POLITECNICO DI MILANO
UNIVERSITY CAMPUS OF COMO

School of Industrial and Information Engineering

Master of Science in

Computer Engineering

Using deep autoencoders for gene clustering

Supervisor:

PROF. STEFANO CERI

Assistant Supervisors:

DR. ARIF CANAKOGLU

DR. MICHELE LEONE

Master Graduation Thesis of:

DANIELE SPACCAPELI

Candidate ID no. 860794

Academic Year
2016-2017

Ai miei genitori.

ABSTRACT

One of the biggest scientific achievement of the last few decades in Bioinformatics was the introduction of High-throughput sequencing (HTS) methods. We can now at a relatively low cost, reveal the presence and quantity of RNA in a biological sample at any given moment in time. This, plus the introduction of new analytical techniques, brought us insights in biological and medical research. Despite all these advances, given the complexity of human biology, a lot of molecular interactions are still unknown. Pathway is the name given to series of interactions among molecules in a cell that lead to a change. Some of the most common biological pathways are involved in the regulation of gene expression and play a vital role in studies of genomics. The aim of this thesis is to introduce a method that highlights new genes to target, in particular for cancer research. Since multiple pathways are dysfunctional in cancer, and cancer accumulates new mutations as it progresses, researchers require research tools to identify relevant cancer-related genes.

It is here presented a novel way of using deep neural networks to extract gene modules from gene expression data. The thesis pipeline is divided in two parts. The first focuses on the validation of the model by comparing the performances to other well-established methods. The second part instead is concerned with the extraction of gene modules, some data exploration, and the validation of the results. A Deep Autoencoder is trained with different types of cancer data taken from The Cancer Genome Atlas project. By leveraging the autoencoder's network topology, a ranking is calculated between the most relevant genes for each reduced dimension to create the modules. The effectiveness of the resulting gene sets is then tested, by checking for the over-representation of specific genes in each module with a method called enrichment analysis. Significant enriched terms related to cancer have been found in the extracted modules. The unknown genes in these modules can be highlighted as relevant target for functional analysis. The results are promising and are intended to primarily aid biologists and researchers in the investigation of new gene interactions and pathways.

SOMMARIO

Uno dei più grandi traguardi scientifici degli ultimi anni nel campo della Bioinformatica è stata l'introduzione di metodi di sequenziamento ad elevato parallelismo. Possiamo adesso, a relativo basso costo, rilevare la presenza e quantità di RNA in un campione biologico. Questo fatto, con l'introduzione di nuove tecniche di analisi, ha portato a numerose scoperte nella ricerca biologica e medica. Nonostante questi traguardi, data la complessità della biologia umana, molteplici interazioni molecolari sono ancora sconosciute. Via biologica è il nome dato a una serie di interazioni tra molecole in una cellula che portano a un determinato cambiamento. Alcune delle vie biologiche più comuni riguardano la regolazione dell'espressione genica e svolgono un ruolo essenziale negli studi di genomica. Lo scopo di questa tesi è illustrare un nuovo metodo che individui geni da analizzare, in particolare per la ricerca oncologica. Dato che molteplici vie biologiche sono disfunzionali nel cancro e che il cancro progredendo produce nuove mutazioni, i ricercatori hanno la necessità di acquisire strumenti per identificare geni rilevanti legati ad esso.

In questo lavoro è introdotta un'applicazione innovativa di reti neurali profonde per l'estrazione di gruppi di geni da dati di espressione genica. La tesi si divide in due parti. La prima si concentra sulla verifica della bontà del modello tramite il confronto delle prestazioni con altri metodi noti. La seconda invece riguarda l'estrazione dei moduli di geni, l'esplorazione dei dati, e la verifica dei risultati. Abbiamo addestrato un autoassociatore profondo con dati di differenti tipologie di cancro presi dal progetto "The Cancer Genome Atlas". Sfruttando la topologia della rete, troviamo un ordinamento fra i geni più importanti per ogni dimensione ridotta, i quali costituiscono un singolo modulo. L'efficacia di questi gruppi di geni (moduli) è poi verificata cercando una alta rappresentanza di specifici geni con un metodo chiamato 'analisi da arricchimento'. Nei moduli estratti sono stati trovati termini significativi legati al cancro. Gli altri geni all'interno dei moduli possono essere evidenziati come obiettivi importanti per una successiva analisi funzionale. I risultati sono promettenti e sono pensati in primo luogo per l'utilizzo da parte di biologi nella ricerca di nuove interazioni fra geni.

TABLE OF CONTENTS

List of figures **xiii**

List of tables **xv**

1 Introduction - Objective **1**
1.1 Thesis overview 4
1.2 Summary 6

2 Background **9**
2.1 Introduction to Bioinformatics 10
2.1.1 DNA sequencing 10
2.1.2 Genome annotation 12
2.1.3 Analysis of gene expression 12
2.2 RNA sequencing 13
2.2.1 Gene expression 14
2.3 TCGA 15
2.3.1 Dataset used in the Thesis 16
2.4 Neural Networks 18
2.4.1 Model description 19
2.4.2 How learning works 20
2.4.3 Deep Neural Networks 22
2.5 Autoencoders 23
2.5.1 Structure 24
2.6 Dimensionality reduction 26
2.6.1 Principal Component Analysis 27
2.6.2 Kernel-Principal Component Analysis 30
2.7 Tools used 31
2.8 Summary 32

3 Model **33**
3.1 Architecture . . . 33
3.1.1 Parameters explained . . . 35
3.2 Training . . . 37
3.2.1 Preprocessing . . . 38
3.2.2 Results . . . 38
3.3 Validation through Classification . . . 40
3.3.1 Support Vector Classifier . . . 42
3.3.2 Classification Results . . . 43
3.4 Comparison with other cancer types . . . 44
3.5 Summary . . . 45

4 Gene Module **47**
4.1 Definition . . . 47
4.2 Extraction . . . 48
4.3 Data Exploration . . . 51
4.3.1 Module size . . . 51
4.3.2 Properties . . . 52
4.4 Summary . . . 55

5 Enrichment Analysis **57**
5.1 Overview . . . 58
5.1.1 Enrichr . . . 59
5.2 Evaluation . . . 59
5.2.1 Breast Cancer . . . 61
5.2.2 Lung Cancer . . . 63
5.3 Summary . . . 65

6 Evaluation **67**
6.1 Stability . . . 67
6.1.1 Possible solutions . . . 69
6.2 Related works . . . 70
6.3 Summary . . . 72

7 Conclusions and future works **75**

References **79**

Appendix A Biology Handbook **87**

Appendix B Enrichment Results **97**

Appendix C Tools prooduced **105**

LIST OF FIGURES

1.1 Dimensionality reduction performed by the *encoder* 2
1.2 The *decoder* returns the vector back to the original input space 2
1.3 From the data to the extracted modules 3
1.4 Graphical representation of one-hot ranking with the decoder 4
1.5 Thesis Overview: Pipeline for the experiments 8

2.1 The evolution of the cost of DNA sequencing compared to Moore's Law. . . 12
2.2 Summary of the RNA-Seq process. 13
2.3 Central dogma of molecular biology: from genetic code to proteins. 14
2.4 ID tag breakdown for the samples in the TCGA project. 17
2.5 Example of a simple neural network having only one hidden layer. 19
2.6 Example of the gradient evolution over time on the surface error (contour line). 21
2.7 Example of a deep neural network with two hidden layers. 24
2.8 Layout of an architecture of a deep autoencoder 26
2.9 Example of a 2-dimensional manifold embedded in a three dimensions . . . 28
2.10 PCA of a multivariate Gaussian distribution. 30

3.1 Example of a "two" layer deep autoencoder. It has 3 hidden layers. 34
3.2 *One* Layer models. Training and test loss plot (above), Δ evolution (down). On the left `ReLU` and on the right `TanH`. 40
3.3 *Two* Layer models. Training and test loss plot (above), Δ evolution (down). On the left `ReLU` and on the right `TanH`. 41
3.4 *Four* Layer models. Training and test loss plot (above), Δ evolution (down). On the left `ReLU` and on the right `TanH`. 41
3.5 *Eight* Layer models. Training and test loss plot (above), Δ evolution (down). On the left `ReLU` and on the right `TanH`. 42
3.6 Separating hyperplanes examples. In this case H_3 is the maximum margin classifier. 43

4.1 Picture of a gene regulatory network. 48

6.1 Ordered variance for each of the 20000 genes in the data. Breast (left), Lung (right) 69

A.1 Model of the differences between DNA and RNA molecules 88
A.2 Focus on the Nucleotide and Base pairs 89
A.3 A visual overview of all the chromosomes 90
A.4 The gene expression process 91
A.5 An example of C-A polymorphism 94
A.6 How alternative splicing works 95

C.1 Example of a report produced by the developed support tool 106

LIST OF TABLES

2.1 The largest RNA-seq datasets in TCGA 16

3.1 Final loss between models with different activation functions. 39
3.2 Accuracies of the classifiers 44
3.3 Confusion Matrix - Autoencoder features 44
3.4 Training loss on different type of cancer 45

4.1 Table form of the histogram relative to module size 50 55
4.2 Table form of the histogram relative to module size 100 55

5.1 Example of an Enrichr term-value pair 60
5.2 Module #13 enrichment, against `GO Biological` 61
5.3 Module #228 enrichment, against `GO Biological` 62
5.4 Module #147 enrichment, against `GO Biological` 62
5.5 Module #115 enrichment, against `GO Biological` 62
5.6 Module #358 enrichment, against `GO Biological` 63
5.7 Module #370 enrichment, against `GO Biological` 63
5.8 Module #77 enrichment, against `GO Biological` 64
5.9 Module #92 enrichment, against `GO Biological` 64
5.10 Module #232 enrichment, against `GO Biological` 64
5.11 Module #233 enrichment, against `GO Biological` 65
5.12 Module #308 enrichment, against `GO Biological` 65
5.13 Module #331 enrichment, against `GO Molecular` 65

B.1 Module #100 enrichment, against `GO Biological` 97
B.2 Module #125 enrichment, against `GO Biological` 97
B.3 Module #6 enrichment, against `GO Cellular` 98
B.4 Module #21 enrichment, against `GO Biological` 98
B.5 Module #28 enrichment, against `GO Molecular` 98

B.6 Module #42 enrichment, against GO Molecular 99
B.7 Module #44 enrichment, against GO Biological 99
B.8 Module #100 enrichment, against GO Biological 99
B.9 Module #198 enrichment, against GO Molecular 99
B.10 Module #231 enrichment, against GO Cellular 100
B.11 Module #302 enrichment, against GO Biological 100
B.12 Module #32 enrichment, against GO Biological 100
B.13 Module #36 enrichment, against GO Biological 101
B.14 Module #37 enrichment, against GO Biological 101
B.15 Module #114 enrichment, against GO Biological 101
B.16 Module #144 enrichment, against GO Biological 101
B.17 Module #243 enrichment, against GO Molecular 102
B.18 Module #323 enrichment, against GO Biological 102
B.19 Module #56 enrichment, against GO Cellular 102
B.20 Module #104 enrichment, against GO Biological 103
B.21 Module #151 enrichment, against GO Biological 103
B.22 Module #386 enrichment, against GO Biological 103

CHAPTER 1

INTRODUCTION - OBJECTIVE

The discovery of new gene interactions is key to solve problems in medicine and biology. One of the main difficulties is that a gene interacts with multiple other genes at the same time. This is because a gene can participate in multiple cell activities. For example gene TP53, a well known tumor suppressor, regulates the transcription of proteins involved in different cellular activities [1][2]. Given the high complexity of molecular interactions and the large number of the elements at play, much is still left to be discovered. In biological research is important to understand what gene and for which function to focus on.

In this thesis the **aim** is to present an innovative method for gene grouping (or more specifically clustering) based on deep neural networks. To do this clustering, a neural network is built and a way to extract the groups (also called modules) is presented. Each step is validated from multiple points of view and at the end, the modules are evaluated from a biological perspective (with software tools, but also aided by a biologist).

The types of neural network used in this thesis are **autoencoders** [3], which are a class of neural networks used for efficient codings and dimensionality reduction [4]. The autoencoder is actually composed of two connected parts: an *encoder* (Fig. 1.1) and a *decoder* (Fig. 1.2). They are trained by approximating the identity function, meaning that the output is forced to be the same as the input. Where the dimensionality reduction comes at play is the middle layer of the network where its intermediate result (also known as code) is of a smaller size than the input. The encoder gives a way of going from the high dimensional space to the lower dimensional one ($n \to m$, $m \ll n$), effectively reducing it in size. The following part of the network, the decoder, performs the reverse function, returning again the original vectors to the higher dimensional input space. At the end of the training the weights of the network contain, in their numerical values and structure, a way of synthesizing the patterns in the data.

Each reduced dimension (m in total) is composed by different contribution of the original variables (n, genes). To each of reduced dimensions correspond a cluster of the input variables (genes) that are most influencing over the them. These clusters, or gene modules, are intended to represent specific patterns of the original data. The idea is that it is possible to group the genes that collaborate to form a single reduced dimension by ranking their individual contribution.

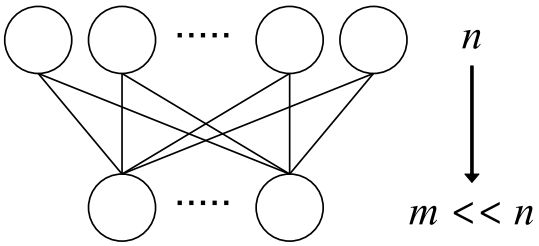


Fig. 1.1 Dimensionality reduction performed by the *encoder*

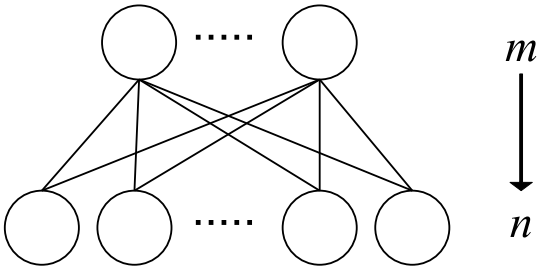


Fig. 1.2 The *decoder* returns the vector back to the original input space

There are three main sections this thesis is divided into:

1. The autoencoder model is constructed and trained with samples of gene expression of cancer tissues coming from the TCGA project [5]. Data from healthy and cancerous samples is used at this stage. The features extracted from the encoder are validated against other well known methods for feature extraction. Our model features perform on the same level as the other methods in their ability to discern cancer and healthy samples in a classification scenario.

2. The autoencoder model is then retrained with only cancerous data to better capture tumor activity. In order to understand the contribution of each gene to each reduced dimension, **one-hot vectors** (vectors with only one non-zero element) are used as inputs of the decoder. These vectors activate only a specific dimension and not the others. The dimensions of the output of the decoder are again the genes. By reading the

decoder's output we can understand how much each genes affect the said dimension (Fig. 1.4) and construct a ranking from it. The top genes for each reduced dimension compose the final gene module relative to that specific dimension.

The main **research questions** here are:

> *Can a deep autoencoder group genes which are involved in a shared biological function? Or simply, given a clustering method built upon an autoencoder trained on cancer data, is it possible to get useful gene sets? Furthermore, will these functions be related to the specific cancer used to train the model?*

3. To answer these questions the extracted modules are checked for disproportionately high presence of genes that are known to participate in a common activity. The method used is called enrichment analysis. The whole process, from training to enrichment, is depicted in Fig. 1.3. The enrichment analysis returns for each gene sets (modules) a list of terms, that represent a specific biological activity, and their associated P-values, which indicate their statistical significance. Most of our modules have significant terms (small P-values) associated with them.

   These results validate the ability of the deep autoencoder to group related genes. The last step entails answering our final question: do the modules relate to activities connected to the specific data used to train the model (cancer)? For this step a manual biological evaluation of the terms is necessary: that is linking the terms to cancer with domain-specific knowledge. Hence a subset of modules, having the most significant terms, is chosen and studied. The findings resulted in numerous modules connected to cancerous activities with a subset of them strictly related to the type of cancer used during the training the model.

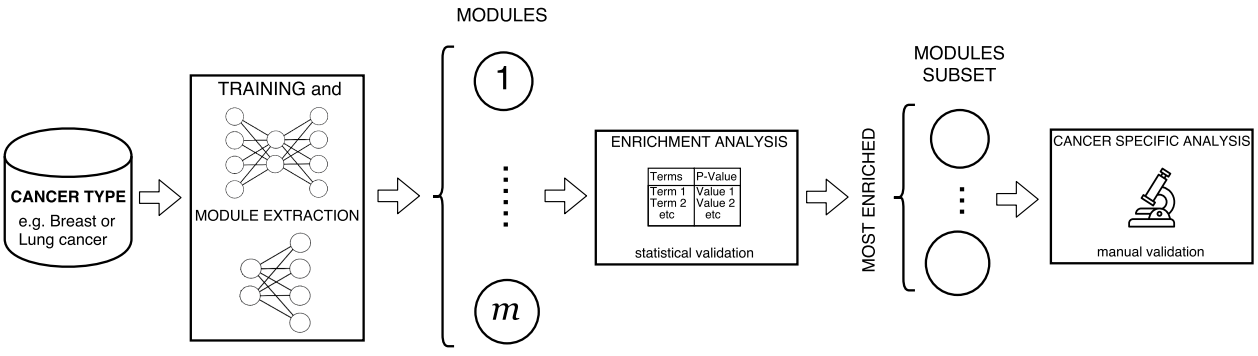


Fig. 1.3 From the data to the extracted modules

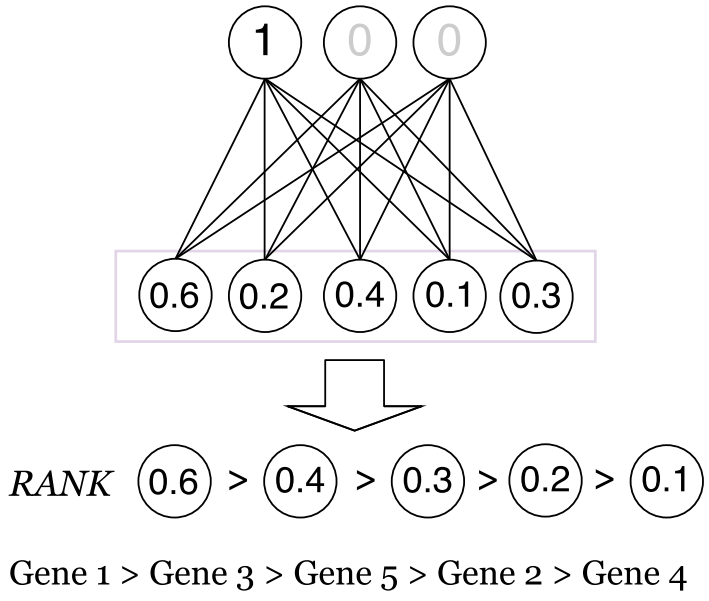


Fig. 1.4 Graphical representation of one-hot ranking with the decoder

1.1 Thesis overview

The process explained above is here presented more in detail. The pipeline for the experiments done in this thesis is divided in **three parts**/activities. To guide the reader through all the various steps a schema of the pipeline was created. The schema can be seen in Figure 1.5, with each part tagged with its respective number. The first part is about the validation of the model and model parameters, the second part explains how gene sets are extracted from the network while the third and final part validates the goodness of the modules from a biological perspective. Now a more detailed explanation.

The dataset, upon which the models are built, comes from the TCGA project which collects and analyzes information relative to multiple types of cancer. The data used, in particular, is a collection of samples from cancer patients represented as expression levels coming from RNA-seq experiments. Each dimension of the vectors (samples) in the data represents the expression level of a specific gene at a specific time.

A **preprocessing step** was performed in which the data was **split** in two. The TCGA RNA-seq dataset contains informations relative not only to cancerous cells, but it also includes control and healthy cell samples for each tissue. For each cancer type it has a limited amount of samples overall e.g. breast cancer has around 1200 instances with ~ 1000 of cancerous type. In the model validation part we use all the data available (healthy plus cancerous) to tune the parameters of the network, both for the scarcity constraints and because of the underlying assumption that all gene expression data share *high level features* (biggest

discriminator). What is of particular interest is identifying modules connected to cancerous activities and not so much in other (normal) activities. For this reason the autoencoder is retrained with the data that includes *only* samples taken from cancerous cells. From this new autoencoder model the gene modules are extracted. The reader could imagine the healthy samples as acting as a sort of noise to the true objective of finding tumor-gene interactions. Healthy cells won't present cancerous patterns.

Following the pipeline, in the **first part** ① (the leftmost section of the picture under "training" and "model validation") the aim is to build an autoencoder model. This is done by finding the appropriate parameters, such as: *number of layers* and *activation functions*. The number of layers in particular affects the minimization of the cost function. The cost function measures the difference between the input and the output, in our case the difference between the input and the output. What is of interest in this part is the quality of the reduced features, this is because the features are directly related to the gene sets that are going to be extracted (a module is extracted from each of them). The goodness of the encoded features of the model is validated by comparing them to ones found by using a PCA and a k-PCA in a classification scenario. The two models are, of course, trained on the same data. The reason for the use of those methods for comparison is because of their popularity in the literature and known results to be targeted. The extracted features from each of these three models are fed to a Support Vector Classifier that tries to predict whether a sample comes from a cancer cell or not, successively the results are evaluated. Similar results are expected across all the methods and if that is going to be the case it would confirm the ability of the autoencoder to have *learned meaningful features* (they would at least preserve the ability to discern cancer to non-cancer cells).

In the **second part** ② of the experiments a new autoencoder is trained using the same "validated" parameters, but this time using strictly cancer samples. In the diagram this part is represented by the upper part of the right column. This new training is done because, as written above, the objective is for the model to capture first and foremost processes related to tumor activities. By using cancer samples exclusively the attempt is to try to target those specific functions without noising the model with healthy ones. Only the decoder part of the newly trained network is used for gene module extraction. The decoder is a generative model, it can be used to reconstruct a code into its original form. The idea underlying gene module extraction is of using one-hot vectors to get the corresponding gene expression profile vector to a specific dimension. This way it is possible to isolate the individual contribution of each gene (corresponding to one original input dimension) to each reduced feature. After having done this for each of the reduced dimension, it's possible to rank individual gene contribution (Fig. 1.4). The *top ranked* genes for each outputted vector for the decoder are going to

constitute the gene modules and are the ones that represent better the specific dimension. In this part an analysis is done over the genes present across all the modules. Their properties are visualized to evaluate the appropriate following next steps to take. A promising result over all the modules, for example, would be an high frequency of gene involved with cancer activities. Furthermore it would be desirable to have an high degree of separation between each module (cluster). If the groups have many genes in common it wouldn't be a good indication for the method's ability do discern *different* functions.

In the **third and final part** ③ the module are validated from a biological standpoint (Fig. 1.3). What is interesting here is to discover whether the extracted modules have an higher representation of genes that are known to participate in a common activity. These already known gene sets are taken from different knowledge bases and are compiled with genes that were discovered to be involved in the same/similar function. If our modules have a significant presence of these genes they could be investigated for their relationship to that specific activity. Gene set enrichment analysis is the name for the class of methods built for this specific purpose. In the thesis' case Enrichr [6], a software tools with a Web API, is used. The enrichment is done to each module to verify the quality of the grouping. The results from Enrichr give us an indication of how "*valuable*" a module is, and which activities it is involved with, but they do not give an immediate indication if those activities (terms) are related to cancer. For that purpose, a manual step of verification is necessary. That is: checking each modules for what the enriched terms actually do (if there is literature that links them to cancer) and if they're related between each other. This part will conclude with some consideration regarding the method's strengths and weaknesses, especially on the stability of the extracted modules.

After the previous section, some software tools are produced and presented. These tools are built to further support third parties' work (such as biologist) in the exploration/investigation of the genes in the extracted modules. For example, one could be interested in knowing whether a particular gene is present in some module or which are the genes which are differentially expressed in the training data and by how much. The thesis will finish with some concluding remarks and mentions of possible future developments.

1.2 Summary

In this introductory chapter the objective and main research questions have been discussed. The pipeline for the experiments (Fig. 1.5) done during the thesis work is also presented in detail.

The thesis is structured as follows:

- *Chapter 2*, lays the ground for the concepts and the theory of the methods used in later chapters. This overview concerns both Bioinformatics and biological aspects and the background of machine learning related material.
- *Chapter 3*, presents the used model, its training and the validation of the results by comparing it to other well-known tools.
- *Chapter 4*, shows how to extract the modules from the model, by leveraging the generative part of the neural network (decoder). Also some data exploration is done to check from desired properties from a macro-level perspective.
- In *Chapter 5*, the grouping are validated with enrichment analysis. Enrichment and the software tool used are discussed. The latter part of the chapter is dedicated to the in-depth manual analysis of the most significant results.
- In *Chapter 6*, the method is evaluated with a focus on the stability of the results and a review of similar methods in literature is presented.
- In *Chapter 7*, the thesis is wrapped-up with a quick summary and some considerations for future works.

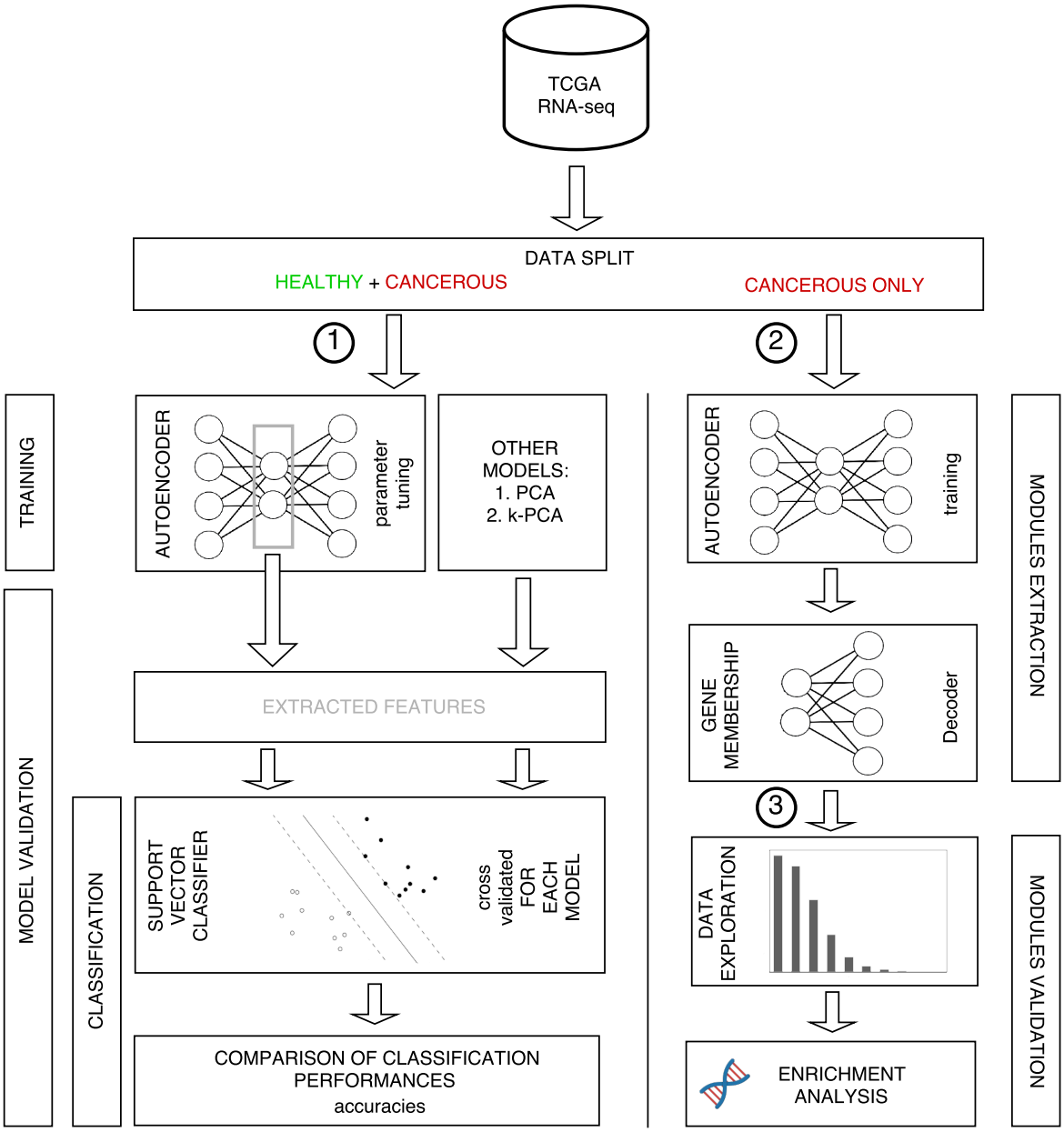


Fig. 1.5 Thesis Overview: Pipeline for the experiments

CHAPTER 2

BACKGROUND

Bioinformatics is a field that combines the more classical data science approaches to biology, hence it requires at the very least a brief overview of its main ideas. It's very important for any sound engineering job to have a good understanding of the underlying domain-specific *knowledge.*
For this reason ***Appendix A*** *was created as a glossary of the most used terms related to biology. A star ($\star$) is put next to the first appearance in the text of such terms.*

In this chapter the readers are presented to the necessary background material. Several main concepts are explained which constitute the basis of the thesis. The chapter is structured in two parts with the **first part** devoted to the field of Bioinformatics. DNA sequencing is described with its current uses. Gene expression and its profiling is also introduced, with the last section of this first part being devoted to the presentation of the dataset over which the used models are built upon and the project that produced it.

In the **second part** the readers are exposed to Machine Learning related concepts. The model of neural networks is introduced, from the mathematical constructs to different possible typologies. The next part is devoted to autoencoders, the centerpiece to understand this thesis' foundation. Autoencoders are a type of neural networks used for efficient codings and dimensionality reduction, hence an introduction to dimensionality reduction is also given. In particular, a focus is put on the comparison between different models (PCA and k-PCA) that try to achieve similar results. These two models will be then used in the next chapter to assess the training quality of the final autoencoder. In the last part of the chapter, a brief mention of the tools and library used is also included.

Part I: Bioinformatics

2.1 Introduction to Bioinformatics

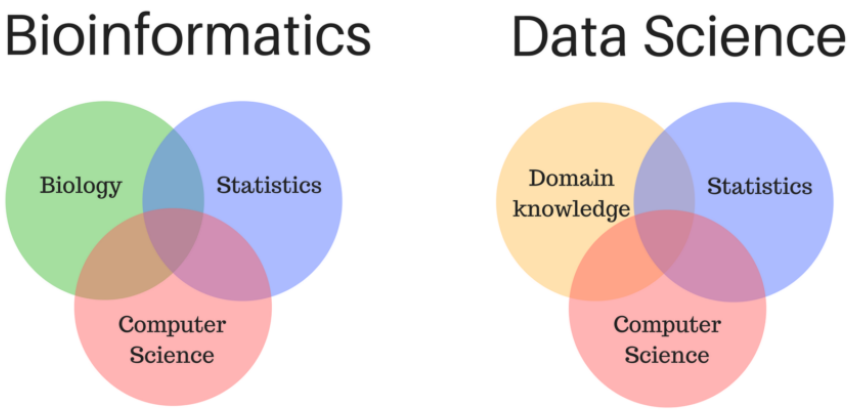
Over the past few decades, and more specifically starting in the mid-1990s, driven by the Human Genome Project and by advances in **DNA*** sequencing technology, the field of bioinformatics experienced explosive growth. Rapid developments in genomics and other molecular research technologies have combined with developments in computer science to produce new approaches to get a myriad of biological insights. Bioinformatics is the name given to these approaches and it is, in fact, a multi-disciplinary field that combines Computer Science, Biology, Mathematics and Engineering to analyze and interpret biological data. In other words, it can be seen as Data Science applied to the domain knowledge of biology. The size and the complexity of biological data, raises many challenges for the scientific community.

The primary activities of bioinformatics revolve around the creation and betterment of databases, algorithms, statistical techniques, and theories to solve problems arising from the management and analysis of biological data, particularly DNA, **RNA***, and protein sequences. The main goal being the increase in our understanding of biological processes. What differentiates bioinformatics from previous approaches is its focus on developing and applying computationally intensive techniques to achieve the aforementioned goal. The current primary research efforts in the field include: sequence alignment, **gene*** finding (locate a gene within a sequence), drug discovery, protein structure alignment, protein structure/function prediction, prediction of gene expression and others. Let's now look at some of these areas in depth.

2.1.1 DNA sequencing

Thanks to new sequencing methods, the cost of sequencing the genome from a biological sample is now relatively small in terms of both money and time. As mentioned, the introduction of rapid sequencing methods has accelerated biological and medical research, but before any analysis can happen there is the need to extract data from the biological samples.

DNA sequencing is the process of determining the arrangement of **nucleotides*** within a DNA strand (guanine, cytosine, adenine, thymine in humans). Sequencing can be used to determine the sequence of individual genes, larger genetic regions or full chromosomes of any organism. Despite all the latest advances, the process still remains difficult, as the extracted data can be noisy or contain missing data (maximum accuracy is of 99%). A



major research area in computational biology involves developing statistical tools to separate signal from noise in high-throughput gene expression studies. The main applications of this technology nowadays are:

- *Medicine*, e.g. to determine the risk of genetic diseases.
- *Forensics*, with profiling used extensively in investigations. It's highly unlikely that two people have the exact same DNA. Other uses include paternity testing.
- *Molecular biology*, to study the genomes and the proteins therein encoded. It's especially useful in identifying gene-phenotype association and drug targets.
- *Evolutionary biology*, to study how different organisms are related and (co-)evolved.
- *Meta-genomics*, which is the identification of organism in certain environments. Particularly useful in the study of ecology, epidemiology, microbiology.

The high demand for sequencing at a low-cost and major research efforts such as the Human Genome Project have driven the development of high-throughput sequencing technologies. This methods parallelize the sequencing process by producing thousands or millions of sequences concurrently. High-throughput (or next-generation) sequencing technologies managed to lower the cost of DNA sequencing beyond what was possible with normal methods. The cost of sequencing the whole genome of a human is going down drastically to few thousands of dollar (from *millions*). Let's compare this trend in the following graph depicting Moore's law (Figure 2.1), an observation which illustrates the trend in computer

hardware of doubling the power every two years. Every progress as fast as Moore's Law is considered to be doing extraordinarily well.

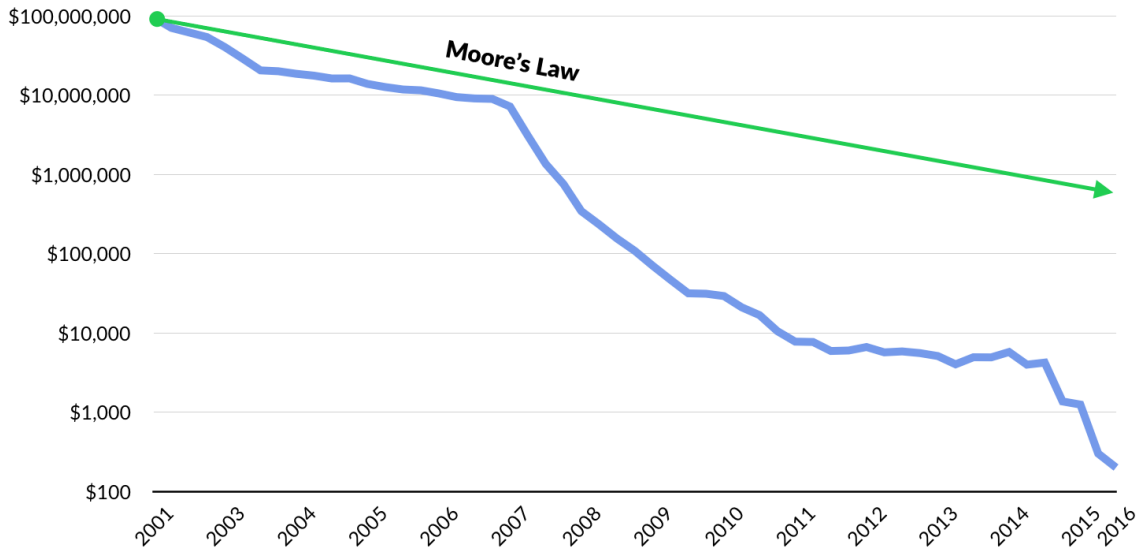


Fig. 2.1 The evolution of the cost of DNA sequencing compared to Moore's Law.

2.1.2 Genome annotation

DNA or genome annotation is the process of identifying the genes (coding regions) and their functions in a DNA sequence. It help us make sense of the **genome**⋆. This process needs to be automated since the genomes of most animals are too big for manual annotation (there are 6,469.66 Mbp, Mega base-pairs for humans). Annotations are made possible by the fact that genes have recognizable start and stop regions (start and stop codons). After the end of the Human Genome Project, a new project started by the National Human Genome Research Institute: the ENCODE project. The project is a collaborative effort for the collection of the functional elements of the human genome.

2.1.3 Analysis of gene expression

The expression of genes can be assessed by measuring **mRNA**⋆ levels. Studies of gene expression are often used to determine the genes implicated in a particular disorder: one might compare microarray data (results of microarray experiments) from cancerous cells to data from non-cancerous cells to determine the genes that are up-regulated and down-regulated (more or less expressed than usual) in a particular type of **cancer**⋆. One can then

apply, for example clustering algorithms to that expression data, to determine which genes are co-expressed i.e. upstream regions can be searched for over-represented regulatory elements.

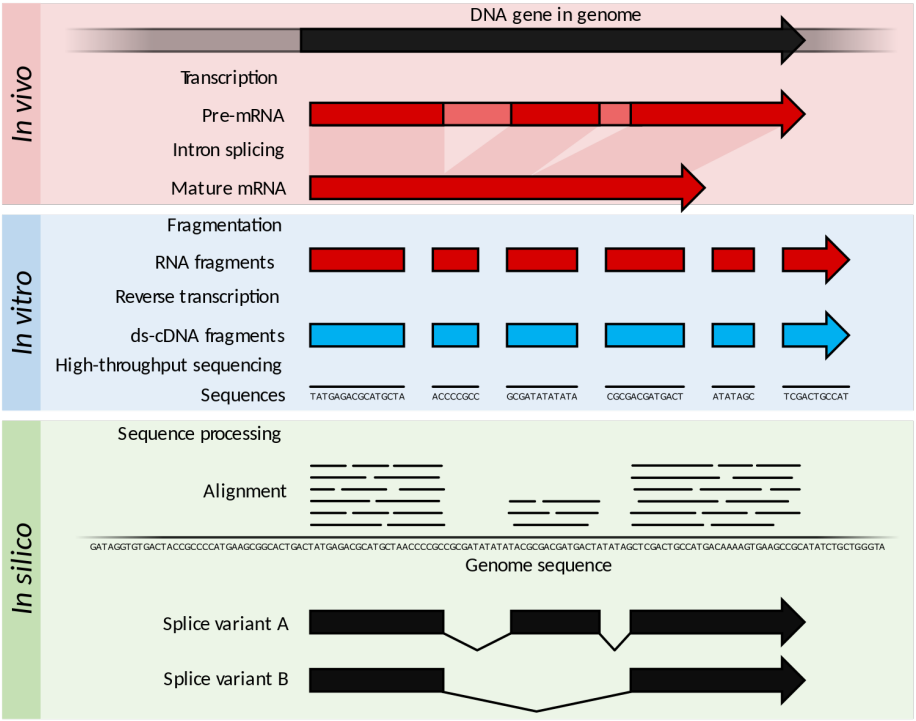


Fig. 2.2 Summary of the RNA-Seq process.

2.2 RNA sequencing

What will be of particular interest for this thesis is RNA sequencing and gene expression. The former is also called Whole **transcriptome**⋆ shotgun sequencing (WTSS, RNA sequencing or RNA-Seq)and it is the new technology that is used for gene expression studies, replacing hybridization-based microarrays. RNA sequencing uses high-throughput sequencing to reveal the presence and quantity of RNA in a biological sample or determine **exon/intron**⋆ boundaries (see Figure 2.2). The problems that microarrays had with the quantification of genes and production of artifacts, are now mostly solved with WTSS which is not anymore based on chemical tagging, but on **cDNA**⋆ sequencing. To note here is that RNA-Seq isn't only limited to mRNA, but can look at all the different types of RNA such as: tRNA, rRNA, and tmRNA.

2.2.1 Gene expression

What is the difference between studying the genome and gene expression levels of some cells? The genome tells us what the cell can potentially do, while the expression profile tells us what the cell is *actually* doing at any point in time. The information contained in a gene is translated during the synthesis of a gene product in a process called gene expression, a process used by all known life forms. Gene product the usually refers to proteins, or functional RNA in non-protein coding (transfer RNA or small nuclear RNA).

Regulation of gene expression is critical to an organism's development. Cells have several mechanisms that are used to increase or decrease the amount of gene product synthesized: transcription, **RNA splicing***, translation, and post-translational modification of a protein. Gene regulation gives the cell control and is the basis for cellular differentiation and the adaptability of any organism (it's linked with evolutionary functions). Gene expression is the most basic level at which the genotype gives rise to the phenotype (observable trait). These traits are often expressed by the synthesis of proteins that control the organism's shape and size. In multi-cellular organisms, gene regulation drives, for example, morphogenesis in the embryo, leading to the creation of different cell types with different gene expression profiles from the same genome.

Expression is quantified in a RNA-Seq experiment by counting the number of reads that map to each locus, in a step during the analysis called transcriptome assembly. Expression can be quantified for exons or genes using **contigs*** or reference transcript annotations.

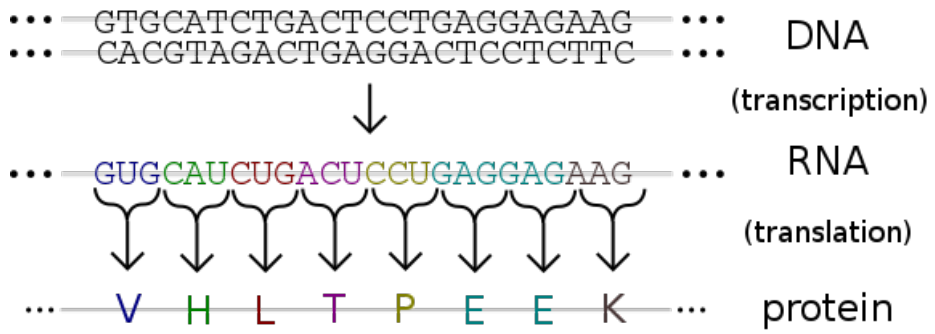


Fig. 2.3 Central dogma of molecular biology: from genetic code to proteins.

Gene expression profiling

Genes contain the instructions for making messenger RNA, but at any given moment each cell makes mRNA *only* from a fraction of the genes it carries. If a gene is used to produce

mRNA, it is said to be "on", otherwise it is "off". Many factors determine whether a gene is on or off: metabolism, its local environment, and chemical signals from other cells.

Gene expression profiling is the name given to the activity that consists in the collective measurement of thousands of genes, to create a global overview of cellular functions. An expression profile allows one to deduce a cell's type, state, environment, etc ... Expression profiling experiments consist in measuring the relative amount of mRNA expressed in two or more conditions/time. The reason for delayed measurements is because altered levels of a sequence of mRNA suggest a different need for the protein coded by the mRNA (homeostatic response or *pathological condition*). Example: if breast cancer cells have levels of mRNA associated with a particular receptor higher than normal cells, it might be that this receptor plays a role in breast cancer. A drug that interferes with this receptor may prevent or treat breast cancer. This example shows how one of the main uses for gene expression profiling may be as an important diagnostic test.

2.3 TCGA

The Cancer Genome Atlas (TCGA) is a project, started in 2005, with the goal to catalog genetic mutations causing for cancer, using genome sequencing to demonstrate that advanced genomic technologies could be utilized to generate biologically significant conclusions from the genomic datasets generated. TCGA applies next generation genome analysis techniques to improve the ability to diagnose, treat, and prevent cancer via a better understanding of the genetic basis of this disease.

TCGA is supervised by the National Cancer Institute's (NCI) Center for Cancer Genomics and the National Human Genome Research Institute (NHGRI) funded by the United States government. The first stage of the project lasted three-year, from 2006, and focused on characterization of three types of cancers: glioblastoma multiforme (GBM), lung cancer (LC), and ovarian cancer (OV). In 2009 the project expanded to phase 2, which planned to complete the genomic characterization and sequence analysis of 25 different tumors by 2014. In this phase, the project was performing whole exome and whole transcriptome sequencing on 100% of the samples and whole genome sequencing on 10% of the cases. The goal was surpassed, ending up cataloging 33 types of cancer. Funding is split between genome characterization centers, which perform the sequencing, and genome data analysis centers, which perform the bioinformatic analyses. The project scheduled 500 patient samples (more than most precedent genomics studies). Techniques for the analysis include gene expression profiling, copy number variation profiling, SNP (single nucleotide polymorphism) genotyping, genome wide DNA methylation profiling, microRNA profiling, and exon sequencing of at

least $1,200$ genes. TCGA is sequencing the entire genomes of some tumors using hybrid-capture technology for at least $6,000$ candidate genes. The TCGA dataset, comprising more than 2 petabytes of genomic data, has been made publicly available, and this genomic information helps the cancer research community to improve the prevention, diagnosis, and treatment of cancer.

2.3.1 Dataset used in the Thesis

Table 2.1 The largest RNA-seq datasets in TCGA

Dataset	Cancer	Healthy	Σ
BRCA	1102	113	1215
LUAD	513	58	571
LUSC	502	51	553
KIRC	534	72	606
HNSC	521	43	564
THCA	513	59	572
PRAD	498	52	550
	4183	448	= 4631

The datasets used in this thesis comes from TCGA public data. The focus here is on the RNA-seq of breast invasive carcinoma (BRCA), the most common form of breast cancer. To notice is that TCGA data contains control (healthy samples), and this will be important when training the model for architecture validation and gene module extraction. The reason for the decision to focus on breast cancer is that is the dataset with the largest sample size. Nonetheless, as discussed in Chapter 3, also other types of cancer are used for validation. Those other type of cancer are:

- **LUAD**, Lung Adenocarcinoma;
- **LUSC**, Lung Squamous Cell Carcinoma;
- **KIRC**, Kidney Renal Clear Cell Carcinoma;
- **HNSC**, Head-Neck Squamous Cell Carcinoma;
- **THCA**, Thyroid Cancer;
- **PRAD**, Prostate Adenocarcinoma.

This TCGA data in our case is given in a matrix format (genomic matrix) tracking $20,530$ individual genes. It can be seen as a two-dimensional matrix with one dimension pertaining to the gene expression level and the other to the patient. As said before the data contains tumoral and control data, it's possible to differentiate cancer samples from controls by parsing their ID which is structured as follows:

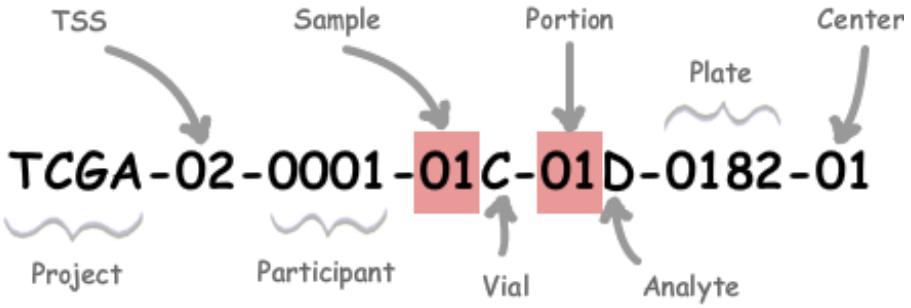


Fig. 2.4 ID tag breakdown for the samples in the TCGA project.

What important is the "Sample" part of the ID, that can be used to discern the two types of patient sample. Here *Tumor* types are represented by a range from 01 - 09, while *Normal* types from 10 - 19 and control samples from 20 - 29. The gene expression profile was measured experimentally using the Illumina Hi-Seq 2000 RNA Sequencing platform by the University of North Carolina TCGA genome characterization center. This dataset shows the gene-level transcription estimates, as in RSEM normalized count (RNA-Seq by Expectation-Maximization). The scaled estimate value is the estimated frequency of the gene amongst the total number of transcripts of each genes that were sequenced (there are multiple ones, alternative splicing). Newer versions of RSEM, call this value (multiplied by $1e+6$) TPM - Transcripts Per Million. The values are subsequently divided by the 75-percentile and multiplied by 1000. The reason for this last part is to make the values a bit more comparable between experiments (discounting outliers). Genes are then mapped onto the human genome coordinates using UCSC cgData HUGO probeMap.

Part II: Machine Learning

In this second part the theory of the Machine Learning part is introduced. The model of neural networks is presented, from the mathematical constructs to different possible typologies. The next part is devoted to autoencoders, the centerpiece to understand the thesis' method. Autoencoders are a type of neural networks used for efficient codings and dimensionality reduction, hence an introduction to dimensionality reduction is also given. In particular, a focus is put on the comparison between different models (PCA and k-PCA) that try to achieve similar results. These two models will be then used in the next chapter to assess the training quality of the final autoencoder model. Also, in the last part of the chapter, a brief mention of the tools and library used is also included.

2.4 Neural Networks

Neural networks (or more aptly ANNs, artificial neural networks) are computing devices inspired by the biological networks found in animal brains. These networks improve performance on specific tasks by "observing" a growing number samples, and they work generally without task-specific programming. The first introduction to this type of models was made by Warren McCulloch and Walter Pitts [7] in the early 40s. The model was inspired by neurons and consisted of a single unit with a single output, which quickly demonstrated its limitations. Nowadays when people talk about neural networks they think about models organized in layers with different layers possibly performing different kinds of transformations on their inputs. As an example a model can be found in Fig. 2.5. Each layer of a neural network is based on a collection of connected units or nodes called "neurons". Each connection (which is inspired synapses in brains) between neurons can transmit a signal from one to the other. Patterns in the form of data are presented to the network via the "input layer" (the group of **red** neurons in the figure), which communicates to one or more "hidden layers" where the bulk of the processing is done via a system of "weighted" connections. The hidden layers (**blue** group in the picture) then link to an "output layer" (**green**) where the output values of each neuron represent the answer. In image recognition, for example, ANNs can learn to identify images that have cats in them by taking images that have been labeled (as having a cat or not having it) and using the results to identify cats in other images.

Usually, the signal/connection between artificial neurons is a real number, and the output of each neuron is calculated by a function (they're called activation functions and can be linear or non-linear e.g. hyperbolic tangent or sigmoid functions) of the weighted sum of its inputs. The weight increases or decreases the strength of the signal at a connection and

finding the optimal set of weights for the specific model is the main goal. Additionally, artificial neurons may sometimes have a threshold such that only if the signal (result of the summation) crosses that threshold the signal sent (e.g. step function).

Despite at the beginning the goal of the artificial neural network approach was to solve problems in the same way that a human brain would, nowadays, the research is focused on matching specific abilities, deviating from biology. ANNs are used on a variety of tasks, such as: computer vision, speech recognition, machine translation, social network filtering, playing board/video games and performing medical diagnosis.

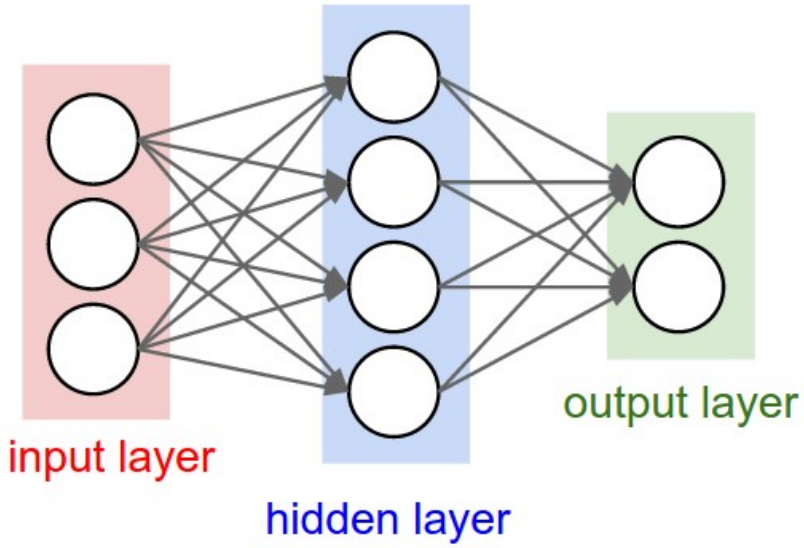


Fig. 2.5 Example of a simple neural network having only one hidden layer.

2.4.1 Model description

This section is going to define in a more rigorous way each of the main component in a neural network:

Neuron A neuron j with label $p_j(t)$ receiving an input from predecessor neurons consists of the following components:

- an **activation** $a_j(t)$, depending on a discrete time parameter, this time dependence isn't always present;
- a **threshold** θ_j, which stays fixed unless changed by a learning function;

- an **activation function** f that computes the new activation at a given time $t+1$ from $a_j(t)$, θ_j and the net input $p_j(t)$ giving rise to the relation

$$a_j(t+1) = f(a_j(t), p_j(t), \theta_j)$$

- and an **output function** f_{out} computing the output from the activation $o_j(t) = f_{out}(a_j(t))$.

An input neuron has no predecessor but serves as input interface for the whole network. Similarly an output neuron has no successor and thus serves as output interface of the whole network.

Connections and weights The network consists of connections between layers. Each connection takes the output of a neuron i and links it to the input of a neuron j (i is said to be the predecessor of j). To each connection a weight w_{ij} is assigned, this will be what is going to be trained. Optimal weights of the network minimize the loss function over samples.

Propagation function The propagation function computes the input $p_j(t)$ to the neuron from the outputs $o_i(t)$ of predecessor neurons and is basically always in the form

$$p_j(t) = \sum_i o_i(t) w_{ij}$$

Learning rule The learning rule is an algorithm which modifies the weights of the neural network, to obtained the desired results. An example of a learning rule is Back propagation, but more on that later.

2.4.2 How learning works

Let's look how does *learning* works, neural networks are mainly used for pattern recognition and this learning possibility is the thing that has attracted the most interest. Given a specific task to solve, there exist a class of functions $f^\star \in F$ which solves the task with some criteria of optimality. This implies that there must be a cost function (also loss or error) $C : F \to \mathbb{R}$ such that, for the optimal function $f^\star$:

$$C(f^\star) \leq C(f)$$

$\forall f \in F \setminus f^\star$. In other words, there is no solution which has a cost less than the cost of the optimal solution.

The cost function C is an important concept, it calculates the difference between the model output and its desired output or, in other words, how far away a solution is from an optimal solution to the problem. Learning algorithms search through the solution space: in ANNs usually usually using gradient-based methods over the surface error, to find a function (dictated by the set of weights) that has the smallest possible error. Gradient based methods usually repeatedly minimize the cost function by changing the parameters (weights in the neural network case) following the opposite direction of the gradient. A two-dimensional example is given in Fig. 2.6 where the "weight" vector x (x_i being the value of the vector x at time i) is repeatedly updated towards the directions of the local minimum over the surface of the cost function.

As an example, consider the most widely used of such cost functions, the Mean Squared Error (MSE):

$$C = E\left[(f(x) - y)^2\right]$$

in which the learning problem consists in finding the model f, which minimizes for data pairs.

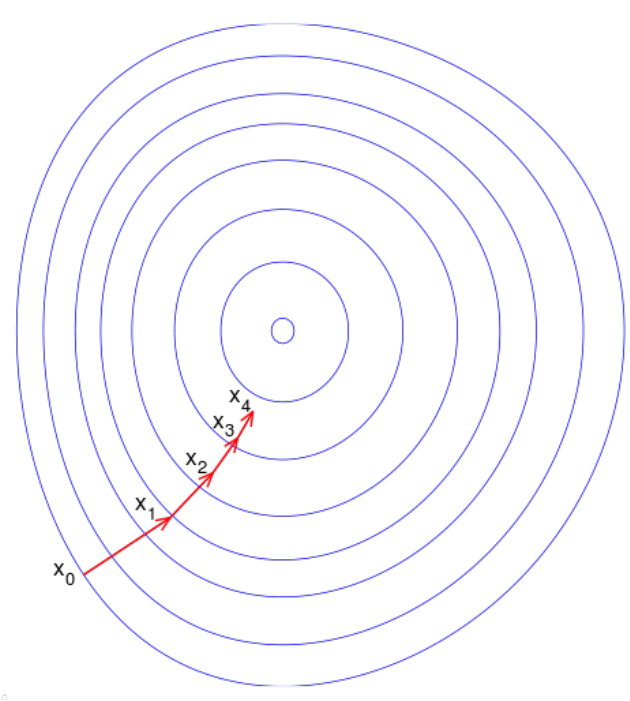


Fig. 2.6 Example of the gradient evolution over time on the surface error (contour line).

Learning algorithm: Backpropagation

Back propagation (or Backprop) is a method to calculate the gradient of the cost function with respect to the weights in a layers. The name stands for backward propagation of errors, because the error is calculated at the output and distributed back through the network layers in a similar way as the chain rule for computing the derivative of the composition of two or more functions. The weight updates of back propagation can be done via several methods, the classic being gradient descent which is based on:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\frac{1}{n}\sum_{i=1}^{n} \nabla_w C(z_i(t), w_{ij}(t))$$

where, η is the learning rate (step size), C is the cost function. The method used in the thesis is a popular variant of the gradient descent method called, mini-batch stochastic gradient descent. This method has good convergence property, being much faster albeit still a bit more noisy given the different update rule. The latter being now:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\nabla_w C(z(t), w_{ij}(t)).$$

This is the *stochastic* version of gradient descent because each iteration (being a batch, subset of the dataset) updates the value of the gradient and therefore learns "on-line" instead of waiting for an entire pass-through of the training set.

2.4.3 Deep Neural Networks

A deep neural network is deep learning [8] (hierarchical learning) method that uses a neural network with multiple hidden layers between the input and output layers to model complex relationships. The term is actually kind of contentious because there was not much usage of networks with only one layer overall: the famous McCulloch–Pitts model. This model soon demonstrated its limitation and the field only got traction with new discovery.

After having parted way with the simple one neuron model the field had a resurgence in the late 1990s when higher computational power permitted the use of more complex (multi layered) models. The literature subdivide the neural network category in other two terms: *shallow* networks and *deep* networks. Shallow network refers to ANN with only one hidden layer, while DNN has multiple ones. See Fig. 2.7 for an example of a simple deep neural network. This is similar to Fig. 2.5, but with the addition of another hidden layer (there are two **blue** groups of layer connected between each other).

DNNs architectures generate compositional representations of the data where lower layers map low level (more specific) type of features. For example the first layer would represent high order feature (such as edges in an image). DNNs are typically feed-forward, meaning that the data flows from the input layer to the output layer without going backwards, nonetheless some architecture exists which loop back (e.g. Recurrent Neural Networks, RNNs). Each architecture is better suited for specific domains, for example Convolutional Neural Networks are used especially in the field of Computer Vision.

DNNs naturally share, many of the same faults of numerous other machine learning methods: over-fitting and computation time:

- **Over-fitting** is an issue for DNNs because of the added layers, by adding too many layers the risk is to have specific weights dedicated to every single example in the dataset. To compensate this side-effect there exist several regularization methods such as weight decay (ℓ_1-regularization) or sparsity constraints (ℓ_2-regularization) which can be applied during training. An addition which seemed to have found success is also using dropout neurons, this consist in randomly omitting, during training, units from the hidden layers.

- On the **computational** side, DNNs have several training parameters to consider: number of layers and number of units per layer, the learning rate, initial weights, loss function and activation function. Performing a proper grid search through the parameters for the optimal values may not be possible considering due to the cost in time and computational resources. Various speedups are possible, such as using batching (computing the gradient on a subset of samples rather than on individual ones). One significant improvement during the last decade was the use of GPU, graphical processing units, to enable quick matrix and vectors operations which are used intensively during training. To notice here it is also the choice of proper activation functions with non-linear ones to be treated carefully given the vanishing gradient effect. For this reason the main function used in deep network is ReLU (rectified linear unit) [9] that do not suffer from this problem.

2.5 Autoencoders

Autoencoder [3] refers to a particular layout for artificial neural networks which is used in particular for unsupervised learning of efficient codings. This means that the network

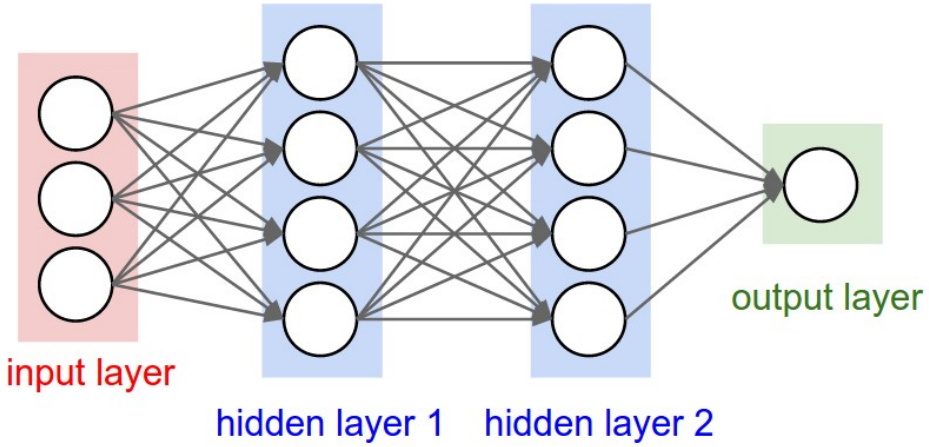


Fig. 2.7 Example of a deep neural network with two hidden layers.

objective is to learn a representation (code) of a set of data of reduced dimension, hence the main use of autoencoders is dimensionality reduction [4][10]. Basically after the training the ANN will be able to take an high input vector and give as an output a vector of smaller size. Autoencoder usage is not limited to dimensionality reduction, but they're also used as generative models, in this thesis *both* of these aspects are leveraged.

As the name could give out, the autoencoder is actually composed of two parts: an *encoder* and a *decoder*. The first part takes the high dimensional vector and reduce the size, while the decoder part returns again the high input vector. In fact the training works jointly between the two parts to approximate the identity function. After that only one of the two (encoder or decoder) is usually employed. An annotated picture of the model can be found in Fig. 2.8.

2.5.1 Structure

The whole process of an autoencoder can be defined mathematically as:

$$\phi : \mathbf{X} \to \mathbf{F},$$

$$\psi : \mathbf{F} \to \mathbf{X},$$

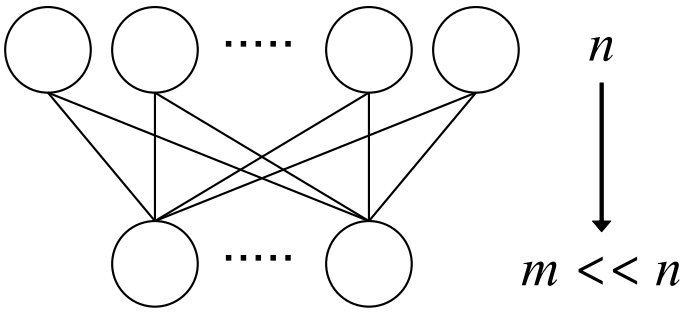
$$\phi, \psi = \arg\min_{\phi,\psi} \|\mathbf{X} - (\psi \circ \phi)\mathbf{X}\|^2$$

Let's take a look more precisely on the simplest case possible, an autoencoder with only one hidden layer, the *encoder* part takes the input $\mathbf{x} \in \mathbf{X} = \mathbb{R}^n$ and maps it to $\mathbf{z} \in \mathbf{F} = \mathbb{R}^m$ (where

$m < n$) using this relation:

$$\mathbf{z} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

The image/output $\mathbf{z}$ is what it is referred to as *code*, latent variables, or reduced dimensions. In the above formula σ is an element-wise activation function such as a sigmoid function or a `ReLU` (rectified linear unit). $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ is a bias vector and are the parameters to be found via training.

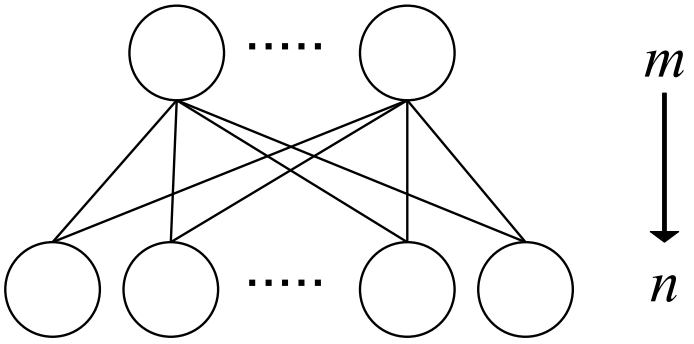


Dimensionality reduction performed by the *encoder*.

After the encoder, the *decoder* section of the network maps $\mathbf{z}$ to a reconstruction $\mathbf{x}'$ of of the same size as the input $\mathbf{x}$:

$$\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{z} + \mathbf{b}')$$

where σ', $\mathbf{W}'$, and $\mathbf{b}'$ for the decoder may differ, depending on the network architecture, from the corresponding σ, $\mathbf{W}$, and $\mathbf{b}$ of the encoder. Sometimes as a regularization method Tied Weights (the weight matrices are one the transpose of the other) between encoder and decoder are used, and that way you trade-off expressiveness for performance.



The *decoder* returns the vector back to the original input space

Autoencoders are trained to minimize the reconstruction errors between the input and output. Let's take a look at the squared error:

$$\mathscr{L}(\mathbf{x},\mathbf{x}') = \|\mathbf{x}-\mathbf{x}'\|^2 = \|\mathbf{x}-\sigma'(\mathbf{W}'(\sigma(\mathbf{W}\mathbf{x}+\mathbf{b}))+\mathbf{b}')\|^2$$

where $\mathbf{x}$ is averaged over some inputs of the training set. If the feature space $\mathbf{F}$ has lower dimensionality than the input space $\mathbf{X}$, then the feature vector ϕ_x can be regarded as a compressed representation of the input $\mathbf{x}$. If the hidden layers are larger than the input layer, an autoencoder can possibly learn the identity function (over-fit) and become useless. Although in this last case some experimental results have shown that autoencoders may still learn new features.

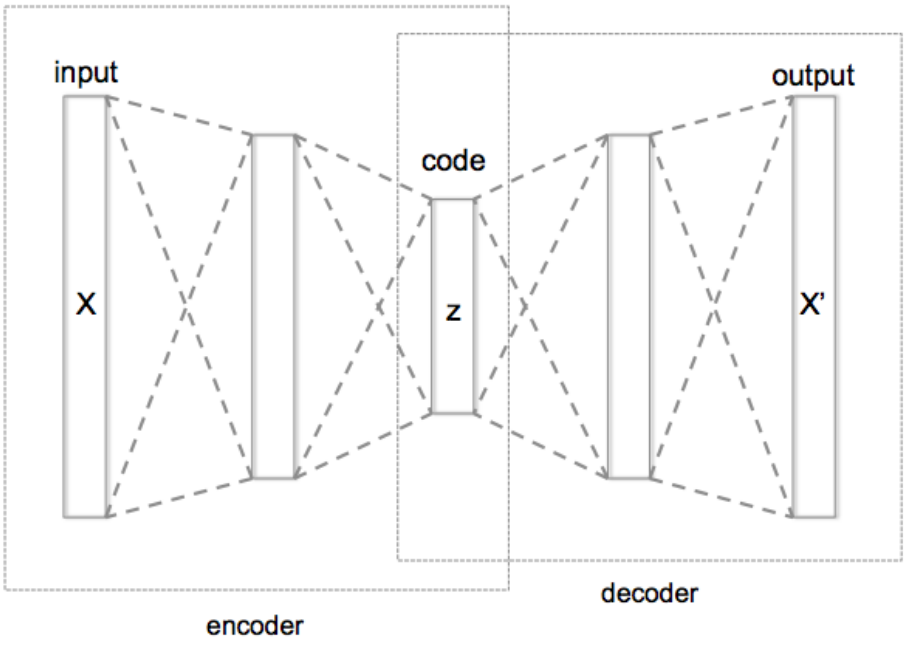


Fig. 2.8 Layout of an architecture of a deep autoencoder

2.6 Dimensionality reduction

Dimensionality reduction is the process in Machine Learning involving the reduction of the number of variables to be taken in consideration, and it can be divided in two sub-fields: (1) feature selection and (2) feature extraction. The former concerns in finding a meaningful

subset of features from the all with various methods, adding features/dimensions one by one and making considerations on the information gain or eliminating some of them from the totality. Feature extraction, the one approach used in the thesis, is all about constructing new features from a combination of the original ones. These new feature will be fewer and *hopefully* maintain the most important properties.

The data dealt with in this thesis is gene expression data which is particularly high-dimensional (see the Introduction). The typical result from microarray experiments include around $20,000$ genes (sample's dimensions). These genes interact with each other in a myriad of way, each gene codes from some proteins and there are about $2,000,000$ of them. The complexity of their interaction is apparent even by just taking in consideration the sheer numbers.

Dimensionality reduction can be linear or non linear. The linear case is based on the idea of doing a linear projection of the data over some chosen lower dimensional hyperplane. While this approach can be very powerful and obtain very good generalization results, the method is limited in the description of complex data interaction such as in our genetic case. The non-linear approach instead assumes that the data of interest lie on an embedded non-linear manifold within the higher-dimensional space. (Fig. 2.9 represent a two dimensional manifold embedded in a three dimensional space) There are multiple ways of doing dimensionality reduction in a non-linear fashion. Some of these methods to mention: Self-organizing map [11], Isomap [12], Kernel principal component analysis (k-PCA) and finally, Autoencoders.

In the next chapters an autoencoder with non-linear activation functions is used for the aforementioned purpose. When used for dimensionality reduction, one of the hidden layers (usually the middle one, output of the decoder) is limited to contain only a small number of network units. This way, the network must learn to encode the vector into a small number of dimensions and then decode it back into the original space. The learned encoding is found by minimizing the loss function therefore by trying to detect regularities between dimensions (genes) across all the training set. Let's look now at some other methods which will be used as a basis for evaluating performance of the autoencoder model which will be then used as a basis for module extraction.

2.6.1 Principal Component Analysis

One of the staple methods for the analysis of gene expression data is PCA [13]. Principal Component Analysis is a procedure that uses orthogonal transformations to convert data belonging to a set of presumptively correlated variables into a new set of values of *linearly* uncorrelated variables called principal components. Dimensionality reduction in this case

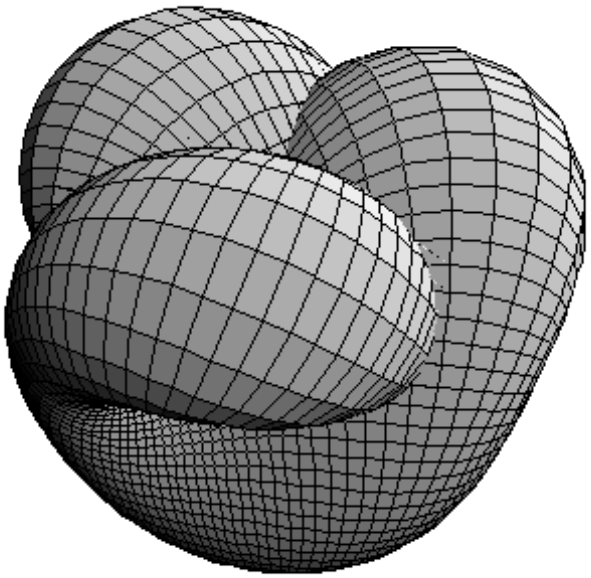


Fig. 2.9 Example of a 2-dimensional manifold embedded in a three dimensions

is achieved by using a smaller number of principal components than the original number of dimensions of the data. The new features in the PCA case are obtained via a *linear* transformation of the axes such that that it maximizes the variance of the data (so that it accounts for as much of the variability in the data as possible). Each of the succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. One of the main use of PCA is data visualization, but also to extract new (fewer) features that can represent the data.

Mathematically, the transformation done by PCA is defined by a set of p-dimensional vectors of weights or "*loadings*" $\mathbf{w}_k = (w_1 \dots w_p)_k$ that map each row vector $\mathbf{x}_i$ of $\mathbf{X}$ (data matrix, zero-centered) to a new vector of principal component scores $\mathbf{t}_i = (t_1 \dots t_m)_i$ given by:

$$t_{ki} = \mathbf{x}_i \cdot \mathbf{w}_k \qquad \text{for} \qquad i = 1 \dots n, \qquad k = 1 \dots m$$

in such a way that the individual variables $t_1 \dots t_m$ of $\mathbf{t}$ considered over the data set successively inherit the maximum possible variance from $\mathbf{x}$, with each loading vector $\mathbf{w}$ constrained to be a unit vector. In order to maximize variance, the first loading vector w_1 thus has to satisfy:

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} \left\{ \sum_i (t_1)_i^2 \right\} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \sum_i (\mathbf{x}_i \cdot \mathbf{w})^2 \right\} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \right\}$$

Since w_1 has been defined to be a unit vector, it equivalently also satisfies

$$\mathbf{w}_1 = \arg\max \left\{ \frac{\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$$

With w_1 found, the first principal component of a data vector x_i can then be given as a score $t_{1i} = x_i \cdot w_1$ in the transformed co-ordinates, or as the corresponding vector in the original variables, $\{x_i \cdot w_1\} w_1$.

The k-th component can be found by subtracting the first $k-1$ principal components from $\mathbf{X}$:

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X} \mathbf{w}_s \mathbf{w}_s^{\mathrm{T}}$$

and then finding the weight vector which extracts the maximum variance from this new data matrix

$$\mathbf{w}_{(k)} = \underset{\|\mathbf{w}\|=1}{\arg\max} \left\{ \|\hat{\mathbf{X}}_k \mathbf{w}\|^2 \right\}$$

The transformation $\mathbf{T} = \mathbf{X}\mathbf{W}$ maps a data vector x_i from an original space of p variables to a new space of p variables which are uncorrelated over the dataset. See an example in Fig. 2.10 to visualize the principal components (vectors) of a two dimensional gaussian distributed data.

Not all the principal components need to be used. This is especially important for dimensionality reduction purposes, in which only the new dimensions which better explained the data are kept. Keeping only the first L principal components, produced by using only the first L loading vectors, gives the truncated transformation

$$\mathbf{T}_L = \mathbf{X}\mathbf{W}_L$$

An autoencoder neural network with a linear hidden layer is similar to PCA. Upon convergence, the weight vectors of the K neurons in the hidden layer will form a basis for the space spanned by the first K principal components. Unlike PCA, this technique will not necessarily produce orthogonal vectors.

One of the limit of PCA for the detection of collaborating genes (dimensions) is that it's underlying assumption is linearity, but very rarely in the real world one can make such assumption. Given the high-dimensionality and complexity of human biology the objective is to try perform dimensionality reduction (and corresponding features extraction) assuming non-linearity, in an attempt to better capture the interactions between genes. For this reason let's see a variant of PCA that into account.

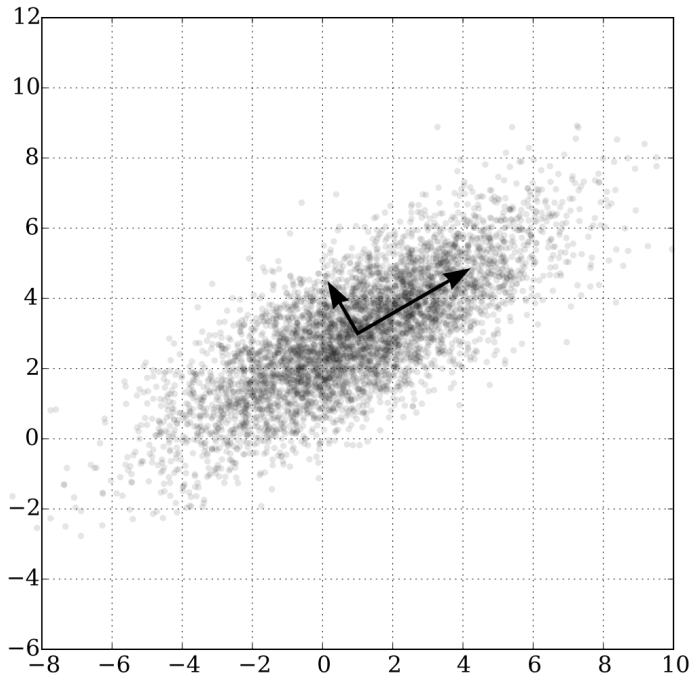


Fig. 2.10 PCA of a multivariate Gaussian distribution.

2.6.2 Kernel-Principal Component Analysis

Kernel-principal component analysis [14] (or kernel PCA) is an extension of principal component analysis that uses techniques of kernel methods. It is therefore a combination of Principal component analysis and the kernel trick. In PCA $\mathbf{X}^T\mathbf{X}$ can be recognized as proportional to the empirical sample covariance matrix of the dataset $\mathbf{X}$, so it operates by diagonalizing the covariance matrix (finding the eigen-decomposition). PCA works by computing:

$$C = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}}$$

. It then projects the data onto the first k eigenvectors of that matrix. Instead k-PCA in the first step it begins by computing the covariance matrix of the data after being transformed into a higher-dimensional space,

$$C = \frac{1}{m}\sum_{i=1}^{m} \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^{\mathbf{T}}$$

It then projects the transformed data onto the first k eigenvectors of that matrix, just like PCA. Important to notice is that Φ is never explicitly calculated, but it dealt by replacing the inner product with another function, to each function it's associated a specific "true" feature space ($\Phi(\mathbf{x})$-space) corresponding to Φ. So basically it performs a linear projections for the points in the usually higher dimensional space (Φ), but this results in non linear decomposition in the original space.

The difference here with respect to linear PCA, it's possible to use the eigenvalues to rank the eigenvectors based on how much of the variation of the data is captured by each principal component. This is useful for data dimensionality reduction and it could also be applied to k-PCA. However, in practice there are cases that all variations of the data are same. This is typically caused by a wrong choice of the kernel scale.

2.7 Tools used

The main tools and libraries are given a brief overview in this section. This is by no mean an exhaustive list of what was used, but it will give an idea of the most important pieces. The main machine on which the experiments were run, such as the training of the neural networks, is not equipped with a GPU. This constraint guided some design choices and focus on the experiment pipeline. Not having the possibility of running experiments on GPUs results in slower time and less options in the parameter selection. Nonetheless as it is discussed later this didn't resulted in significant degraded performances. Let's have a brief overview of the main tools:

- **Python** is an interpreted high-level programming language. It features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python is widely used in any type of Data Science settings for the wide availability of libraries and frameworks. The fact that Python is an interpreted language lends itself for quick prototyping and data exploration.

- **Keras** [15] is an open source neural network library written in Python. It is an interface capable of running on top of several neural network frameworks (it's more of an engine than an end-to-end machine-learning framework) such as MXNet, Deeplearning4j, TensorFlow, Microsoft Cognitive Toolkit or Theano. Keras is designed to enable fast experimentation with deep neural networks being minimal, modular and extensible. Our implementation relies on TensorFlow, which is an open source software library

for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The architecture is flexible allowing to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

- **Libraries** Several libraries were used, but the main ones were: Pandas [16] and Matplotlib [17]. Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard-copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

2.8 Summary

In this chapter the field of Bioinformatics is introduced with its main activities and successes. Then the concepts of gene expression and gene expression profiling are explained to prime the reader with the theoretical foundation that underlies the domain knowledge of the data dealt with. The TCGA project and the data that are taken from it are also discussed before focusing on the Machine Learning background. This second part introduced what a neural network is, its mathematical constructs and their different typology. It is explained what learning is and what it is meant by Deep Neural Networks. Then the specific ANN that was used in the thesis is introduced: the Autoencoder. They're a type of neural networks used for efficient codings and dimensionality reduction. The topic of dimensionality reduction is discussed, and two alternative methods are presented: Principal Component Analysis and kernel Principal Component Analysis. These two methods are discussed mathematically and in comparison between each other since they will be used in the next chapter to assess the training quality of the autoencoder model. A brief part is then dedicated to discussing the software tools used for the experiments.

CHAPTER 3

MODEL

This chapter is devoted to the training and validation of the autoencoder model. Experiments are made to find the parameters necessary for the definition of the network architecture, such as, number of hidden layers and activation functions. What is important here is the capability of the reduced target dimensions of the autoencoder to capture true latent variables. To do this two stages were undertook: in the first there's a comparison between different parameters through model validation. In the second part PCA and k-PCA models are used with the same number of target reduced variables as comparison in a classification scenario. The reason for the choice of PCA and k-PCA is because they're well known methods used in the fields with known performance targets to be met. It will be shown that the proposed model manages to perform on the same level as the other two. In the last part the autoencoder trained from breast cancer data is compared with other autoencoders (fixing the parameters) trained on other type of cancer. The goal is to see how different dataset impact the performances, especially for what regards the size. It's important to remind the reader that the focus to breast cancer is done for two reasons: the first being that it is the one type with the largest available dataset and second, because as shown later on, the evaluation of the target modules requires knowledge of the specific tissue and focusing on more than one at a time would disperse the efforts.

3.1 Architecture

In the introduction, the mentioned goal is to have a hierarchical model that explain the complexity of biological data. How should one decide how many layers should our final model have? Which non-linear activation function should be chosen for the layers' nodes? In this and the following sections these questions will be answered. To find the optimal parameters several training sessions were performed for different types of network parameters

and have chosen the best performing one. As is going to be discussed it's not true that the more the layer the merrier, but we need to be cognizant of over-fitting and diminishing returns.

Before getting into an explanation of the main parameters for the network and the training process, here a little notation that is used in the text. When talking about a model of two layer in the following material, the focus is on the number of layers relative to either the encoder or the decoder minus the layer they both have in common. The notation does not refers on the total number of layers of the network which would instead be: $2n+1$. The reason for this this is for practicality, reasoning about the size of an individual part rather than the whole (the network is symmetric). Now an example to clarify what we've written in Figure 3.1. The `dense` attribute in this case means that the layer are fully connected: there's a connection from each neuron of a layer to each neuron of the previous layer. The output "`dense_2`" layer is the code, while the "`inputLayer`" is a placeholder for the input data.

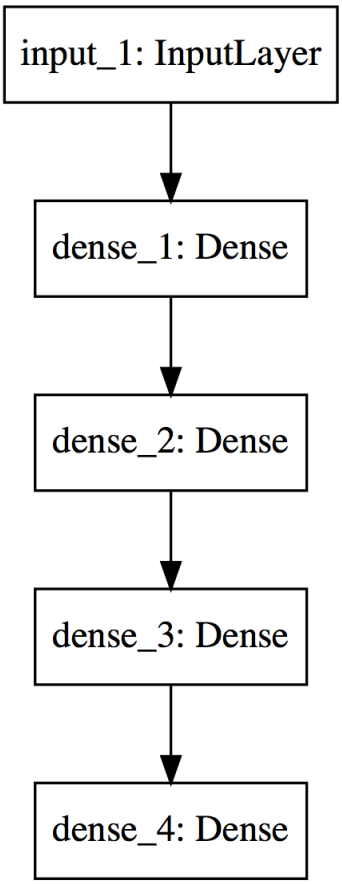


Fig. 3.1 Example of a "two" layer deep autoencoder. It has 3 hidden layers.

Important to note here are the constraints the experiments were under. Training a neural network model, especially a deep one, is a very computational intensive process. For this reason to perform quick iterations and validations, the setups usually include a computer provided with a GPU. Unfortunately this was not the case in this thesis and the fine-tuning suffered a little bit all considering, limiting the training time. Nonetheless as it is going to be discussed, good results were achieved and future works could involve more model tweaking.

3.1.1 Parameters explained

This subsection will go over the main concepts involving the neural network. Included are not only parameters descriptive of the network architecture, but also ones which explain the training process.

Architecture parameters:

- **Activation function** (also called *transfer function*) of a node defines the output of that node given an input or set of inputs. An electronic circuit for example can be seen as a network of activation functions that can be either 1 or 0. They can be linear and non-linear, some examples of the most popular are:
  - Identity:
    $$f(x) = x$$
  - Sigmoid or Logistic:
    $$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$$
  - Rectified linear unit (`ReLU`):
    $$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$
  - TanH:
    $$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$
  - *etc* ...
- **Number of layers** refers to how many layers are present in a neural network. In particular hidden layers are all the layers in between the input (first) and output (last) layer. In our case, since the network is symmetric, the number of layers in the autoencoder is defined as the number of layers for the encoder/decoder minus one.

- **Loss function** calculates the difference between the model output and its desired output. Example: Mean Squared Error (MSE) which is given by:

$$E\left[(f(x)-y)^2\right]$$

- **Connectivity** refers to the number of (active) connections between neurons in the layers. When it's not mentioned, the default is usually a fully connected (dense) network.

Training parameters:

- **Epoch** describes the number of times the algorithm sees all dataset (*one pass-through*). Each time the algorithm has seen all samples in the dataset, an epoch has passed.
- **Batch size** refers to how many samples the algorithm sees before computing the gradient update. This is quite an important concept. In this case the optimizer to minimize the objective function used is called Stochastic Gradient Descent method. Technically though given that new weights are calculated at the end of each batch and not after each training samples the method is called **Mini-Batch Gradient Descent**. This is because it's really in between Gradient Descent and Stochastic Gradient Descent, it has overall better properties: more robust convergence, memory and computationally more efficient.
- **Shuffle** indicates the shuffling of the training data before the start of each epoch. It can leads to better generalization properties given that the introduced randomness reduces the probability of having imbalanced training batches.
- **Early stopping policy** [18] refers to a policy that stop the training once your loss changes in a certain way. For example to avoid protracted training times the training is stopped after the validation loss has stopped improving by a certain margin (*see Minimum delta*) after some number of subsequent epochs (*see Patience*). This does not necessarily mean that the model will not improve more, but it's just a computational issue.
- **Validation loss** refers to the value of the cost function computed over the validation set. The latter is the set of samples used to tune the parameters of the model.
- **Minimum delta** is a threshold that allows us to evaluate a loss at some epoch as an improvement or not. If the difference of loss is below this delta it is quantified as not improving.

- **Patience** is a parameter that represents the number of epochs before you have to stop once you obtain the minimum delta. If small batches or large learning rates are used, the loss will be noisy (jump a lot) so it's better set a large patience. If instead large batches' sizes are used (or small learning rates) the loss will be smoother and one therefore can use a smaller patience argument.

Here it's necessary to mention also the **target reduced dimensions** (number of modules), or in other words: the size of the *middle layer*.

Important to note The terms: reduced/target dimensions, latent variables and modules are here used interchangeably to identify a grouping of genes without consideration to their biological significance. Each grouping is identified to a reduced dimension because we will find a one-to-one mapping between the two. The functional analysis that will link some modules to a correspondent biological activity will be the subject of chapter 5 called "Enrichment Analysis"

The size of the middle layer was chosen by taking inspiration to a similar paper that used autoencoders. Comparatively the number of modules (target latent variables) is reduced even more to 400. The reason this was done is because a latter part of manual verification for each module was envisioned: aiming for too many modules would have been counter-productive. Beside the aforementioned problem: there's a trade-off between the number of modules and the encoding quality. The less the modules the less the encoding ability (reconstruction results), but increasing the number of modules, the specificity of each modules (capability of extract a specific activity) is reduced. In other words, this parameter can balance not only the quality but also whether one wants to extract high or low level activity.

The dimension of each subsequent layer of the encoder (decoder) was reduced (increased) linearly until the target was reached. Example: if the starting vector is of $20,000$ features and it's going to be reduced to 400 with 4 layers then the mid-step sizes for each layers would be: 15100, 10200, 5300.

Important to note is that the effect of decreasing slowly the size of each subsequent layer generally leads to better performances. [19] It can intuitively be understood that diminishing the size abruptly wouldn't let the model learn for non-trivial reduction. The other side of the medal here would be a greater risk of over-fitting (it's related to the number of overall connections) and an increase of compute time.

3.2 Training

In the training the "raw" performance (over the cost function) of different network was evaluated architectures. The comparison is between two **activation functions**: `ReLU` (linear)

and `TanH` (non-linear). The `Sigmoid` function was also taken into consideration, but results are here not included. The performance of this functions in preliminary trials was always worse than the `TanH` by at least an order of magnitude. The main reason for this could be that its output is not zero-centered; antisymmetric activation functions can yield faster convergence than a similar process with non-symmetric activation functions. [20]

The **number of layers** tried were: 1, 2, 4 and 8. Conceptually the training stage is divided in two parts. The first evaluates the performance of different activation functions while the second one the effect of layers increase. It's also investigated if keeping all the parameters frozen but the number of layers would yield better results.

As far as training parameters go, as said the **optimizer** is Mini-Batch Gradient Descent with batch size of 64, [21] and the **loss function** is the mean squared error. The **epochs** number chosen is 200. While the epochs' number is chosen for time constraints the batch size is chosen following suggestions in related literature. As far as the stopping policy goes: the **minimum delta** is chosen to be $5e-5$, with **patience** of 100 to still let fast converging model the possibility of jumping out of local minima.

3.2.1 Preprocessing

Both healthy and cancer cell samples are used during this part of the training. This choice was done to use the largest possible set. Since, unfortunately, a vast amount of data is not available, using all the available data at hand is vital. In the second part of the thesis the approach is changed and exclusively cancer samples are used, because it's more interesting to capture cancer-related interactions. In that case it's like healthy samples are treated as "noise", furthering us to the target of identifying mainly tumor activity.

The data is processed before being used in training. One of such operation performed is min-max normalization on $[0,1]$ to make the training faster and reduce the chances of getting stuck in local optima. A hold-out set of 20% is removed to verify the results at the end of the process with unseen data. The split between validation and training set is $10:90$. The reason for this choice are similar as above, maintaining the training set as large as possible given the relatively small dataset.

Important to note, to perform the splits of the data, *stratified sampling* was used to balance the amount of cancer and healthy samples present in each set.

3.2.2 Results

The first experiments for what concerns the network architecture were regarding the activation functions. The aim was to validate our assumption that the RNA-seq data would be better

suited for non-linear reductions. The training is undertook by keeping all the parameters the same but the type of activation function in each neuron. The only difference between the models are the activation functions `ReLU` and `TanH`. To get as of a complete picture as possible the same setup was tried with increasing model depth. Results are reported in Table 3.1, the values are taken for the autoencoder relative to the last epoch. The Δ indicates the difference between the Training loss and the Validation loss. See also the training and validation plots at Fig. 3.2-3.3-3.4-3.5. For the plots on the upper part of each figure, the X axis represent the epoch number during training and on the Y axis the value of the loss for both training and validation. For the plots on the bottom of each figure, the X axis represent the epoch number during training and on the Y axis the value is the difference between the training and the validation loss.

Important to note here is that the validation loss is smaller than the training loss given that the sign of Δ is always positive. At the beginning it seems odd, but it's mostly due to the way Keras (the programming library used) calculates the loss. The training loss is calculated as the average of the losses over each batch of training data. Because the model is improving over time, the loss over the first batches of an epoch is generally higher than over the last batches. On the other hand, the validation loss for an epoch is computed using the model as it is at the end of the epoch, resulting in a lower loss. Furthermore the difference is really relatively small, hence it could also be due to statistical noise and the inherent difficulty of having a *balanced* split between training and test set in *unlabeled* data. it's important to stress more on the focus on the improvement in the trend, see Table 3.1 underneath where the value of the training loss is reported for various number of layers and activation function:

Table 3.1 Final loss between models with different activation functions.

Layer\Metric	`ReLU`		`TanH`	
	Training Loss	Δ	Training Loss	Δ
1	5.99e-02	1.06e-07	4.08e-02	3.11e-04
2	5.71e-02	3.70e-05	4.47e-03	1.46e-05
4	5.49e-02	1.28e-04	2.97e-03	7.44e-05
8	5.47e-02	1.05e-04	2.92e-03	6.19e-05

It's possible to observe here that using a shallow autoencoder (1 layer) gives us comparable results between the different activation functions. This is to be expected given the reduction is really simple in both cases. What is surprising though is that by adding layers for the linear activation the loss improves only slightly, not enough to warrant much attention. Interestingly the progression observed for the non-linear case by adding multiple layers is significant. An autoencoder of size 2 improves the loss by an order of magnitude and by doubling it to 4 another time it's halved. This significant decrease to the rate of improvement

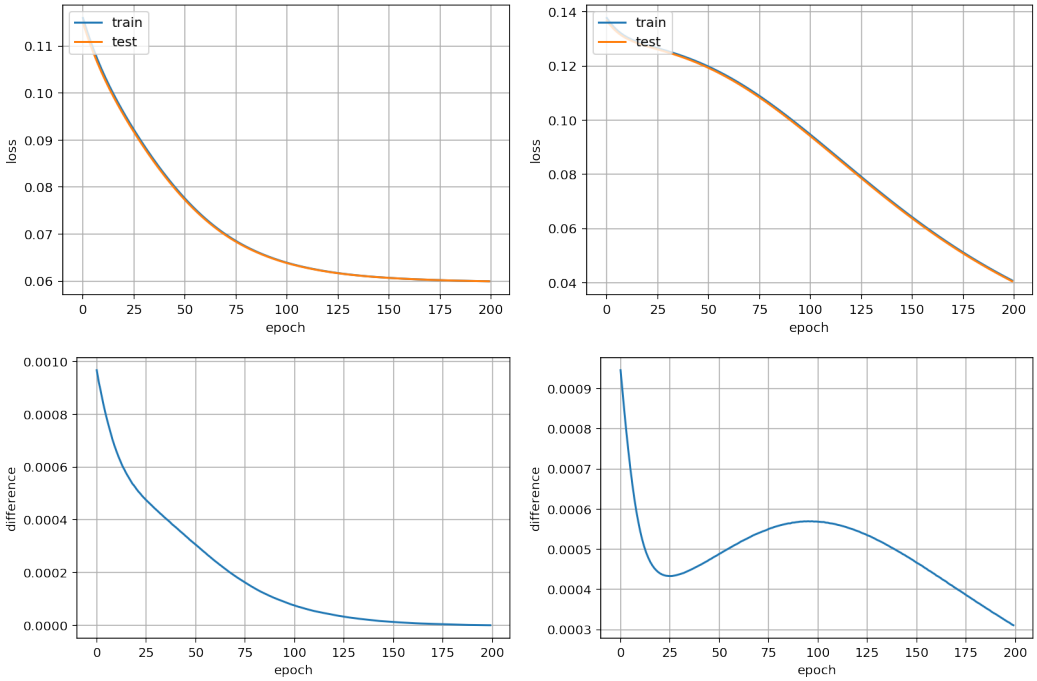


Fig. 3.2 *One* Layer models. Training and test loss plot (above), Δ evolution (down). On the left `ReLU` and on the right `TanH`.

would suggest that the optimal network size is being reached. By using 8 layers one can see that the model does not improve by much. It appears that at this points, for our training setup, significantly diminished returns are reached compared to the training time necessary.

The sweet-spot between performances and training time for our experiment pipeline is therefore a model with ***4 layers***.

3.3 Validation through Classification

After having settled with those parameters, the next step is validating the fact that the model have learned meaningful features. In other words can these reduced features represent the data in a classification scenario? This is going to be put to test by feeding an Support Vector Classifier with features obtained from the encoder, a PCA and a k-PCA (a variant of PCA that enables non-linearity, here with a radial basis function kernel) model. Of course all of them have the same number of latent variables. The reason why of the use of the other two models, as mentioned before, is because there are other papers that used them and it's possible to have clear targets (performance-wise) that are known to be expected. [22] This step can be seen as a double verification step for the model after the classic training process, by using state-of-the-art works. Achieving high accuracies in the results would validate the

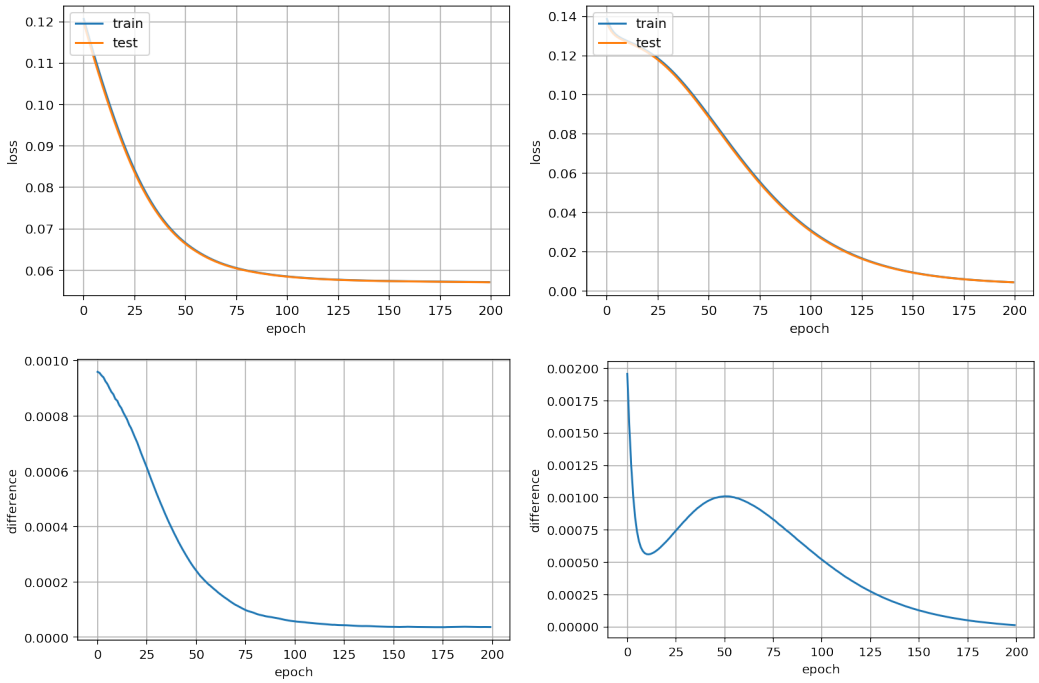


Fig. 3.3 *Two* Layer models. Training and test loss plot (above), Δ evolution (down). On the left `ReLU` and on the right `TanH`.

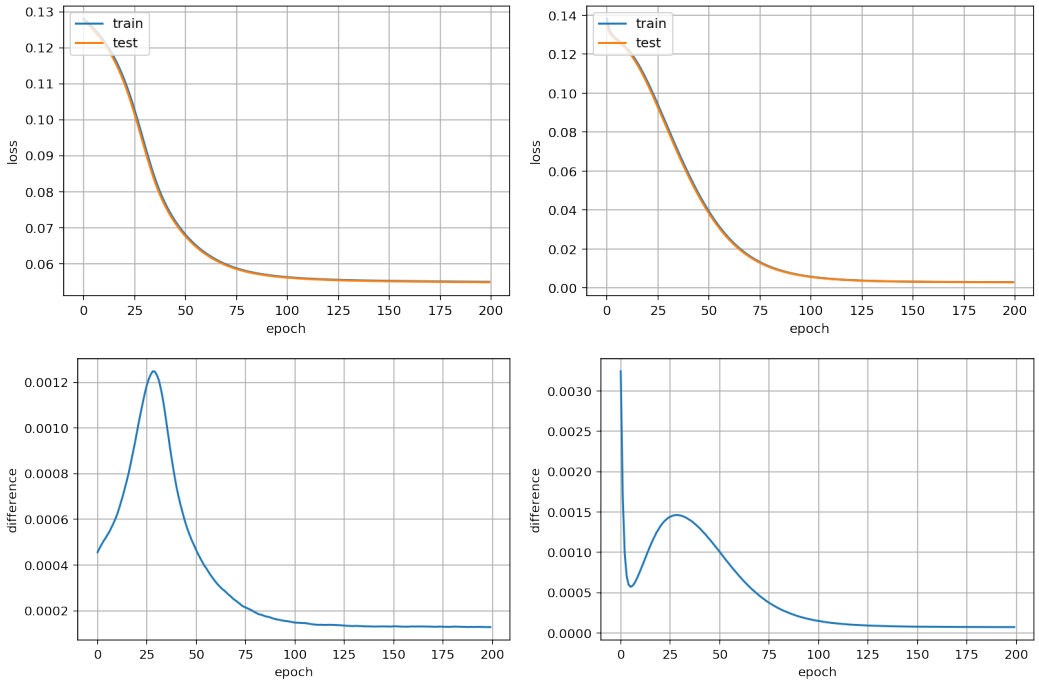


Fig. 3.4 *Four* Layer models. Training and test loss plot (above), Δ evolution (down). On the left `ReLU` and on the right `TanH`.

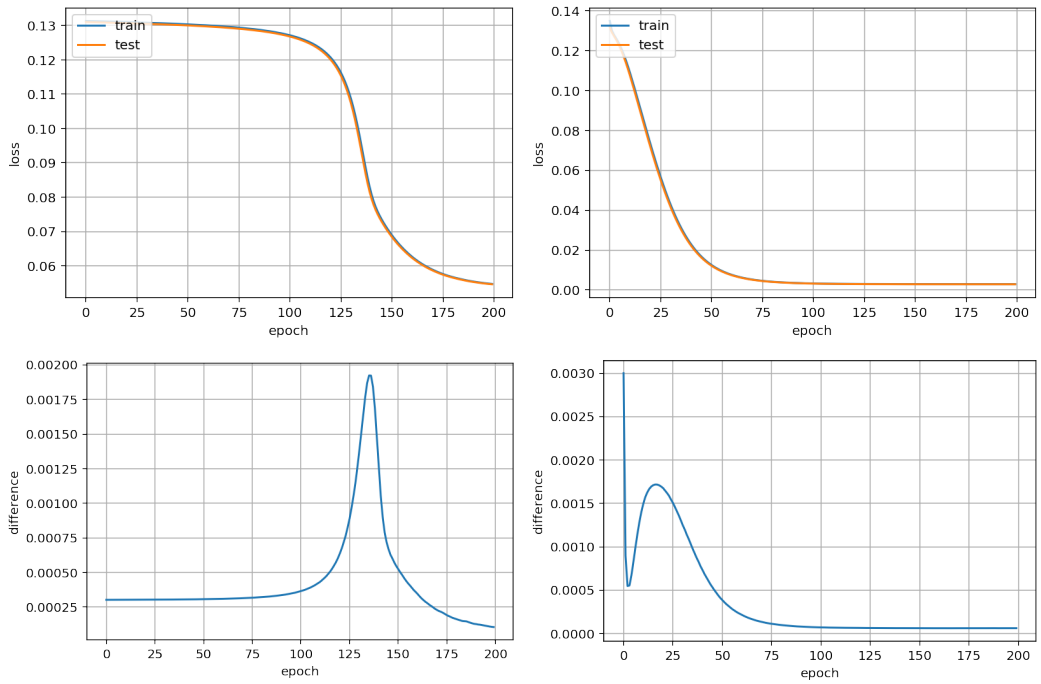


Fig. 3.5 *Eight* Layer models. Training and test loss plot (above), Δ evolution (down). On the left `ReLU` and on the right `TanH`.

ability of autoencoders to extract meaningful new features that can summarize/replace the original ones.

3.3.1 Support Vector Classifier

In this subsection the concept of a support vector classifier (SVC) [23] is introduced. SVCs are supervised learning models that construct an hyperplane or set of hyperplanes in a higher or infinite-dimensional space to separate data of different classes. A good separation is said to be achieved by the hyperplane that has the largest distance to the nearest training point belonging to any class (so-called *functional margin*, see Fig. 3.6 for an example of classification through different margin). This is done intuitively because the larger the margin the lower the generalization error of the classifier.

Following an explanation on the case of linear classification. Any hyperplane can be identified as the set of points $\vec{x}$ that lie on it, in other words: $\vec{w} \cdot \vec{x} - b = 0$ where $\vec{w}$ is the normal vector to the hyperplane. The support vector methods can be extended to cases in which the dataset isn't linearly separable, by introducing the *hinge loss function*: $\max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b))$. This function is zero if $\vec{x}_i$ lies on the correct side of the margin.

For data on the wrong side of the margin, the function's value is proportional to the distance from the margin.

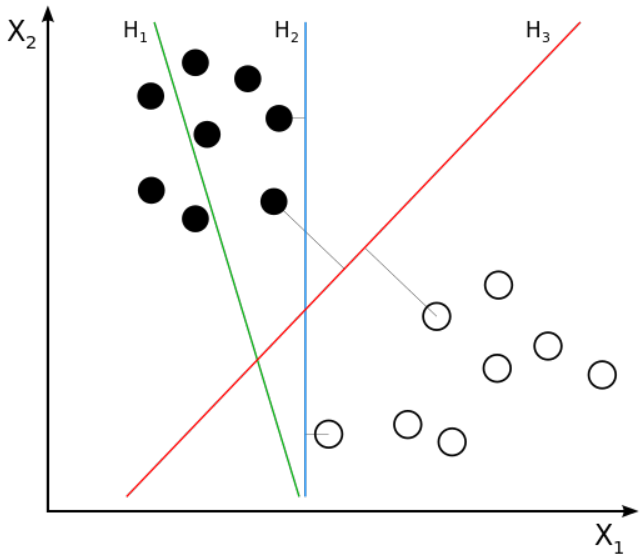


Fig. 3.6 Separating hyperplanes examples. In this case H_3 is the maximum margin classifier.

The maximum margin classifier is then obtained by minimizing

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0,1-y_i(\vec{w}\cdot\vec{x}_i-b)\right)\right]+\lambda\|\vec{w}\|^2$$

where the parameter λ determines the trade-off between increasing the margin-size (more generalization) and ensuring that the $\vec{x}_i$ lie on the right side of the margin.

For non-linear classification the resulting algorithm is similar, except that every dot product is replaced by a nonlinear *kernel* function. (it was explained in the background section) This allows the SVM to fit the maximum-margin hyperplane in a non-linearly transformed feature space (of possibly higher dimension). The classifier is a hyperplane in the transformed feature space, but in the original input space it may be nonlinear. Multiple kernel functions are possible, but in this case the Gaussian radial basis function is used, itt is defined as:

$$k(\vec{x}_i,\vec{x}_j)=\exp(-\gamma\|\vec{x}_i-\vec{x}_j\|^2),\quad \text{for}\quad \gamma\to 0.$$

3.3.2 Classification Results

The classifier was trained via 10 fold cross-validation and the confusion matrices was computed over an ***hold-out*** (unseen) set. The accuracies results are in the Table 3.2 below.

Table 3.2 Accuracies of the classifiers

PCA	k-PCA	Autoencoder
99.5%	99.5%	99.1%

Here it can be seen the Autoencoder performs on par as the other methods. This results validate the goodness of the reduced (encoded) features from our method. High accuracies results (above $\sim 95\%$) were expected given previous literature. The results difference between the best performing and the others is only off by one prediction. The reason may be because of some parameters/tweaking, since the grid search of the best parameters were performed with the same value ranges.

Table 3.3 Confusion Matrix - Autoencoder features

Samples		True condition	
		Positive	Negative
Predicted condition	Positive	224	2
	Negative	0	17

3.4 Comparison with other cancer types

This section is will show the reader to the results obtained on different type of cancer data. All the training process is the same as the one presented so far. The datasets were presented in the *Background* with the exception of LU. This dataset is the combination of two types of lung cancer: LUASC and LUAD. Using the two datasets combined would get us a bigger dataset, almost of the size of the biggest one BRCA, while still referring to the same tissue type. The results are relative to the losses at the last epoch of training. All of the autoencoders were trained with the same fixed parameters obtained in the previous section, this of course isn't necessarily ideal, but is nonetheless representative given that the commonality between the data.

From Table 3.4 one can see how the value of the cost function changes by varying the type of cancer used as data. It can be noticed that the loss is roughly inverse linearly proportional to the number of samples present: by doubling the dataset size the loss gets halved. These results do not directly correlate with the quality of the encoding, but they can be somewhat indicative of the rate of possible improvement.

Table 3.4 Training loss on different type of cancer

Dataset	Size	Loss
BRCA	1102	2.9e-03
LU	1015	3.6e-03
KIRC	534	7.4e-03
HNSC	521	8.5e-03
THCA	513	7.2e-03
PRAD	498	9.7e-03

3.5 Summary

In this chapter the process that brought us to choose the autoencoder model, that is going to be used for gene module extraction, is introduced. The functional parameters were explained with respect to the architecture and training process. Multiple different options are possible and the reason for each choice is given relative to the problem at hand.

The training pipeline focuses mainly on two parameters: the activation function and the number of layers used. A trade-off between performance and the training time is done to enable for quick prototyping. The final model with is a four layered fully connected network. The activation function used is the TanH (non-linear) which significantly outperformed the ReLU (linear) function.

To further validate the extracted features of the autoencoder their performance in a classification scenario is compared against features obtained by two well-known methods: PCA and k-PCA. A Support Vector Classifier is trained with the data of reduced dimension obtained by the three methods. Results were on par of between all the methods and as expected from previous literature. At the end of this two stages there is a validation regarding the capability of our training model from the data standpoint. Biological considerations will be drawn further into the thesis work.

Later in this chapter it is shown, using different types of cancer, how the error changes with varying sample sizes. This was done to provide some insights into to which degree of possible improvement on the loss it's possible to obtain by having more input data.

CHAPTER 4

GENE MODULE

In this chapter the readers are presented to the concept of gene module and how the modules are extracted from the autoencoder model. Subsequently the extracted modules are visualized and filtered to understand the quality of the grouping especially on the top K genes for each modules. Also the appropriate desired number of genes for each modules is discussed. The most frequent genes over all the modules (most active in the encoding) are researched from a biological standpoint. The assumption would be that the high frequency genes in the encoding are telling of the types of modules found.

4.1 Definition

Before going more in-depth in the thesis, it is important to explain what a gene module is. In previous chapters the discussion was limited to "reduced" or "target" dimensions when talking about the middle layer of the autoencoder. The aim of the autoencoder is to find a reduced representation of the features found in the input data. Hence the input variables are mapped onto *latent* variables that can explain the data. How good do these latent variables explain the data depends on the quality of the encoding. This was explained in the previous chapter. The interesting part, around which this thesis is centered, is to open the black box and find out how this mapping works. In other words, which are the main contributors (genes) to each target dimensions (modules).

The concept of gene module exploits the modularity recognized as a design principle in biological systems. [24] The assumption (later verified) underlying the thesis work is that genes present in the same modules, extracted from the neural network, participate towards a common biological function. Genes interacts with each other in multiple ways and participate in multiple regulatory networks. A gene regulatory network (see Fig. 4.1) is a collection of molecular regulators that interact with each other and with other substances in

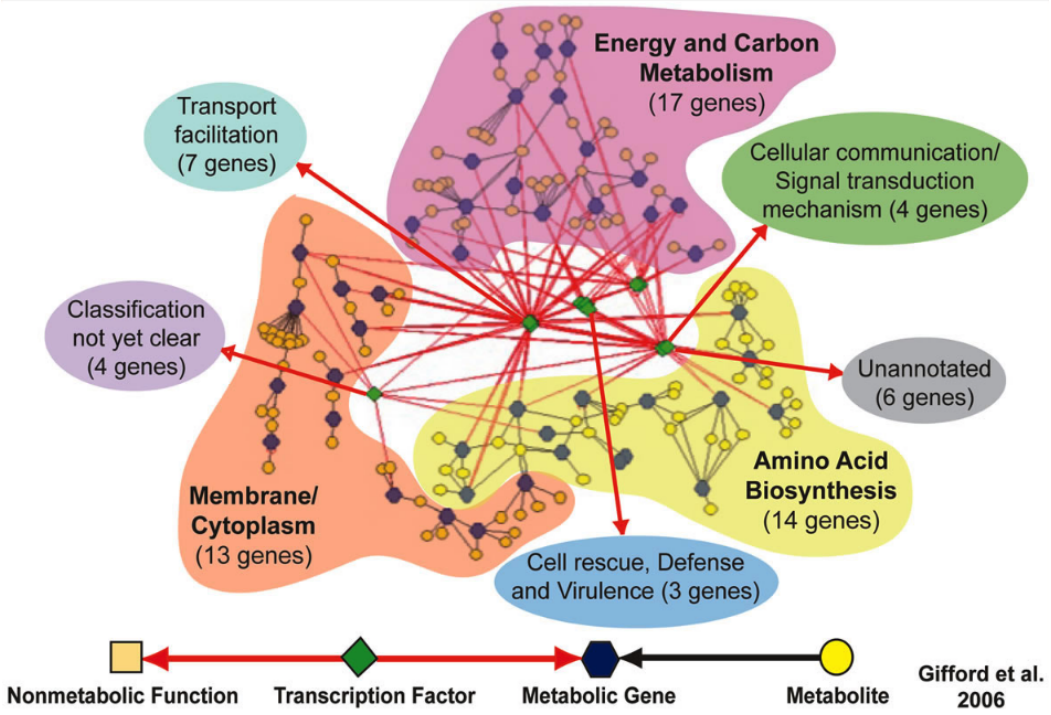


Fig. 4.1 Picture of a gene regulatory network.

the cell to govern the gene expression levels of mRNA and proteins. Modules are groupings of genes that belong to the same regulatory network. A gene can be present in *multiple networks/modules*, it can collaborate in several biological functions at the same time. With alternative splicing (see *Appendix A*) a gene can produce different proteins that contributes to different outcomes in the cellular life-cycle (there are around 20.000 genes, but 100.000 proteins).

4.2 Extraction

In this section the reader is introduced to how modules are extracted from the neural network. As explained in the chapter devoted to the *Background* the autoencoder is divided in two parts: an encoder (recognition model) and a decoder (generative model). The decoder, it's what is of interest here, it maps the code to the original input space. The code is represented, as explained in the *Model* chapter, by an array of size 400 with each dimension being calculated as a non-linear function of the starting dimensions. *Note* each dimensions in the starting feature space correspond to a gene, with the value being its expression level at a specific moment in time.

The network weights contain a representation of how genes interact with each other, combining into forming the reduced dimensions. The reduced dimensions/latent variables are hence constructed from the original variables, but how is it possible to understand the genes that contribute the most to each of them. To achieve this the focus is shifted to the decoder part of the network, which represent the inverse function of the operation performed by the encoder. To pass from the original dimensions to the reduced ones it's necessary to go through the encoder which represents the function f. Instead to go from the reduced dimensions back to the original dimensions, one has to use the decoder which represents the function f^{-1}. In fact the objective of the training process is to make the network approximate $f \circ f^{-1} = I$.

To assess the contribution of each gene (original dimension) to the latent variables (grouping) **one-hot vectors** are used as inputs of the decoder. One-hot vectors are vectors where all the elements, minus one, are equal to zero:

$$\lambda e_i = \lambda \begin{pmatrix} 0 \\ \vdots \\ 1_i \\ \vdots \\ 0 \end{pmatrix}, \qquad \lambda \in \mathbb{R}.$$

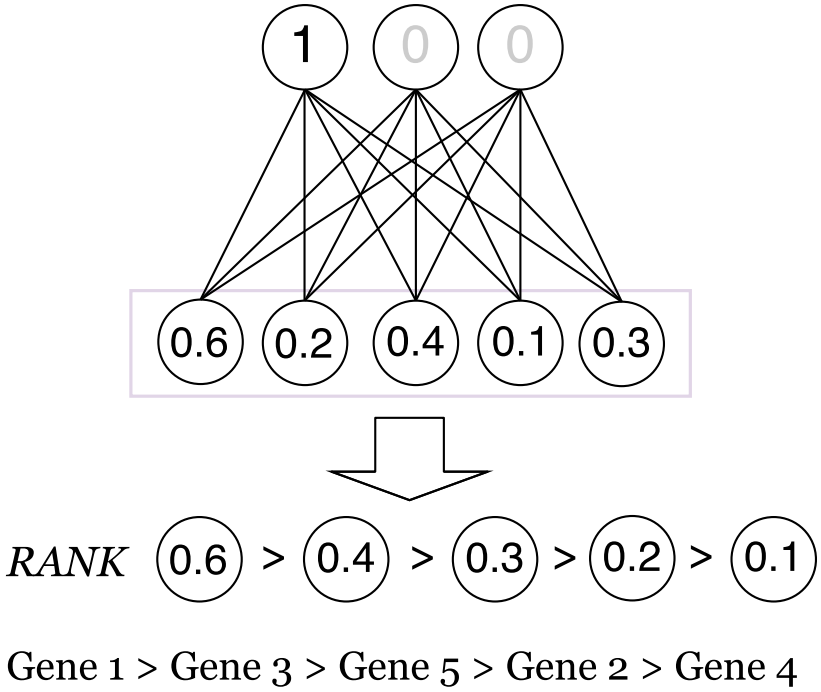
These vectors can be imagined as activating only a specific dimension and not the others. The vector e_i "queries" for the genes that contribute to the reduced dimension. Therefore by reading the output of the decoder we can gather quantitative data on how much each genes affects the said dimension. *For example*, to gather the contribution of the genes (original dimensions) to the **first dimension**/group, the one-hot vector:

$$\lambda e_1 = \lambda \begin{pmatrix} 1 \\ \vdots \\ 0_n \end{pmatrix}, \qquad \text{with the previously described model we would have } n = 400$$

will be used as an input for decoder (f^{-1}). The most "expressed" genes in the newly generated output vector would be the one with more influence over the encoding result. This operation is done for every module, obtaining:

$$r_i = f^{-1}(\lambda e_i), \qquad \forall i \in [1, n]$$

The dimensions of each vector r_i are ranked in decreasing order according to their values. The topmost dimensions of each ranking are then taken to create the final gene modules (see the *Module size* part in the next section for more informations).



Graphical representation of one-hot ranking with the decoder

Given the multiple non-linear operations applied by the autoencoder at each layer and to have a clearer numerical picture of the method (a sort of sanity-check for the method), it was tested whether the ranking is univocal between different high input values of the one-hot vector (values of λ). By "high input values" is meant that λ will take on smaller multiples of 1 (being that the largest value).

There are two motives for why 1 is used as the highest value possible, both connected to each other:

1. the **vertical asymptotes** for the activation function of each neuron (`TanH`) are ± 1;

2. the **input values** for each neuron are all positive, ranging $[0, 1]$ (the data was normalized, optimizing for faster and more stable convergence properties). It would not make much sense to use values of λ greater than the highest value ever encountered during training. The network would *not* reliably respond to such input values. This can be sees as a sort of *extrapolation-interpolation* conundrum.

The values of λ used in the test ranged between 0.6 and 1. The experiments showed that the ranking of the top influencing genes to each modules tends to stabilize with high values of λ tried. This means that intersection between each subsequent ranking grows, approaching 100% with the highest $\lambda = 1$. The last ranking, associated with the highest λ, is used as the final one from the use in the next steps.

As written, each generated vector ranks the contribution of *every* gene to each dimension/module. For enrichment analysis, though, it's not useful to use all the ranked genes. All the genes will be present and the enrichment analysis can not find an over-representation of some gene compared to another. It's necessary to limit the size of each gene sets, cutting the size at some specific rank e.g. the enrichment set has to be limited to the first K genes per module, for some value of $K \leq$ number of genes $\simeq 20.000$.

How to decide at which point to cut the ranking to produce the modules? Is there any options conducive to the next step in the experiment pipeline? Can we yet say anything qualitative about the grouping obtained in this step? Are the most frequent genes in all the modules related with each other or some specific activity (i.e. cancer in general or the type of cancer used during the training of the autoencoder)? These and other questions will be answered in the next section by visualizing and and studying the properties of the modules.

4.3 Data Exploration

In this section the reader can find an overview of the properties of the extracted modules. For brevity's sake only the case in which breast cancer data was used as starting data is taken into account. Similar consideration can nonetheless be made for other types of cancer. This section is divided in three parts. The first part will discuss about the idea of module size and how it was chosen. In the second part properties about the genes contained in the modules are explored especially for what regards the intersection between them. Finally the most frequent genes overall in the whole encoding are studied more in depth to find if there's any immediate linkage between them and cancer.

4.3.1 Module size

After having obtained the ranking of contributing genes for each modules one should decide the functional number of genes to be selected amongst the top one to be used on successive experiments. There are two possible ways of doing it: *ex ante* and *ex post*. The latter is a verification done through biological experimentations. This means that after having chosen a particular K for the cut we study the best performing modules and retroactively

tweak the value. It's the usual type of verification that is done in classical machine learning models, where a metric is optimized to indicate the update direction. This qualitative type of verification is unfeasible both for the manual intensive work done by experts and because it would void the exploratory type of analysis this method is geared towards.

> *The method presented in the thesis is to be intended as a tool. The reader could think of it as a clustering technique with two parameters: number of clusters (reduced dimensions) and cluster size (top gene per modules).*

There are nonetheless several possible ways to choose the value for K. One of them is using **expert knowledge**, by weighing in the degree of exploration: the bigger the size of the modules the larger is the number of related genes to study. On this front inspired by other works and after having asked a biologist for an expert opinion it was decided to settle on **50 genes per module**. Here 50 is chosen as a middle-ground between being too restrictive with the number of modules to enrich and taking too many of them. Using less gene would give us less surrounding genes to target for biological analysis.

Another method to choose an appropriate K, could use make some statistical consideration regarding the differential between the expression value related to each gene in the decoded vector. It's hence possible to use the standard deviation relative to the distribution of the rankings' values. In this case we can stop the module to the gene with a ranking in which the difference between successive value is greater than 2σ of the positively contributing genes. The median of the stopping index for all the 400 modules in this case is 59. (close to the *expert chosen value*)

4.3.2 Properties

The following part will be about the properties of the resulted modules. For comparison results include also the case for target module size 100. This double analysis is done also to consider how the properties evolve with increasing module size. It's interesting to see whether the modules share a large number of genes because it would be an indication that the method cannot separate gene contribution (what we are going to test from a biological standpoint with enrichment analysis). Keeping the comparison to cluster method, it's clear that in most cases, a method which returns cluster with high overlaps between all the modules is not a good one (very often overlaps are not even permitted, being the membership a non-fuzzy property). Following an overview regarding both cases of K :

- $\mathbf{K = 50}$ With this cut value, across all the modules there are 9673 unique genes present. *Remember that* there are about 20.000 overall in the dataset, that would

mean that $\sim 50\%$ of the genes are present in the final modules. Still, this does not give any indication about how many shared genes are present in each modules. The maximum percentage of genes in common between any two given modules is: 8%. The intersection is small even as an absolute value: at maximum over a module of 50 genes only 4 will be found in a different one.

- $\mathbf{K = 100}$ In this case there are 17367 unique genes present. This means that almost all the genes are present in the final representation. This could indicate a dispersion in the module specificity (capability of targeting a specific function). At maximum, this time, the modules have 5% genes in common.

From this preliminary analysis the results are comforting in the capability of the network to achieve a good separation between module (good cluster distinction). This is desirable; an high intersection/overlap would mean that the module, if related to a function, would belong to similar pathways or to the same one. In the next section the genes are looked for biological significance.

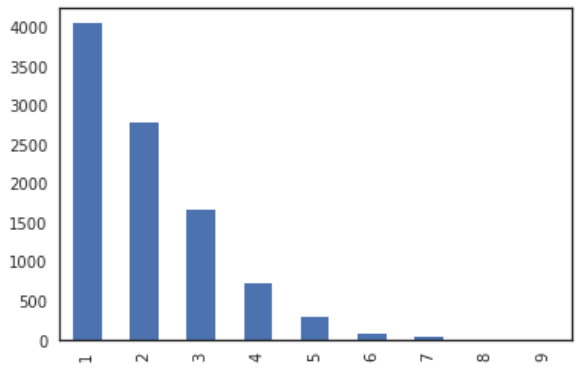
Most frequent genes

Another analysis it's possible to perform at this stage is to look at the most frequent genes present in the encoding between all the modules. In this last part of the section the distribution of genes in between the modules is investigated.

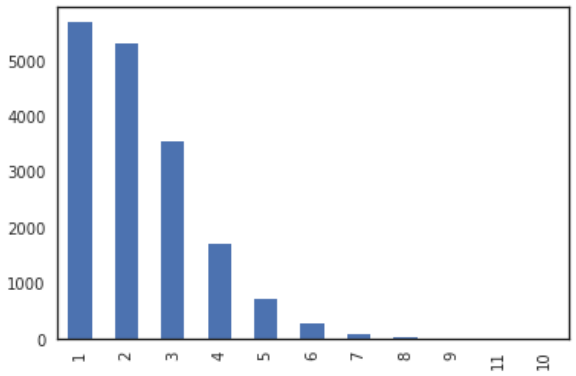
For the case in which K is equal to 50 the graph is plotted and presented in Fig. 4.2a and when it's equal 100 the result is in Fig 4.2b. The X axis here indicates the number of times a gene is present and the Y axis how many genes repeat themselves that many times. This visual representation of the data can show that that the majority of genes are never present more than 2 times between all the modules. A more detailed (readable) description of the same data is given in Table 4.1 (module size is 50) and Table 4.2 (module size 100). In the former case 4 genes are present 9 times (maximum) while in the latter there are fewer genes that are very frequent with one present 11 times and only another one 10 times. Now an in depth look to them; an evaluation to see if the most frequent genes are linked to the data the methods was trained with (*breast cancer and cancer more broadly*).

A list of the most frequent genes present when the module size is **50**:

- **PDE4DIP** was recently proposed as an highly relevant gene in breast cancer [25], it's also amongst the top 50 genes differentially expressed in invasive stroma compared to in situ stroma. [26]



(a) Most frequent genes over the top 50



(b) Most frequent genes over the top 100

- **MCTP1** possibly cancer related, it seems to be regulated in cell-lines which are drug resistant. [27] With expert knowledge (biologist) one could possibly narrow the specific interaction.
- **SNX10** is expressed in all cancer tissue in the TCGA dataset. The gene encodes a member of the sorting nexin family and plays a role in the secretion of MMP9 [28] which drives the progression of cancer. [29]
- **ATE1** encodes an arginyltransferase, an enzyme that is involved in posttranslational conjugation of arginine to N-terminal aspartate or glutamate residues. The gene is expressed in all TGGA cancer tissues and seems to act as a tumour suppressor [30] affecting DNA mutagenesis by regulating stress response [31]

Most frequent genes present when the module size is **100** (*Note that* ATE1 is the most frequent also with this choice of K):

- **CTNNA1** encodes a member of the catenin family of proteins that play an important role in cell adhesion process by connecting cadherins located on the plasma membrane to the actin filaments inside the cell. Several studies found a link between this gene and breast cancer. [32] [33] [34]
- **ATE1**, same consideration as the previous case. The repeated presence of this gene is reaffirming the capability of the method. Independently of the cutoff chosen, the most frequent gene, is cancer related.

The most frequent genes in all the modules are related to cancer and some strictly to breast cancer. Previous cited works validate the preliminary analysis over the quality of the grouping extracted from the raw breast cancer data. In the next chapter single modules are

studied more in depth, trying to linked them to a specific function, with a statistical method well used in the field of bioinformatics: enrichment analysis.

Table 4.1 Table form of the histogram relative to module size 50

Number of appearances	1	2	3	4	5	6	7	8	9
Number of genes	4048	2780	1665	733	293	95	46	9	4

Table 4.2 Table form of the histogram relative to module size 100

Number of appearances	1	2	3	4	5	6	7	8	9	10	11
Number of genes	5687	5297	3546	1707	727	267	90	31	13	1	1

4.4 Summary

This chapter introduced the concept of a gene module and in particular of gene regulatory networks. The method used to extract the individual modules is explained. It consists in using one-hot vectors as output to the decoder part of the network to obtained a measure of the magnitude of contributions of each specific gene to the extracted feature (identified as modules).

Furthermore the question of choosing an appropriate module size is investigated with one suggested and justified. The issue is equated to the one of choosing the number of clusters that are targeted in a clustering algorithm e.g. K means, or an algorithm that let the user choose the maximum cluster size. The properties of the modules are also studied, in particular: the intersection between gene modules and the amount of unique genes overall. As expected the maximum amount of shared genes between any two modules is never higher than 8% (around 4). An high percentage of shared genes between modules would have indicate a poor ability of the method to separate modules and modules' functionality.

Finally the genes which are more present overall in the encoding are studied individually. It was found that all of the top genes by frequency were also present in previous cancer study literature. Some of the genes in particular are present in papers discussing directly breast cancer, which represents the training data for the autoencoder. These results are promising as validation of the method so far, but in the next chapter this bird's-eye view is put aside and the singular modules are studied for their ability to represent/target a specific, hopefully cancer-related, functionality.

CHAPTER 5

ENRICHMENT ANALYSIS

In this chapter the modules extracted by the thesis' method are individually validated for their capability of representing a specific functionality. It's worth reviewing all the steps taken so far to check for the effectiveness of the method. Firstly the autoencoder was trained, by minimizing a cost function, to reduce the starting high-dimensional space. The reduction is tested by using the obtained dimensions in a cancer classification scenario, achieving similar results to other well-known methods. Next the model is retrained using only cancer data and a module is extracted from each dimension. The extracted modules are studied both visually and statistically, verifying that the genes are well separated: only a small number of genes are shared between any two given groups. Finally a preliminary analysis from a biological perspective was undertook, to check the function of the genes which are most frequent over all the encoding. The most frequent genes were linked to cancer or breast cancer activity by cited published works. These steps were necessary to guide the experiments, because if any of those would have yield poor results it would have been a sign that something was not correct in the initial assumptions.

The steps the reader is presented in this chapter regard the biological analysis of the modules, and can be divided in two: the **first**, test for statistical over-representation of a class of genes in the modules. The gene sets that are used for this test are already known for contributing to a specific activity. This part can be automated with software tools that query multiple well known genomic databases and perform what's called enrichment analysis. The **second step** consist in a manual verification of the enrichment results to see if the terms obtained from the enrichment are connected between each others and are related to some specific function. If there are terms (with high enrichment score) connected to cancer it would be a strong indication that the other genes in the group contribute to the same function, thus *answering* the thesis' research question. That was the original objective: providing an additional list of genes which are participating to some specific activity. The

other genes in the group then would ideally be subject to exploratory analysis (strictly from a biological/experimental standpoint) targeted for the discovery of novel gene interactions.

5.1 Overview

This section will introduce the concept of enrichment analysis and will then describe more in the depth the tool used in this thesis: Enrichr. Results from the enrichment and the biological analysis of each cluster are presented in the next section *Evaluation*.

The completion of the Human Genome Project resulted in an enormous amount of genetic information to be further studied by researchers. The problem of how to interpret and analyze this incredible amount of data represent a significant challenge for the research community. In order to seek out genes associated with diseases, researchers utilized DNA microarrays, which measure the amount of gene expression in different cells. Researchers would perform these microarrays on thousands of different genes, and compare the results of two different cell categories, e.g. normal cells versus cancerous cells. They needed a way to compare groups of genes by leveraging a previous knowledge base and to do this in a systematic way: this is done by a method called enrichment.

But what is enrichment? **(Gene) Set enrichment analysis** (GSEA) (or functional enrichment analysis) is a method used to identify classes of genes that are over-represented in a large set of genes. The analysis uses statistical methods to identify significantly enriched groups of genes. Researchers performing next-generation sequencing experiments that return sets of genes (e.g. genes that are differentially expressed under particular conditions) often want to retrieve a functional profile of that gene group, to understand better the underlying biological processes. This can be done by comparing the input gene set, that we want to *enrich*, to each of the classes (terms, such as a pathway name or a transcription factor that regulates the genes) in the gene library of reference. GSEA uses *a priori* gene sets that have been grouped together by their involvement in the same biological pathway, or by proximal location on a chromosome (human or otherwise). A statistical test can then be performed for each class to see if it is present significantly in the input genes from the experiments.

There are multiple openly accessible tools on-line to perform this analysis. In the following section the focus will shift on the tool that is used in the thesis, Enrichr, which provided a practical API that makes possible to automatically perform several enrichments at a time.

5.1.1 Enrichr

Enrichr [6] [35], our tool of choice, is a web-based application that includes several gene set libraries, multiple ways to rank enriched terms, and various visualization approaches to display results. The main indicators from its result that are used in this thesis are the resulting pairs of Term and P-value. For each gene set in the knowledge bases there's a **Term**/annotation that indicates to which activity the genes in the set are involved in. The associated **P-value** is used in order to quantify the statistical significance of the term. That is the gene in our groups are linked to the one in the knowledge base, if the associated term has low P-value. The P-value is computed using a standard statistical method: the hyper-geometric test (Fisher's exact test [36]), a test that is implemented in most gene list enrichment analyses programs. This is a binomial proportion test that assumes a binomial distribution and independence for probability of any gene belonging to any set.

In this work gene enrichment of the extracted modules is performed against *four* public gene libraries named: `'GO Molecular Function 2015'`, `'GO Cellular Component 2015'`, `'GO Biological Process 2015'` and `'KEGG 2016'` (Kyoto Encyclopedia of Genes and Genomes). *GO* stands for 'Gene Ontology' and contains gene sets created from the three gene ontology trees [37]. The Gene Ontology Consortium, the group that maintains the GO library, uses graphs as a structure for related terms, but they can be seen as a tree since all the terms in a domain can trace their parentage to the root term. To create the relative gene set libraries, the Enrichr team "cut" the tree at either the third or fourth level and created a gene set from the terms and their associated genes downstream of the cut. The reasoning behind this was avoiding the presence of very high level functions. *KEGG* instead is a collection of pathways under the then ongoing Japanese Human Genome Program. The pathways associated gene set libraries, in Enrichr were created from each of the databases by converting members of each pathway to a list of human genes.

Example

In Table 5.1 the readers can find an example of what is intended by term-value pair from the Enrichr output. The terms are ranked by the order of significance (P-value) and have not only their full name description, but also the reference code (on Gene Ontology in this case).

5.2 Evaluation

In the following section, results of the enrichment of modules obtained from two different models are evaluated. The *first model* that is presented here was trained strictly with breast

Table 5.1 Example of an Enrichr term-value pair

Term	P-value
vitamin transport (GO:0051180)	5.13154851693e-05
folic acid transport (GO:0015884)	0.000169869167294
modified amino acid transport (GO:0072337)	0.00124782046916
neurotrophin TRK receptor signaling pathway (GO:0048011)	0.000595734193651
neurotrophin signaling pathway (GO:0038179)	0.00063607205649

cancer (BRCA, Breast Invasive Carcinoma) data; while the *second model* was trained from lung cancer (LUSC, Lung Adenocarcinoma, and LUAD, Lung Squamous Cell Carcinoma) data. What was found in both cases was that:

> *the majority of modules had statistically significant P-values, showing that the method successfully managed to cluster genes for common functions.*

As written in the previous section, small P-values indicate that the groups can be linked to the relative gene set from the known databases with high probability. Hence part of the original *research question* is verified. Now the focus is shifted on the biological meaning of each cluster. What is meant by it is: after having found that the modules are able to meaningfully cluster genes, the question is, are these modules enriched for a function associated to the original data used for training? For example, are the modules obtained from the first model enriched with terms that are related to breast cancer. And for how many of the modules this holds true? This process entails manually checking each module's enriched terms and verifying: what the terms mean and if the terms are related to each other. The latter phenomenon would further help in the validation of the quality of the modules since it would mean that multiple gene sets, known for the participation into a common activity, are represented at the same time. To render this manual verification possible it was necessary to have a way of filtering the enrichment results. The modules where filtered in the P-value of their most meaningful term ($\leq e-05$). This reliably resulted in less than ~ 40 modules per each of the two model to be studied.

The rest of this section is divided in an overview of the two cancer cases with some statistics about the number of significant modules and a look over some specific modules. In both cases, thanks to the **biologist** of our laboratory, some samples of modules' enrichment are tagged with a reference to a published article that shows a proven link from the module's terms to cancer. While mostly all of the top 5 terms in each modules are related to each other not all are cancer related (or related to the specific cancer type). For brevity's sake the selection of modules presented here in the thesis is limited. Only 3 modules are focused on per each paragraph. This is solely to give to the reader an idea of what the important terms

were and the references to support their validity. In ***Appendix B*** the results for the modules that are not included here are left for the reader to view.

Important to note For the modules manually verified the explanation of their relevance to cancer was based on a conservative estimate. Meaning, any association was tried not to be made if possible, because this way the connection is as strong as possible.

5.2.1 Breast Cancer

In this part the modules extracted from the model trained on breast cancer data are going to be discussed. In the process 400 modules were enriched "against" the 4 databases mentioned in the previous section. This results in exactly 1600 tables of enrichment results. For the manual verification it was decided to focus on modules with highest enriched term having a P-value under a certain cutoff. This was done to render the manual work manageable and to include also results relative to another cancer type. The cutoff was chosen to be $e-04$ because it reliably (across different runs) gave us a smaller list of around 40 modules to go through.

Of all the 400 modules 372 had the most enriched term's value under $e-04$. The number of modules with P-value smaller than the cutoff ($e-05$) are 35. Out of all the 35 modules: 19 are cancer related with 25% of them strictly breast cancer related.

Breast Cancer related

Following an overview of three modules that were found to be related to breast cancer.

Table 5.2 Module #13 enrichment, against `GO Biological`

Term	P-value
histone H3-K4 methylation (GO:0051568)	3.2528722798e-05
histone lysine methylation (GO:0034968)	1.32739164569e-05
histone methylation (GO:0016571)	3.49558385378e-05
protein alkylation (GO:0008213)	0.000118038410346
protein methylation (GO:0006479)	0.000118038410346

The terms of Table 5.2 are all related to each other and in particular the role of histone H3-H4 acetylation dynamics was found to help defining breast cancer subtypes [38].

Table 5.3 Module #228 enrichment, against `GO Biological`

Term	P-value
single organism cell adhesion (GO:0098602)	4.38763554598e-05
wound healing, spreading of cells (GO:0044319)	0.00062991870488
negative regulation of smooth muscle cell migration (GO:0014912)	0.000397852306904
single organismal cell-cell adhesion (GO:0016337)	0.000295567877423
regulation of cell-substrate adhesion (GO:0010810)	0.000522076580859

For module 228 (Table 5.3) many enriched terms mention cell adhesion. Cell-adhesion is an important phenomenon in cancer because during its progression, cells lose their original tissue contacts and ultimately form new tumors. Tumor cells inevitably experience alterations in cell adhesion. Multiple sources discussed this phenomena particularly in relation to breast cancer [39].

Table 5.4 Module #147 enrichment, against `GO Biological`

Term	P-value
regulation of leukocyte activation (GO:0002694)	4.92711408497e-05
regulation of cell activation (GO:0050865)	7.85085852027e-05
positive regulation of leukocyte mediated immunity (GO:0002705)	4.30172751565e-05
hypothalamus development (GO:0021854)	0.000469437940862
regulation of lymphocyte activation (GO:0051249)	0.000208137182627

In reference to the terms of module 147 (Table 5.4) leukocytes related dynamics are present in the top 3 terms. Leukocytes are cells that contribute to immune response, here in particular breast cancer [40].

Cancer related

The following modules are connected to cancer more in general.

Table 5.5 Module #115 enrichment, against `GO Biological`

Term	P-value
negative regulation of response to DNA damage [...] (GO:2001021)	3.60646862779e-06
negative regulation of DNA repair (GO:0045738)	0.000169869167294
regulation of response to DNA damage stimulus (GO:2001020)	0.000260008410174
negative regulation of intrinsic apoptotic [...] (GO:1902230)	0.00177125637478
regulation of intrinsic apoptotic [...] (GO:1902229)	0.00289597063475

The reader can see how all the top terms in the enrichment result of module 115 (Table 5.5) are all related to DNA damage and apoptosis. Apoptosis is the process of cell death, an insufficient amount of it results in uncontrolled cell proliferation, such as cancer. (here an article linking the terms and cancer [41])

Table 5.6 Module #358 enrichment, against `GO Biological`

Term	P-value
negative regulation of cysteine-type endopeptidase [...] (GO:0043154)	2.65058180031e-05
negative regulation of cysteine-type endopeptidase [...] (GO:2000117)	2.96812852382e-05
negative regulation of signal transduction by p53 [...] (GO:1901797)	0.00177125637478
negative regulation of apoptotic signaling pathway (GO:2001234)	0.00121829704709
regulation of cysteine-type endopeptidase activity [...] (GO:0043281)	0.00119441970837

In module 358, clear links were found between the role of P53 and apoptosis [42] (Table 5.6).

Table 5.7 Module #370 enrichment, against `GO Biological`

Term	P-value
leukocyte mediated cytotoxicity (GO:0001909)	3.67068918761e-05
negative regulation of smooth muscle contraction (GO:0045986)	0.00054680280059
cell killing (GO:0001906)	0.000169094883171
natural killer cell mediated cytotoxicity (GO:0042267)	0.00062991870488
natural killer cell mediated immunity (GO:0002228)	0.00062991870488

The terms in module 370 (Table 5.7) were found to strongly relate to cancer [43].

5.2.2 Lung Cancer

The same treatment for breast cancer is done here for lung cancer. Following there are included the results relative to an autoencoder trained on lung cancer data: 389 modules had the most enriched term's value under $e-04$, and 38 with P-value smaller than $e-05$. Out of all the 38 modules: 20 are cancer related with 60% of them strictly lung cancer related.

Lung Cancer related

Here an overview of a sample three modules that were found to be related to lung cancer.

Table 5.8 Module #77 enrichment, against `GO Biological`

Term	P-value
purine deoxyribonucleotide metabolic process (GO:0009151)	8.04663174679e-06
2'-deoxyribonucleotide metabolic process (GO:0009394)	6.29042976962e-05
deoxyribose phosphate metabolic process (GO:0019692)	7.60844583344e-05
nucleoside diphosphate biosynthetic process (GO:0009133)	0.000272131942618
deoxyribonucleotide metabolic process (GO:0009262)	9.90436680677e-05

For module 77, nucleic acid metabolism is present in all the most important terms (see Table 5.8). Nucleic acid metabolism influences the induction of lung adenoma [44].

Table 5.9 Module #92 enrichment, against `GO Biological`

Term	P-value
mitochondrial fission (GO:0000266)	1.62655776435e-05
regulation of mitochondrial fission (GO:0090140)	0.000397852306904
positive regulation of mitochondrial fission (GO:0090141)	0.000218054077011
regulation of mitochondrion organization (GO:0010821)	0.000100907596367
mitochondrial fragmentation involved in [...] (GO:0043653)	0.000218054077011

All the enriched terms for module 92 (Table 5.9) references mitochondrial fission which is connected to lung cancer progression [45].

Table 5.10 Module #232 enrichment, against `GO Biological`

Term	P-value
negative regulation of intracellular steroid [...] (GO:0033144)	6.92895518836e-05
negative regulation of circadian rhythm (GO:0042754)	0.000469437940862
regulation of glucocorticoid secretion (GO:2000849)	0.000272131942618
entrainment of circadian clock by photoperiod (GO:0043153)	0.000913492369971
protein-chromophore linkage (GO:0018298)	0.000469437940862

In reference to module 232 in Table 5.10, links between circadian clock, glucocorticoid secretion and pulmonary inflammation was recently found [46].

Cancer related

The following 3 modules are connected to cancer more in general, not strictly lung cancer.

Table 5.11 Module #233 enrichment, against `GO Biological`

Term	P-value
regulation of transcription from RNA [...] (GO:0043618)	3.26900359274e-06
protein hydroxylation (GO:0018126)	8.04663174679e-06
regulation of DNA-templated transcription [...] (GO:0043620)	5.21939846522e-06
regulation of transcription from RNA [...] hypoxia (GO:0061418)	5.13154851693e-05
4-hydroxyproline metabolic process (GO:0019471)	0.000469437940862

Hypoxia is a key regulatory factor in tumour growth [47] (Module 233, Table 5.11).

Table 5.12 Module #308 enrichment, against `GO Biological`

Term	P-value
positive regulation of cell-matrix adhesion (GO:0001954)	2.66485589452e-06
regulation of cell-matrix adhesion (GO:0001952)	4.30172751565e-05
positive regulation of cell-substrate adhesion (GO:0010811)	8.56590675595e-05
regulation of gene expression by genetic imprinting (GO:0006349)	0.00101933263419
positive regulation of focal adhesion assembly (GO:0051894)	0.000913492369971

A similar module to 308 was found in the breast cancer case. Cell-matrix adhesion is a very important factor in cancer progression [48]. (Table 5.12)

Table 5.13 Module #331 enrichment, against `GO Molecular`

Term	P-value
intramolecular oxidoreductase activity [...] (GO:0016864)	2.18955063625e-05
protein disulfide isomerase activity (GO:0003756)	2.18955063625e-05
oxidoreductase activity, acting on a sulfur group [...] (GO:0016667)	0.000249794059724
protein disulfide oxidoreductase activity (GO:0015035)	0.00177125637478
intramolecular oxidoreductase activity (GO:0016860)	0.000351965055354

In reference to Table 5.13 of module 331, all the terms can be found to be related to cancerous activity [49].

5.3 Summary

In this chapter the concept of enrichment analysis is presented along with an overview of Enrichr, the enrichment tool used. The evaluation methodology used on the enrichment of the extracted modules is explained for each cancer type.

After the enrichment tool validated the capability of the model to group useful gene sets, the focus is shifted in interpreting the terms that were associated with the modules. A subset of the most enriched modules was studied from a biological perspective: that is, verifying that the significant terms for each modules are related to the type of data used to train the network. What was found was that a significant percentage of the most enriched modules are related to cancer and some directly linked to the specific cancer used in the training. For each case three examples are presented to the reader with a published article validating the claims. The rest of the modules that weren't found to be related to cancer still had the most enriched terms all related to each other. This indicates that the method can function more generally because it also identifies normal cell activities that of course are still present in tumor cells.

CHAPTER 6

EVALUATION

The first section of this chapter regards the stability of the method for gene clustering used in the thesis. By stability it is meant the reoccurence of modules which are enriched by the same functions across multiple versions of the model. A certain degree of repeatability would be desirable for the use of the method as a research tool. The problem is discussed with some consideration on its effects and possible solutions.

Furthermore the final section of the chapter will present the reader to similar approaches from the literature to the same problem of gene clustering. Various methods that use autoencoders are analyzed and summarized to give an overview of the differences and common points between the works.

6.1 Stability

This section is devoted to some consideration regarding the stability of method. All the models trained during our experiments yielded modules that were significant, but considerations have to be done on the consistency of the results across different runs (models trained) on the same data. The results of two single run on breast cancer data didn't return meaningful functional overlap over the most significant modules. This means that, at least, in the first ~ 40 modules obtained from two different models it wasn't possible to find the same enriched function. This it's not an indication on the stability per se, but more on the not-immediate possibility of recognizing stability. Anyway the method seems to be not as stable as desired, following some considerations.

The **main issue** with stability is inherent with the small number of data that is available. The model training process has many variables that depends on random initialization. For example, the samples are subdivided randomly in two sets: the *training set*, used for effectively tweaking the weights of the network, and the *validation set*, used to check the progression

of the training at each update. With a dataset of limited size, as the one used, the impact on the training results can be significant due to the high intrinsic variance of the training set (it can happen by excluding important samples for certain activities). Another parameter affected by random initialization is the set of weights of the neural network, hundreds of thousands, which are sampled randomly from a limited uniform distribution. Theses issues are compounded in such an high dimensional space by the difficulties of dealing with saddle points and multiple local minima [50].

The combination of the random initial conditions and the limited number of samples results in fluctuations in the final groupings.

Without changing the model so far, these problems could be solved in two ways:

1. Increasing the **number of samples**: $n \to \infty$ would stabilize the method. With an higher number of samples the initial split of the data would not affect much the results, reducing the variance of the splitted data [51]. This is because more samples containing specific patterns would be available and the gradient wouldn't fluctuate as much. While this approach is certainly the best one, at the moment it is not foreseeable to obtain a dataset of much bigger size.

2. Increasing the **training sessions**: $t \to \infty$. The alternative to having more samples would be verifying that enrichment results *converge* across multiple trained models. This approach presents other issues. For starter, the hardware used is a significant handicap to the iteration speed of the training pipeline. The training of each individual model requires multiple hours at a time and the enrichment analysis is also a slow process (it queries for all the extracted modules, 4 on-line databases). The **biggest bottleneck** though, is the manual validation of the data that let us link the terms in a module to cancerous activities. Small differences between any two given modules would result in different P-values and modules that were present under a certain cutoff of the P-value could now not be considered. The manual comparison between two runs would require, to be as thorough as possible, to check $4 \times (400 \times 400)$ couples of results.

To conclude the only thing that can be stated is that the method finds meaningful groupings with a certain level of significance (P-value) to which is possible associated a biological activity. Following some attempts to achieve stability are explained.

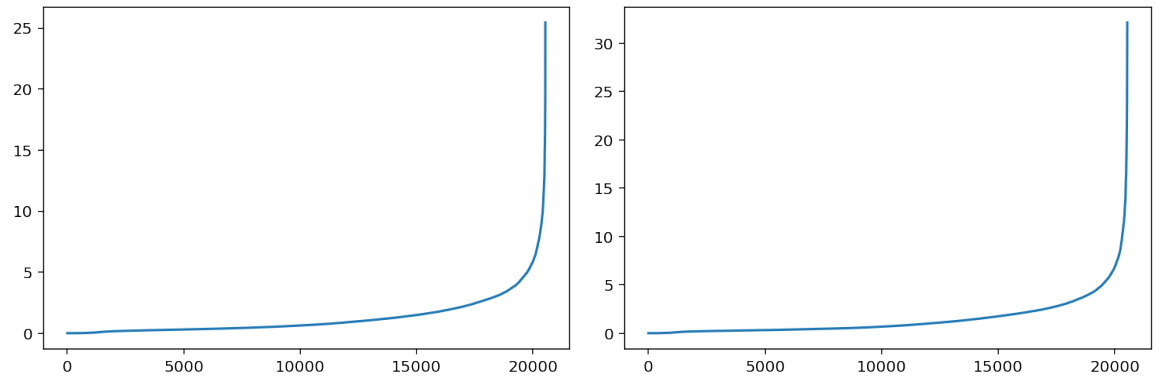


Fig. 6.1 Ordered variance for each of the 20000 genes in the data. Breast (left), Lung (right)

6.1.1 Possible solutions

Multiple attempts were undertook to stabilize the results of our method. While they are intended for future works here some preliminary results are presented. The hope is that the following assessments could serve as a basis for improvements.

The **first** idea is about additional preprocessing for the dataset. This entails diminishing the dimensionality of the input data for the model via *feature selection*. For example, by removing feature with low variance, which don't affect much the reduced dimensions, it would be possible to make the training process faster and minimize the risk of getting stuck during it. The ordered variance of each of the 20000 features is plotted in Fig. 6.1 for both dataset. It is possible to see that approximately a quarter of the features are the one that accounts for the majority of the variance. Hence reducing the data in the beginning to the features that have more variance could prove beneficial.

The **second** possibility would consists in inducing *sparsity* in the output of the middle layer. By sparsity is meant that only one dimension is active at any single time. This would have two effects: orthogonalize the extracted dimensions to minimize their interdependence, thus increasing module separation; and possibly reducing the solution space, making the results more stable. In theory the L_0 penalty ("norm") would have to be the added term to the objective function, where the L_0 ($\lambda \sum_{i=1}^{n} |o_i|^0$) is the number of non-zero components of the output vector. Optimization though in this case reduces to a combinatorial search and its time complexity is very difficult to optimize. L_1 norms ($\lambda \sum_{i=1}^{n} |o_i|^1$) are the ones which are usually used in its place; they function as proxies to L_0 norms and they work by promoting sparsity because of the shape of their feasible set of solutions.

The above ideas were **implemented** with some successes: they increased the inter-modules overlap between different runs on the same dataset and they resulted in faster convergence. Some preliminary enrichment results suggests high functional overlap not

only between module of different runs, but also between the modules in a single run. While these results are promising they require more experiments, and biological evaluations of the enrichment results, which are left for future works.

6.2 Related works

Following there is an overview of related works from literature that used deep autoencoders applied to the discover of relevant genes. All articles reaffirm that autoencoders have the capability to successfully identify input combinations that affect the overall set of values. [4] [52]. Across the different works, the commonalities and the differences with our approach are summarized to give a better support for the theoretical concepts discussed in the thesis.

Here a list of the articles with a short summary:

1. **ADAGE-Based Integration of Publicly Available Pseudomonas aeruginosa Gene Expression Data with Denoising Autoencoders Illuminates Microbe-Host Interactions** [53] [54]

   This article has a very similar overall process to the one presented in the thesis, but it is applied on a different dataset. The main difference with our method is the way they extract the membership of each gene to the clusters. Even though the method is comparable, they make considerations on the weights between the layers instead than directly measuring the output of the decoder.

   The target dimensions for the middle layer are fewer: "A model with 50 nodes was chosen to balance reconstruction error with the need to manually interpret the ADAGE model, and our subsequent analyses demonstrated that networks of this size are capable of adequately extracting major global transcriptional patterns".

   They expect a larger number of nodes to improve the results (*like it is done in this thesis* 400 *nodes*), writing: "We demonstrated the biological relevance of a 50-node ADAGE model, and we expect that increasing the node number will allow for further separation of distinct processes with independent transcriptional signatures.".

   They also mention what we referred as *stability* without seemingly solving it.

2. **Using neural networks for reducing the dimensions of single-cell RNA-Seq data** [55]

   The model validation of this training resulted in an autoencoder with mostly the *same parameters* as ours: same activation function, loss function, weight initialization,

optimizer (mini-batch gradient descent). Also a similar number of training epochs for the training were undertook (100 v. 200).

The main difference in this approach is in the number of hidden layers of their model (2, while in the thesis we use 4) and the number of reduced dimensions (100, while in the thesis we use 400).

Our choice of using more hidden layers is supported by their conclusion: "While the results are encouraging, there are several directions for future work which we would like to explore. These include testing more involved (deeper) architectures".

Unlike our method they don't justify the cutoffs used beside just stating: "top 10 most highly weighted (hidden layer) nodes", "nodes with a high (absolute value) weight"

3. **Boosting Gene Expression Clustering with System-Wide Biological Information: A Robust Autoencoder Approach** [56]

   This article is also similar conceptually to the thesis' work, but uses yet another method for extracting the clusters (it is based on network considerations).

   Furthermore, the researchers don't justify the number of target modules "[...] we set the number of clusters to be 100" (as the second paper). The results they provide are not validated biologically by individually checking the enrichments terms, unlike in our case.

4. **A Deep Learning Approach For Cancer Detection And Relevant Gene Identification.** [22]

   In this case the researchers used the *same method for validating the autoencoder parameters* (comparison in a classification setting). The dataset used in this case is limited to breast cancer and the preprocessing is different: they used over-sampling to balance healthy and cancerous samples for the training (in our case we limit the data to cancer samples).

   Yet another difference is their method for extracting genes from the network. It's much more similar to the one used in the first article, but instead of finding gene modules they find a collection of the most influencing genes over all the encoding.

5. **Learning influential genes on cancer gene expression data with stacked denoising autoencoders** [57]

   Very similar to the previous article. It uses the *same number* of target dimensions as done in *this* thesis: "The number of features for the final representations was chosen to

be 400 for comparison with other methods as the number of final features should not be higher than the number of samples.".

The enrichment results seem to be worse overall. This is probably because of some parameters chosen for the model. The results are not validated by a specialist/biologist: "we did not have the opportunity to validate the results with a thyroid cancer specialist".

6. **Learning structure in gene expression data using deep architectures, with an application to gene clustering** [58]

   This is the most different article in this list. It limits the use of autoencoders as a preprocessing step for another method. The modules extraction is different (based on another work [59]) as the dataset used.

From the previous literature overview it can be gathered that there are multiple similarities between their method and ours. These are not only in the parameters used in the autoencoder's architecture, but also in the use of classification for the preliminary testing of the reduced features. All the methods at the end concluded with some type of enrichment analysis, but most of them lacked in providing any biological evaluation of the enriched terms.

6.3 Summary

In this chapter some considerations regarding the method were made: in particular the issue with the stability of the results was discussed along with the comparison of our approach to similar works in the literature.

The first section is dedicated to a discussion on the *stability*. While the results of autoencoders trained on the same data across different runs all contain meaningful connected terms in the majority of the modules; the functional terms do not reoccur between different models. Some motivations were given to this phenomenon and their relation to the inherent difficulties given by the problem's dimensionality and the effects of the randomized initialization of the network parameters. Possible *solutions* were also discussed and implemented to evaluate if any improvement could be witnessed. The initial results are promising and could help the development of future attempts to the stabilization.

The last section contains an overview of similar methods/articles in the literature to provide a better foundation to the overall presented process. A brief summary of each one of the articles is attached to provide a quick resource for the differences and similarities between the approaches. Similar network parameters (number of layer, activation functions, ...) to the one found in this thesis are used over multiple works, validating the approach and

the parameters selected in our model validation. The use of a large number of neurons, such as 400, in the middle layer was also suggested by multiple articles in their conclusions and future works' suggestions.

CHAPTER 7

CONCLUSIONS AND FUTURE WORKS

Some tools were produced as a way of making the presented method more accessible and comprehensive. The tools are described in ***Appendix C****.*

In this thesis the problem of gene clustering with deep autoencoders has been addressed. Gene clustering is the task of grouping multiple genes that collaborate in a shared biological function, these genes form a so called 'gene module'. Autoencoders, a particular class of neural network architectures used for dimensionality reduction, were leveraged to extract the modules. The research questions that were posed at the beginning of this thesis were: *Can a deep autoencoder group gene that are involved in a common biological activity? And will these activities be linked to the type of cancer used as training data of the model?* To answer these questions we have proposed a method for extracting gene modules from each of the reduced dimensions of the middle layer of the autoencoder. The dataset used is composed by healthy and cancer samples coming from RNA-seq experiments of different types of cancer. The data is high dimensional, containing ~ 20000 features, each representing the expression level of a specific gene. The later verified assumption that was made initially was that deep architectures are able to better match the complex interactions of human biology with respect to simpler models. The whole pipeline of the experiments done in the thesis can be divided in three parts, with each step functioning also as validation of the various design choices.

In the *first part* the autoencoder model is constructed by using all the data available to find the optimal parameters. The model's parameters are found by multiple experiments that ended up in the choice of the ones belonging to the autoencoder that was able to better minimize the cost function without loosing the ability to generalize on unseen samples. The resulting architecture was validated by confronting the performance of the reduced features to the one belonging to other well known methods in a classification scenario. The autoencoder's features performed on the same level ($> 95\%$ accuracy) as the one of the other

methods, reaching the expected performance targets from previous literature results. These results prove the capability of the model to preserve information, at least the one relative to whether the sample is cancerous or not.

After the viability of autoencoder extracted features was verified, the model was retrained using only cancerous samples. This was done to better capture tumor activities, since that is our primary goal. The gene modules are then extracted from the neural network. This was done by using only the decoder part of the autoencoder. The decoder takes a vector from the reduced space and returns one from the original space. By using one-hot vectors (e_i, vectors with only one non-zero element) we gathered a ranking of the most expressed genes in the resulting output. The top 50 genes for each dimension were taken to form a module, ending up with a total of 400 modules. Preliminary analysis over the most frequent genes in all the modules was done, verifying that all of them are related to cancer. The analysis proved beneficial in being a sort of sanity check for the modules' significance.

In the *final part*, the quality of the genes in the modules was verified by using a method called Enrichment Analysis. This method checks whether some set of genes known to contribute to a specific function is over-represented in our modules. If there are more genes from a specific set than one could normally expect in a random grouping, that set is said to be 'enriched' for that specific function (term) with an associated significance level (given by its P-value). Breast and lung cancer were the focus for this section; an autoencoder was trained using each of the two different datasets and modules extracted from it. The *majority* of the modules extracted from each of the two models had significant P-value ($\leq e-04$) answering positively to our first research question about the autoencoder's ability of grouping genes into 'useful' sets.

To answer the second research question regarding the biological interpretation of each module, a step of manual verification is needed. This step entails *manually* checking for relatedness of each module's most enriched terms and a possible link to, if possible, cancer activity via some published articles. To do this, the set of modules under examination was filtered by the P-value of their most significant term ($\leq e-05$) to reduce the number of modules to be manually studied. Pretty much all of the resulting modules have the most significant terms related to each other. Around half of the modules, in both the breast and lung cancer case, contain cancer related terms, with a subset (25% for the breast cancer case and 60% for the lung cancer one) strictly related to the type of cancer the autoencoder was trained with. The other half of the modules contains terms which are more general, being related to 'generic' cell activities. These last two facts answers our second and final initially posed question: the method for the extraction of cluster of genes from the autoencoder not

only can group related genes, but also with a non-trivial subset of them *related to the specific type of cancer the data belonged to*.

To conclude the thesis' work we evaluated the method against other similar approaches in the literature and we found several similarities. Many of our findings on the optimal parameters were also present in other articles. The choice of a relatively 'large' size for the middle layer (400) was also mentioned across multiple articles as a possible future development to increase the quality of the resulting gene sets. The issue of the stability of results, across different autoencoder models trained on the same data, was also explained in the previous chapter. Multiple possible solutions were proposed and implemented. The stability of the results improved significantly, but other experiments are needed.

For the future it would be interesting to solve the stability of the method. Possible improvements could be achieved with the use of transfer learning techniques to make up for the limited number of samples. This would allow the method to be included as a mainstay in biological research settings.

REFERENCES

[1] Chun-Qi Li, Ana I Robles, Christin L Hanigan, Lorne J Hofseth, Laura J Trudel, Curtis C Harris, and Gerald N Wogan. Apoptotic signaling pathways induced by nitric oxide in human lymphoblastoid cells expressing wild-type or mutant p53. *Cancer research*, 64(9):3022–3029, 2004.

[2] Ana I Robles, Nicole A Bemmels, Amy B Foraker, and Curtis C Harris. Apaf-1 is a transcriptional target of p53 in dna damage-induced apoptosis. *Cancer research*, 61(18):6660–6664, 2001.

[3] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

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

[5] Katarzyna Tomczak, Patrycja Czerwińska, and Maciej Wiznerowicz. The cancer genome atlas (tcga): an immeasurable source of knowledge. *Contemporary oncology*, 19(1A):A68, 2015.

[6] Edward Y Chen, Christopher M Tan, Yan Kou, Qiaonan Duan, Zichen Wang, Gabriela Vaz Meirelles, Neil R Clark, and Avi Ma'ayan. Enrichr: interactive and collaborative html5 gene list enrichment analysis tool. *BMC bioinformatics*, 14(1):128, 2013.

[7] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[9] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000.

[10] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[11] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[12] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[13] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[14] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[15] François Chollet et al. Keras, 2015.

[16] Wes McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, pages 1–9, 2011.

[17] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[18] Rich Caruana, Steve Lawrence, and C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408, 2001.

[19] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[20] Yann LeCun, Ido Kanter, and Sara A Solla. Second order properties of error surfaces: Learning time and generalization. In *Advances in neural information processing systems*, pages 918–924, 1991.

[21] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[22] Padideh Danaee, Reza Ghaeini, and David A Hendrix. A deep learning approach for cancer detection and relevant gene identification. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017*, pages 219–229. World Scientific, 2017.

[23] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[24] Ziv Bar-Joseph, Georg K Gerber, Tong Ihn Lee, Nicola J Rinaldi, Jane Y Yoo, François Robert, D Benjamin Gordon, Ernest Fraenkel, Tommi S Jaakkola, Richard A Young, et al. Computational discovery of gene modules and regulatory networks. *Nature biotechnology*, 21(11):1337, 2003.

[25] Ritika R. Samant and Rajeev S. Samant. Abstract 1503: Discovery of new breast cancer relevant genes using the cancer genomics public database. *Cancer Research*, 76(14 Supplement):1503–1503, 2016.

[26] Xiao-Jun Ma, Sonika Dahiya, Elizabeth Richardson, Mark Erlander, and Dennis C. Sgroi. Gene expression profiling of the tumor microenvironment during breast cancer progression. *Breast Cancer Research*, 11(1):R7, Feb 2009.

[27] Radosław Januchowski, Karolina Sterzyńska, Piotr Zawierucha, Marcin Ruciński, Monika Świerczewska, Małgorzata Partyka, Katarzyna Bednarek-Rajewska, Maciej Brązert, Michał Nowicki, Maciej Zabel, et al. Microarray-based detection and expression analysis of new genes associated with drug resistance in ovarian cancer cell lines. *Oncotarget*, 8(30):49944, 2017.

[28] Chun Zhou, Ying Wang, Jin Peng, Cuixian Li, Peiqing Liu, and Xiaoyan Shen. Snx10 plays a critical role in mmp9 secretion via jnk-p38-erk signaling pathway. *Journal of Cellular Biochemistry*, 118(12):4664–4671, 2017.

[29] Christine Mehner, Alexandra Hockla, Erin Miller, Sophia Ran, Derek Radisky, and Evette Radisky. Tumor cell-produced matrix metalloproteinase 9 (mmp-9) drives malignant progression and metastasis of basal-like triple negative breast cancer. 5, 05 2014.

[30] Reena Rai, Fangliang Zhang, Kristen Colavita, Nicolae Adrian Leu, Satoshi Kurosaka, Akhilesh Kumar, Michael D Birnbaum, Balázs Győrffy, Dawei W Dong, Michael Shtutman, et al. Arginyltransferase suppresses cell tumorigenic potential and inversely correlates with metastases in human cancers. *Oncogene*, 35(31):4058, 2016.

[31] Akhilesh Kumar, Michael D Birnbaum, Devang M Patel, William M Morgan, Jayanti Singh, Antoni Barrientos, and Fangliang Zhang. Posttranslational arginylation enzyme ate1 affects dna mutagenesis by regulating stress response. *Cell death & disease*, 7(9):e2378, 2016.

[32] David W Craig, Joyce A O'Shaughnessy, Jeffrey A Kiefer, Jessica Aldrich, Shripad Sinari, Tracy M Moses, Shukmei Wong, Jennifer Dinh, Alexis Christoforides, Joanne L Blum, et al. Genome and transcriptome sequencing in prospective metastatic triple-negative breast cancer uncovers therapeutic vulnerabilities. *Molecular cancer therapeutics*, 12(1):104–116, 2013.

[33] LI Ding, Matthew J Ellis, Shunqiang Li, David E Larson, Ken Chen, John W Wallis, Christopher C Harris, Michael D McLellan, Robert S Fulton, Lucinda L Fulton, et al. Genome remodelling in a basal-like breast cancer metastasis and xenograft. *Nature*, 464(7291):999, 2010.

[34] Teresa L Mastracci, Suzanna Tjan, Anita L Bane, Frances P O'Malley, and Irene L Andrulis. E-cadherin alterations in atypical lobular hyperplasia and lobular carcinoma in situ of the breast. *Modern Pathology*, 18(6):741, 2005.

[35] Maxim V Kuleshov, Matthew R Jones, Andrew D Rouillard, Nicolas F Fernandez, Qiaonan Duan, Zichen Wang, Simon Koplev, Sherry L Jenkins, Kathleen M Jagodnik, Alexander Lachmann, et al. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. *Nucleic acids research*, 44(W1):W90–W97, 2016.

[36] Ronald A Fisher. On the interpretation of χ 2 from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, 85(1):87–94, 1922.

[37] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25, 2000.

[38] Terri L Messier, Jonathan AR Gordon, Joseph R Boyd, Coralee E Tye, Gillian Browne, Janet L Stein, Jane B Lian, and Gary S Stein. Histone h3 lysine 4 acetylation and methylation dynamics define breast cancer subtypes. *Oncotarget*, 7(5):5094, 2016.

[39] Alaa Afify, Phillip Purnell, and Laura Nguyen. Role of cd44s and cd44v6 on human breast cancer cell adhesion, migration, and invasion. *Experimental and molecular pathology*, 86(2):95–100, 2009.

[40] Sabina Sangaletti, Claudio Tripodo, Sara Sandri, Ilaria Torselli, Caterina Vitali, Chiara Ratti, Laura Botti, Alessia Burocchi, Rossana Porcasi, Andrea Tomirotti, Mario P Colombo, and Claudia Chiodoni. Osteopontin shapes immune suppression in the metastatic niche. *Cancer Research*, 2014.

[41] Constantinos G Broustas and Howard B Lieberman. Dna damage response genes and the development of cancer metastasis. *Radiation research*, 181(2):111–130, 2014.

[42] Feng Chen, Daniel Chang, Meidee Goh, Sergey A Klibanov, and Mats Ljungman. Role of p53 in cell cycle regulation and apoptosis following exposure to proteasome inhibitors. *Cell Growth and Differentiation-Publication American Association for Cancer Research*, 11(5):239–246, 2000.

[43] Jun Wu and Lewis L Lanier. Natural killer cells and cancer. *Advances in cancer research*, 90(1):127–56, 2003.

[44] Alvin M Kaye and Nathan Trainin. Urethan carcinogenesis and nucleic acid metabolism: factors influencing lung adenoma induction. *Cancer research*, 26(10):2206–2212, 1966.

[45] Jalees Rehman, Hannah J Zhang, Peter T Toth, Yanmin Zhang, Glenn Marsboom, Zhigang Hong, Ravi Salgia, Aliya N Husain, Christian Wietholt, and Stephen L Archer. Inhibition of mitochondrial fission prevents cell cycle progression in lung cancer. *The FASEB Journal*, 26(5):2175–2186, 2012.

[46] Julie Gibbs, Louise Ince, Laura Matthews, Junjie Mei, Thomas Bell, Nan Yang, Ben Saer, Nicola Begley, Toryn Poolman, Marie Pariollaud, et al. An epithelial circadian clock controls pulmonary inflammation and glucocorticoid action. *Nature medicine*, 20(8):919, 2014.

[47] Adrian L Harris. Hypoxia—a key regulatory factor in tumour growth. *Nature Reviews Cancer*, 2(1):38, 2002.

[48] Judith Katto and Ulrich Mahlknecht. Epigenetic regulation of cellular adhesion in cancer. *Carcinogenesis*, 32(10):1414–1418, 2011.

[49] Tsutomu Tanaka, Goro Kutomi, Toshimitsu Kajiwara, Kazuharu Kukita, Vitaly Kochin, Takayuki Kanaseki, Tomohide Tsukahara, Yoshihiko Hirohashi, Toshihiko Torigoe, Yoshiharu Okamoto, et al. Cancer-associated oxidoreductase ero1-α drives the production of vegf via oxidative protein folding and regulating the mrna level. *British journal of cancer*, 114(11):1227, 2016.

[50] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[51] Damien Brain and G Webb. On the effect of data set size on bias and variance in classification learning. In *Proceedings of the Fourth Australian Knowledge Acquisition Workshop, University of New South Wales*, pages 117–128, 1999.

[52] Alex Krizhevsky and Geoffrey E Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, 2011.

[53] Jie Tan, John H Hammond, Deborah A Hogan, and Casey S Greene. Adage-based integration of publicly available pseudomonas aeruginosa gene expression data with denoising autoencoders illuminates microbe-host interactions. *MSystems*, 1(1):e00025–15, 2016.

[54] Jie Tan, Georgia Doing, Kimberley A Lewis, Courtney E Price, Kathleen M Chen, Kyle C Cady, Barret Perchuk, Michael T Laub, Deborah A Hogan, and Casey S Greene. Unsupervised extraction of stable expression signatures from public compendia with an ensemble of neural networks. *Cell systems*, 5(1):63–71, 2017.

[55] Chieh Lin, Siddhartha Jain, Hannah Kim, and Ziv Bar-Joseph. Using neural networks for reducing the dimensions of single-cell rna-seq data. *Nucleic acids research*, 45(17):e156–e156, 2017.

[56] Hongzhu Cui, Chong Zhou, Xinyu Dai, Yuting Liang, Randy Paffenroth, and Dmitry Korkin. Boosting gene expression clustering with system-wide biological information: A robust autoencoder approach. *bioRxiv*, page 214122, 2017.

[57] Vítor Teixeira, Rui Camacho, and Pedro G Ferreira. Learning influential genes on cancer gene expression data with stacked denoising autoencoders. In *Bioinformatics and Biomedicine (BIBM), 2017 IEEE International Conference on*, pages 1201–1205. IEEE, 2017.

[58] Aman Gupta, Haohan Wang, and Madhavi Ganapathiraju. Learning structure in gene expression data using deep architectures, with an application to gene clustering. In *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, pages 1328–1335. IEEE, 2015.

[59] Ka Yee Yeung and Walter L. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[60] Eva Wertheimer, Alvaro Gutierrez-Uzquiza, Cinthia Rosemblit, Cynthia Lopez-Haber, Maria Soledad Sosa, and Marcelo G Kazanietz. Rac signaling in breast cancer: a tale of gefs and gaps. *Cellular signalling*, 24(2):353–362, 2012.

[61] Sarah C Baumgarten and Jonna Frasor. Minireview: inflammation: an instigator of more aggressive estrogen receptor (er) positive breast cancers. *Molecular Endocrinology*, 26(3):360–371, 2012.

[62] Elisabet E Manasanch and Robert Z Orlowski. Proteasome inhibitors in cancer therapy. *Nature Reviews Clinical Oncology*, 14(7):417, 2017.

[63] Qinghua Xiong, Qian Shi, Xiangdong Le, Bailiang Wang, and Keping Xie. Regulation of interleukin-8 expression by nitric oxide in human pancreatic adenocarcinoma. *Journal of Interferon & Cytokine Research*, 21(7):529–537, 2001.

[64] Elitsa Ananieva. Targeting amino acid metabolism in cancer growth and anti-tumor immune response. *World journal of biological chemistry*, 6(4):281, 2015.

[65] Sonia B Jakowlew, Terry W Moody, and Jennifer M Mariano. Transforming growth factor-beta receptors in human cancer cell lines: analysis of transcript, protein and proliferation. *Anticancer research*, 17(3C):1849–1860, 1997.

[66] Min Shen, Sara Schmitt, Daniela Buac, and Q Ping Dou. Targeting the ubiquitin–proteasome system for cancer therapy. *Expert opinion on therapeutic targets*, 17(9):1091–1108, 2013.

[67] Massimo Bonora and Paolo Pinton. The mitochondrial permeability transition pore and cancer: molecular mechanisms involved in cell death. *Frontiers in oncology*, 4:302, 2014.

[68] Matteo Parri and Paola Chiarugi. Rac and rho gtpases in cancer cell motility control. *Cell Communication and Signaling*, 8(1):23, 2010.

[69] Peter Friedl and Darren Gilmour. Collective cell migration in morphogenesis, regeneration and cancer. *Nature reviews Molecular cell biology*, 10(7):445, 2009.

[70] Linara Gabitova, Andrey Gorin, and Igor Astsaturov. Molecular pathways: sterols and receptor signaling in cancer. *Clinical Cancer Research*, 20(1):28–34, 2014.

[71] Subhash C. Kukreja, Walter P. Shemerdiak, Thomas E. Lad, and Patricia A. Johnson. Elevated nephrogenous cyclic amp with normal serum parathyroid hormone levels in patients with lung cancer. *The Journal of Clinical Endocrinology & Metabolism*, 51(1):167–169, 1980.

[72] Haruhisa Kitano, Joon-Yong Chung, Kyung Hee Noh, Young-Ho Lee, Tae Woo Kim, Seok Hyung Lee, Soo-Heang Eo, Hyung Jun Cho, Chel Hun Choi, Shuhei Inoue, et al. Synaptonemal complex protein 3 is associated with lymphangiogenesis in non-small cell lung cancer patients with lymph node metastasis. *Journal of translational medicine*, 15(1):138, 2017.

[73] Tatjana Mijatovic, Isabelle Roland, Eric Van Quaquebeke, B Nilsson, Anne Mathieu, Frank Van Vynckt, Francis Darro, Gustavo Blanco, Vincenzo Facchini, and Robert Kiss. The $\alpha 1$ subunit of the sodium pump could represent a novel target to combat non-small cell lung cancers. *The Journal of pathology*, 212(2):170–179, 2007.

[74] Takashi Takahashi, Marion M Nau, Itsuo Chiba, Michael J Birrer, Richard K Rosenberg, Michelle Vinocour, Mark Levitt, Harvey Pass, Adi F Gazdar, and John D Minna. p53: a frequent target for genetic abnormalities in lung cancer. *Science*, 246(4929):491–494, 1989.

[75] Roy M Bremnes, Robert Veve, Fred R Hirsch, and Wilbur A Franklin. The e-cadherin cell–cell adhesion complex and lung cancer invasion, metastasis, and prognosis. *Lung cancer*, 36(2):115–124, 2002.

[76] Yun-Neng Tang, Wei-Qiao Ding, Xiao-Jie Guo, Xin-Wang Yuan, Dong-Mei Wang, and Jian-Guo Song. Epigenetic regulation of smad2 and smad3 by profilin-2 promotes lung cancer growth and metastasis. *Nature communications*, 6:8230, 2015.

[77] Ioannis A Voutsadakis. Peroxisome proliferator activated receptor-γ and the ubiquitin-proteasome system in colorectal cancer. *World journal of gastrointestinal oncology*, 2(5):235, 2010.

[78] Nicholas Turner and Richard Grose. Fibroblast growth factor signalling: from development to cancer. *Nature Reviews Cancer*, 10(2):116, 2010.

[79] Pascal Mariot, Natalia Prevarskaya, Morad M Roudbaraki, Xuefen Le Bourhis, Fabien Van Coppenolle, Karine Vanoverberghe, and Roman Skryma. Evidence of functional ryanodine receptor involved in apoptosis of prostate cancer (lncap) cells. *The Prostate*, 43(3):205–214, 2000.

[80] Maryam K Mohammed, Connie Shao, Jing Wang, Qiang Wei, Xin Wang, Zachary Collier, Shengli Tang, Hao Liu, Fugui Zhang, Jiayi Huang, et al. Wnt/β-catenin signaling plays an ever-expanding role in stem cell self-renewal, tumorigenesis and cancer chemoresistance. *Genes & diseases*, 3(1):11–40, 2016.

APPENDIX A

BIOLOGY HANDBOOK

DNA

Deoxyribonucleic acid or DNA is a thread-like chain of nucleotides. DNA stores biological information and more specifically it carries the genetic instructions used in the development, functioning, growth and reproduction of all known living organisms and viruses. DNA and ribonucleic acid (RNA) are nucleic acids; alongside proteins, lipids and complex carbohydrates (polysaccharides), they are one of the four major types of macromolecules that are essential for all known forms of life. Most DNA molecules consist of two biopolymer strands coiled around each other to form a double helix. The DNA backbone is resistant to cleavage (bond splitting), and both strands of the double-stranded structure store the same biological information. This information is replicated as and when the two strands separate. A large part of DNA (more than 98% for humans) is non-coding, meaning that these sections do not serve as patterns for protein sequences, but they may have some other functions (such as regulators, enhancers, etc. . .). The two strands of DNA run in opposite directions to each other and are thus antiparallel. Attached to each sugar is one of four types of nucleobases (informally, bases). It is the sequence of these four nucleobases along the backbone that encodes biological information.

RNA

Ribonucleic acid or RNA is a polymeric molecule (nucleic acid) essential in various biological roles, such as: coding, decoding, regulation, and expression of genes. It can be of several types. Many viruses encode their genetic information using an RNA genome. Some RNA molecules play an active role within cells by catalysing biological reactions, controlling gene expression, or sensing and communicating responses to cellular signals. One of these active

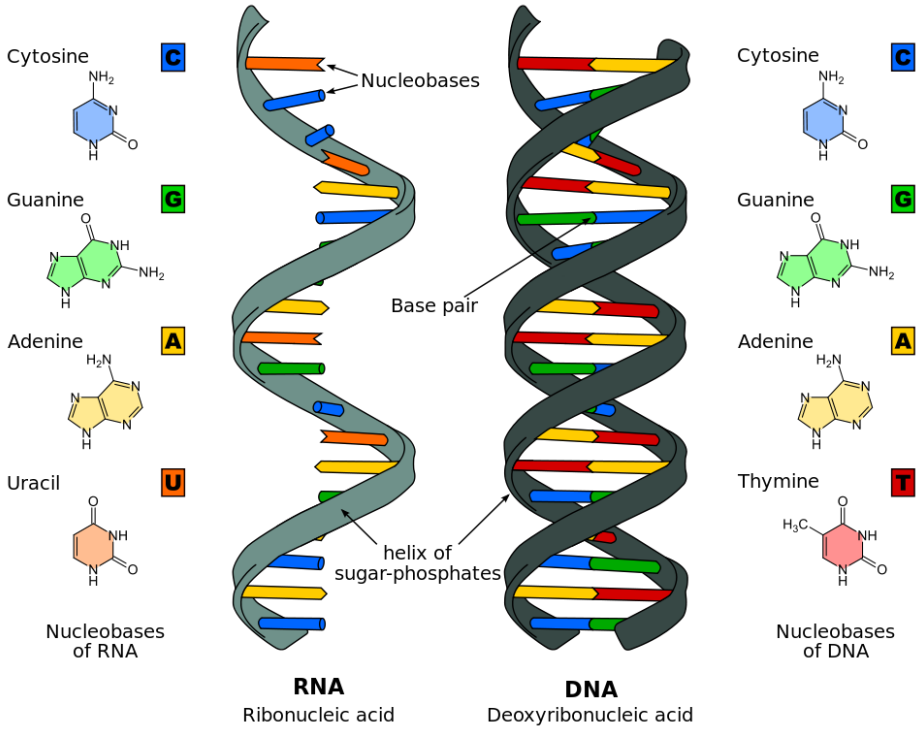


Fig. A.1 Model of the differences between DNA and RNA molecules

processes is protein synthesis, a universal function where RNA molecules direct the assembly of proteins on ribosomes. This process uses messenger RNA (mRNA) to convey genetic information that directs the synthesis of proteins, transfer RNA (tRNA) molecules to deliver amino acids to the ribosome, where ribosomal RNA (rRNA) then links amino acids together to form proteins. The chemical structure of RNA is very similar to that of DNA, but differs in three main ways: Unlike double-stranded DNA, RNA is a single-stranded molecule in many of its biological roles and has a much shorter chain of nucleotides. However, RNA can, by complementary base pairing, form intra-strand double helixes, as in tRNA. While DNA contains deoxyribose, RNA contains ribose, which is more prone to hydrolysis. The complementary base to adenine in DNA is thymine, whereas in RNA, it is uracil.

Nucleotide

A nucleotide is an organic molecule. It is the monomer comprising DNA or RNA biopolymer molecules. Each nucleotide consists of a nitrogenous heterocyclic base (or nucleobase), which can be either a double-ringed purine or a single-ringed pyrimidine; a five-carbon pentose sugar (deoxyribose in DNA or ribose in RNA); and a phosphate group.

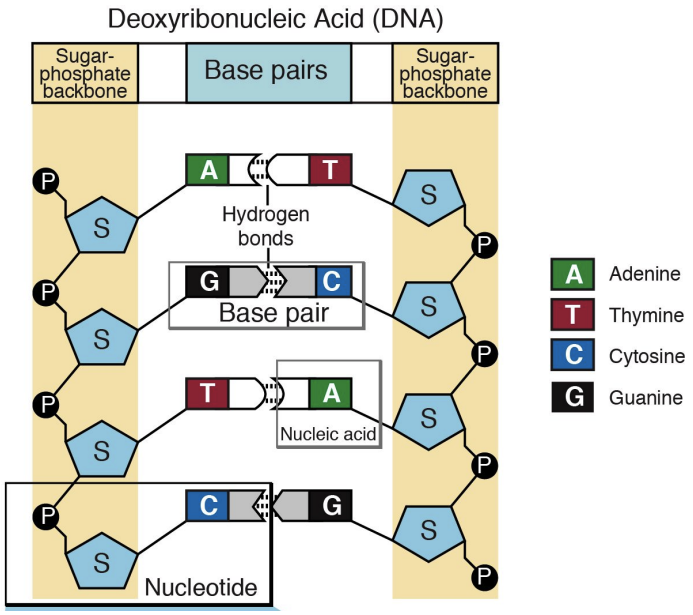


Fig. A.2 Focus on the Nucleotide and Base pairs

Gene

A gene is a sequence in the DNA or RNA which codes for a molecule that has a function. During gene expression, the DNA is first copied into RNA. The RNA can be directly functional or be the intermediate template for a protein that performs a function. Genes make up different DNA sequences called genotypes. Genotypes along with environmental and developmental factors determine what the phenotypes will be. Most biological traits are under the influence of polygenes (many different genes) as well as gene–environment interactions. Some genetic traits are instantly visible, such as eye color or number of limbs, and some are not, such as blood type, risk for specific diseases, or the thousands of basic biochemical processes that constitute life. Genes can acquire mutations in their sequence, leading to different variants, known as alleles (brown and blue eye colours), in the population. These alleles encode slightly different versions of a protein, which cause different phenotypical traits. Usage of the term "having a gene" (e.g., "good genes," "hair colour gene") typically refers to containing a different allele of the same, shared gene. Genes evolve due to natural selection or survival of the fittest of the alleles. The concept of a gene continues to be refined as new phenomena are discovered. For example, regulatory regions of a gene can be far removed from its coding regions, and coding regions can be split into several exons. Some viruses store their genome in RNA instead of DNA and some gene products are functional non-coding RNAs. Therefore, a broad, modern working definition of a gene is any discrete

locus of heritable, genomic sequence which affect an organism's traits by being expressed as a functional product or by regulation of gene expression.

Genome

Fig. A.3 A visual overview of all the chromosomes

A genome sequence is the complete list of the nucleotides (A, C, G, and T for DNA genomes) that make up all the chromosomes of an individual or a species. Within a species, the vast majority of nucleotides are identical between individuals, but sequencing multiple individuals is necessary to understand the genetic diversity. In terms of modern molecular biology and genetics, a genome is the genetic material of an organism. It consists of DNA (or RNA in RNA viruses). The genome includes both the genes (the coding regions) and the noncoding DNA, as well as the genetic material of the mitochondria and chloroplasts.

mRNA

Messenger RNA (mRNA) is a large family of RNA molecules that convey genetic information from DNA to the ribosome, where they specify the amino acid sequence of the protein products of gene expression. Following transcription of primary transcript mRNA (known as pre-mRNA) by RNA polymerase, processed, mature mRNA is translated into a polymer of amino acids: a protein. As in DNA, mRNA genetic information is in the sequence of nucleotides, which are arranged into codons consisting of three base pairs each. Each codon

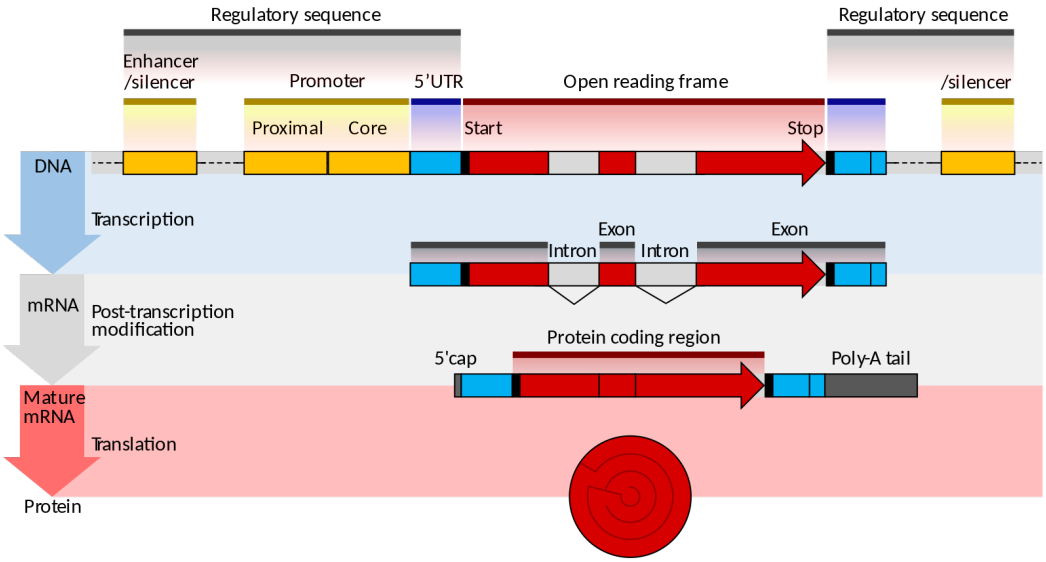
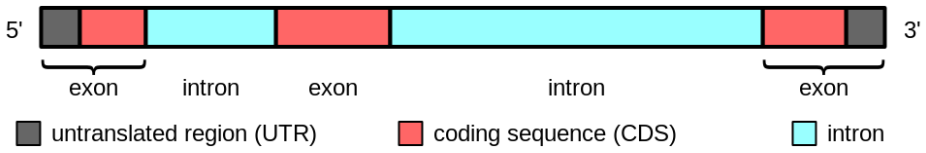


Fig. A.4 The gene expression process

encodes for a specific amino acid, except the stop codons, which terminate protein synthesis. This process of translation of codons into amino acids requires two other types of RNA: Transfer RNA (tRNA), that mediates recognition of the codon and provides the corresponding amino acid, and ribosomal RNA (rRNA), that is the central component of the ribosome protein production.

Exon/Intron



An exon is any part of a gene that will encode a part of the final mature RNA produced by that gene after introns have been removed by RNA splicing. The term exon refers to both the DNA sequence within a gene and to the corresponding sequence in RNA transcripts. In RNA splicing, introns are removed and exons are covalently joined to one another as part of generating the mature messenger RNA. Just as the entire set of genes for a species constitutes the genome, the entire set of exons constitutes the exome. An intron is any nucleotide sequence within a gene that is removed by RNA splicing during maturation of the final RNA product. The term intron refers to both the DNA sequence within a gene and the corresponding sequence in RNA transcripts. Introns are found in the genes of most

organisms and many viruses, and can be located in a wide range of genes, including those that generate proteins, ribosomal RNA (rRNA), and transfer RNA (tRNA). When proteins are generated from intron-containing genes, RNA splicing takes place as part of the RNA processing pathway that follows transcription and precedes translation.

Transcriptome

The transcriptome is the set of all RNA molecules in one cell or a population of cells. It is sometimes used to refer to all RNAs, or just mRNA, depending on the particular experiment. It differs from the exome in that it includes only those RNA molecules found in a specified cell population, and usually includes the amount or concentration of each RNA molecule in addition to the molecular identities.

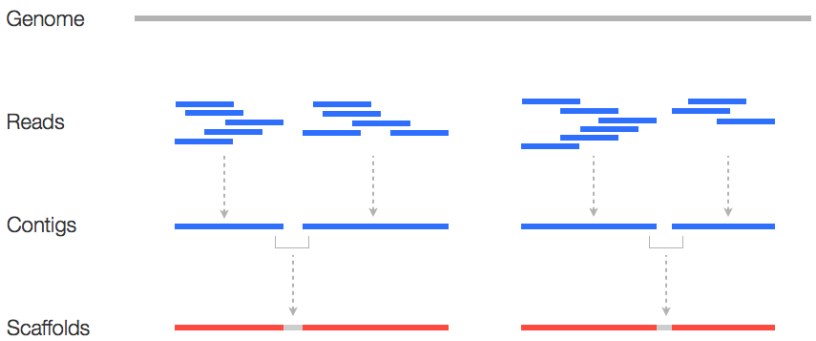
Reverse Transcriptase

A reverse transcriptase (RT) is an enzyme used to generate complementary DNA from an RNA template, a process termed reverse transcription. It is mainly associated with retroviruses. However, non-retroviruses also use reverse transcriptase. RT inhibitors are widely used as antiretroviral drugs. Reverse transcriptase activities are also associated with the replication of chromosome ends (telomerase) and some mobile genetic elements (retrotransposons).

cDNA

In genetics, complementary DNA (cDNA) is DNA synthesised from a single stranded RNA (e.g., messenger RNA (mRNA) or microRNA) template in a reaction catalysed by the enzyme reverse transcriptase. cDNA is often used to clone eukaryotic genes in prokaryotes. When scientists want to express a specific protein in a cell that does not normally express that protein, they will transfer the cDNA that codes for the protein to the recipient cell. cDNA is also produced naturally by retroviruses (e.g. HIV-1) and then integrated into the host's genome, where it creates a provirus. The term cDNA is also used, in a bioinformatics context, to refer to an mRNA transcript's sequence, expressed as DNA bases (GCAT) rather than RNA bases (GCAU).

Contig



A contig (from contiguous) is a set of overlapping DNA segments that together represent a consensus region of DNA. In bottom-up sequencing projects, a contig refers to overlapping sequence data (reads); in top-down sequencing projects, contig refers to the overlapping clones that form a physical map of the genome that is used to guide sequencing and assembly. Contigs can thus refer both to overlapping DNA sequence and to overlapping physical segments (fragments) contained in clones depending on the context.

Cancer

Cancer is a large family of diseases involving abnormal cell growth with the potential to invade or spread to other parts of the body. They are in contrast to benign tumours, which do not spread to other parts of the body. Possible signs and symptoms include a lump, abnormal bleeding, prolonged cough, unexplained weight loss, and a change in bowel movements. While these symptoms may indicate cancer, they may have other causes. They form a subset of neoplasms. A neoplasm or tumour is a group of cells that have undergone unregulated growth and will often form a mass or lump, but may be distributed diffusely. All tumour cells show the six hallmarks of cancer. These characteristics are required to produce a malignant tumour. They include: Cell growth and division absent the proper signals Continuous growth and division even given contrary signals Avoidance of programmed cell death Limitless number of cell divisions Promoting blood vessel construction Invasion of tissue and formation of metastases Over 100 types of cancers affect humans. In 2015, about 90.5 million people had cancer. About 14.1 million new cases occur a year. It caused about 8.8 million deaths (15.7% of deaths). The most common types of cancer in males are lung cancer, prostate cancer, colorectal cancer and stomach cancer. In females, the most common types are breast cancer, colorectal cancer, lung cancer and cervical cancer. If skin cancer other than

melanoma were included in total new cancers each year, it would account for around 40% of cases. In children, acute lymphoblastic leukaemia and brain tumours are most common except in Africa where non-Hodgkin lymphoma occurs more often. In 2012, about 165,000 children under 15 years of age were diagnosed with cancer. The risk of cancer increases significantly with age and many cancers occur more commonly in developed countries. Rates are increasing as more people live to an old age and as lifestyle changes occur in the developing world. The financial costs of cancer were estimated at $1.16 trillion USD per year as of 2010.

Single Nucleotide Polymorphism

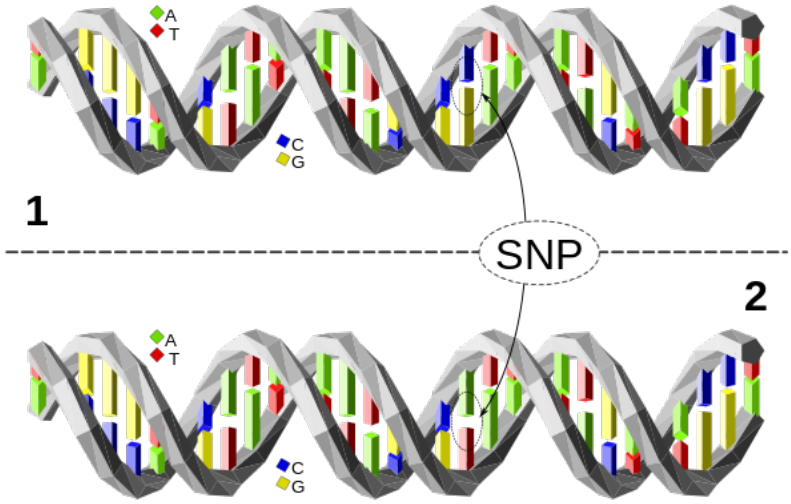


Fig. A.5 An example of C-A polymorphism

A single-nucleotide polymorphism (SNP), is a variation in a single nucleotide that occurs at a specific position in the genome, where each variation is present to some appreciable degree within a population. For example, at a specific base position in the human genome, the G nucleotide may appear in most individuals, but in a minority, the position is occupied by an T. This means that there is an SNP at this specific position, and the two possible nucleotide variations are said to be alleles for this position. SNPs underlie differences in our susceptibility to disease e.g. sickle-cell anemia, β-thalassemia and cystic fibrosis result from SNPs.The severity of illness and the way our body responds to treatments are also manifestations of genetic variations. For example, a single-base mutation in the Apolipoprotein E gene is associated with a higher risk for Alzheimer's disease.

Alternative Splicing

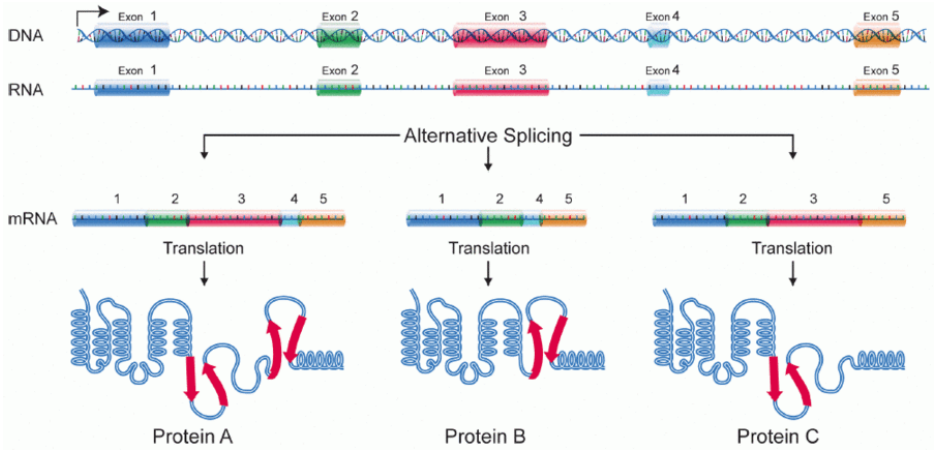


Fig. A.6 How alternative splicing works

Alternative splicing is a regulated process during gene expression that results in a single gene coding for multiple proteins. In this process, particular exons of a gene may be included within or excluded from the final, processed mRNA produced from that gene. Consequently, the proteins translated from alternatively spliced mRNAs will contain differences in their amino acid sequence and, often, in their biological functions. Notably, alternative splicing allows the human genome to direct the synthesis of many more proteins than would be expected from its 20,000 protein-coding genes. Alternative splicing occurs as a normal phenomenon in eukaryotes, where it greatly increases the biodiversity of proteins that can be encoded by the genome. There are numerous modes of alternative splicing observed, of which the most common is exon skipping. In this mode, a particular exon may be included in mRNAs under some conditions or in particular tissues, and omitted from the mRNA in others.

APPENDIX B

ENRICHMENT RESULTS

In this appendix, the remaining results from the manual verification of the extracted modules are included. In the chapter devoted to the *Enrichment Analysis* only 6 modules are presented for each cancer type. The rest of them also have interesting insights which are left here for the reader.

Breast Cancer

Breast Cancer related

Table B.1 Module #100 enrichment, against `GO Biological`

Term	P-value
epithelial tube branching involved in lung [...] (GO:0060441)	1.62655776435e-05
regulation of Rac protein signal transduction (GO:0035020)	0.000469437940862
aerobic respiration (GO:0009060)	0.00137041296548
positive regulation of lipid catabolic process (GO:0050996)	0.00113078452929
acylglycerol catabolic process (GO:0046464)	0.00124782046916

Related article: "Rac signaling in breast cancer: a tale of GEFs and GAPs" [60].

Table B.2 Module #125 enrichment, against `GO Biological`

Term	P-value
stem cell division (GO:0017145)	3.67068918761e-05
intracellular estrogen receptor signaling pathway (GO:0030520)	0.000913492369971
somatic stem cell division (GO:0048103)	0.000718757572453
positive regulation of acute inflammatory response (GO:0002675)	0.00191580216511
regulation of T cell differentiation (GO:0045580)	0.00183755032274

Related article: "Minireview: inflammation: an instigator of more aggressive estrogen receptor (ER) positive breast cancers" [61].

Cancer related

Table B.3 Module #6 enrichment, against `GO Cellular`

Term	P-value
proteasome accessory complex (GO:0022624)	9.75370366188e-06
proteasome regulatory particle (GO:0005838)	0.000397852306904
proteasome complex (GO:0000502)	0.000525323124367
cytoplasmic vesicle part (GO:0044433)	0.00207755248182
cytoplasmic vesicle membrane (GO:0030659)	0.00255261632643

Related article: "Proteasome inhibitors in cancer therapy" [62].

Table B.4 Module #21 enrichment, against `GO Biological`

Term	P-value
small molecule catabolic process (GO:0044282)	4.57962365495e-05
nitric oxide biosynthetic process (GO:0006809)	0.00054680280059
negative regulation of interleukin-8 production (GO:0032717)	0.00054680280059
arginine metabolic process (GO:0006525)	0.00062991870488
arginine catabolic process (GO:0006527)	0.000169869167294

Related articles: "Regulation of interleukin-8 expression by nitric oxide in human pancreatic adenocarcinoma." [63] and "Targeting amino acid metabolism in cancer growth and anti-tumor immune response" [64].

Table B.5 Module #28 enrichment, against `GO Molecular`

Term	P-value
ATP binding (GO:0005524)	5.555057613e-05
transforming growth factor beta-activated [...] (GO:0005024)	0.000813291421624
transmembrane receptor protein serine/threonine [...] (GO:0004675)	0.000813291421624
transforming growth factor beta binding (GO:0050431)	0.000718757572453
transforming growth factor beta receptor binding (GO:0005160)	0.00137041296548

Related article: "Transforming growth factor-beta receptors in human cancer cell lines: analysis of transcript, protein and proliferation." [65].

Table B.6 Module #42 enrichment, against `GO Molecular`

Term	P-value
ubiquitin thiolesterase activity (GO:0004221)	9.90436680677e-05
cysteine-type peptidase activity (GO:0008234)	0.00089369059915
collagen binding (GO:0005518)	0.00050118705903
thiolester hydrolase activity (GO:0016790)	0.000575802113062
ubiquitinyl hydrolase activity (GO:0036459)	0.00230383353688

Related article: "Targeting the ubiquitin–proteasome system for cancer therapy" [66].

Table B.7 Module #44 enrichment, against `GO Biological`

Term	P-value
positive regulation of mitochondrial outer [...] (GO:1901030)	8.33007195229e-05
regulation of mitochondrial outer membrane [...] (GO:1901028)	0.00014657676169
positive regulation of mitochondrion organization (GO:0010822)	0.000550191771545
regulation of protein insertion into [...] (GO:1900739)	0.00206576853123
positive regulation of protein insertion into [...] (GO:1900740)	0.00206576853123

Related article: "The mitochondrial permeability transition pore and cancer: molecular mechanisms involved in cell death" [67].

Table B.8 Module #100 enrichment, against `GO Biological`

Term	P-value
epithelial tube branching involved in lung [...] (GO:0060441)	1.62655776435e-05
regulation of Rac protein signal transduction (GO:0035020)	0.000469437940862
aerobic respiration (GO:0009060)	0.00137041296548
positive regulation of lipid catabolic process (GO:0050996)	0.00113078452929
acylglycerol catabolic process (GO:0046464)	0.00124782046916

Related article: "Rac signaling in breast cancer: a tale of GEFs and GAPs" [60].

Table B.9 Module #198 enrichment, against `GO Molecular`

Term	P-value
Rac GTPase binding (GO:0048365)	5.69169031669e-05
inositol trisphosphate kinase activity (GO:0051766)	0.000169869167294
exonuclease activity (GO:0004527)	0.0008746486678
Rho GTPase binding (GO:0017048)	0.000629283942131
ATPase binding (GO:0051117)	0.00518076445826

Related article: "Rac and Rho GTPases in cancer cell motility control" [68].

Table B.10 Module #231 enrichment, against `GO Cellular`

Term	P-value
contractile fiber (GO:0043292)	9.90436680677e-05
focal adhesion (GO:0005925)	0.000235574451193
cell-substrate adherens junction (GO:0005924)	0.000257966040442
cell-substrate junction (GO:0030055)	0.000273802975808
adherens junction (GO:0005912)	0.000497256653527

Related article: "Collective cell migration in morphogenesis, regeneration and cancer" [69].

Table B.11 Module #302 enrichment, against `GO Biological`

Term	P-value
embryonic cranial skeleton morphogenesis (GO:0048701)	5.13154851693e-05
pathway-restricted SMAD protein phosphorylation (GO:0060389)	0.000469437940862
response to cholesterol (GO:0070723)	0.00113078452929
response to sterol (GO:0036314)	0.00149853462764
regulation of lipid transport (GO:0032368)	0.00117355122158

Related article: "Molecular pathways: sterols and receptor signaling in cancer" [70].

Lung Cancer

Lung Cancer related

Table B.12 Module #32 enrichment, against `GO Biological`

Term	P-value
cGMP metabolic process (GO:0046068)	2.86757414113e-05
cAMP catabolic process (GO:0006198)	0.000718757572453
cyclic nucleotide catabolic process (GO:0009214)	0.000913492369971
cyclic nucleotide metabolic process (GO:0009187)	0.000411783643815
cAMP metabolic process (GO:0046058)	0.00345743152997

Related article: "Elevated nephrogenous cyclic AMP with normal serum parathyroid hormone levels in patients with lung cancer" [71].

Table B.13 Module #36 enrichment, against GO Biological

Term	P-value
placenta blood vessel development (GO:0060674)	5.69169031669e-05
synaptonemal complex assembly (GO:0007130)	0.00101933263419
negative regulation of alcohol biosynthetic process (GO:1902931)	0.00062991870488
synaptonemal complex organization (GO:0070193)	0.00124782046916
negative regulation of steroid biosynthetic process (GO:0010894)	0.00124782046916

Related article: "Synaptonemal complex protein 3 is associated with lymphangiogenesis in non-small cell lung cancer patients with lymph node metastasis" [72].

Table B.14 Module #37 enrichment, against GO Biological

Term	P-value
positive regulation of sodium ion transport (GO:0010765)	6.29042976962e-05
positive regulation of sodium ion [...] (GO:2000651)	0.000332074180163
positive regulation of sodium ion [...] (GO:1902307)	0.00062991870488
regulation of sodium ion transport (GO:0002028)	0.00045507583593
nucleus localization (GO:0051647)	0.00124782046916

Related article: "The α1 subunit of the sodium pump could represent a novel target to combat non-small cell lung cancers" [73].

Table B.15 Module #114 enrichment, against GO Biological

Term	P-value
dolichol-linked oligosaccharide biosynthetic process (GO:0006488)	8.33007195229e-05
negative regulation of intrinsic apoptotic [...] (GO:1902166)	0.000397852306904
negative regulation of intrinsic apoptotic [...] (GO:1902254)	0.00062991870488
regulation of intrinsic apoptotic signaling [...] (GO:1902165)	0.00054680280059
granulocyte differentiation (GO:0030851)	0.00062991870488

Related article: "p53: a frequent target for genetic abnormalities in lung cancer" [74].

Table B.16 Module #144 enrichment, against GO Biological

Term	P-value
regulation of cell adhesion mediated by integrin (GO:0033628)	9.90436680677e-05
positive regulation of cell adhesion mediated [...] (GO:0033630)	0.000718757572453
regulation of catenin import into nucleus (GO:0035412)	0.00149853462764
regulation of cell adhesion (GO:0030155)	0.00148127650608
negative regulation of peptidyl-tyrosine [...] (GO:0050732)	0.00307789877528

Related article: "The E-cadherin cell–cell adhesion complex and lung cancer invasion, metastasis, and prognosis" [75].

Table B.17 Module #243 enrichment, against `GO Molecular`

Term	P-value
beta-tubulin binding (GO:0048487)	9.09499482584e-05
SMAD binding (GO:0046332)	0.0125610214202
RNA polymerase II transcription cofactor activity (GO:0001104)	0.016298707591
tubulin binding (GO:0015631)	0.022350641825
cyclase regulator activity (GO:0010851)	0.0222807196156

Related article: "Epigenetic regulation of Smad2 and Smad3 by profilin-2 promotes lung cancer growth and metastasis" [76].

Table B.18 Module #323 enrichment, against `GO Biological`

Term	P-value
chromosome organization involved in meiosis (GO:0070192)	9.90436680677e-05
synaptonemal complex assembly (GO:0007130)	0.00101933263419
synaptonemal complex organization (GO:0070193)	0.00124782046916
activation of phospholipase C activity (GO:0007202)	0.000433082063756
positive regulation of phospholipase C activity (GO:0010863)	0.000629283942131

Related article: "Synaptonemal complex protein 3 is associated with lymphangiogenesis in non-small cell lung cancer patients with lymph node metastasis" [72].

Cancer related

Table B.19 Module #56 enrichment, against `GO Cellular`

Term	P-value
ubiquitin ligase complex (GO:0000151)	9.21549527451e-05
cullin-RING ubiquitin ligase complex (GO:0031461)	0.000132203089205
Cul3-RING ubiquitin ligase complex (GO:0031463)	0.0016321581628
microbody membrane (GO:0031903)	0.00494766354779
peroxisomal membrane (GO:0005778)	0.00494766354779

Related article: "Peroxisome proliferator activated receptor-γ and the ubiquitin-proteasome system in colorectal cancer" [77].

Table B.20 Module #104 enrichment, against GO Biological

Term	P-value
cellular response to fibroblast growth factor stimulus (GO:0044344)	9.21549527451e-05
response to fibroblast growth factor (GO:0071774)	0.000102031457675
regulation of fibroblast migration (GO:0010762)	0.00101933263419
fibroblast growth factor receptor signaling pathway (GO:0008543)	0.000731357860307
O-glycan processing (GO:0016266)	0.000371234763507

Related article: "Fibroblast growth factor signalling: from development to cancer" [78].

Table B.21 Module #151 enrichment, against GO Biological

Term	P-value
regulation of ryanodine-sensitive [...] (GO:0060314)	3.2528722798e-05
negative regulation of ryanodine-sensitive [...] (GO:0060315)	0.000272131942618
regulation of calcium ion transmembrane transport (GO:1903169)	0.000234913166441
regulation of calcium ion transmembrane (GO:1901019)	0.000234913166441
negative regulation of cation channel activity (GO:2001258)	0.000913492369971

Related article: "Evidence of functional ryanodine receptor involved in apoptosis of prostate cancer (LNCaP) cells" [79].

Table B.22 Module #386 enrichment, against GO Biological

Term	P-value
regulation of nucleocytoplasmic transport (GO:0046822)	8.08634213906e-05
regulation of intracellular protein transport (GO:0033157)	0.000213389819348
cranial suture morphogenesis (GO:0060363)	0.000272131942618
negative regulation of catenin import into nucleus (GO:0035414)	0.000218054077011
regulation of hexokinase activity (GO:1903299)	0.000218054077011

Related article: "Wnt/β-catenin signaling plays an ever-expanding role in stem cell self-renewal, tumorigenesis and cancer chemoresistance" [80].

APPENDIX C

TOOLS PROODUCED

In this appendix the tools produced as support to the method are introduced.

The **first** tool performs a *lookup* over all of the modules for a specific gene. The query consists in a `JSON` file containing the name of the gene to be searched and the cancer type over which the desired autoencoder was trained. Here is an example of the input:

```
{
    "Gene": "KRAS",
    "Cancer": "BREAST"
}
```

The output of the previous query returns the number identifying the module that contains the desired gene and its position in it, with a snapshot of the other leading genes:

```
The 384-th module contains the gene.
The gene KRAS was found in the module's first 7 most relevant genes.
The module, rounded to the nearest multiple of 10, is the following:
['KIAA0664P3' 'USF2' 'F5' 'DICER1' 'CDNF' 'IQCA1' 'KRAS' ...
... 'C18orf32' 'CRYZL1' 'HIST1H2BI']
```

The **second** tool returns, for each module, a report containing all the known tumor suppressors, proto-oncogenes and oncogenes in the module; a list of the genes in the module with their differential expression levels; and the results of the enrichment analysis performed on it. The following page include a sample module's report (Fig C.1).

Report for the extracted Module 6

Lists for known Genes associated to Cancer *

	1
Oncogenes	TRIP11

* It only includes genes found in the module that were found in known lists for Tumour Suppressors (TS), Proto-Oncogenes (PO) and Oncogenes (OG).

Positive Differential Gene Expression **

	log (delta_median)	delta_mean	cancer_mean	healthy_mean	cancer_median	healthy_median
CAPN9	0.518310	1.701730	5.683437	3.339799	6.10670	3.6367
GK3P	0.171859	1.202445	4.352217	3.619473	4.39165	3.6982
C1orf112	0.147251	1.169303	8.118501	6.943028	8.08780	6.9804
C19orf24	0.105037	1.109261	9.088028	8.192865	9.01775	8.1186
PSMC4	0.079359	1.089878	10.839941	9.946011	10.77180	9.9500
CCL4L2	0.075011	1.047990	5.952785	5.680193	6.06640	5.6280
HNRNPA3P1	0.070426	1.077116	6.625893	6.151515	6.58500	6.1372
TBC1D2	0.060658	1.058849	8.850133	8.358255	8.92020	8.3952
HS6ST1	0.050375	1.061858	10.628033	10.008904	10.63570	10.1132
FIZ1	0.046077	1.049203	8.656202	8.250265	8.61285	8.2250
PSMC5	0.032699	1.037332	11.392847	10.982834	11.31335	10.9494
MIPEP	0.032324	1.032937	9.032271	8.744261	9.03415	8.7468
PSMD1	0.029434	1.029305	11.274311	10.953327	11.25835	10.9318
ANKRD50	0.024867	1.028934	10.463742	10.169498	10.45190	10.1952
ARPC2	0.021922	1.018594	12.075147	11.854722	12.07820	11.8163
NECAP1	0.013279	1.017128	9.656776	9.494161	9.63875	9.5116
ELP2P	0.013232	1.033212	3.369181	3.260882	3.32835	3.2846
IRF2BP2	0.009113	1.010180	12.458605	12.333056	12.46185	12.3488
PRPF40B	0.005270	1.018732	8.441611	8.286390	8.51445	8.4697
SEPN1	0.004491	1.002423	11.765410	11.736976	11.77235	11.7196
WDR92	0.003409	1.004394	8.022053	7.986958	8.02130	7.9940
CHRNB3	0.000000	1.471308	0.074089	0.050356	0.00000	0.0000
RNF113B	0.000000	1.565917	0.242291	0.154727	0.00000	0.0000
LRTM1	0.000000	1.836561	0.131942	0.071842	0.00000	0.0000
KRTAP12-1	0.000000	1.102061	0.017032	0.015455	0.00000	0.0000
GOLGA8E	0.000000	0.701630	0.037521	0.053477	0.00000	0.0000

Negative Differential Gene Expression **

	log (delta_median)	delta_mean	cancer_mean	healthy_mean	cancer_median	healthy_median
RBM12	-0.001907	1.001045	10.974426	10.962965	11.00330	11.0243
ANXA2	-0.008513	0.985521	13.644667	13.845129	13.66785	13.7847
QARS	-0.011721	0.984666	11.756424	11.939509	11.76430	11.9030
CAB39	-0.014259	0.981006	10.929002	11.140603	10.95980	11.1172
CRY1	-0.014428	0.981468	8.884937	9.052701	8.89020	9.0194
WNT5A	-0.017885	0.973340	8.423780	8.654511	8.53655	8.6906
ATP6V0A1	-0.021638	0.976342	10.344883	10.595555	10.36420	10.5909
ELF1	-0.022082	0.977316	10.627054	10.873719	10.70410	10.9431
MRPL46	-0.023485	0.978989	8.447082	8.628372	8.38710	8.5864
(OG) TRIP11	-0.024459	0.977589	10.006738	10.236139	10.09280	10.3427
ZNF512	-0.033634	0.960255	9.629408	10.027972	9.72475	10.0574
PERP	-0.039039	0.976835	12.113093	12.400347	12.19885	12.6845
FAM21C	-0.039757	0.961206	10.363512	10.781781	10.39385	10.8154
PPP3CA	-0.052234	0.946272	10.243354	10.824958	10.24285	10.7921
HEG1	-0.074758	0.918443	10.541059	11.477101	10.66820	11.4963
SLC12A4	-0.097033	0.900097	9.423333	10.469236	9.49390	10.4613
KSR1	-0.115446	0.893816	8.051356	9.007850	8.04625	9.0309
USP51	-0.129171	0.860474	5.194900	6.037256	5.30590	6.0375
TTC23	-0.133677	0.873317	7.886451	9.030461	7.91900	9.0516
MEST	-0.146222	0.832906	9.514442	11.423185	9.68730	11.2126
LPHN2	-0.171780	0.845638	9.039430	10.689474	9.03600	10.7295
ARMC4	-0.259497	0.905685	2.568210	2.835656	2.15085	2.7881
C10orf140	-0.356972	0.769765	3.945225	5.125233	3.63885	5.1999

** The list is sorted by descreasing order of the differential expression calculated from the median.

Enrichment Results ***

	Overlap	P-value	Adjusted P-value	Old P-value	Old Adjusted P-value
Term					
proteasome accessory complex (GO:0022624)	3/17	0.000010	0.001102	0.000012	0.001310
proteasome regulatory particle (GO:0005838)	2/12	0.000398	0.019787	0.000442	0.016662
proteasome complex (GO:0000502)	3/63	0.000525	0.019787	0.000434	0.016662
cytoplasmic vesicle part (GO:0044433)	5/363	0.002078	0.057689	0.001286	0.036328
cytoplasmic vesicle membrane (GO:0030659)	4/229	0.002553	0.057689	0.001775	0.040119

*** Results obtained by the Enrichr tool.

Fig. C.1 Example of a report produced by the developed support tool