able to organize documents, topic modelling can be applied to finding patterns in images, social networks, and genetic data [21]. That is the reason why we also tried out this method to see whether they are comparable with others.

2.2.3.1 Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is the most basic topic model. It assumes that each document exhibits several topics in some proportions [21]. Using statistical inference, it tries to capture the generative process that created the documents from the topics [21].

To get more into detail, a topic is a distribution over a fixed vocabulary [21]. In our research, the vocabulary is the 96 mutation types and a topic is a signature over the words with some distribution. For example, one topic might have 0.2 of ACA > ATA and 0.8 of ATG > AAG and 0.0 for others. It is also assumed that these topics are fixed before data generation. Then for each document (or sample in our case) the words (mutations) are generated in a two-stage process [21]:

1. Randomly choose a distribution over topics
2. For each word in the document
   (a) Choose a topic randomly based on the topics distribution from step 1
   (b) Choose a word randomly based on the vocabulary distribution from topic chosen in step 2.a

This is the way how the documents are created. According to the article ref. [21], since we actually have the documents as evidence and need to find the distributions, we need to solve the reversed generative process by fixing the number of topics and finding the posterior probability. To find the exact posterior, all the possible topic structures should be considered, which is exponentially large and intractable to compute. That is why an approximation is found. The main methodology for that is using sampling, for example Gibbs sampling [21].

The implementation we used is done by Blei, who is also the author of the article [21], so it should be certain that it works as stated. To begin with, the data

set had to be converted into a certain format for each sample

$$[M] \quad [term_1] : [count] \quad [term_2] : [count] \quad ... \quad [term_N] : [count]$$

where $[M]$ is the number of distinct mutation types per sample, $[term_1]...[term_N]$ are mutation types that occur in that sample and count shows how many times each term occurred. Ordering of mutations has to be fixed and same over the whole data set. In addition, a settings file has to be used, which specifies the convergence criterion or number of maximum iterations. Signature proportions over samples are modelled as Dirichlet distributions, which take parameter α that determines the shape of the distribution. The settings file has to also include whether user wants a fixed α or this is also estimated during training from some starting α [22]. We did the second option, which worked quite well. The settings we used are in Table 2.2, where first two lines are variational inference parameters, second two for variational EM and the fifth for alpha. More detailed descriptions are available in ref. [22].

Table 2.2 – Parameters used for LDA.

var max iter	-1
var convergence	1e-8
em max iter	80
em convergence	1e-7
alpha	estimate

2.2.3.2 Hierarchical Latent Dirichlet Allocation

One of our goals was also to find relations between signatures. According to the defined 30 mutational signatures by COSMIC, there are some signatures that are found in all the different types of cancer [13]. Because of that, our another hypothesis is that there might be some kinds of signatures that are more general. That is why we also try hierarchical methods to uncover if this is true.

The method most suited for this is the hierarchical latent Dirichlet allocation (hLDA) [10]. According to the paper ref. [10], it considers an infinitely branching

infinitely deep tree, where each node is a topic. More general topics are near the root and they get more specific down the tree. For example in the article, the root topic had words - "the", "of", "and", "to", "a".

To elaborate more, based on ref. [10], this method uses a generative process where for each document, a path from root of the tree to a leaf is considered. Choice of where to go next is done according to the nested Chinese restaurant process (nCRP), which is a stochastic process that assigns probability distributions to trees. More specifically and according to the restaurant idea, we have N customers in a sequence in the first restaurant. The first customer sits at the first table and the table for the nth customer is drawn from the distribution [10]:

$$\text{p(occupied table } i \mid \text{previous customers)} = \frac{nr_i}{\gamma+n-1}$$
$$\text{p(next unoccupied table} \mid \text{previous customers)} = \frac{\gamma}{\gamma+n-1}$$

where γ is the parameter that controls how often new table is chosen compared to the previous tables and nr_i is the number of people at table i. On every table is a card with the name of another restaurant where customers on this table should go next, making this process nested and forming a tree. Now to make the method more related to our problem — each restaurant is a distribution over topics, each table is a topic and according to the nCRP, it is chosen whether a new topic is created for the sample or some previous one is used. [10]

Although the structure is unbounded and depends on the data, we need to make some assumptions for efficiency and to guide the result in the right direction. The assumptions lay in the four main parameters, which are in more detail explained in ref. [10]. Firstly, η for each level controls the shape of the topics drawn from the Dirichlet distribution. Also, these values are relative to each other, e.g., η = [1.0, 0.5, 0.25], leads to large proportion of words in the root distribution, fewer terms in the middle level, and even smaller mass of terms in the leaf distributions. Secondly, γ is the parameter in nCRP that was described in the previous paragraph that controls how often new tables are chosen. These two combined control the size of the tree. Since a small η leads to fewer words in each topic to have a

high probability, a model with small η and large γ usually generates a tree with more topics. Also, large γ means that documents are likely to choose new paths down the tree. Another distribution, named Griffiths-Engen-McCloskey [23], that controls the distribution over levels, has parameters m (ranging in [0, 1]) and $\pi(>0)$, where the first one controls the proportion of general words relative to specific words (for example, with $m = 0.5$, likely more words from samples will be assigned to higher up the tree) and π reflects how strictly the documents should follow these proportions. A large π results in more general trees. Settings these up correctly is the most difficult part of hLDA and is the deciding factor if the method is successful or not. In Table 2.3 are the parameters we got the best results for 21 sample data set.

Table 2.3 – Parameters used in hLDA for producing Figure 3.5.

Parameter	Value(s)
DEPTH	2
ETA (η)	[0.05, 0.05]
GAM (γ)	1
GEM_MEAN (m)	0.3
GEM_SCALE (π)	100

According to Blei et al., it is also important to know that trees are not as easy to identify using hLDA in data sets where there is lots of similarity between topics. Also, compared to ordinary hierarchical clustering, where the intermediate nodes are the summaries of the child nodes, with hLDA, intermediate nodes show the shared terms between the documents that have paths through them [10].

2.2.4 Neural networks

Artificial neural networks (NN) are a group of methodologies used in machine learning for solving prediction, classification and function approximation problems [24]. According to the article ref. [24], they are inspired from biological neural networks and like those, consist of simple computing nodes, called neurons. These neurons operate as non-linear summing units, which are joined by weighted

connections. These neurons form layers. The first layer is the input layer, where data are fed in, the last one is output layer where approximations to the target values are located and in between are one to many hidden layers, that control the capacity and complexity of the model.

Neural networks were developed already more than 50 years ago, but back then, computational resources were too limited and amounts of data were not sufficient enough [24]. Nowadays, these limitations are basically gone and there are networks with millions of neurons that are trained to solve different problems [24].

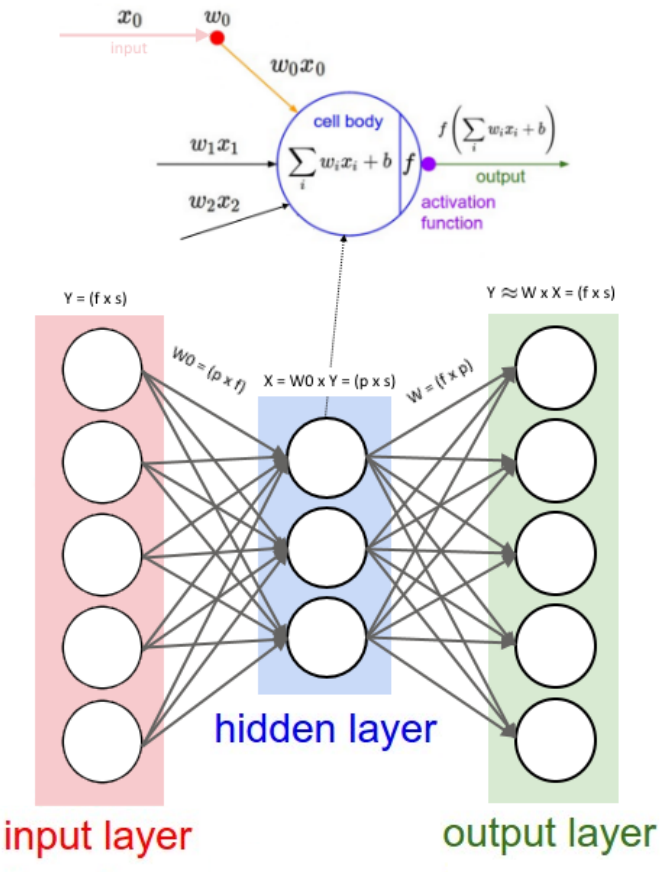

Figure 2.4 – A neural network, more specifically an autoencoder, with explanation of a neuron on the top, where x stands for input, w is weight, and f is activation function. In the middle are the shapes of layers with f meaning number of features, p number of signatures and s number of samples [25].

An example neural network can be seen in Figure 2.4. On the top is a precise explanation of what goes on inside a neuron: all the inputs x are multiplied with their respective weights w, they are summed together, a bias b is added, non-linear activation function f is applied to the sum and that is the output that will be sent to the next layer until the output layer is reached. We removed the bias parameter b for our experiments, because we want only direct interactions between weights and data, so it would look more like a matrix decomposition into two.

It is very important that the activation function is non-linear, since there are problems that cannot be solved with a linear capability [24]. For example in a square, if on the ends of one diagonal, there are black dots and on the other are red, these cannot be separated by a line, but can with certain weights and a curve [24]. Finding good activation functions is an active field of study. When previously sigmoidal function was the default, now ReLU and modifications of it, are more popular, because of faster convergence and positive activations [25].

During training, the data is input and the weights are adjusted using back-propagation and gradient descent such that the final output would be close to the desired output [24]. For example, if the problem would be classification, then the weights are adjusted so that the output is the correct class [24].

To explain some more according to the article ref. [24], training the networks starts off with randomly initialized weights (from uniform distribution in the implementation we used) and during forward propagation, using the input values and the weights, all neurons values are calculated at each level until the output layer is reached. There, a loss function (for example mean squared error) is calculated between the predicted result of the network and the true labels of the data. Then, during backpropagation, this error is propagated backwards through the network and the weights are updated according to how much error were their responsibility. Contribution to the total error of each weight is calculated using the gradient and the chain rule of the system and since the gradient points to the maxima of the system, subtracting some amount in the direction of the gradient leads to the finding the good local minimum on the error surface [24]. To reach faster to local optima, normalization is often used before every layer [26]. Since we do not have

specific labels in our data sets, we minimize Frobenius error of the difference of original data set and the reconstruction at each iteration.

In addition to the core principles of neural networks, there are additional features that have proven useful for the training result to generalize better to unseen data and with less overfitting. One of these methods is dropout, where in every iteration, a proportion of neurons is turned off [26]. Another way is partially corrupting the input, making it more noisy and in this way more generalized [27]. We also experiment with these additions.

2.2.4.1 Autoencoder

Autoencoder is a neural network with one constraint - the output layer has to have the same amount of neurons as the input [28]. A reconstruction of the original data is created, with one or several layers in the middle that have fewer neurons. With this, data is compressed in hidden layers, but still as much information as possible is stored in the weights, so the difference between the original and the output is minimal [28]. In this thesis, when referring to a neural network, it always means an autoencoder, because that is the kind of architecture used in our experiments. Also, we used the simplest setup with one hidden layer in between that determines the number of signatures, numerically 96 — number of signatures — 96.

We used the implementation Deeplearntoolbox in Matlab by R. B. Palm [29]. It had the necessary ReLU activation function for non-negative outputs, but was otherwise basic to modify and improve ourselves. Because we needed non-negativity also for weight matrix W and not only for X, we started by adding absolute values around W initialization and then during every iteration, applied ReLU to W also. This enforced our constraints to the method. After that, we needed to fix the parameters, which are explained in next section.

2.2.4.2 Bayesian optimization

One of the most difficult parts about training neural networks is the tuning of hyperparameters. Usually there are around five or even more of them and getting them right is "black art" requiring expert experience, rules of thumb, or sometimes brute force search of the whole grid of different combinations [30]. An automatic approach that can optimize the performance faster than brute force would be useful. One of the most widespread methods for this is the framework of Bayesian optimization that models the algorithm as a sample from a Gaussian process [30]. This way, it often takes many fewer runs of the algorithm to find near optimal results [30].

Table 2.4 – Different hyperparameter values used with Bayesian optimization for neural networks.

Learning rate	[1, 0.1, 0.01, 0.001, 0.0001, 0.00001]
RMSprop	[0, 1]
Decay rate	[0.5, 0.9, 0.99, 0.999, 0.9999, 1]
Scaling learning rate	[0.5, 0.9, 0.99, 0.999, 0.9999, 1]
Momentum	[0, 0.5, 0.9, 0.99, 0.999]

We iterated 300 times over the possible values in Table 2.4 to find the best combination of hyperparameters. According to Stanford course in ref. [31], these are the rules of thumb that are often used, but for specific problems a search over these possible values should always be made. Learning rate is the initial rate at which gradient descent step is made. RMSprop is an adaptive learning rate method that is often employed, which uses decay rate parameter to decrease learning rate iteratively. Scaling learning rate is a more general and robust parameter that also decreases learning rate. Momentum is another modification to the vanilla gradient descent update that can be used alone or with RMSprop together. These all play an important role in finding the optima of the search space. [31]

Table 2.5 – Parameters used for training a neural network.

Learning rate	0.0001
RMSprop	1
Decay rate	0.999
Scaling learning rate	0.9999
Momentum	0.9

The optimal hyperparameters we found can be seen in the Table 2.5. These were the result of Bayesian optimization and they are used in all of the succeeding experiments where neural networks are used.

3 Results

In this section, all the results achieved with different methods are presented and compared with each other. The main criteria for Alexandrov et al. [2] was reconstruction error and signature stability. These were the values that were important for us also. In addition, greater sparseness which would improve interpretation and quicker computations were also our goal. One of the methods was also suited for discovering hierarchies between signatures, which is biologically very important information and we expanded our research to that also. Frobenius norm was the measure of reconstruction error. The formula is $\sqrt{\mathrm{Tr}(A^\top \times A)}$, where A is the difference matrix between the original data matrix and the reconstructed matrix after applied methods.

First, signatures for organic data sets of 21 and 119 breast cancer samples and 506 samples from different cancer types are learned. After that, synthetic data sets of different noise levels are decomposed. The implementations with modifications we had to make and separate scripts for calling the methods are available on Github with the link available in Appendix B and also as ref. [20].

3.1 Validation

We began our research with checking, whether the results published by Alexandrov et al. were correct, stable and reproducible.

In 3.1 are the signatures that were extracted using NMF method. By comparing to the results in their article, Figure 4, the results are exactly the same, which was quite expected [2]. In their code, there are some parameters of how many iterations

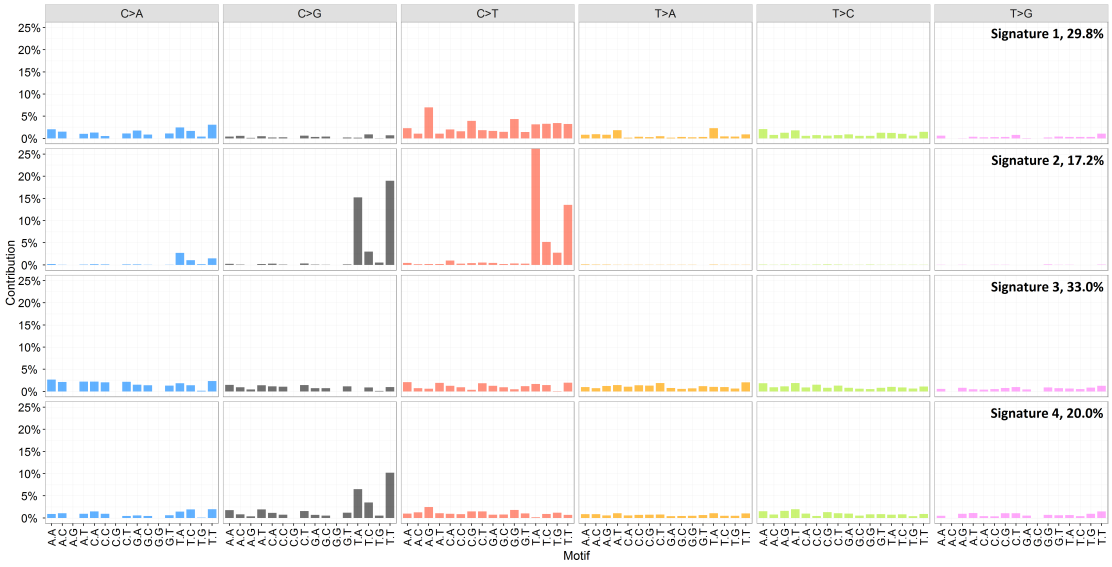

Figure 3.1 – 4 extracted mutational signatures using Alexandrov et al. method on the y-axis over 96 mutation types of the x-axis.

to perform and what cutoff values to use, but by using the default ones, we got the same results. The signatures are ordered according to ordering in COSMIC [13]. On the plot are also the average signature proportions over samples, meaning sample proportions are first normalized to one and then the mean is taken. As can be seen, actually the third signature is the most important, although it looks rather flat and indistinctive. The signatures are normalized to one, so each shows a distribution over mutation types. Usually, most of the mutation types are quite insignificant and only some higher peaks distinguish the signature. That is why the notion of sparsity is also important. The more completely zero mutations, the cleaner the signature might be.

In Table 3.1 is the comparison between different data sets using NMF method. As can be seen, the number of found signatures gets larger when having more data in the case of breast cancers, when there should only be certain underlying processes for a fixed cancer type. In the latter case with nine different cancer types combined, we assumed that more signatures would be found, but in reality only 7. After that, the stability started to decrease a bit more while having exactly the same reconstruction error. Also, the problem of larger matrices is that the

Table 3.1 – Results of different data sets for validation method of NMF.

Data set	21 breast cancer genomes	119 breast cancer genomes	506 cancer genomes
Signatures	4	6	7
Reconstruction error	505.7	1363.6	6128.732
Stability	0.94	0.987	0.995
Sparsity (%)	0.07	0.05	0.06
Time taken (h)	7	8	30

calculations take too much time (time taken value shows how many CPU core hours the calculation took). This could be improved with the use of GPU-s, but the method itself should be faster also. Results of all the possible number of signatures can be seen in Appendix A, Table 4.1.

3.2 Rectified Factor Networks

The first novel method we started investigating was RFN. It is newest of the methods here, published in June 2015 and it claims to produce sparser results while maintaining similar error rates as neural networks (that we will try later) and factor analysis methods [9]. Before achieving the results, some modifications to the algorithm had to be made that is described in the methods chapter and in the limitations section of discussion chapter.

At first we tried running the algorithm 100 times and then clustering the signatures into groups, just like Alexandrov et al. did, but the averaged signatures and distributions were a lot worse than the reconstruction error of NMF. Luckily, we also found the option of retraining previous models. With that we just iterated over 100 resampled data sets and always improved one model, which should semantically be very similar to just averaging over all the different models.

The results for three data sets can be seen in Table 3.2. The results are rather similar to NMF, but not quite. The reasons for picking these signature models

Table 3.2 – Results of different data sets for RFN.

Data set	21 breast cancer genomes	119 breast cancer genomes	506 cancer genomes
Signatures	4	5	6
Reconstruction error	506.7	1551.3	5917.6
Sparsity (%)	0.09	0.20	0.21
Time taken (h)	0.45	1.3	1.8

were due to the fact that next models had too much correlation or cosine similarity between signatures but also visual inspection had to be made, because sometimes even if the model with one more signature also seems to be suitable according to similarity measures, the new signature might be a combination of the others, which is a sign of overfitting. Another way of validation is by comparison to the verified signatures by COSMIC [13]. An important benefit of this method is the greater sparsity of larger data sets and also the huge speed improvement. Results of the whole signature range are available in Appendix A, Table 4.3.

Figure 3.2 – Distribution of 5 mutational signatures over 96 mutation types from 119 samples using RFN.

In Figure 3.2 can be seen the result of RFN for the medium sized data set.

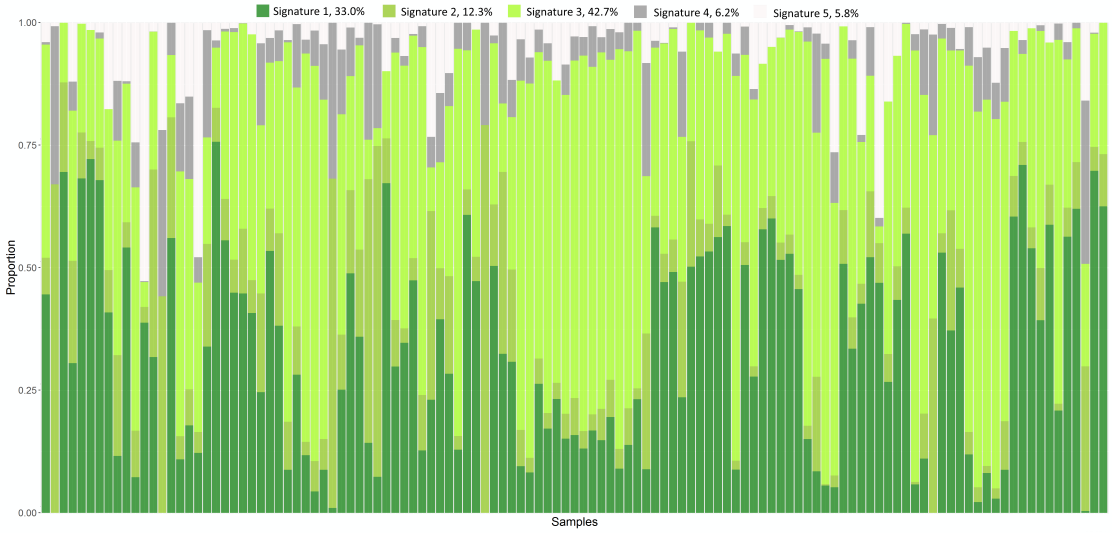

Figure 3.3 – Distribution of 5 mutational signatures over 119 samples using RFN.

Since the medium and smaller sized data sets are both breast cancer samples, it makes sense that similar signatures are found. Compared to the results shown in Figure 3.1 containing results of NMF for 21 samples, the first four signatures here are very similar to the ones there. Thanks to more data, the first, second and forth signature here are actually more accurate when compared with COSMIC signatures in ref. [13] where they are named as the first, second and thirteenth. The fifth signature here does not seem to be validated there as biologically significant. When comparing the average proportions of signatures, these are not as uniform as in Figure 3.1. Also, it would have made sense that the first four have roughly the same ratio as in the previous Figure, but the third signature has started to dominate and fourth is a lot less important. To check if this has something to do with the different methods, we compared results for 21 sample set, where both NMF and RFN were the best with four signatures and there the average signature proportions were very similar. So the difference is purely based on this larger data set. In this thesis we just evaluate the performance of different methods, but a research for deeper understanding of this difference could also be very important for biologists.

In Figure 3.3 is also visible the other side of the decomposition, where signatures

interact with samples. As can be seen, even though these are all breast cancer samples, the proportions of each signature vary quite a lot. Also, there are almost distinguishable two groups - one where there are lots of first and one where there are lots of third signature mutations.

Table 3.3 – Sparsity results of different data sets for RFN.

Data set	21 breast cancer genomes	119 breast cancer genomes	506 cancer genomes
Signatures	4	5	6
Original reconstruction error	506.7	1551.3	5917.6
New reconstruction error	518.3	1419.9	5940.4
Original Sparsity (%)	0.09	0.20	0.21
New Sparsity (%)	0.10	0.20	0.22
Decay rate	0.3	0.5	0.15

The location where we were able to enforce non-negativity of W matrix as described in the discussion chapter, was the parameter "L1_weightdecay". When the initial code only subtracted a portion from the weights to decay them towards zero, after our modifications, it would just set all negative elements to zero. To extrapolate this idea, it is possible to set all the values to zeros that are only slightly positive. This should increase the sparsity of the result. This is exactly what we did next. By increasing the decay 0.5 at a time and iterating with the chosen number of signatures, we trained a lot more models. When analyzing them, we allowed 20% more error and then compared the results that can be seen in Table 3.3. As can be seen, the results stayed quite the same. All the models that produced a lot more sparsity also had too large errors. Interestingly, the error actually improved for 119 sample data set. When training the model on 119 sample data set with zero sparsity again, to see if the worse performance was down to not iterating enough and our modifications, it turned out that the result with 5 signatures is correct and really is so much worse compared to bigger sparsity threshold. So to conclude, "L1_weightdecay" should be considered as another hyperparameter, that effects both sparsity and reconstruction error.

3.3 Topic modelling

Topic modelling is a method that tries to determine the underlying distributions of topics (for us, signatures) that most probably created the data. The most outstanding researcher in that field is Blei [21], who has also made C implementations of the methods that we are going to use [22].

3.3.1 Latent Dirichlet Allocation

LDA is the most basic topic modelling algorithm, where in theory each word in each document is generated according to underlying distributions. In practice, the reversed thing has to be done, meaning the distributions have to be found.

Training process started with Monte Carlo resampling as in NMF and RFN, to avoid overfitting. This way we created 50 new data sets. Next, we iterated over the possible number of signatures ranging from 2 to lower dimension to the matrix and for each of them, trained a model with the 50 data sets.

Table 3.4 – Results of different data sets for LDA.

Data set	21 breast cancer genomes	119 breast cancer genomes	506 cancer genomes
Signatures	4	6	8
Reconstruction error	505.7	1406.0	6128.732
Sparsity (%)(10^{-6} cutoff)	0.0(0.08)	0.0(0.13)	0.0(0.09)
Time taken (h)	5.0	8.5	15.3

The results of latent Dirichlet allocation can be seen in Table 3.4. For data set of 21 breast cancers, the resulting error is very similar to the one achieved using NMF. But it is important to note that while Frobenius error of decomposition using NMF hit a plateau, with LDA, the results kept decreasing a lot longer, almost reaching error of 400. The exact results can be seen in appendix A, Table

4.2. The reason why we did not choose a model with a better accuracy is that Pearson correlation and cosine similarity started to increase between signatures and it was mainly just breaking signatures up into smaller and more specific parts, but keeping compactness and similarity to the validated signatures from COSMIC [13] is more important. As can be seen from the Table, sparsity is always 0 with LDA. When comparing the weight matrices of NMF and LDA, there exist a lot lower valued weights for LDA even reaching 10^{-100}, while smallest weights for NMF are until 10^{-5}. That is why, it would make sense to set a cut off value for LDA, at 10^{-6} for example. Since the proportion counts in matrix X are not as large, these would influence Frobenius error by less than five points, which is acceptable. Sparsity values inside the brackets show the result when this was done. Time taken to complete the computations was significantly more than for RFN and similar to NMF. Also, we used slightly less (50) training sets than for NMF(80) or RFN(100).

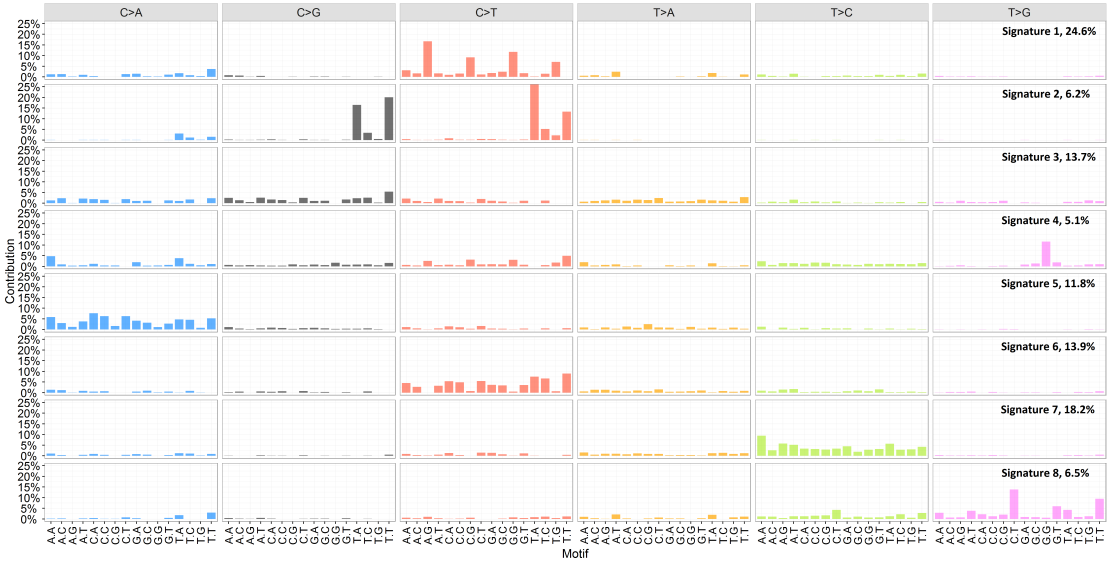

Figure 3.4 – 8 mutational signatures extracted with LDA for 506 sample data set with signatures on the y-axis and mutation types on the x-axis.

In Figure 3.4 is the result of LDA of the largest data set. When compared to the previous one, first three are found the same as in smaller data sets before. Fourth one was also found in middle sized data set. The last four are similar to the ones we found with other methods using this data set, but these are not validated in [13], but look to be forming certain clusters of mutation types. An interesting

observation regarding this data set is that even though we chose decomposition using 8 signatures as the best, because for 9 signatures, rather flat and indistinguishable signature was added, when we moved to 10 signatures, there appeared one that had quite a specific shape. So even if all the signatures are not good, it might be a good idea to check a larger decomposition also.

3.3.2 Hierarchical latent Dirichlet allocation

In this section we will give the results of a slightly different type of learning of mutational signatures, where we try to uncover hierarchical structure of the signatures. For this we used hierarchical latent Dirichlet allocation. We used again the implementation by D. Blei. Although in the article, he presented a totally unbounded tree structure for the method, in reality, still a parameter indicating the depth of the tree was used. Also other parameters that were explained in Methods had to be used.

The resulting hierarchy for 21 breast cancer samples data set can be seen in Figure 3.5. It is obvious that in the first level is only the root signature, but the number of signatures in the second level depends on the hyperparameters and latent variables underlying the data and in this case equals four. These hyperparameters are very data dependent and for example using the same parameters for 119 samples data set resulted in 118 signatures, which is totally wrong. What can be seen in the plot is that the first level signature is very similar to the second signature produced using "flat" models in Figure 3.1 with NMF and 3.2 using RFN and forth signature using LDA in 3.4. When looking at the proportion values in those figures, that signature does not seem to be the most common. That is because those value are the average signature proportions over samples, which are normalized. When looking at the original mutation counts, this signature dominates and for that reason it makes sense that it is the root of the hierarchy and in a sense is the most general. On the second level all the signatures correspond to the signatures resulting from other methods using 21 sample set. The biggest difference is with the last signature where there are too many mutations in the

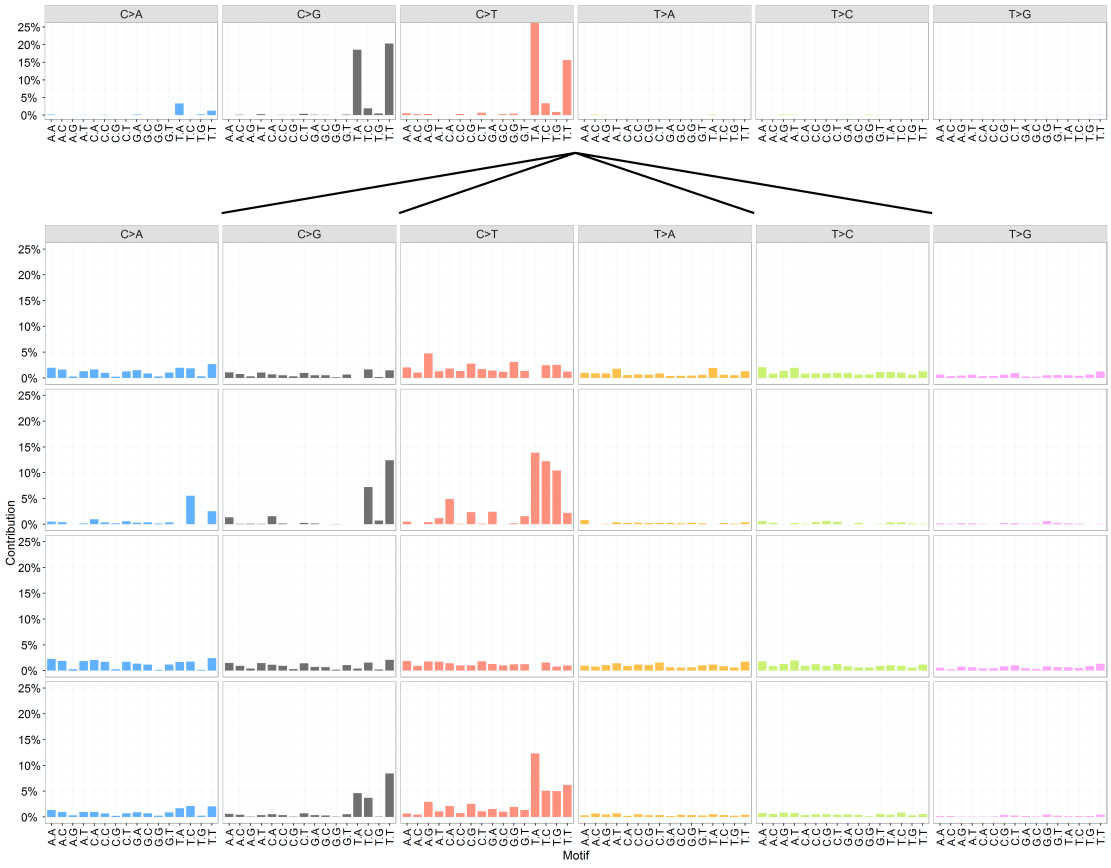

Figure 3.5 – Hierarchical representation of mutational signatures using hLDA for 21 breast cancer genomes with one signature as root and four signatures as leaves.

C > T section. Also there seems to be some bleeding across the signatures. It is important to note, that achieving this result took a lot of attempts with different parameter combinations and this was the simplest data set. That is why this method still needs to be improved. A further analysis on the limitations of this method is in the discussion chapter.

3.4 Neural networks

As the last method, we experimented with neural networks. Again, we had to use retraining one model with new Monte Carlo resamples added after some iterations.

We trained each model using 100 data sets where for the first one, we trained for 20 000 epochs for it to reach the right area of minima on the error surface and then for consequent ones for 6000 epochs such that the model would adjust for the new resampled data. When usually neural networks work best using normalized data that is centered around zero, we could not do that because we do not allow negative values in our problem and when just scaling the values to range [0, 1], the results were the same. So in the end we left it unnormalized. We also removed the bias, so all the terms would be interacting with the data.

In Table 3.5 are given the results for different data sets. Most notable are the very large sparsities. Almost half of the weight values in W are zeros. Our stopping criteria was once again cosine similarity, correlation and visual inspection. Inside the brackets are the biggest number of signatures that could have been the final results because the similarity and correlation were still below 0.7. Visual inspection and comparison with results of other methods were the reasons why we stopped earlier. Downside of this large sparsity is worse reconstruction error. This can be best seen with the smallest data set where the value is around 65 points worse and 119 sample set where it is even around 200 points worse, but since the error itself is also larger, the ratio is similar for the two data sets. With biggest data set, the error was similar with other methods.

Table 3.5 – Results of different data sets for neural networks.

Data set	21 breast cancer genomes	119 breast cancer genomes	506 cancer genomes
Signatures	4(6)	5(7)	7(14)
Reconstruction error	572.4(538.4)	1666.64(1605.1)	5760.1(4713.5)
Sparsity (%)	0.43(0.60)	0.54(0.67)	0.43(0.71)
Time taken (h)	0.6	1	1.75

In Figure 3.6 is displayed a different kind of decomposition that neural network does as compared to other methods. When increasing the number of signatures, then other methods often take a fraction of counts from almost all mutation types from some signatures and form a new signature out of those. Autoencoders on the other hand, only take a few mutation types and completely place it into a new

signature. In the upper part of the Figure is 4 signature decomposition and on the bottom is decomposition with 5 for 21 sample data set. For both of them, first three signatures are very similar and on the bottom, the forth and fifth combined make the original forth and very clearly there is a lot of sparsity and mutation types are almost separated between the two signatures.

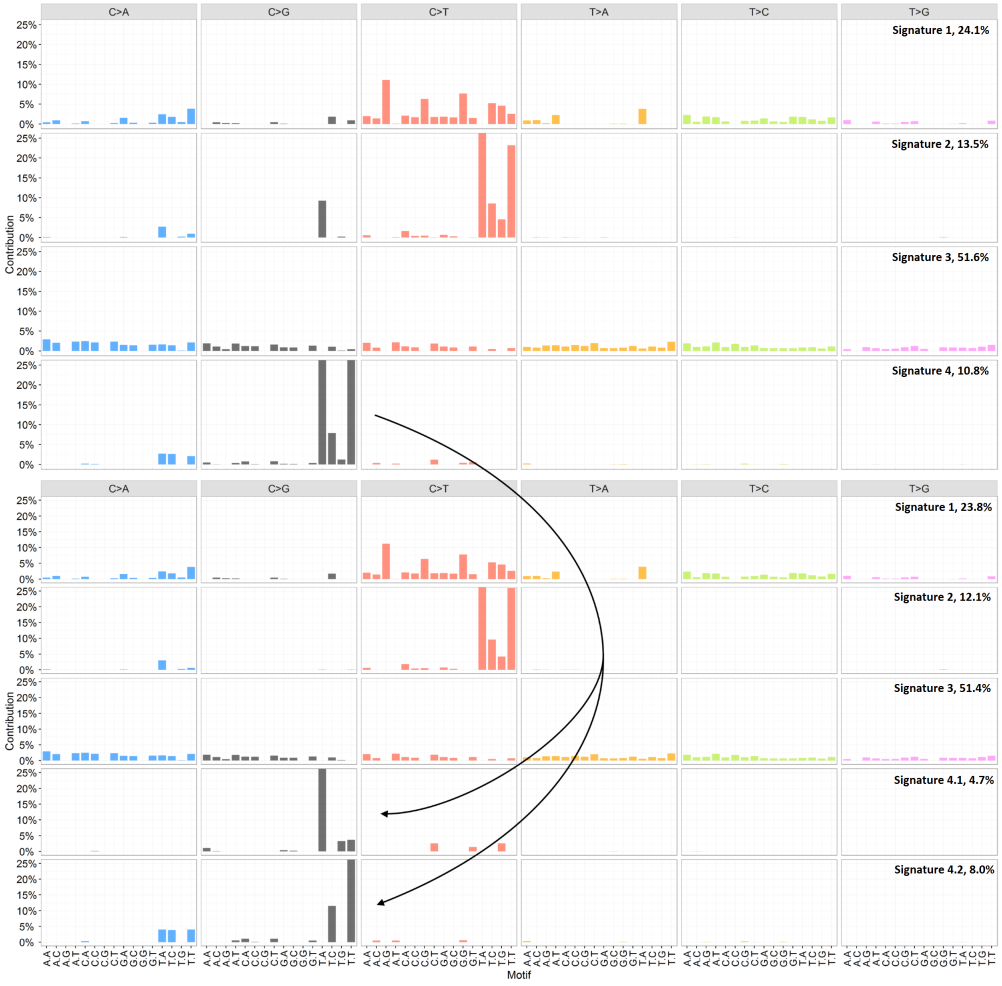

Figure 3.6 – 4 and 5 mutational signatures extracted using NN for 21 samples.

Neural networks also have extra features that in a lot of cases help to generalize the model better and make it more robust. Two of these features that we tried are explained in the Methods section. Firstly, dropout with the rate of 0.1% and 0.01% and also the denoising feature with 0.01% and 0.1% which adds noise while training. Unfortunately, although often improving the results, for us, these extras

gave a lot worse results, often producing several times higher Frobenius error and that is why we did not use these in the end.

3.5 Synthetic data

After we finished experimenting with organic data and were not satisfied with the precision and the variance between methods, we decided to make also synthetic data sets where we know the ground truth and thanks to that can verify which methods produce the best results.

In the next tables it can be seen how noise affects the overall decomposition error of all four methods. Firstly, in Table 3.6 are the results for data set that most closely resembles ground truth - the one where all mutations are sampled from signature distribution. In the first column is the method used, in the second one, the number of signatures extracted, third one contains Frobenius errors of two runs of each method. This way it can roughly be measured what is the variance inside the method. In the brackets is the difference between the first and second run. Forth column contains sparsity of resulting W matrix and inside the brackets is the result when 10^{-3} cutoff is applied to W. CPU core hours that took to calculate result is in the fifth column and in the last one is the stability metric for NMF. To remind the reader, 5 was the correct number of signatures.

As can be seen in the Table, when there is not enough signatures, the error is too great, but from 5 signatures and up, the result stays constant. With such an unnoisy data, it is easy to distinguish that 5 is the correct number even if it would not be known. This also corresponds to stability results of NMF. Results of neural network also flatten out starting with 5 signatures, but the value is 100 points larger. This is due to a lot greater sparsity that seems to be intrinsic to this method. We chose this specific cutoff value to calculate another kind of sparsity by considering that the mutation counts are usually below 100, and this way the 10^{-3} weight has little effect. The result is that we got a lot better sparsity for two methods with an increase in error usually less than 10 points. RFN and NN that

both use ReLUs have this kind of sparsity built in, so the results did not improve. The method to prefer according to this data set would probably be RFN, with a fast computation time, good error and considerable sparsity.

Table 3.6 – Results for all methods using synthetic data by sampling 2000 mutations.

Method	Nr	Frobenius error	Sparsity (Cutoff) (%)	CPU time(h)	Stability
NMF	3	1837.34 (+0.32)	0.04(0.11)	1.6	0.97
	4	1285.89 (-0.20)	0.04(0.19)	1.9	1.00
	5	972.34 (-0.16)	0.04(0.21)	2.0	1.00
	6	972.00 (+27.00)	0.04(0.22)	3.1	0.83
	7	979.42 (-8.30)	0.04(0.20)	4.0	0.63
RFN	3	1728.72 (+19.06)	0.05(0.05)	0.1	-
	4	1285.00 (+33.01)	0.09(0.09)	0.12	-
	5	981.09 (+8.17)	0.08(0.08)	0.12	-
	6	972.20 (+9.64)	0.12(0.12)	0.15	-
	7	972.24 (+2.50)	0.15(0.15)	0.20	-
LDA	3	1833.10 (+2.45)	0.00(0.16)	0.25	-
	4	1291.86 (-0.36)	0.00(0.27)	0.28	-
	5	985.65 (-0.88)	0.00(0.34)	0.6	-
	6	984.80 (+2.40)	0.00(0.38)	1.3	-
	7	984.64 (+1.47)	0.00(0.46)	1.5	-
NN	3	1853.53 (-3.03)	0.33(0.35)	0.30	-
	4	1343.37 (-2.12)	0.40(0.41)	0.40	-
	5	1090.58 (-4.45)	0.49(0.51)	0.40	-
	6	1082.46 (-25.39)	0.57(0.58)	0.45	-
	7	1058.22 (-13.67)	0.57(0.58)	0.43	-

Next we did the learning with all four methods on data set with medium noise level generated by sampling 200 times from the distributions and then multiplying by 10. These results can be seen in Table 3.7. With the added noise, error keeps decreasing even after the actual number has been reached. The methods are basically overfitting. Because the error keeps dropping (similarily to organic data) it is difficult to distinguish what should be the correct number of signatures if we would not know it beforehand. Stability of NMF gives the correct result that it should be 5 and also very interestingly the results of neural network start

fluctuating a lot more with above optimal number of signatures. This can be seen with all three data sets in Tables 3.6, 3.7 and 3.8 and should definitely be investigated more in the future to see, if this could be used as a metric or a heuristic for determining the number of signatures. Once again the result is lowest for RFN and now the other three methods are similar and perform a bit more poorly. It is very interesting that autoencoder is now fully equal with NMF and LDA while having 50% sparsity.

Table 3.7 – Results for all methods using synthetic data by sampling 200 mutations.

Method	Nr	Frobenius error	Sparsity (Cutoff) (%)	CPU time (h)	Stability
NMF	3	3459.15(-3.09)	0.04(0.15)	1.4	0.99
	4	3165.71 (-0.13)	0.04(0.26)	1.64	1.00
	5	3033.36 (-0.06)	0.04(0.29)	2.0	1.00
	6	3031.76 (+5.50)	0.04(0.26)	2.32	0.76
	7	3070.74 (-7.63)	0.04(0.22)	3.0	0.50
RFN	3	3333.85 (+19.40)	0.08(0.08)	0.08	-
	4	3101.35 (+16.91)	0.11(0.12)	0.10	-
	5	2964.98 (+5.75)	0.16(0.17)	0.08	-
	6	2902.88 (-0.71)	0.19(0.19)	0.10	-
	7	2851.15 (-2.25)	0.22(0.23)	0.12	-
LDA	3	3465.84 (-15.37)	0.00(0.15)	0.22	-
	4	3164.84 (+2.46)	0.00(0.25)	0.55	-
	5	3035.22 (+2.73)	0.00(0.32)	1.00	-
	6	3018.41 (-0.66)	0.00(0.41)	1.3	-
	7	2992.19 (+4.10)	0.00(0.43)	1.6	-
NN	3	3361.14 (-1.64)	0.26(0.29)	0.40	-
	4	3146.41 (-2.26)	0.42(0.44)	0.45	-
	5	3036.58 (-1.64)	0.50(0.51)	0.45	-
	6	3001.46 (-33.92)	0.52(0.54)	0.45	-
	7	2917.48 (+23.27)	0.56(0.57)	0.50	-

Lastly, in Table 3.8 are result for the noisiest data set. The smallest error with 5 signatures is now achieved by the neural network, which is quite surprising. So it could be said, that it handles noisy data the best. This is all achieved while still having one of the best sparsity levels. Also differences between two runs of

other methods are significantly larger than NN. Even with data set with this much noise, there still exists the tendency that until the correct number of signatures, the difference between the two runs of NN is small, and with excessive signatures, the difference gets a lot larger. With this data set, the stability calculation of NMF does not show as good results any more, which could also be related to the large difference in results of two runs. All in all, this data set was most difficult to decompose and NN and RFN got significantly lower Frobenius error, while NN had the more stable result of the two.

Table 3.8 – Results for all methods using synthetic data by sampling 20 mutations.

Method	Nr	Frobenius error	Sparsity (Cutoff) (%)	CPU time(h)	Stability
NMF	3	9932.62 (-1.83)	0.04(0.16)	1.8	0.66
	4	9690.95 (-0.00)	0.04(0.38)	2.0	0.90
	5	9596.02 (+23.57)	0.04(0.30)	2.4	0.72
	6	9558.16 (+45.48)	0.04(0.29)	2.3	0.53
	7	9552.96 (+69.47)	0.04(0.28)	2.4	0.35
RFN	3	9526.02 (-44.72)	0.10(0.10)	0.10	-
	4	9356.57 (-86.07)	0.28(0.29)	0.10	-
	5	9155.55 (-67.54)	0.28(0.29)	0.10	-
	6	8984.44 (-52.20)	0.27(0.29)	0.12	-
	7	8826.96 (-55.15)	0.31(0.32)	0.12	-
LDA	3	9929.10 (+44.228)	0.01(0.37)	0.25	-
	4	9698.81 (-115.24)	0.00(0.46)	0.25	-
	5	9660.90 (+39.94)	0.00(0.56)	0.25	-
	6	9479.32 (-80.30)	0.00(0.66)	0.30	-
	7	9383.19 (-179.24)	0.00(0.71)	0.50	-
NN	3	9546.48 (-0.50)	0.18(0.18)	0.35	-
	4	9345.27 (-1.17)	0.24(0.25)	0.40	-
	5	9139.79 (-0.88)	0.24(0.25)	0.42	-
	6	9032.01 (-56.28)	0.33(0.34)	0.40	-
	7	8835.30 (-29.23)	0.36(0.38)	0.47	-

We also tried comparing the errors of too few and too many signatures with the ground truth of 5 signatures. This means only compared the W matrices, not the whole reconstruction. For that we had two options - if we had too few

signatures, we either drop one from the ground truth or sum two signatures into one. By testing, it turned out, that summing is more accurate. For example with 4 signatures and neural networks, result of summing two signatures, the final result is 0.079 and by dropping it is 0.127. Since the error using 5 signatures is 0.106, it definitely is better to sum signatures since it conserves the mutational info better. Similar characteristics occurred in all of the methods.

Table 3.9 – Frobenius error between original and trained weight matrix W between different amounts of signatures for 2000 mutation synthetic data set.

Signatures	NMF	RFN	LDA	NN
3 vs 5	0.037	0.061	0.032	0.063
4 vs 5	0.044	0.048	0.042	0.079
5 vs 5	0.021	0.024	0.040	0.106
6 vs 5	0.080	0.098	0.045	0.203
7 vs 5	0.108	0.082	0.134	0.300

The results with all the methods using this approach on synthetic data set generated from 2000 sampled mutations can be seen in Table 3.9. This time, once again NN performs the poorest. Of course this is natural since these results have a strong relation to results in Table 3.6. That is why, a more unbiased and useful information from this table is that NMF and RFN produce similar results in the sense that the smallest error is with 5 signatures and both less and more signature make the result worse. This is orthogonal to LDA and NN, where methods keep overfitting and result with 3 signatures is the best. It is important to remember that when training with less signatures, ground truth is compressed, so in principle, even with just two or one signature, the result could improve further. This just does not have any meaning or interpretation. Overall the results are rather good and considering the difference between the ground truth and sampled data according to Table 2.1 is 0.005, result of around 0.02 for NMF and RFN is very acceptable. These were the experiments we conducted using synthetic data sets. In the next chapter is the discussion over the results we achieved.

4 Discussion

In this chapter we will give summary of the results and make an overall comparison of the methods. In addition, limitations of methods will be explained and directions what to do next in this field of research are suggested.

4.1 Summary and comparison

In this thesis we tested three new methods of rectified factor networks, topic modelling and neural networks for mutational signature decomposition in human cancer genomes. We compared them with the method of Alexandrov et al. [2] that was mainly based on non-negative matrix factorization. In the first part of the research, we used the same data sets as they did and we achieved similar results when validating their method and also analyzing new methods. Second part of the research was based on synthetic data sets where signature and sample distributions were known. In general, based on Frobenius error, all the methods performed quite equally. For noisier and more complicated data sets, neural networks and RFN seemed to be better, while NMF outperformed them by a small margin in simpler data sets. In addition, RFN and especially neural networks, were a lot better regarding sparsity of the output. Also, there seems to be more possibilities for improvement for RFN and NN, so in the future, they may give even better error result. Improvements will be explained in section 4.3.

While working with organic data sets, we decided to combine all the smaller data sets of different cancer types into one, with a lot more samples. According to COSMIC there are some signatures that are characteristic to all different cancer types [13]. Combining samples of different cancer types is a good idea for finding

these types of signatures, but after that, only the types that have the majority of samples and mutation counts seem to be identified. This is an aspect to consider.

Another general observation from the research is that when sparsity is of great importance, this could be achieved by setting a cutoff value for resulting weight matrix. Smaller values than the cutoff can be set to zeros, because they do not affect the result in a large way. When we did it with value 10^{-3}, it greatly increased the sparsity of NMF and LDA while not too negatively affecting the error result. In the next sections are summaries of all the different methods.

4.1.1 Validation

The original method based on NMF [2] performed well on simpler data sets and was a bit worse on other experiments we ran. Still, it is a good benchmark for comparing with other methods. The best quality of this method is the stability metric that by clustering signatures of different iterations and then averaging the clusters finds, what is the inside cluster distance and if it is small, the signature is stable. This was the best method to distinguish the correct number of signatures. In addition, the Frobenius norm was similar to other methods, confirming that it is a good and valid method to use for decomposition of mutation data.

4.1.2 Rectified Factor Networks

RFN is a factor analysis based method . To fit to the biological background, we had to add a non-negativity constraint for the weight matrix that broke the convergence properties. This means that the results were not as stable and a lot of iterations have to be performed to reach a near optimal result. Even with this extra work, it still finished quicker than NMF and produced a similar reconstruction error and for noisier data, according to synthetic data sets, the error was even better than that of NMF. Also the sparsity is better, so if a better metric for determining the correct number of signatures (the one of NMF did not perform well) would be

found and the convergence improved, this would definitely be a good method to use in the future.

4.1.3 Latent Dirichlet Allocation

LDA is a method of topic modelling and is based on finding the underlying distributions of signatures that most likely created the data [21]. Reconstruction error of it is very similar to RFN and NMF. On noisier data, it performs like NMF and is slightly worse than RFN. Because of that and the fact that it runs slower, it is slightly less favoured. An interesting quality is that it has absolutely no sparsity, but by setting a cutoff value, it can reach sparsity levels that are better than RFN and NMF.

4.1.4 Neural Networks

Frobenius error results of neural networks tended to be a bit worse for smaller and less noisy data sets than those of other methods. This was due to the inherent tendency to produce very sparse results. It produced around 30% more sparsity. It also has a different kind of way for decomposing data compared to other methods mentioned in the Results section. Another interesting phenomena of neural networks was discovered during training of synthetic data sets. When the number of signatures to decompose exceeded the true number from which data was generated, the variance between the results increased a lot. When before it was below 5, now it was up to 10 fold bigger. This could be a way to efficiently determine the true number of signatures and should be further analysed in the future. All in all, it could be said that neural networks act the most differently out of the methods, but still produce good results, especially for more difficult data sets and when sparsity is a desirable quality.

4.1.5 Hierarchical Latent Dirichlet Allocation

As a small extra research, we were interested to see, if it is possible to organize signatures into a hierarchical tree structure. For this we used hLDA. By tinkering with the parameters a lot, we were able to get an output that had similar signatures to "flat" models, but just organized into two levels. Our opinion is that there is a lot more room for further development in this area, since finding these hierarchies should be possible in a more natural way.

4.1.6 Performance on synthetic data sets

Using synthetic data sets with known ground truth gave us a lot more information about the methods. We were able to set up three data sets with suitable degrees of noise and difficulty, such that with the easiest data set, NMF produced the best results, but with noisier sets, especially RFN and neural network, but also in a smaller scale LDA emerged as better methods. When these two (or three) could be further developed and made more stable, they would be preferred over NMF in the future.

4.2 Limitations

In this section, a discussion over the limitations of different methods is held. For every method there is a separate subsection, but first let us have a look at the biggest problem regarding all the new methods tested. This is evaluating what is the correct number of signatures. The validation method worked very well in that regard and it is best seen from synthetic data sets where exactly until the correct number of signatures, stability is over 0.99 and when one extra is added, only that signature is assigned with a low stability rating and the first five still have it very high. Only when a lot of noise was added with the last set, this did not hold anymore. With other methods, we were not able to find a reliable metric

to evaluate correctly the right amount of signatures needed. This is a problem that should be investigated in the future. The metrics we used, combined with experience, visual verification and comparison with NMF, led to the decisions of how many signatures to select, but this kind of approach is not suitable when a lot more data sets have to be analyzed.

4.2.1 Validation

Although NMF does not suffer from the major problem of evaluating signatures, it still has its own drawbacks. Firstly, the implementation we used, took the longest of all the methods. Although there are lots of NMF implementations that only take a fraction of time, this one was more thorough to make sure the stability and correctness would be as good as possible. The parallel toolbox by Matlab that was used in the implementation and which should have made the execution a lot faster, also did not seem to work as well as hoped for and could partly be responsible for the slow performance.

Another limitation of NMF was that occasionally, it produced exceptionally large Frobenius errors and made peaks. In our data sets, this was with 119 sample set with 5 signatures. With one less, the error was 2378 and with six signatures, the result was 1363 and with 5 signatures it was 4355. Same thing with stability 0.99, 0.98 and between them was 0.76. So this is an important factor to consider when using this method. Adding more iterations and slightly modifying some parameters made the difference smaller, but did not completely solve the problem.

When analyzing synthetic data sets with 200 and 20 samplings and organic data set with 506 samples, NMF was not able to give as good error results as other methods. This is an important fact, because it is possible that it does not perform as well when it comes to more difficult data sets as other methods.

4.2.2 Rectified Factor Networks

The biggest problem with RFN was that the initial implementation did not impose non-negativity constraints to both resulting matrices. This was the toughest challenge with this method and we spent a considerable amount of time figuring out how to do this. In the article, there are proofs on the convergence of the method, but with our modification this does not hold anymore unfortunately. That is why we had to make more iterations and then roughly 10% of the times, the result was very similar to the ones using NMF or LDA, meaning it converged to the real minima. In other cases, the random initialization of W probably led it to the wrong direction. Using iterations seemed like a good solution, since it gave a correct result and because the overall method is a lot faster than the others, even if we repeated it 20 times, it still outperformed them. To give an idea of the speed, one iteration using the medium organic sized data set and 5 signatures took 30 seconds compared to 14 minutes of NMF and 22 minutes of LDA. During training, since after the first resampled data set, the weight matrix W was already with values leading to a good result, for all the succeeding data sets, all the iterations produced the same result, so it always converged and we just needed one iteration. Thanks to that, we did 200 iterations with first resampling to increase the chances of finding a good optima. All of this made the method quite robust, but figuring out how to keep the original convergence properties, would definitely make RFN even better and perhaps it would also be easier to determine the correct number of signatures. Other than that, there are no notable limitations of the method. It had rather good sparsity and the error results were always at or near the top.

4.2.3 Latent Dirichlet Allocation

LDA was similar to NMF in the sense that the training process took a considerable amount of time. With some exceptions it generally took 5 to 10 or even more times longer than RFN for example.

Another thing to note was that when running with the biggest data set of

506 samples, sometimes the alpha estimation blew up, giving garbage results. If this happened, we skipped that iteration and it luckily was not an often occurring event, so it is not a major limitation, but still it is important to keep it in mind when training models with LDA.

4.2.4 Hierarchical Latent Dirichlet Allocation

Finding hierarchical structure of signatures was done using hLDA. Although the article [10] stated that it is a nonparametric model and even though this kind of model also needs some assumptions in the form of parameters, it is strange that the depth of the tree had to be fixed. Also there were lots of other parameters that played a crucial role in the resulting model. When digging deep into the semantics of the parameters, they make sense, because somehow the general shape of the tree has to be specified. Unfortunately, empirical results show that it was not as easy as explained and the parameters were too sensitive. Slight change in some parts completely altered the tree. This was most evident with γ where having it too large results in lots of topics, but they are very similar, making the model useless. Probably even more thorough understanding of used Dirichlet, Beta and GEM distributions would give more information on what ranges the values could be and that would give a better idea of their effect, but right now, the results were unsatisfactory. In addition, similarly to LDA, it also had the drawback of lengthy computation time.

4.2.5 Neural Networks

The Frobenius reconstruction error result of neural network on easier data sets was significantly worse than those of other methods. This is due to the inherent favoring of sparse results. It would have made more sense that there is a parameter to control the level of sparsity, but instead it was automatically the case. It could also be down to the fact that we also added ReLU function to the weight matrix W and not only the activation matrix X. From that stems the issue that it is difficult

to determine the number of signatures. When for RFN and LDA it was possible to do it using cosine similarity, this was not possible since a lot of values are zeros and because of that, cosine similarity formula has a lot of multiplications where one multiplier is 0, negating the possible difference or similarity of signatures. Also, as mentioned in the results chapter for neural network, it has a different kind of decomposition, which makes comparison of signatures using Pearsons correlation coefficient also have little effect. In the future, other network architectures should be experimented with that perhaps could produce even better results while also having a good way to determine the correct number of signatures.

4.3 Future work

The field of mutational signatures in human cancer genomes is not yet researched too much and further work should be done, because finding underlying processes and reasons behind cancer might give important information for cancer discovery and treatment.

This thesis concentrated on finding and testing out other methods in addition to NMF for extracting the signatures. This is just a small part of all the possible things that could be researched. Closest topics to this thesis would be to test these and other methods even more thoroughly. For example RFN looked very promising, but finding a way to make it converge while having non-negativeness constraints on both decomposed matrices, would make the method a lot better. In addition to algorithmic improvements, these methods should also be analyzed with other kinds of biological features, such as bigger surroundings around mutations; in addition to substitutions, also look at insertions, deletions; consider both DNA strands separately; look at kataegis which is a hypermutation in a small region in a genome [1]; take other genome features like expression, methylation and histone marks into account. All in all, there are very many biological aspects to consider that could help to improve the precision or interpretation even more.

In addition to previously mentioned possible improvement of RFN, both topic

models and neural networks could also be further developed. For example author-topic models are being developed where there is an additional distribution over topics for different authors. Biologically speaking, this could be used to have separate signature distributions over different cancer types or different sites, where cancers were found. Also new information about which signatures are more general and which are more specific could be deducted with this method [21]. As for neural networks, a discovery using synthetic data was that number of signatures that is over the correct number has a lot bigger variance between different runs making it possible to perhaps determine this way what is the correct number. Initial tests on organic data did not prove this theory, but further investigation and perhaps improvement of the methodology, could still make this idea work. Also, we analysed one of the most basic architectures of neural networks, there are several others, that could work better for this kind of problem.

We analyzed rather briefly interactions between signatures and how they could be hierarchically related based on similarities between mutations in different samples. In addition to further developing this method, neural networks are often used to model second and higher order interactions. It would be very interesting to see whether modeling higher order interactions between features would be able to explain relationships between signatures and their underlying processes. For example if co-occurrence of two features leads to a bigger than linear count of mutations. This should also probably first be verified using a synthetic data set, but if it works there, seeing the results on organic data sets could be useful for geneticists and others. A method for this could be a factored 3-way restricted Boltzmann machine [32] or just having a lot more input features generated by multiplying original features to make interactions. Another way could be to make one hot vectors of length four out of the four nucleotides and then apply convolutional neural networks, something experimented with in article by Kelley et al. [5]. There seemed to be interactions between signatures when using too many signatures in our thesis also, which gives some motivation to try and experiment with these ideas more.

Very recently, on 2nd May 2016, an article, which used 560 breast cancer whole-

genome sequences, was published [33]. This means there is now an even larger data set of organic samples that could be used for learning mutational signatures with different methods.

As can be seen, there are still a lot of things that could be done. These were all the ideas that were quite related to this thesis. In addition to that, there are definitely even more aspects in this field of research related to both biology and methodology.

Conclusion

The aim of this thesis was to improve the methodology for learning mutational signatures from genome mutation data. It is an important problem, because it helps finding underlying processes that creates cancer and this aids in getting better information about cancers and also for cancer prevention and therapy. Currently, NMF has been used for learning DNA mutational signatures. We experimented with neural networks, topic modelling and rectified factor networks to see, if some of these produce better results.

We used organic data sets mainly based on breast cancer samples and also created synthetic data sets that quite closely resembled the organic data, but where we knew the ground truth. In addition to experimenting with different set ups on all the methods, our contribution was also the modification of RFN and neural networks algorithms where we set non-negativity constraints to both weights and strengths matrices. Furthermore, we experimented with hierarchical signature structures using hLDA.

Our results reveal a lot of information. On simpler data sets, such as 21 and 119 breast cancer sample data sets, NMF produces the best results, while RFN and LDA also perform very well. Neural networks perform significantly worse on these data sets and also on the least noisy synthetic data set. When data sets get more complicated, especially NMF and to some extent LDA start performing worse than neural networks and RFN. This is especially evident with synthetic data sets were we know the correct number of signatures and Frobenius error results for NMF and LDA are not as good. When comparing sparsity of the methods, then NN is the best, but RFN also produces sparse results. For the others, an additional cutoff step has to be performed to give results similar to RFN. Another important

aspect, computation time, is the smallest for RFN. Neural networks also run quite quickly, while other two take too much time.

The biggest advantage for NMF is precise stability functionality which is able to determine the correct number of signatures a lot better than other methods. This is a critical aspect that should be solved for other methods, because after that, RFN would be the preferred method for learning mutational signatures. Other future directions would be to also solve the convergence problem for RFN, trying out different neural network architectures, taking into consideration other biological aspects of genome data and also experimenting with higher order interactions between signatures. Further improvements are also necessary for hLDA, where currently a lot of different parameter combinations have to be experimented with to achieve a result that is interpretable.

Bibliography

[1] Ludmil B Alexandrov, Serena Nik-Zainal, David C Wedge, Samuel AJR Aparicio, Sam Behjati, et al. Signatures of mutational processes in human cancer. *Nature*, 500(7463):415–421, 2013.

[2] Ludmil B Alexandrov, Serena Nik-Zainal, David C Wedge, Peter J Campbell, and Michael R Stratton. Deciphering signatures of mutational processes operative in human cancer. *Cell reports*, 3(1):246–259, 2013.

[3] Bruce Alberts, Dennis Bray, Julian Lewis, Martin Raff, Keith Roberts, et al. Molecular biology of the cell (3rd edn). *Trends in Biochemical Sciences*, 20(5):210–210, 1995.

[4] George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, USA, 2002.

[5] David R Kelley, Jasper Snoek, and John Rinn. Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *bioRxiv*, page 028399, 2015.

[6] Alkes L Price, Nick J Patterson, Robert M Plenge, Michael E Weinblatt, Nancy A Shadick, and David Reich. Principal components analysis corrects for stratification in genome-wide association studies. *Nature genetics*, 38(8):904–909, 2006.

[7] Oliver Stegle, Leopold Parts, Matias Piipari, John Winn, and Richard Durbin. Using probabilistic estimation of expression residuals (peer) to obtain increased power and interpretability of gene expression analyses. *Nature protocols*, 7(3):500–507, 2012.

[8] Jeffrey T Leek and John D Storey. Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLoS Genet*, 3(9):e161, 2007.

[9] Djork-Arné Clevert, Andreas Mayr, Thomas Unterthiner, and Sepp Hochreiter. Rectified factor networks. In *Advances in Neural Information Processing Systems*, pages 1846–1854, 2015.

[10] David M Blei, Thomas L Griffiths, and Michael I Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7, 2010.

[11] Robert J Robbins. *Molecular Biology Fundamentals*. John Hopkins University, 1995.

[12] Oliver Branderberg, Zephaniah Dhlamini, Alessandra Sensi, Kakoli Ghosh, and Andrea Sonnino. *Introduction to Molecular Biology and Genetic Engineering*. Food and Agriculture Organization of the United Nations Rome, 2011.

[13] COSMIC. Signatures of Mutational Processes in Human Cancer. `http://cancer.sanger.ac.uk/cosmic/signatures`, 2015. [Online; accessed 2016.03.22].

[14] Ludmil B Alexandrov. WTSI Mutational Signature Framework. `http://www.mathworks.com/matlabcentral/fileexchange/38724-wtsi-mutational-signature-framework`, 2012. [Online; accessed 2016.04.06].

[15] HDP Health. How DNA Sequencing Works. `https://www.hdphealth.com/2015/07/07/how-dna-sequencing-works/`, 2015. [Online; accessed 2016.03.22].

[16] COSMIC. COSMIC Mutation Data Download. `http://cancer.sanger.ac.uk/cosmic/download`, 2015. [Online; accessed 2016.04.22].

[17] Daniel D Lee and Sebastian H Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[18] Sorana-Daniela Bolboaca and Lorentz Jäntschi. Pearson versus spearman, kendall's tau correlation analysis on structure-activity relationships of biologic active compounds. *Leonardo Journal of Sciences*, 5(9):179–200, 2006.

[19] Thomas Unterthiner. librfn - Rectified Factor Networks. `https://github.com/untom/librfn`, 2015. [Online; accessed 2016.04.06].

[20] Lauri Tammeveski. mutationalsignaturesNCSUT. `https://github.com/Tamme/mutationalsignaturesNCSUT`, 2015. [Online; accessed 2016.04.13].

[21] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[22] David M Blei. Topic Modeling Software. `http://www.cs.columbia.edu/~blei/topicmodeling_software.html`, 2015. [Online; accessed 2016.04.06].

[23] Jim Pitman et al. Combinatorial stochastic processes. *Technical Report 621, Dept. Statistics, UC Berkeley, 2002. Lecture notes for St. Flour course*, 2002.

[24] Judith E Dayhoff and James M DeLeo. Artificial neural networks. *Cancer*, 91(S8):1615–1635, 2001.

[25] Andrej Karpathy. CS231n Convolutional Neural Networks for Visual Recognition. Neural networks 1. `http://cs231n.github.io/neural-networks-1`, 2015. [Online; accessed 2016.04.13].

[26] Andrej Karpathy. CS231n Convolutional Neural Networks for Visual Recognition. Neural networks 2. `http://cs231n.github.io/neural-networks-2`, 2015. [Online; accessed 2016.04.13].

[27] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[28] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[29] Rasmus B Palm. DeepLearnToolbox. `https://github.com/rasmusbergpalm/DeepLearnToolbox`, 2015. [Online; accessed 2016.03.22].

[30] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[31] Andrej Karpathy. CS231n Convolutional Neural Networks for Visual Recognition. Neural networks 3. `http://cs231n.github.io/neural-networks-3`, 2015. [Online; accessed 2016.04.13].

[32] Alex Krizhevsky, Geoffrey E Hinton, and Ranzato Marc'Aurelio. Factored 3-way restricted boltzmann machines for modeling natural images. In *International conference on artificial intelligence and statistics*, pages 621–628, 2010.

[33] Sandro Morganella, Ludmil B Alexandrov, Dominik Glodzik, Xueqing Zou, Helen Davies, et al. The topography of mutational processes in breast cancer genomes. *Nature Communications*, 7, 2016.

Appendix A

Table 4.1 – Results of different data sets for NMF.

Nr	Stability 21	Error 21	Stability 119	Error 119	Stability 506	Error 506
1	0.999	13618.6	0.999	20264.3	0,999	43343.3
2	0.997	910.4	0.999	3023.9	0.741	33372.9
3	0.975	763.3	0.963	2572.6	0.999	9900.5
4	0.940	505.6	0.994	2378.5	0.999	8712.9
5	0.499	564.1	0.769	4355.3	0.998	7727.7
6	0.475	504.1	0.986	1363.5	0.998	6618.1
7	0.317	542.6	0.917	1319.0	0.995	6128.7
8	0.259	497.6	0.901	1268.1	0.944	6045.3
9	0.248	485.5	0.666	1504.6	0.959	6170.8
10	0.128	538.9	0.577	1473.0	0.740	8027.0
11	0.099	521.2	0.528	1380.7	0.857	6245.9
12	0.106	515.1	0.468	4969.6	0.854	6893.1
13	0.040	524.5	0.263	8893.3	0.788	8558.7
14	0.050	520.4	0.395	8160.1	0.789	7581.6
15	-0.01	9794.3	-	-	-	-

Table 4.2 – Results of different data sets using LDA.

Nr	Error 21	Error 119	Error 506
1	-	-	-
2	219001.0	3009.6	28468.1
3	740.9	2580.0	9697.0
4	507.9	2358.8	8465.8
5	476.0	2232.5	7453.6
6	479.3	1406.0	6297.8
7	459.1	1322.1	5802.9
8	430.4	1295.9	5722.0
9	419.2	1245.2	5434.6
10	432.1	1212.1	5052.4
11	407.7	1187.0	4873.8
12	409.2	1151.3	4663.5

Table 4.3 – Results of different data sets using RFN.

Nr	Error 21	Error 119	Error 506
1	20569.3	24742.8	36241.5
2	853.5	2766.4	31496.9
3	680.7	2150.5	9000.1
4	506.7	1829.2	7464.8
5	481.6	1551.3	6441
6	474.9	1403.6	5917.6
7	500	1377.3	5581.6
8	461.1	1283.2	5237.6
9	473.1	1288	5064.2
10	461.4	1259.8	5287.6
11	461.6	1599.3	5042.8
12	571.9	1437.5	4613.8
13	511.4	1315.5	5170.4
14	570.7	1722.2	4545.2
15	607.2	2009.8	-

Table 4.4 – Results of different data sets using neural networks.

Nr	Error 21	Sparsity 21	Error 119	Sparsity 119	Error 506	Sparsity 506
1	2814.57	0.07	5980.56	0.05	35748.56	0.06
2	880.29	0.21	2831.95	0.32	15089.42	0.14
3	759.34	0.44	2056.41	0.51	9141.33	0.24
4	572.41	0.43	1743.74	0.46	8585.94	0.4
5	564.13	0.55	1666.64	0.54	8220.49	0.46
6	546.79	0.6	1639.8	0.63	6826.08	0.44
7	533.66	0.65	1605.1	0.67	5760.13	0.43
8	526.9	0.7	1383.2	0.65	5375.65	0.54
9	516.57	0.66	1332.37	0.67	5276.85	0.58
10	521.28	0.74	1285.2	0.7	5176.66	0.62
11	521.04	0.76	1233.54	0.72	5129.68	0.65
12	510.21	0.74	1221.74	0.74	4842.04	0.67
13	498.84	0.74	1138.94	0.72	4829.37	0.71
14	519.41	0.77	1133.83	0.74	4713.54	0.71
15	519.37	0.8	1130.56	0.75	4671.46	0.72

Appendix B

Code repository with our code is available here:
https://github.com/Tamme/mutationalsignaturesNCSUT

Also an acknowledgement to the people who implemented NMF, RFN, LDA, hLDA and neural networks [14, 19, 22, 29].

Appendix C

Non-exclusive licence to reproduce thesis and make thesis public

I, **Lauri Tammeveski** (date of birth 14.10.1991),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Learning DNA mutational signatures using neural networks,

supervised by Raul Vicente Zafra, Leopold Parts, Tambet Matiisen and Ardi Tampuu

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **19.05.2016**