Cancer Classification using Gene Expression Data with Deep Learning

Güray Gölcük

Supervisor: Prof. Stefano Ceri

Advisor: Dr. Arif Çanakoğlu

Department of Electronics, Informatics and Bioengineering
Polytechnic University of Milan

This dissertation is submitted for the degree of
Master of Science

December 2017

Abstract

Technological advances in DNA sequencing technologies allow sequencing the human genome for a low cost and with in a reasonable time span. This advance conduces to a huge increase in available genomic data, enabling the establishment of large-scale sequencing data projects. Producing genomic datasets, which describe genomic information in particular, we concentrate on our attention on gene expression datasets which describe healthy and tumoral cells for various cancer types. The purpose of this thesis is to apply deep learning to classification of tumors based on gene expression.

Two different Deep Learning approaches for analyzing the genomic data. First one is to create a feed-forward network (FFN) with supervised learning, second one is using ladder network with semi-supervised learning [42] . Main purpose of both approaches is to perform binary classification, cancerous or healthy as the outcome, over the Cancer Genome Atlas (TCGA) database.Two cancer types selected from TCGA. Breast cancer is selected because it has the highest available amount of sample in all cancer types in TCGA. The reason for kidney cancer to be selected is because, it has the one of the highest mortality rate among rest. Moreover, three feature extraction method, PCA, ANOVA and random forests, employed to preprocess the selected datasets. Experiments show that, FFN reaches the acceptable accuracy rate but fails to reach a stabilization. On the other hand, ladder network, outperforms the FFN in both accuracy and stabilization meaning. Effects of feature extraction method shows us that, main goal as accuracy is reached by PCA & ladder network combination, main goal as performance is reached by ANOVA & ladder network combination.

Abstract

Le innovazioni riguardanti le tecniche di sequenziamento del DNA stanno permettendo analisi del genoma umano sempre più veloci ed economiche. Ciò ha provocato un aumento considerevole dei volumi di dati disponibili riguardanti il genoma umano, permettendo l'instaurazione di progetti di sequenziamento su larga scala. In questo contesto di database genomici, ci concentreremo sui dati di espressione genica, i quali descrivono cellule tumorali e sane provenienti da diversi tipi di tessuto. Lo scopo di questa tesi è di applicare il deep-learning alla classificazione dei tumori in base all'espressione genica.

Ci siamo concentrati su due approcci fondamentali per l'analisi dei dati genomici. Il primo consiste nella creazione di una rete Feed-Forward (FFN) per l'apprendimento supervisionato, il secondo utilizza una Ladder Network tramite apprendimento semi-supervisionato. L'obiettivo principale di entrambi gli approcci è di eseguire classificazione binaria, avente tumore o sano come risultato. Come dataset abbiamo utilizzato The Cancer genome Atlas (TCGA) e in particolare ci siamo concentrati su due tipi specifici di tumore: tumore della mammella e ai reni. Questa scelta è principalmente dettata dalla più alta disponibilità di samples per questi due tipi di tumore. Abbiamo utilizzato tre differenti tipi di feature extraction, PCA, ANOVA and random forest.

Gli esperimenti hanno mostrato che FFN raggiunge un accuratezza accettabile ma fallisce in quanto a stabilità dei risultati. D'altra parte la Ladder Network sorpassa FFN sia in accuratezza che in stabilità. Gli effetti della feature extraction mostrano che la migliore combinazione per quanto riguarda l'accuratezza è PCA con Ladder Network, mentre per quanto riguarda le performance la combinazione ANOVA e Ladder network prevale.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 DNA Sequencing

The DNA sequencing is a process of detecting the exact sequence of the nucleotides (adenine, guanine, cytosine, and thymine) that creates the DNA. The advancements in the DNA sequencing technologies greatly effects the discoveries the in biological and medical science [24, 32]. Developments in DNA sequencing also helps biotechnology and forensic studies significantly [18].

Sequencing a whole DNA has been very complex and an expensive task. Breaking DNA into smaller sequences and reassembling it into a single line is still complex, even so, improvements into computer science and discoveries of new methods over past two decades, makes it cheaper and faster.

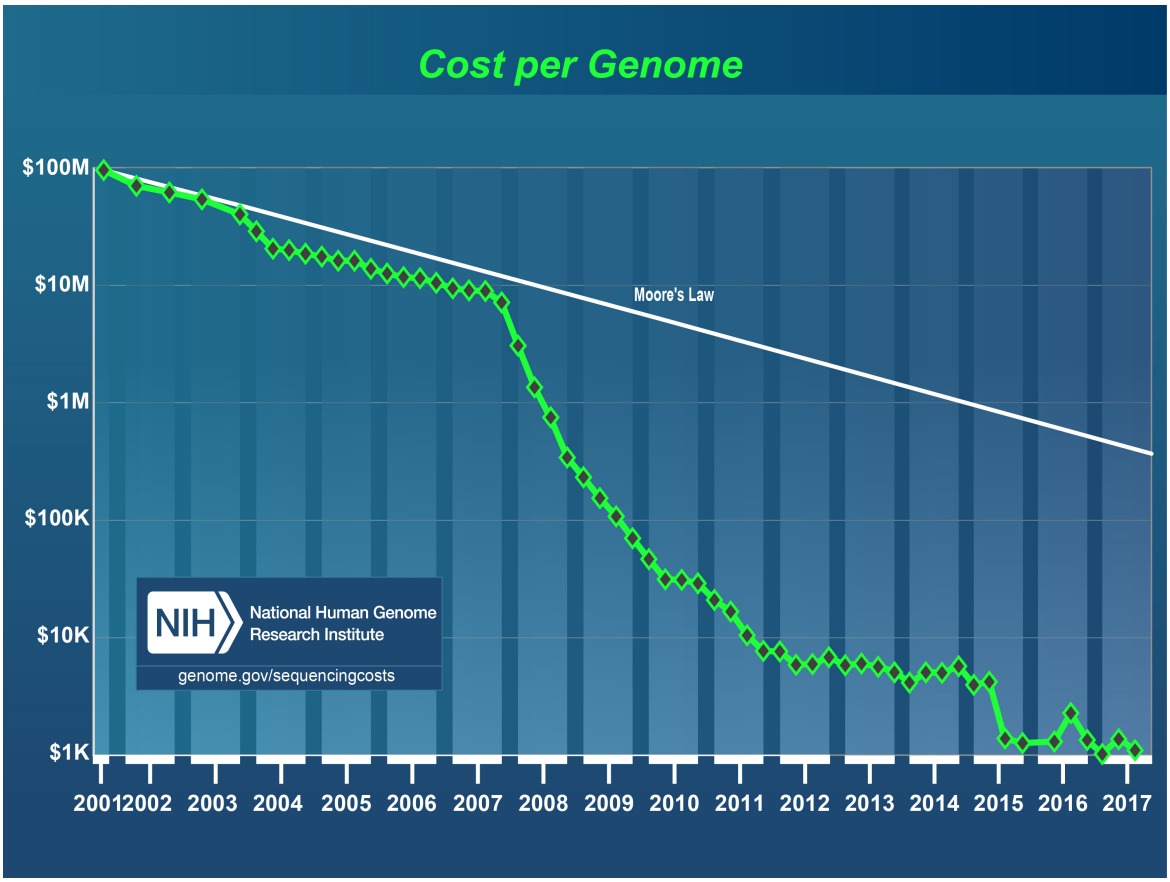


Fig. 1.1 The cost of genome sequencing over the last 16 years [27]

It is evident from the Figure 1.1 that in 2001, DNA sequencing per genome cost 100M dollar however after 2015, the cost has been significantly drop down to the order of 1K dollar. Improvements in DNA sequencing technology keep up with Moore's law. [1] till 2007. After the introducing of Next Generation Sequencing (NGS) technologies, sequencing cost fall sharply and reach the value of 1K dollar today [44].

[1] According to Moore's law the number of transistors in dense integrated circuits meaning the performance of the chips approximately doubles every two years. [41]

1.2 Analysis of Genomic Data

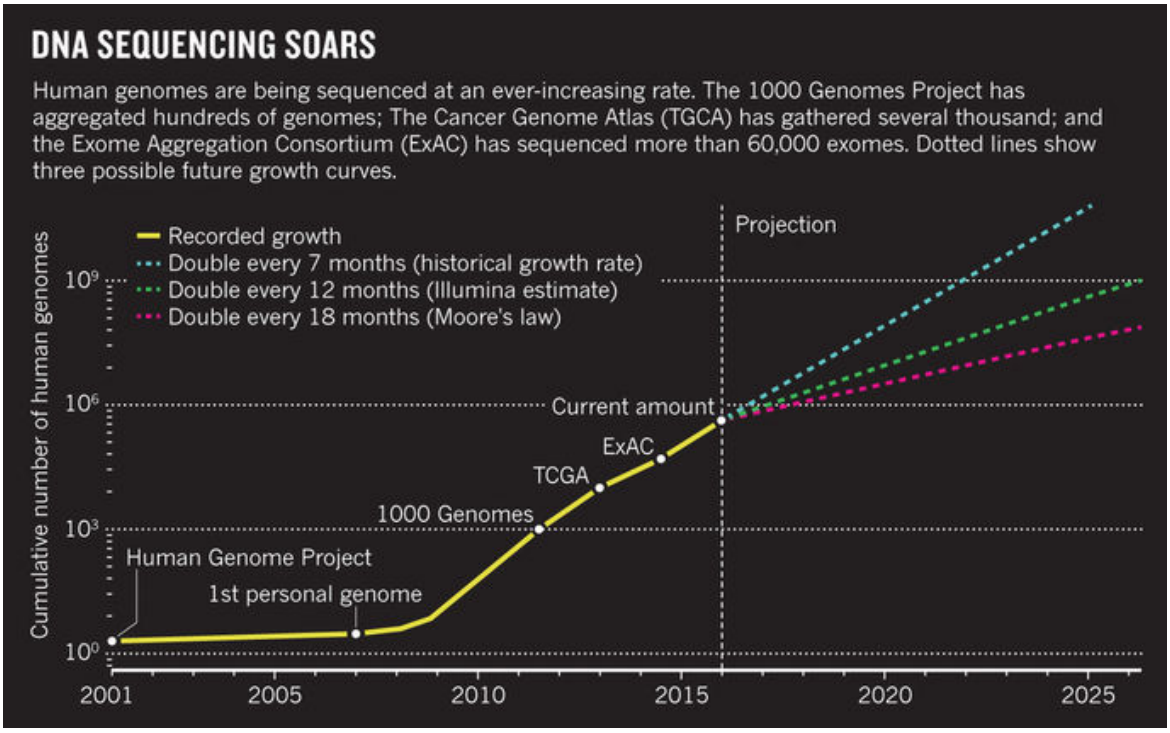


Fig. 1.2 The amount of sequenced genomes over the years [17]

The drop in the sequencing cost allows a huge increase in sequenced genome data as shown in the Figure 1.2. Increase in sequenced genomic data made it possible to establish huge genomic projects such as Human Genome Project [9], The Encyclopedia Of DNA Elements (ENCODE) [11] and the 1000 Genomes Project Consortium [10]. These projects ceaselessly gather and store sequencing data. Importance of big data analyzing techniques shines after this point. Big data analysis are essential to utilize collected data efficiently.
Sequence Data analysis can be broadly separated into three categories [37].

- **Primary Analysis (Base Calling):** Converts raw data into nucleotides with respect to the changes in electric current and light intensity.
- **Secondary Analysis (Alignment & Variant calling)** Maps the short sequence of nucleotides into reference sequence to determine variation from.
- **Tertiary Analysis: (Interpretation)** Analyses variant to estimate their origin, impact, and uniqueness.

Out of these three categories, the tertiary analysis is the most important one, since knowledge gathering from sequenced data handled in the tertiary analysis.

GenData 2020 project, which introduced in 2013, was concentrate on tertiary analysis of genomic sequence data. GenoMetric Query Language (GMQL) and Genomic Data Model (GDM) is the most important outcomes of this project. GMQL is query language that works with heterogeneous datasets such as sequenced genomic data while GDM is the general data model for mapping genomic features with its metadata. GMQL also have interface for Python an R which are most commonly used in data analysis. More information on GMQL can be found on Chapter 2.

1.3 Machine Learning with Genomic Data

Machine learning is a data analysis method that allows the automation of model building and learning with given input data. It is a subset of artificial intelligence based on the idea that machines should learn through experience. With machine learning, computers could use pattern recognition to find hidden insights without any explicit programming. In comparison to the traditional biological computational methods, machine learning has been put into practice with the research on the binary or multiple cancer classification with genomic data [49, 53], For instance, Stacked Denoising Autoencoder (SDAE) [13] , Fuzzy Support Vector Machine (FSVM) [33] and Deep Belief Networks (BDN) [3] have been proposed to do binary and multi-class classification to genomic data.

Deep learning is a branch of machine learning with more complicated algorithms which can model features with high-level abstraction from data. It has achieved the state-of-the-art performance in several fields such as image classification [29, 46, 50], semantic segmentation [31] and speech recognition [25]. Recently, deep learning methods have also achieved success in computational biology [47, 4]. This thesis is the continuation of similar work of Tuncel [52], classification of cancer types with machine learning algorithms, through the deep networks. The goal of this thesis is to analysis the effects of the feature selection methods on the final accuracy of the created Deep Neural Networks. The study can be divided into two part. First part is the creation of the feed forward network and effects of the feature selection algorithm on final accuracy and second part is implementation of the ladder network structure for binary classification, and effects of feature selection algorithms. The input data is the breast and cancer data from the Cancer Genome Atlas (TCGA) database.

This thesis is arranged as follows: In Chapter 2 background for GMQL method is presented. In Chapter 3 information on the input data is presented. Chapter 4 theoretical background for data analyzing is explained, it includes machine learning, neural networks

and used methods for feature extracting. Chapter 5 describes the created feed forward network and the results achieved. Chapter 6 introduces the ladder network, and explains implementation for binary classification. Chapter 7 compares the final result of the both network and effects of the feature extraction on them. Finally, conclusions and possible future work are discussed in Chapter 8.

Chapter 2

Summary of Data Extraction Method

2.1 Genomic Data Model (GDM)

GDM acts as a general schema for genomic repositories. The GDM datasets are a collection of samples. Each sample has two parts, *region data*, and *metadata*. Metadata describes the sample-specific properties while region data describes the portions of the DNA. [7]. Each GDM dataset has a corresponding data schema that has few fixed attributes to represent the coordinate of the regions and identifier of the samples. The fixes attribute which represents the region information consists of a chromosome, the region that chromosome belongs to, left and right ends of the chromosome and the denoting value of the DNA strand that contains the region. There may be other attributes besides fixed ones that have information on DNA region. Metadata stores information about the sample with format-free attribute-value tuples. Figure 2.1 illustrates an excerpt of GDM region data. As seen, First five columns are the fixed attributes of the region data that represents the region information. The last column, on the other hand, is the p-value of the region significance in this case. Figure 2.2 illustrates sample-specific metadata properties. The first column of both figures (id) maps the region data and metadata together.

id	chr	left	right	strand	*p*-value
1	2	2476	3178	*	0.00000000200
1	2	15 235	15 564	*	0.00000000052
1	5	8790	11 965	*	0.00000000009
1	5	75 980	76 342	*	0.00000000037
2	16	862	923	*	0.00000000018
2	16	1276	1409	*	0.00000000006
2	20	3852	4164	*	0.00000000031

Fig. 2.1 An excerpt of region data

id	attribute	value
1	antibody_target	CTCF
1	cell	HeLa-S3
1	cell_karyotype	cancer
1	cell_organism	human
1	dataType	ChipSeq
1	view	Peaks
2	antibody_target	JUN
2	cell	H1-hESC
2	cell_organism	human
2	dataType	ChipSeq
2	view	Peaks

Fig. 2.2 An excerpt of metadata

2.2 GenoMetric Query Language (GMQL)

GMQL is high-level query language which is designed to deal with large-scale genomic data management. The name *genometric*, come from its ability to deal with genomic distances. GMQL is capable to deal with queries with has over thousands of heterogeneous dataset and it is suitable for efficient big data processing [52].

GMQL combines the traditional algebraic operation with domain-specific operations of bioinformatics, which are spastically designed for genomics. Therefore it supports the knowledge discovery between millions of biological or clinical samples, which satisfies the biological conditions and their relationship to experimental [34].

The inclusion of metadata along with processed data in the many publicly available experimental datasets, such as TCGA, makes the initial ability of GMQL to manipulate the metadata is exceptionally important. GMQL operation forms a closed algebra meaning result are denoted as new dataset derived from operand. Hence, region-based operations build new regions, metadata based operations traces the root of each sample outcome. GMQL query, expressed as a sequence of GMQL operations, follows the structure below.

```
<variable> = operation(<parameters>) <variables>
```

Each variable indicates GDM dataset. The operator can apply one or more operand to construct one result variable. Each operator has its own parameters. Most of the GMQL operations are *relational operations*, which are mostly algebraic operations which are modified to fit the need of genomics data.

2.2.1 Relational GMQL Operations

- `SELECT` operator applies on metadata and selects the input samples that satisfy the specified metadata predicates. The region data and the metadata of the resulting samples are kept unaltered.
- `ORDER` operator orders samples, regions or both of them; the order is ascending as default and can be turned to descending by an explicit indication. Sorted samples or regions have a new attribute order, added to the metadata, regions or both of them; the value of `ORDER` reflects the result of the sorting.
- `PROJECT` operator applies on regions and keeps the input region attributes expressed in the result as parameters. It can also be used to build new region attributes as scalar expressions of region attributes (e g., the `length` of a region as the difference between its `right` and `left` ends). Metadata are kept unchanged.

- `EXTEND` operator generates new metadata attributes as a result of aggregate functions applied to the region attributes. The supported aggregate functions are `COUNT` (with no argument), `BAG` (applicable to attributes of any type) and `SUM, AVG, MIN, MAX, MEDIAN, STD` (applicable to attributes of numeric types).

- `GROUP` operator is used for grouping both regions and metadata according to distinct values of the grouping attributes. For what concerns metadata, each distinct value of the grouping attributes is associated with an output sample, with a new identifier explicitly created for that sample; samples having missing values for any of the grouping attributes are discarded. The metadata of output samples, each corresponding a to given group, are constructed as the union of metadata of all the samples contributing to that group; consequently, metadata include the attributes storing the grouping values, that are common to each sample in the group.

- `MERGE` operator merges all the samples of a dataset into a single sample, having all the input regions as regions and the union of the sets of input attribute-value pairs of the dataset samples as metadata.

- `UNION` operator applies to two datasets and builds their union, so that each sample of each operand contributes exactly to one sample of the result; if datasets have different schemas, the result schema is the union of the two sets of attributes of the operand schemas, and in each resulting sample the values of the attributes missing in the original operand of the sample are set to null. Metadata of each sample are kept unchanged.

- `DIFFERENCE` operator applies to two datasets and preserves the regions of the first dataset which do not intersect with any region of the second dataset; only the metadata of the first dataset are maintained.

2.2.2 Domain-specific GMQL Operations

Domain-specific operations are created specifically to respond the genomic management requirement needs.

- `COVER` operation is widely used in order to select regions which are present in a given number of samples; this processing is typically used in the presence of overlapping regions, or of replicate samples belonging to the same experiment. The grouping option allows grouping samples with similar experimental conditions and produces a single sample for each group. For what concerns variants:

  - `FLAT` returns the union of all the regions which contribute to the `COVER` (more precisely, it returns the contiguous region that starts from the first end and stops at the last end of the regions which would contribute to each region of the `COVER`).
  - `SUMMIT` returns only those portions of the result regions of the `COVER` where the maximum number of regions intersect (more precisely, it returns regions that start from a position where the number of intersecting regions is not increasing afterwards and stops at a position where either the number of intersecting regions decreases, or it violates the max accumulation index).
  - `HISTOGRAM` returns the nonoverlapping regions contributing to the cover, each with its accumulation index value, which is assigned to the `AccIndex` region attribute.

- `JOIN` operation applies to two datasets, respectively called **anchor** (the first one) and **experiment** (the second one), and acts in two phases (each of them can be missing). In the first phase, pairs of samples which satisfy the joinby predicate (also called meta-join predicate) are identified; in the second phase, regions that satisfy the **genometric predicate** are selected. The meta-join predicate allows selecting sample pairs with appropriate biological conditions (e.g., regarding the same cell line or antibody).

- `MAP` is a binary operation over two samples, respectively called **reference** and **experiment**. The operation is performed by first merging the samples in the reference operand, yielding to a single set of **reference regions**, and then by computing the aggregates over the values of the experiment regions that intersect with each reference region for each sample in the experiment operand. In other words, the experiment regions are mapped to the reference regions.

  The output of `MAP` operation is called **genometric space**, which is structured as a matrix. In this matrix, each column indicates the experiment sample and each row indicates the reference of the region while matrix entries are scalar. Resulting matrix can be easily inspected with heat maps, which can cluster the columns and/or rows to display the patterns, or processed and analyzed by any other matrix-based analytical process. To summarize `MAP` operation allows the quantitive readings of stored experiments with respect to reference region. If the reference region of the biological data is not known, Map function allows extracting the most interesting reference region out of the candidates.

  Fig. 2.3 illustrate the effect of MAP operation on a small portion of the genome; Input has one reference sample with 3 regions and three corresponding mutation experiment

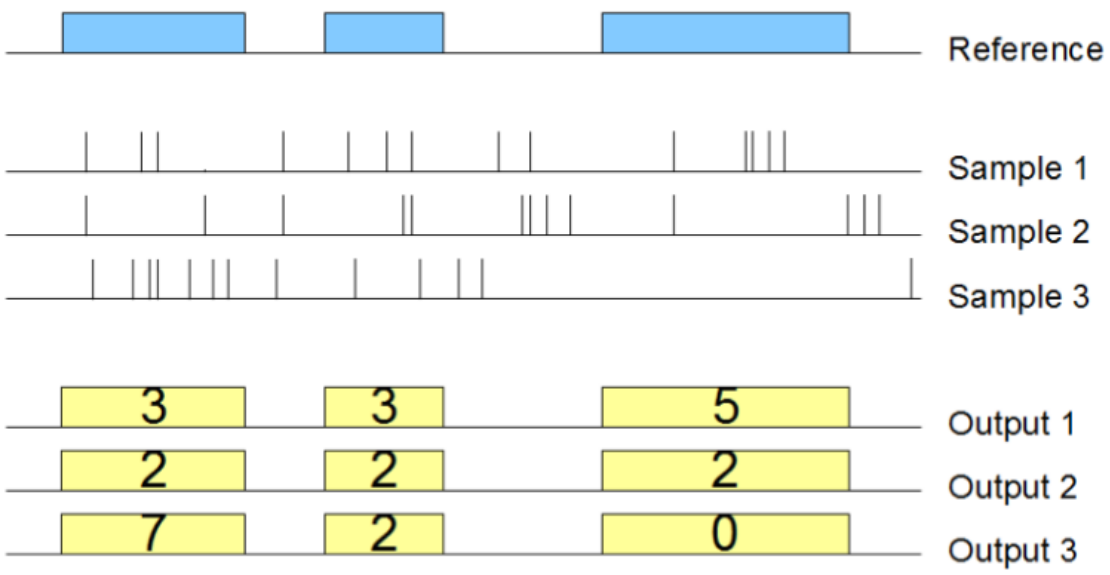


Fig. 2.3 Example of map using one sample as reference and three samples as experiment, using the Count aggregate function.

samples, Output has three samples, which locates int he same region as the reference sample, as well. The features of reference sample correspond to the number mutations that are interacting with those regions. The final result can be explicated as a (3x3) genome space.

2.2.3 Utility Operations

- `MATERIALIZE` operation saves the content of a dataset into the file system, and registers the saved dataset in the system to make it seamlessly usable in other GMQL queries. All datasets defined in a GMQL query are, temporary by default; to see and preserve the content of any dataset generated during a GMQL query, the dataset must be materialized. Any dataset can be materialized, however, the operation is time expensive. Therefore to achieve the best performance it is suggested to materialize the relevant data only [7].

2.2.4 Biological Example

The biological example below uses the MAP operation from domain-specific GMQL operations to count the regions that are peaked in each ENCODE ChIP-seq sample that is intersected with a gene promoter. After that, for each sample, it projects over the promoters

with eat least one intersecting peak and counts these promoters. As the last step, it extracts the top three samples with highest count number of such promoters

```
HM_TF = SELECT(dataType == 'ChipSeq') ENCODE;
PROM = SELECT(annotation == 'promoter') ANN;
PROM1 = MAP(peak_count AS COUNT) PROM HM_TF;
PROM2 = PROJECT(peak_count >= 1) PROM1;
PROM3 = AGGREGATE(prom_count AS COUNT) PROM2;
RES = ORDER(DESC prom_count; TOP 3) PROM3;
```

Further details about GMQL basic operators, GMQL syntax and relevant examples of single statements and a notable combination of them are available at GMQL manual [1] and GMQL user tutorial [2].

2.2.5 Web Interface

In order to make GMQL publicly available and user-friendly for the ones with limited computer science experiment, a web interface designed and implemented by GeCo group. Two services developed for this purpose. REST API and web interface. Both of them have the functionalities to search the genomic features dataset and biological/clinical metadata, which are collected in system repository from ENCODE and TCGA, and build GMQL queries upon them. GMQL interface can efficiently run such queries with thousands of samples with few heterogonous dataset. Moreover, with user management system, private datasets can also be upload to the system repository and used in the same way with the available datasets in the system. GMQL REST API planned to used them with the external systems such as GALAXY [20], which is another data integration system and workflow that is commonly used in bioinformatics, or other systems that can run REST services over HTTP. [3]. Figure 2.4 illustrates the web user interface of GMQL.

[1] GMQL Manual: http://www.bioinformatics.deib.polimi.it/genomic_computing/GMQL/doc/GMQL_V2_manual.pdf

[2] GMQL User Tutorial: http://www.bioinformatics.deib.polimi.it/genomic_computing/GMQL/doc/GMQLUserTutorial.pdf

[3] GMQL REST Services: http://www.bioinformatics.deib.polimi.it/GMQL/interfaces/

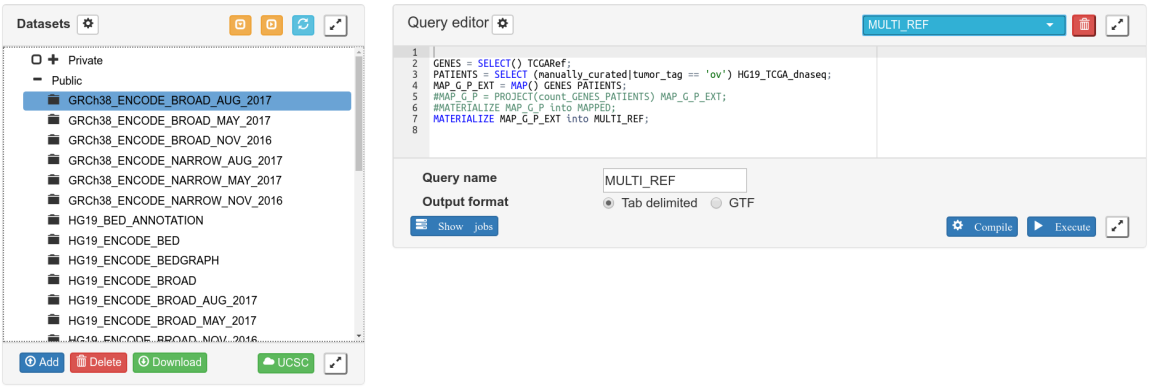


Fig. 2.4 The web graphical user interface of GMQL

2.2.6 Python Interface

Python interface of GMQL, PyGMQL, is an alternative to the web interface of the system. Python library interact with the GMQL through Scala back-end. With PyGMQL it is also possible for users to write GMQL queries with the same syntax as standard Python pattern. PyGMQL can be used locally or remotely. GMQL queries executed on the local computer when it is used locally while it is executed on the remote GMQL server when it is used remotely. It is also possible to switch between remote and local mode during the pipeline analyzing stage. Moreover, PyGMQL introduces efficient data structure to analyze GDM data and provides specifically developed data analysis and machine learning packages to manipulate genomic data. Figure 2.5 describe the interaction between the user and GMQL with PyGMQL in a high-level representation.

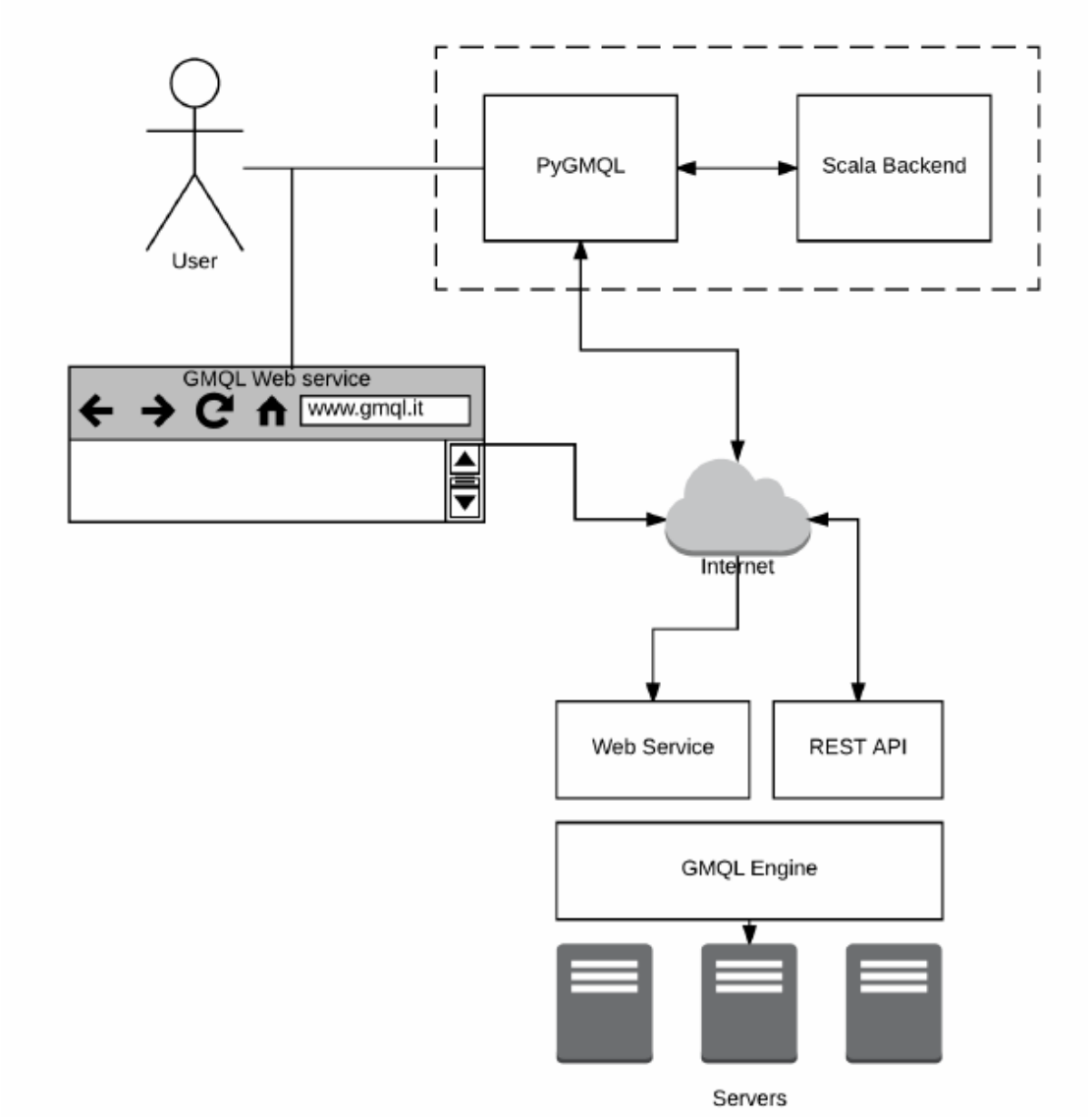


Fig. 2.5 High-level representation of the GMQL system

Chapter 3

Tertiary Analysis of the RNA-Seq Data

Numerous studies made in past years to examine the *transcriptome* [1] to identify healthy or diseased. Study of Golub et al. [21], about distinguishing the types of severe leukemia cancer is one of the pioneering work in this area. After that, many ensuing studies on both supervised and unsupervised on expression data is done. Preceding studies on transcriptome analysis (DNA microarrays) were only available for few types of cancer yet they still did not have enough sample most likely fewer than a hundred. DNA-Seq technology of today gives more precise and complete quantification of the transcriptome with the publicly available dataset. For instance, TCGA dataset contains 33 different types of cancer, including 10 rare cancers and hundreds of samples [19, 39, 2]. Figure 3.1 shows the available cancer types and the amount of samples in TCGA dataset. The interested readers may refer to [54, 40, 23, 45, 38] for a detailed explanation of the RNA-Seq technology.

[1] Transcriptome is the sum of all RNA molecules in a cell or a group of cells that are expressed from the genes of an organism.

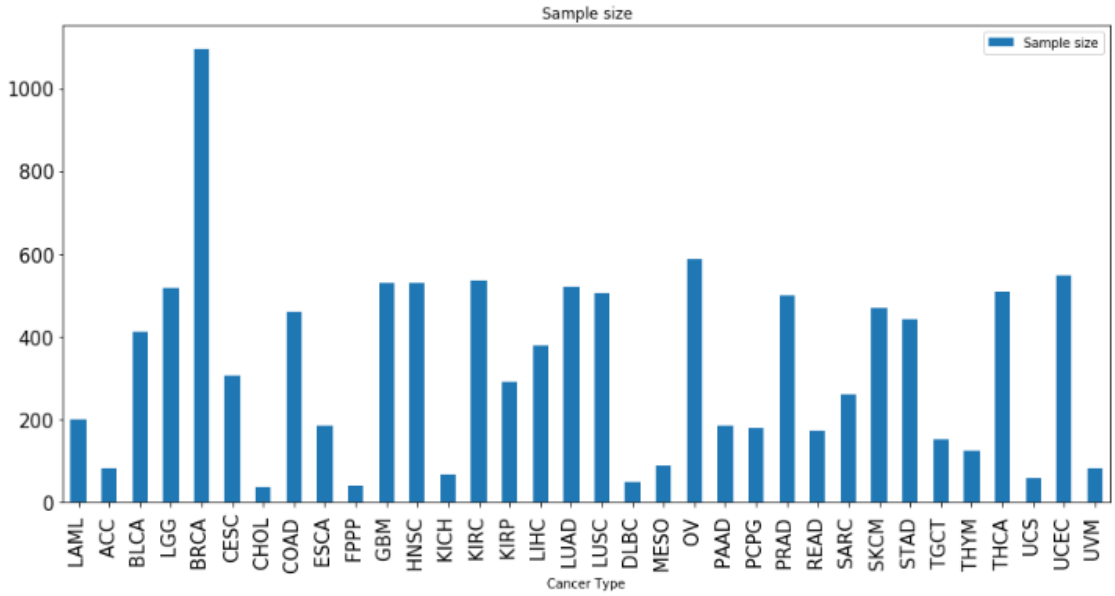


Fig. 3.1 TCGA Cancer Datasets with corresponding sample numbers

3.1 Characteristics of RNA-Seq Data

3.1.1 High Dimensionality

Gene expression data has a considerably small amount of sample while a relatively high amount of features (genes). However, this characteristic is very common an expected in RNA-Seq and DNA Microarray technologies. As it can be clearly seen in Figure 3.1, only few cancer type has more than 500 sample. Despite that, each sample has approximately 20k genes. These phenomena is called the curse of dimensionality. Curse of dimensionality has 2 important traits that make data hard to deal.

1. o have a large number of features which may be irrelevant in a certain comparison of similarity, have corrupt the signal-noise ratio.

2. In higher dimension, all samples "seem similar"

There few feature extraction methods to overcome the high dimensionality problem. Chapter 4 introduces few methods that are used in this thesis.

3.1.2 Biases of the RNA-Seq Data

Having biases is yet another characteristic of RNA-sequential data. Some of the biases need to be taken care before deeply analyzing the data. First bias occurs because of the

technical issues regarding the sequencing depth [51]. Observations in RNA-seq data suffer from a strong dependency on sequencing depth for their differential expression calls, and each observation might have a different number of total reads. Thus, values on the dataset do not only depend on the expression value of the genes but also depends on the depth of sequence matrix. The difference in the length of genes in the RNA-seq data causes another, more critical bias. Longer the genes more the readings it gets. This tendency cause several problems for analysing methods such as classification and clustering [40]. Hence, these need to be solved before analyzing in detailed, otherwise, the final results can not be trusted. Normalization techniques need to be performed in order to clear up the problems with biases. Normalization also helps to improve the convergence speed of various machine learning algorithms. Normalization also necessary for feature reduction techniques such as PCA (explained in Section4.1.5), to allow all genes to have an opportunity to contribute the analysis. PyGMQL implements two methods for data normalization. One for shifting the mean value of every feature (gene) to zero. Another for reducing the variance of every feature to the unit variance. By applying those normalization techniques, we can assure that all of the genes are equally weighted for the classification or clustering.

3.1.3 Missing Values

RNA-Seq data commonly contain missing values like most of the experimental datasets. Classification approaches are not robust in presence of the missing values. Therefore missing value problem should be solved. The most simple solution to deal with the missing problem is to erase the samples to minimize the effects of incomplete datasets. However, considering the amount of sample in TCGAS datasets, erasing might not be a "smart" approach. De Souto et al. [14] pointed that it is common for the gene expression datasets to have up to 5% of missing values, which could affect up to 90% of the genes. Another method is filling the missing values with 0. Yet this approach performs poorly in terms of *estimation accuracy*. More advanced approach is instead of filling with 0, missing values filled by mean or median values of its corresponding column so the effects of the missing values over the general set minimized in computations. Filling missing values, with random values from the original dataset is another approach to solve this problem.

3.1.4 Unbalanced Distribution

Even though the cost of DNA sequencing falls around 1K$, the amount is not small enough for collecting the sequential data casually.Also, finding volunteers that allows sharing their DNA information on public domains is hard to find. As a result, most of the samples in

the available data collections are cancer positive. Unbalancing in the samples make the data unfavorable for direct feed to Deep Learning network. Methods explained in bleow is necessery to sole this problem

3.1.5 Dealing with Imbalanced Data

Imbalanced input data most of the time cause the predictions of DNN model to give prediction highly on one label. There are few methods to prevent them.

Up-sampling / Down-Sampling

Up-sampling and *Down-sampling* basically means to increase or decrease the sample of desired class. *Up-sampling* means, randomly duplicating the samples of the class, which has smaller sample amount until the balanced ratio between the classes is reached while *Down-sampling* means, instead of taking the whole samples from the overly populated class, just takes the same amount of sample with the other class so the final dataset will be balanced. Since the amount of sample in the data is already too small, down-sampling of the cancerous data is not logical. Furthermore, up-sampling of healthy data does not just fail to stabilize the result, but also fails the train data. Training accuracy moves around 50% as same as null hypothesis.
There is also another method named SMOTE [8] published for up-sampling, which introduced a new way to create new samples. SMOTE calculate the k nearest neighbors for each minority class observations. Depending upon the amount of oversampling needed, it creates a synthetic example along the line connecting the observation with k random nearest neighbors. However this method might be risky. There is a possibility for newly created synthetic data to indicate complete new life form, and there is not any way to check it. Because of the this reason, SMOTE is passed.

Penalized weight

Penalized the weight, of the class with superior sample number, in the logistic regression function or in the loss calculation step have possibility to solve the imbalanced data problem.

Proportioned Batch

Imbalanced data may not be a problem if the input batch that is used to feed training step has the same proportion of classes with the whole training data.

3.2 Loading TCGA Data into Repository

The original TCGA data consist of 31 different types of cancer. There are 9.825 samples with 20.271 diverse genes. Only two cancer types (kidney and breast) selected for classification test in deep learning. Breast data selected number of samples , which is the highest amount included in TCGA dataset, and kidney is selected randomly from the rest. In the raw data, samples do not have values for all the genes sequences so these missing values are filled with the mean of that gene expression values to not change the outcome. Also, data normalization is performed in order to remove the biases described in section 3.1.2. How to load and preprocess the data is explained in code snippet 3.1. Whole code can be seen in Github repository [1].

```
import gmql as gl
path = './Datasets/tcga_data/files/'
# the normalized count value is selected
selected_values = ['normalized_count']
# we are only interested in the gene symbol
selected_region_data = ['gene_symbol']
# all metadata are selected
selected_meta_data = []
gs = gl.ml.GenometricSpace()
# to load the data
gs.load(path,selected_region_data,selected_meta_data,
selected_values,full_load=False)
# matrix representation
gs.to_matrix(selected_values, selected_region_data, default_value=None)
# compact representation of region and metadata
gs.set_meta(['biospecimen_sample__sample_type_id',
'manually_curated__tumor_tag','biospecimen_sample__sample_type'])

from gmql.ml.algorithms.preprocessing import Preprocessing
# pruning the genes that contain more than %40 missing values
gs.data = Preprocessing.prune_by_missing_percent(gs.data, 0.4)
# missing value imputation
gs.data = Preprocessing.impute_using_statistics(gs.data, method='min')
# gene standardization
gs.data = Preprocessing.to_unit_variance(gs.data)
```

Code Listing 3.1 Preprocessing of raw TCGA data

Chapter 4

Theoretical Background

This chapter provides basic knowledge for machine learning, neural network and deep learning concepts, that are used during thesis.

4.1 Machine Learning

Machine learning is one of the many areas in artificial intelligence that gives computers to ability to learn without being specifically programmed. The main idea behind the machine learning is, regardless of the instructions to solve a problem, constructing a model with an acceptable amount of data, which can produce valuable predictions of the solutions. In consequence, machines learning is about various models, which use various methods to learn by adapt and improve their result from experience. "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E " Mitchell [36]. Machine learning can be applied in many fields, such as financial services, health-care, customer segmentation , transportation and so on. In this thesis, it is used a binary classification task with TCGA data. Main goal is to create a model which can minimize the differences between predictions $\mathbf{\hat{y}_i} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$ and desired labels $\mathbf{y_i} = (y_1, y_2, ..., y_n)$ of input $\mathbf{x_i} = (x_1, x_2, ..., x_m)$. The i corresponds to instance of the predictions while m and n corresponds to the total number of input and output instances, respectively.

There are three subfields of machine learning; *supervised learning*, *unsupervised learning* and *semi-supervised learning*.

4.1.1 Supervised Learning

Supervised learning is where the input variables and the output variables are already determined and a learning algorithm is used to learn how to map the input to the output. In other words, the supervised learning algorithms learn by inspecting the given inputs $\mathbf{x}$ and desired outputs $\mathbf{y}$ with correcting the predictions$\mathbf{y_i}$. There are several supervised learning algorithms such as Linear Regression (LR), K-nearest Neighbors (KNN), support vector machine, decision trees, Naïve Bayes classifier in the literature.

4.1.2 Unsupervised Learning

Unsupervised learning is where the input data do not contain any information of corresponding output variables. The main goal of unsupervised learning is to find meaningful patterns in distribution or model the underlying structure to increase the understanding of the input data. It is called unsupervised because unlike the supervised learning there is no "correct answer" or supervisor to control the learning process. Algorithms are free to make their own connections and discover new structures in order to interpret the data. Most common use of unsupervised learning is cluster analysis.

4.1.3 Semi-supervised Learning

Semi-supervised learning is placed between supervised and unsupervised learning, it involves the function estimation on labeled and unlabeled data. The main goal of semi-supervised learning is to use unsupervised learning to make predictions for unlabeled data and feed these data into supervised learning to predict the new data. Most of the real world machine learning problems are in the semi-supervised category since collecting labeled data could be expensive, time-consuming while collecting unlabeled data is relatively easier cheaper and easier to collect.

Semi-supervised learning, as an idea is the most similar learning method for the human and animal brain. “We expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object.” LeCun et al. [30]

Regression and classification are the two essential problems of the supervised learning. Regression is used to model the correlation between input samples and targets which are continuous while the targets of classification are categorical. To make it clear with example, classification is classifying the given genomic data as "cancerous" or "healthy", which is the

classification task handled in this thesis, while regression example could be estimating the market price of the certain house.

4.1.4 Linear Regression

Linear Regression is an statistical method that enables to figure out the statistical relationships between two numerical variables. Supposed that the input data is in form of $(x_1, y_1), ..., (x_n, y_n)$, where the x is the input value and y is the corresponding output while x_i and y_i are real numbers with i within 1,...,n and n refers to the sample size, the model that predict the numerical response $f(X) = Y$ by assuming there exist a linear relation between the input and the output. The relation between input and output can be mapped as Equation (4.1).

$$Y = \beta_0 X + \beta_1 \tag{4.1}$$

In Equation (4.1), β_0 and β_1 are the unknown constants that represent the slope and intercept of the predicted regression line. Main goal is, by training the model, finding a best fitting line through all data.

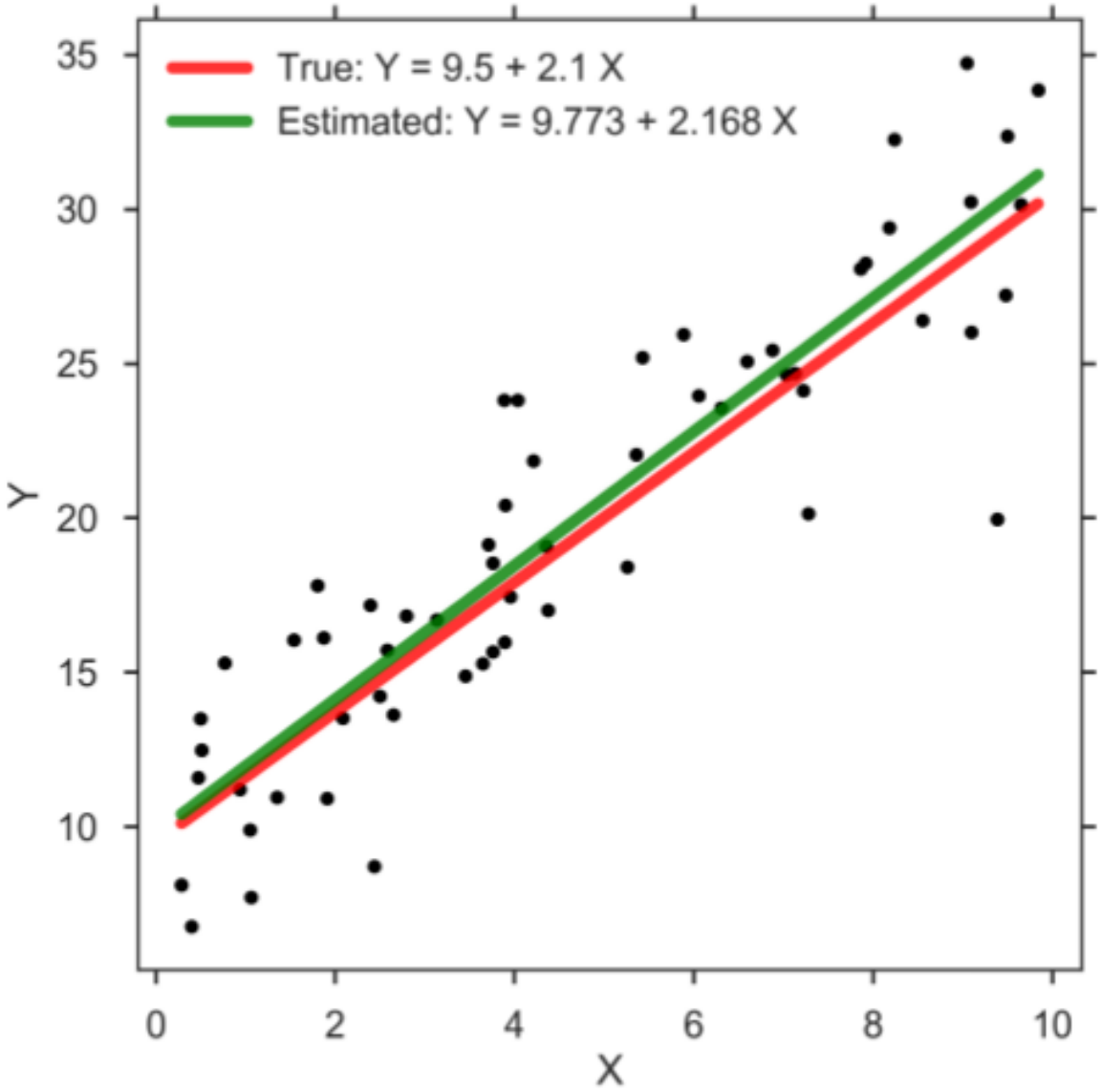


Fig. 4.1 Example of linear regression estimation

An example with one variable linear regression is shown in Figure 4.1 [1]. Figure consistent of multiple black points with two straight lines. The green line consist of the predicted values $\hat{y}_i$ for each possible value of x_i while the red line indicates the correct outputs y_i. The distance between each black point and and green line is called prediction error or loss.

The better estimations is done by minimizing the loss function. The loss function $\mathscr{L}$ is the sum of all individual losses over the instances of X. Assuming $e_i = y_i - \hat{y}_i$ represent the

[1] Source of the picture is https://dzenanhamzic.com/2016/07/25/linear-regression-with-one-variable-in-matlab

i^{th} individual loss, the loss function becomes:

$$\mathscr{L} = \sum_{i=1}^{n} \mathrm{e}_i \tag{4.2}$$

The aim is to find the β_0 and β_1 values, which can minimize the $\mathscr{L}$. Thus the important thing is to choice of selection the best loss function for the problem. The most commonly used one is residual sum of squares (RSS):

$$RSS = \sum_{i=1}^{n} \mathrm{e}_i^2 \tag{4.3}$$

4.1.5 Classification

Classification used for predicting the numerical responses. The main goal is to create a model, which can assign the given input vector into available classes accurately based on the training set and labels. In most of the cases, the input classes are disjoint meaning each sample could only belong to one and only one class. Therefore, input space separates into decision regions which is called *decision boundary* or *decision surface* [6].

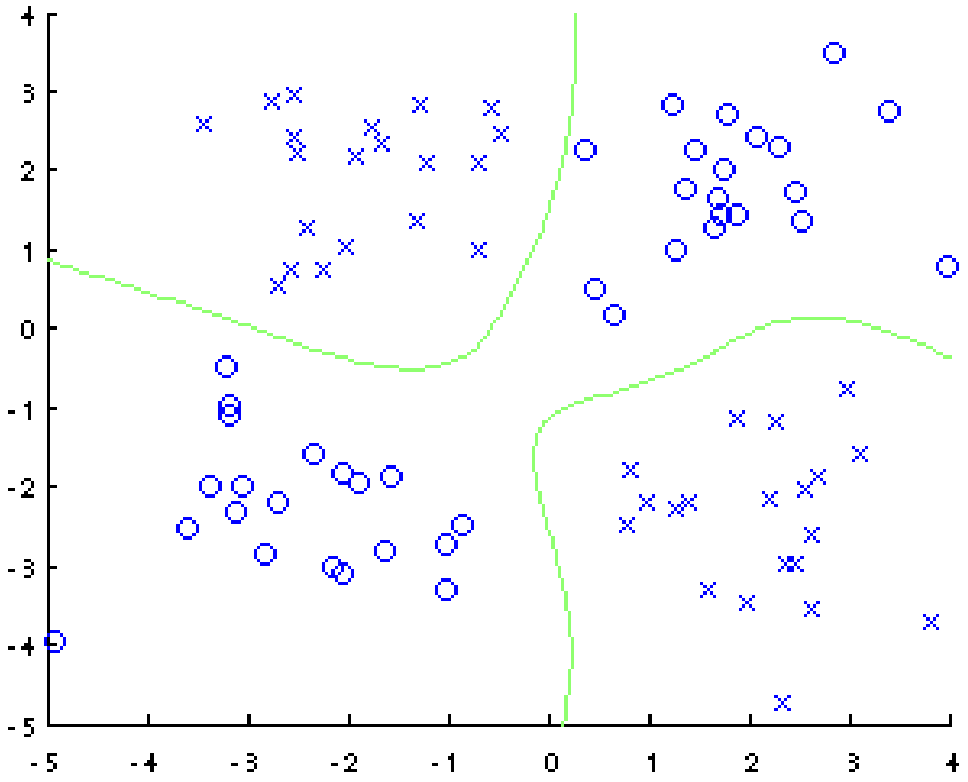


Fig. 4.2 Decision Surface

Figure 4.2 [2] illustrates the decision surface. It consists of 2 different points, 0 and X that indicates the class labels and 2 lines that draws a boundary between points such as when the new data comes If it not visualized in very controversial spot, the class it belongs to will be easy to predict.

Classification task consist of four main phases, *preprocessing*, *feature extraction*, *training* and lastly *classification*.

Preprocessing: In preprocessing stage, raw data, which is gathered from source, handled before employed as an input. This phase is necessary since most of the cases, input data obtained is noisy, incomplete or /and inconsistent. Preprocessing involves data-related tasks such as cleaning, transformation, reduction and so on. Preprocessing steps that is used to prepare the input data for the task handled in this thesis is explained in Section 3.1 to deal with problems of the raw data.

Feature Extraction: Next phase is feature extraction. Features are the domain-specific measurements, which have relative information to create the best possible representation of

[2]Source for the figure is http://www.work.caltech.edu/ htlin/program/libsvm/

the input. For classification, most relevant features need to be extracted from raw data. There is a lot of feature selection methods in literature such as ANOVA, PCA, LDA and etc. all with their own advantages and disadvantages.

As a task in this thesis is a classification problem with input data having a lot of features, three feature extraction algorithms, all chosen from commonly used as feature extracting methods from related genomic classification papers; ANOVA, PCA, Random Forest are introduced in this part.

- ANOVA *(Analysis of Variance)* : ANOVA is the one of the mostly used feature section methods that is used in machine learning to deal with high dimensionality problem. Proposed by Bharathi and Natarajan [5], it uses F-test to select the features that maximize the explained variance.

- Random Forests: *Random forests* is another commonly used feature selection algorithm [16]. Random forests is set of decision tree classifiers. Each node splits dataset into two, based on the condition value (impurity) of every single feature. This way similar instance to falls in the same set. For classification purposes, Gini impurity or information gain/entropy is chosen to be impurity condition. This structure calculates the effect (how much it decrease the impurity) of every feature on the tree. The flowchart for *random forests* can be seen in the Figure 4.3. 7

- PCA *Principal Component Analysis* : Rather than feature selection, *Principal Component Analysis* is a dimensionality reduction technique that used to transform high-dimensional dataset into a smaller dimension. Reducing the dimensionality of dataset also reduces the size of space, number of freedom of hypothesis. Algorithms work faster when they need to deal with a smaller dimension and visualization become simpler.

  PCA transforms the dataset to a lower dimension one with a new coordinate system. In that coordinate system first axis indicates the first principal component which has the greatest variance in data. With PCA it is possible to explain 95-97% of the variance in the input dataset with fewer PCA compare to the original dataset. For instance, 500 PC is enough to explain 98% of the variance in the TCGA kidney dataset, while it only explains 89% of the variance in breast dataset. Since PCA focus to find the principal component with the highest variance, the dataset should be normalized, so that each attribute has an opportunity to contribute the analysis.

Training: In this phase as also explained before, model trains to give the most accurate prediction as possible.

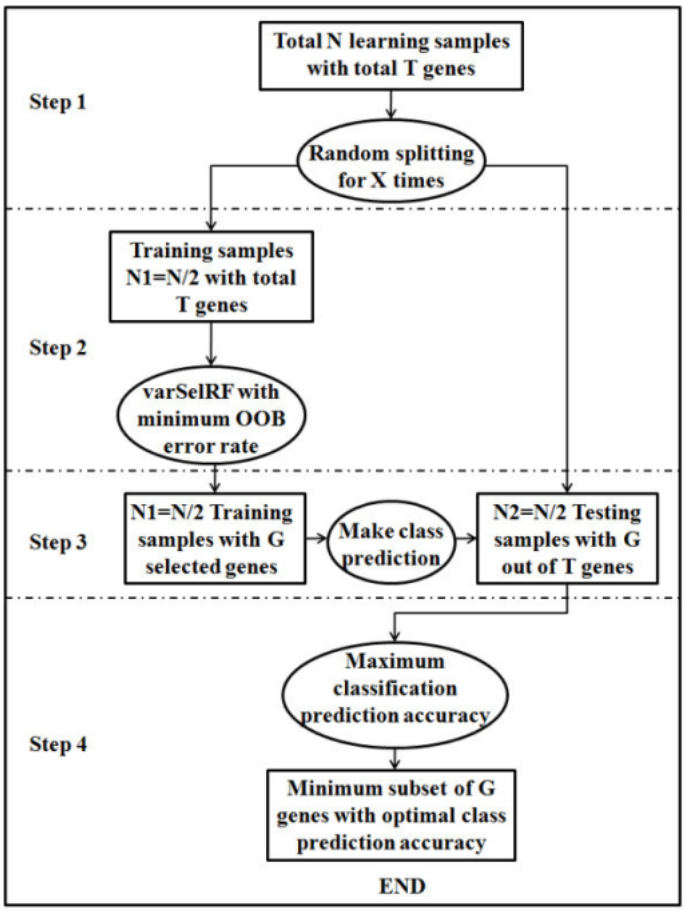


Fig. 4.3 Steps of random forest algoritm [22]

Classification: In classification, the created model assign input into one of the available classes based on created the decision rules.

Classification problem consists of three main parts. First, the frequency of the classes to occur in the input data, probability distribution. Second, characteristic features to define the relationship between input and output for separating the classes. Third, defined loss function which penalized the inaccurate predictions such that the final cost should be minimized as it mentioned in Subsection 4.1.4 (Regression) [35]. Therefore most of the classification tasks in real world, based on the probability theory. Probabilistic model outputs a vector which contains the probability of all possible classes, to show the degree of certainty.

After training model is finalized and the training is done, model can be used to predict the possible class of the new data. After training model is finalized and the training is done, the model can be used to predict the possible class of the unseen data. However, every classifier that is built tries to memorize the training set. Training constantly may cause the model to memorize the training set which will end up with poor performance when dealing with new

data while getting perfect accuracy with the training set. These phenomena called *over-fitting*. What is needed is, to create a model which is capable of generalizing so that it performs well new data as well as the training data. Over-fitting also could occur because of the complexity of the model.It is hard for a model to generalize the inputs with complex structures. There are few methods to detect the over-fitting, one of which is explained in Section 5.3.

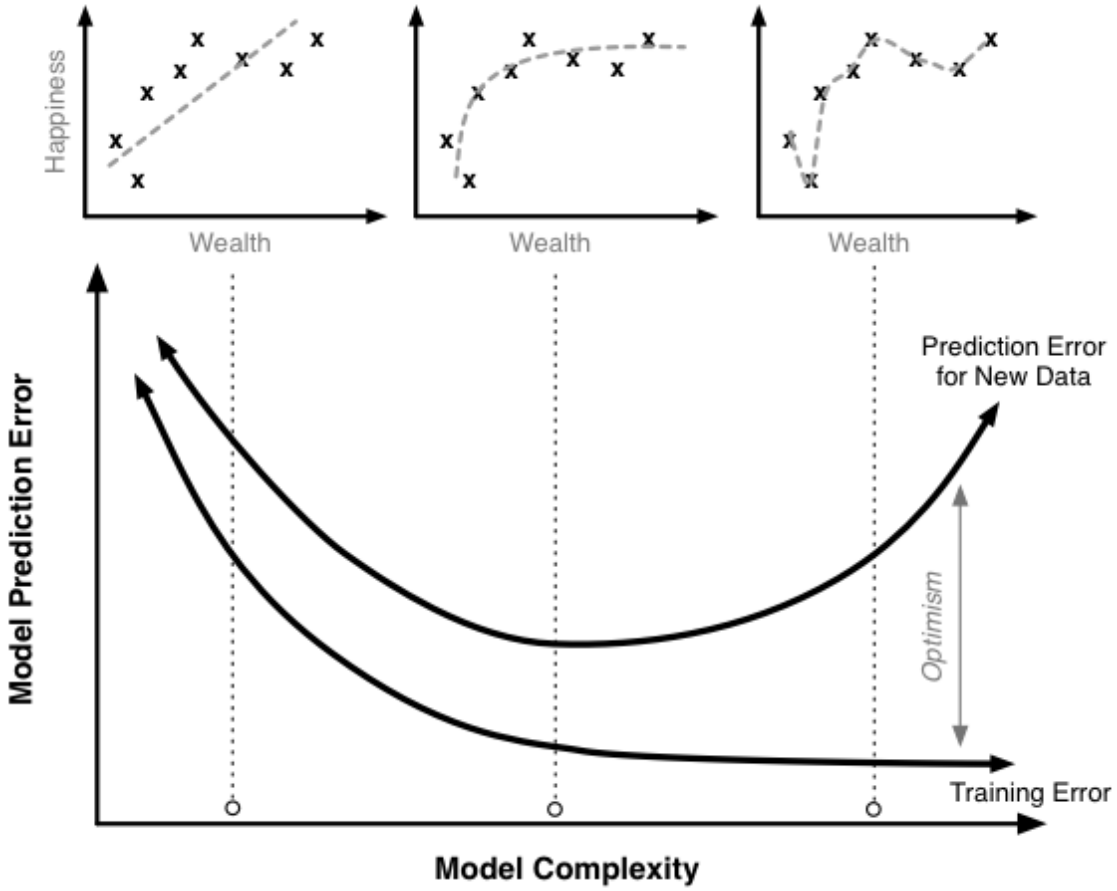


Fig. 4.4 Decision Surface

Figure 4.4 [3] shows three models which are trained with three different complexity level on top. The figure on the bottom shows the training and prediction error rate. Training error rate getting lower and lower which the increase in complexity, however, prediction error rate starts to increase after some point. Top right graphic, shows the over-fitting example which has the perfect curve to include all existing data yet it gets poor performance when to deal with a new sample. On the other hand, the graph in the middle has pattern instead of following each data point which has worse training error but better prediction error. In order

[3] Source for the figure is http: // scott. fortmann-roe. com/ docs/ MeasuringError. html

to create a "good" model which is capable of dealing with new data, the complexity of the model and the amount of training steps should be chosen carefully.

4.2 Deep Neural Network

The deep network is a subsection of machine learning, that models the data using multiple layered artificial neurons. The idea is copying the interactions of the actual nervous system. The figure illustrates the real neural network, as seen, dendrites bring input signals where axons pass the information from one cell to another cell through synapses.

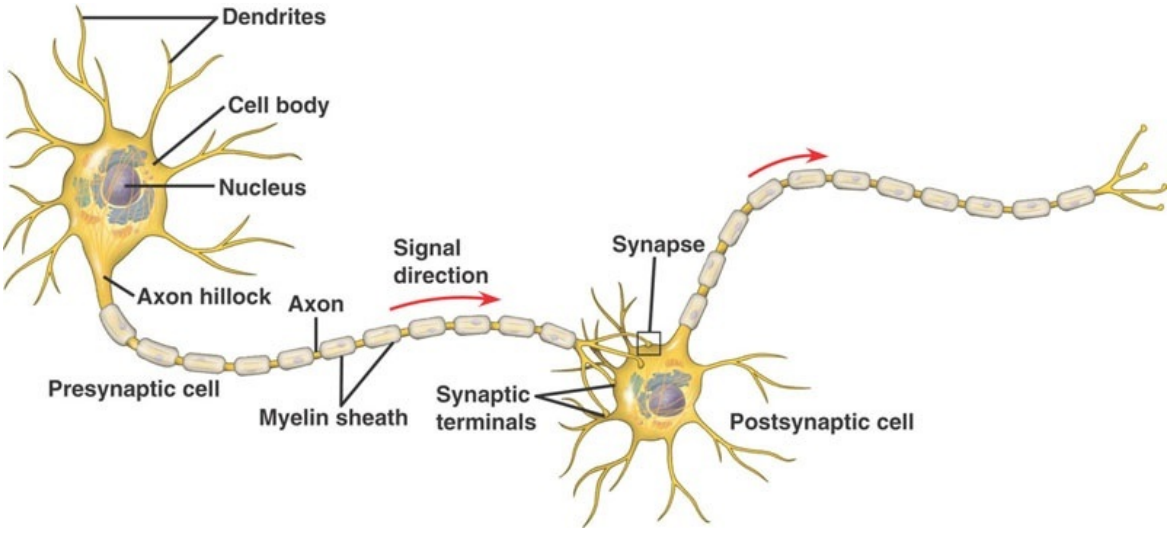


Fig. 4.5 Decision Surface

Artificial network "learns" by adopting changes in the synaptic weights of the network. The architecture has parallelly distributed structure with an ability to learn, which make it possible to solve complex classification tasks with in reasonable time.

.

4.2.1 Perception

Perception is a feed forward network that builds the boundaries of a linear decision, which is a fundamental part of the neural networks. Figure 4.6 [4],illustrates the perception in the ANNs. Input signal can be represented as weights. Transfer function sums the response and feed the resulting signal to the activation function. Based on the activation function's threshold, output is determined. To simplify it can be represent as wighted sum of inputs:

[4]Source for the picture is https: // nl. wikipedia. org/ wiki/ Perceptron

$$y = \sum_{j=1}^{n} w_j x_j + w_0 \tag{4.4}$$

where: $n =$ The number of outputs

$w_0 =$ Value of interception from bias unit $x_0 = +1$

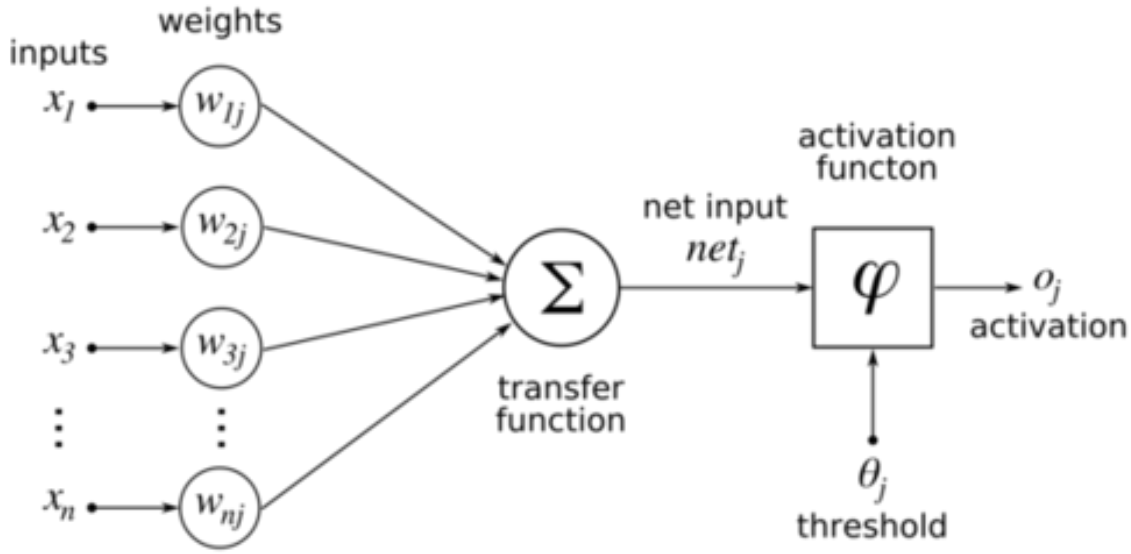


Fig. 4.6 Perception

For linear cases, binary threshold can be used as an activation function (e.g. step function) to determine the output. Then, output of the system decided as a:

$$o(x) = sgn(w \cdot x) \tag{4.5}$$

where:

$$sgn(y) = \begin{cases} +1 & \text{if } y > 0 \\ -1 & \text{otherwise} \end{cases} \tag{4.6}$$

On the other hand, to represent non-linear functions, non-linear activation functions are necessary. There are few non-linear activation functions in the literature. Figure 4.7 illustrates tanh, sigmoid and rectified linear unit (ReLU), which also three of the most common ones.

Sigmoid function compress the input in to [0; 1] range, where 0 means it is not activated and 1 means it has maximum frequency. Tanh, compress it in to [-1 ; 1] range with same logic. Relu has an output of 0 if the input is smaller than 0, output the the raw input otherwise

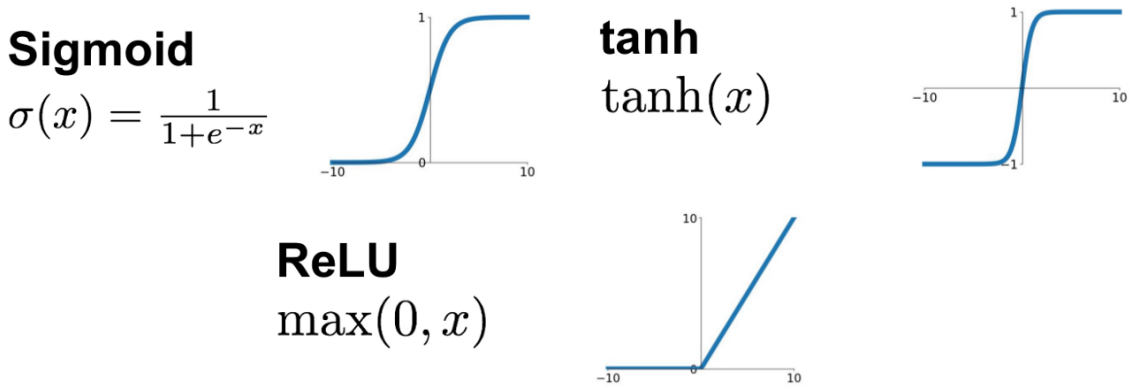


Fig. 4.7 Activation functions: sigmoid, tanh, ReLU

$$sigmoid(x) = \frac{1}{1+e^{-}x} \tag{4.7}$$

$$tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{4.8}$$

$$ReLU(x) = max(x,0) \tag{4.9}$$

Tanh can be considered as a scaled version of sigmoid function:

$$tanh(x) = 2sigmoid(2x) - 1 \tag{4.10}$$

There are a lot of approaches for perception to predict the correct class label by finding the optimum weight vector of the training data. The most common one known as the perception learning rule. If the classification problem is linearly separable, perception learning rule always converges the optimum weights in finite time [43]. For this approach, all weights are initialized with random values between (-1, +1). Then perception applies to each training example. With each wrong classification, weights are updated by adapting them at each iteration with equation (4.11) until the output is correctly classified.

$$w_i < -w_i + \Delta w_i \tag{4.11}$$

where:

$$\Delta w_i = \alpha(t-o)x_i$$

4.2.2 Multilayer Perception

Single perception is limited to linear mapping, so it fails to solve complex tasks. To solve complex tasks, more comprehensive model, which is capable of performing the arbitrary mapping. With perception, it is possible to build larger and more practical structure that can be described as a network of perceptions named multilayer perception (MLP). The basic structure of MLP could also be called as feedforward network (FFN) since it has a direct connection between all layers. General MLP has three main layers; one layer for input, one or more layer as a hidden layer and one layer for the output. Hidden layer is "hidden" from inputs and outputs as it is indicated in the name. Learning task of the complex model by extracting features from input data, handled in the hidden layer. MLP with one hidden layer structure is known as two-layer perception, is shown in Figure 4.8.

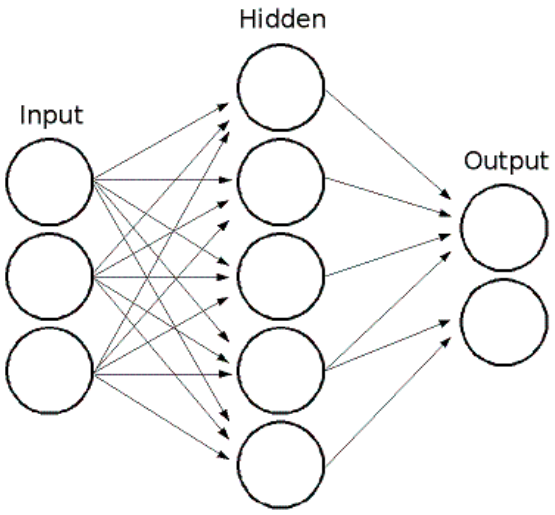


Fig. 4.8 MLP (FFN) with one hidden layer

Furthermore, as proved in [12] and [26], an MLP with a single hidden layer represents a universal function approximation. A two-layer perception can be written mathematically as:

$$y = f(x) = \varphi(b^{(2)} + W^{(2)}(\varphi(b^{(1)} + W^{(1)}x))) \tag{4.12}$$

where:

$W^{(1)}, W^{(2)}$ = Weights

$b^{(1)}, b^{(2)}$ = Biases

φ = activation function

In the equation (4.12), $\varphi(b^{(1)} + W^{(1)}x))$ forms the hidden layer and the rest establish the output layer.

4.2.3 Training an DNN

Deep neural network learns by minimizing the loss function by changing the parameters ($\theta = \{W^{(*)}, b^{(*)}\}$) of the model in training. *stochastic gradient descent* (SGD) is the one of the most common approach to used for learning the parameters. The gradients of a loss function are calculated by using the *back-propagation* (BP) algorithm, then the results fed to the SGD method to update the weights and biases.

Stochastic Gradient Descent

SGD algorithm updates the set of parameters θ incrementally after each epoch. An epoch indicates the number of times all of the training input used to update the parameters. All training samples get through the leaning phase in one epoch before parameters are updated.

SGD calculates the approximation of the true error gradient error based on a single training sample, instead of compute the gradient of the error based on the all training sample like in gradient descent (GD). Thus DNN can train faster with SGD, since calculating the approximation is faster.

$$w_j := w + \Delta w_j \tag{4.13}$$

$$\Delta w_j = \alpha \sum_{i=1}^{n} (target^{(i)} - output^{(i)}) x_j^{(i)} \tag{4.14}$$

where:

α = Learning rate

After each epoch, weights in Equation (4.13), updated as:

$$\Delta w_j = \alpha (target^{(i)} - output^{(i)}) x_j^{(i)} \tag{4.15}$$

Backpropogation

To use SGD in multi-layer networks, gradient of the loss function is needed to be computed. Backpropogation is the most common method used to overcome this problem. In backpropogation, calculating the partial derivatives $\partial L/\partial_w$ of the loss function L with respect to some weight w is enough to analyse the cahnge in the loss with the change of weights. Using mean squared error (MSE) as cost function one output neuron over all n examples is:

$$L = \frac{1}{2}\sum_{j=1}^{n}(t_j - y_j)^2 \tag{4.16}$$

where:

$t =$ target label

$y =$ output of the perceptron

L is scaled by $\frac{1}{2}$ for mathematical convenience of Equation (4.22). Error gradient is calculated in equation (4.17) to use SGD

$$\Delta w_{kj} = -\alpha\frac{\partial L}{\partial w_{kj}} \tag{4.17}$$

where a node in layer k is connected to a node in layer j. The result is taking negative We take the negative because the change of the weights are in the direction of where error is decreasing. because weight changes are in the direction where the error is decreasing. Using chain rule gets us:

$$\frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial y_j}\frac{\partial y_j}{\partial x_j}\frac{\partial x_j}{\partial w_{kj}} \tag{4.18}$$

In Equation (4.18), x_j is the weighted sum of the inputs being passed to j^{th} node and $y_j = f(x_j)$ is the output of the activation function. As a result:

$$\frac{\partial x_j}{\partial w_{kj}} = y_k \tag{4.19}$$

If the sigmoid function is used as an activation function, the derivative becomes:

$$\frac{df(x)}{dx} = f(x)(1 - f(x)) \tag{4.20}$$

Plugging it in to Equation (4.18):

$$\frac{\partial y_j}{\partial x_j} = y_j(1-y_j) \tag{4.21}$$

Finally, the first partial derivative of the remaining part $\frac{\partial L}{\partial y_j}$, which is the derivative of Equation (4.16):

$$\frac{\partial L}{\partial y_j} = -(t_j - y_j) \tag{4.22}$$

Putting the whole thing together, algorithm for the output-layer case become: Overall, putting all things together, we form the algorithm for the output-layer case:

$$\frac{\partial L}{\partial w_{kj}} = -(t_j - y_j)y_j(1-y_j)y_k \tag{4.23}$$

Chain rule for the propagation of the error, :

$$\frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial y_j}\frac{\partial y_j}{\partial x_j}\frac{\partial x_j}{\partial w_{kj}} \tag{4.24}$$

First two partial derivatives in equation (4.24), can be donated as δ_i and the rest is the derivate of the weighted sum of the inputs $w_j i$:

$$\frac{\partial L}{\partial y_j} = -\sum \delta_i w_{ji} \tag{4.25}$$

As a result, it can be written as equation (4.26) for the hidden layers:

$$\frac{\partial L}{\partial w_{kj}} = -\sum(\delta_i w_{ji})y_i(1-y_j)y_k \tag{4.26}$$

In the end, gradient of the calculated error feed to the SGD algorithm.

Chapter 5

FFN Methodology

In this chapter, the details of the implementation steps for the first part of the thesis, creating a Feed-forward network for binary classification of TCGA data is described. This chapter introduces the result of our first attempts to solve the classification problem. Throughout this chapter, the following steps will be covered respectively: data preprocessing, feature extraction and FNN architecture.

5.1 Preprocessing

As briefly mentioned in the Section 4.1.5 preprocessing is an important step to properly analyze the input data. The problems of input data are explained in 3.1. To summarise, input data have three problems to solve in preprocessing phase. It has a bias on expression values, missing values in samples and input data is unbalanced. Normalization function of PyGMQL library is used to normalize the input data to overcome the bias problem of the raw input. Missing values filled with its mean value of the respective column. Sequence of the input data is change to guarantee each batch has the same proportion of the input as whole data.

5.2 Feature Extraction

The most important thing in feature extraction is to select most relevant features, whose has enough information about input to differentiate the different classes easily. Optimal solution is the selecting most important features by hand. However for this case, it is not possible to managed that. So as a feature extracting, PCA with 200, 300, 500, 700 principal components, ANOVA with 200, 300, 500, 700 selected features and random forest with importance level higher than .001, .0005, .0001 selected for experiment to find the optimum

model and the optimum feature selection method. The threshold of PCA algorithm is 606 principal components for kidney data. With 606 principal components, the 99.999% of the variance in the input data is covered. It is not possible to include more principal components. Therefore 606 the highest amount to feature selected for kidney database to make comparison simple.

5.3 Train, Test, Validation split

Deep learning needs two inputs. Once for training the data, one for testing the trained model. It is important to use different samples for training and test to actually trigger *learning* process. Sealing the test information from training cause the model to learn from training data without knowing the answer.
Since input data will be used for classification(in our case), the proportion of the labels in the splits must be preserved. Stratified split algorithms are perfect for this job. Python library sklearn have a method named *stratified shuffle split*[sklearn] that can handle this part. It takes random partitions the data into k independent subsets of the same size.Then it trains the model using k-1 subsets and tests it on the last subset. This procedure is repeated k times assuring that each subset is used once as the test data. The k results are later averaged to produce the final result while preserving the label proportions. Input Data split into 2 parts, 80% *training dataset*, 20% *test dataset* as the most suggested train-test ratio.
However, every system tends to memories the outcomes. After continues training with the same train- test data allow the system to have an idea about the content of test data indirectly. To prevent that another dataset completely separated from train and test is needed. After training done, instead of checking the accuracy of test data (which can be over-fitting because of indirect learning) the accuracy of separate validation dataset gives the correct result. Thus, training data needed to split again. Splitting training dataset into two with 75% , 25% proportion for *validation dataset* give us the final parts of parts 60% for *training dataset*, 20% *validation dataset*and 20% *test dataset*.

Data predation steps explained above (Section 5.1, 5.2, 5.3) are essential for any classification tasks. The resulting output is used for the input feed for both FFN, explained in this chapter, and ladder network explained in Chapter 6.

5.4 FFN Structure

FFN follows the steps mentioned in the Section 4.2. Input TCGA data is fetch from the GMQL server with the GMQL web interface and loaded into the local repository, explained

in Section 3.2. For FFN, expression values of the selected genes are used for the input training feed. Corresponding input training labels, gathered from the TCGA metadata as healthy or cancerous.

Various combinations of different complexity level is tried to prevent over-fitting which is mentioned in Subsection 4.1.5. Best results are gathered by using FFN consist of three layers, one for input, one for hidden and one for output. Nodes in the input layer changes evenly depending on the feature selection method and how many features are extracted. Hidden layer consist of 20 nodes and output layer has nodes same number as output class, which is two in this case. *ReLU* used for activation function. The used batch size is 60, trained with 60 - to 80 epochs. For training *stochastic gradient descent* optimization is used and *cross-entropy* used for the calculation of the loss function. Figure 5.1 illustrates the final structure of used FFN model. $I_1 ... I_n$ indicated the input nodes where the n is the number of the extracted features. $H_1 ... H_m$ indicates the hidden nodes where m is 20 for this case and $O_1 ... O_2$ indicates the output nodes: one for healthy, one for cancerous.

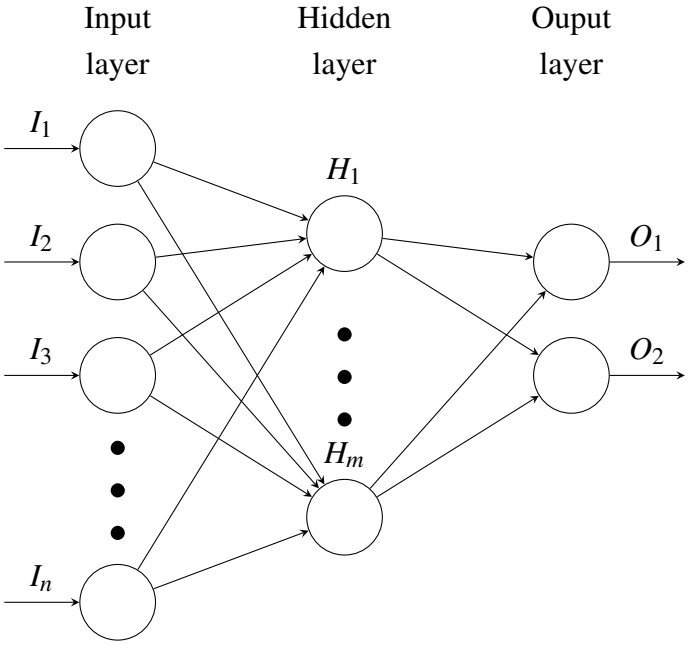


Fig. 5.1 Final structure

5.5 Results & Discussion

Highest achieved validation accuracy can be seen in Figure 5.2 and Figure 5.3. Best results, the lowest error rate, achieved with PCA (300 principal components) as feature extraction method, and 2 layer neural network structure (5.1) as model structure for TCGA kidney

data. PCA (700 principal components) for feature extraction and 4 layered neural network as structure achieves best result, the lowest error rate, for TCGA breast data.

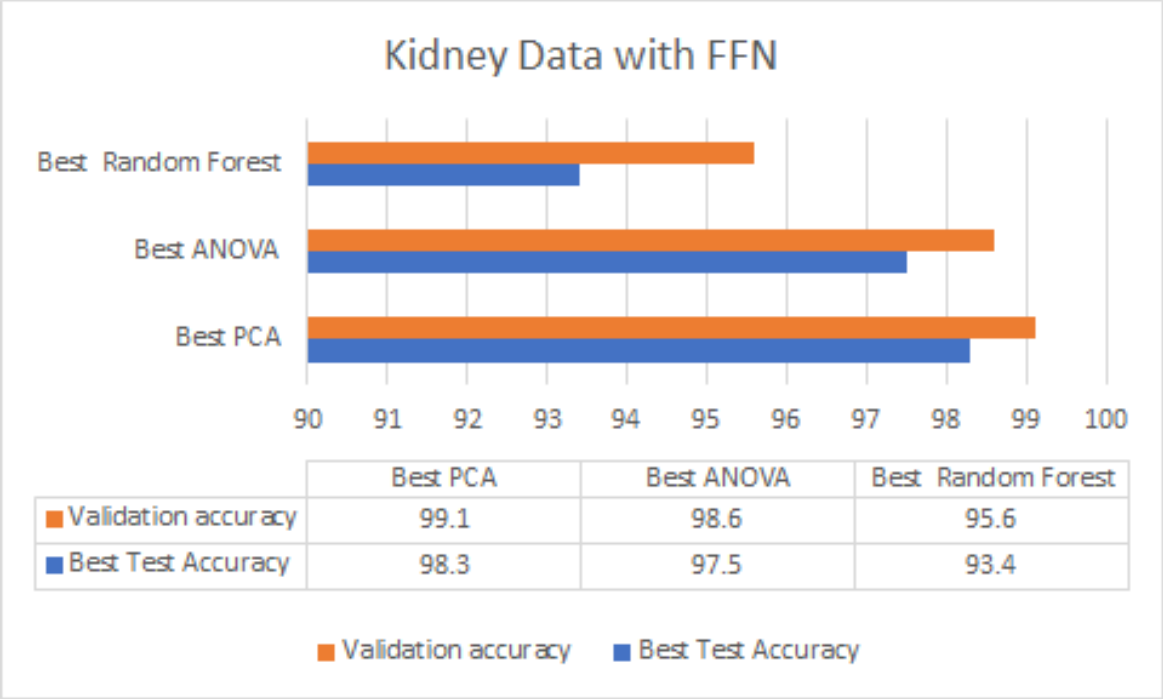


Fig. 5.2 Best accuracy with kidney data

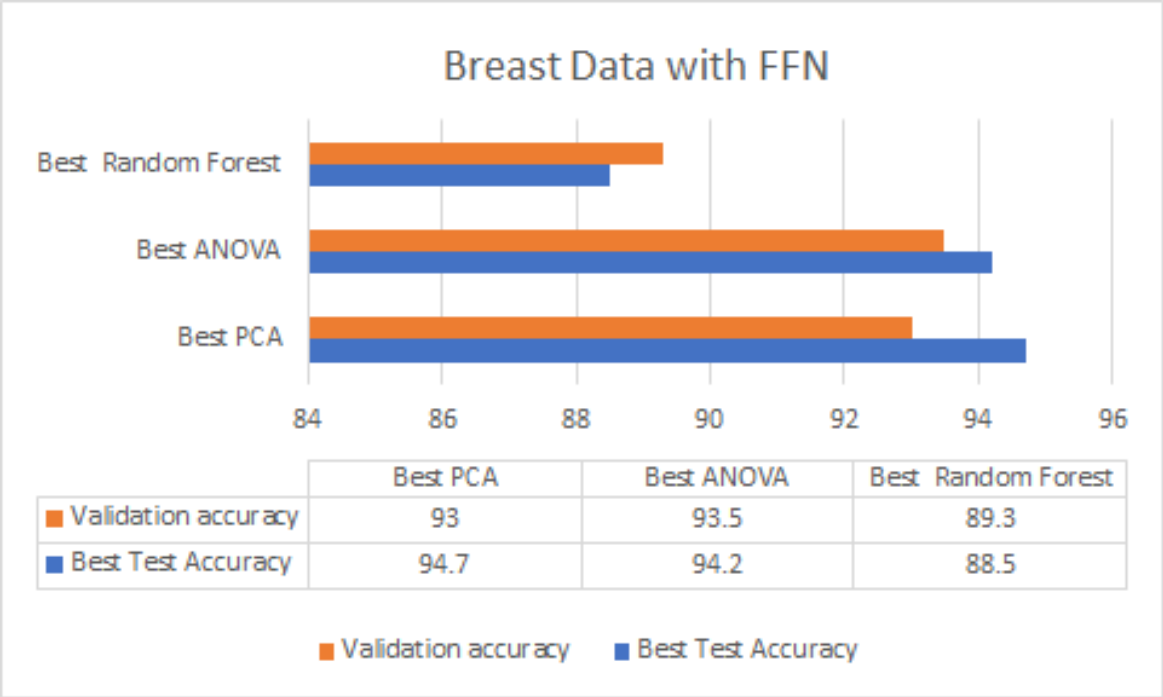


Fig. 5.3 Best accuracy with breast data

The feature extraction methods is absolutely necessary for this level of analysis. Deep learning do not have enough samples to find possible connections between all features. With out any feature selection method, input with all 20206 features, fails to improve the null hypothesis. The amount of features also affecting the learning accuracy, with too much feature involved, the model fails to predict new samples accurately. Figure 5.4 shows the general behavior through the effect of features over models prediction. The number of the features for the classification task is highly dependent on the input data. In other words, around 700 features are necessary for the model to have a good approximation on prediction new samples, in TCGA breast data, while only 300 features are enough with TCGA kidney data.

Taking into consideration that, 300 principal components of PCA can protect 94% of the variance in the kidney dataset, on the other hand it can only protect 82% of breast dataset. Therefore, in order to reach 94% of the variance, 400 additional principal components are required. As it is expected, the number of the features that preserve the accuracy are different for each dataset. The detailed result for each feature selection and method are available in the Appendix A.

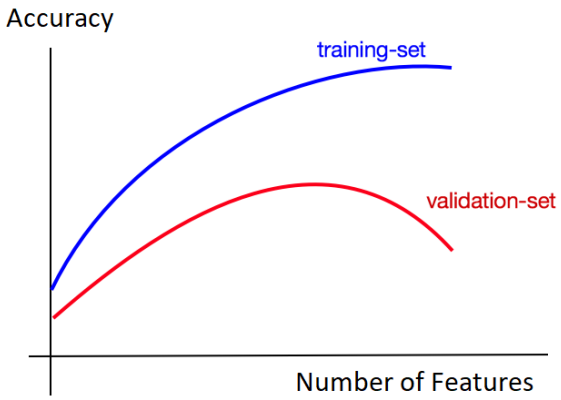


Fig. 5.4 Effects of number of features

The discrepancy in datasets also affects the constructional complexity. 1-layered feed-forward network with small amount of hidden nodes is enough to train kidney data but it is too simple to train breast data. While overfitting/underfitting occurs with more complex structures when training kidney data, it is necessary to reach a decent model with breast data.

To summaries, both input datasets are trained to do binary classification task. Because of the discrepancy in dataset, different structures are necessary to obtain satisfactory results. PCA gives the best performance as a feature extraction method, followed closely by ANOVA. Random forest, on the other hand, has the worst performance, training barely improves the null hypothesis

Final accuracy can be considerable as exceptionally high, especially with kidney data (99%). However, model gives highly unstable results. As it can bee seen on the tables in Appendix A, system generally fails to improve the null hypothesis. Mostly, the model underfits the healthy samples from input, which most probably occurs because of the imbalance in input data. However, stability of the model does not improve with any of the methods mentioned in 3.1.5. All in all, the created feed-forward model is not trustworthy for binary classification of TCGA data.

Chapter 6

Ladder Network with Semi-Supervised learning

The created feed-forward network in Chapter 5 is failed to give a satisfying result. Even though the final accuracy is good, the stability of the system is not acceptable. The final opinion we can infer with is that the resulting model actually is not a trustful model. What is required to achieve a more stable structure that can reach an acceptable accuracy level result with a low amount of input data. Ladder network is satisfying these conditions. It achieves dropping the error rate to 1.06% with only 100 labeled examples with publicly known MNIST (Modified National Institute of Standards and Technology database) [15] dataset.

This chapter will consist of 3 sections: Section 6.1 explains the process of semi-supervised learning. Section 6.2 explains what is ladder network. Section 6.3 shows the ladder structure and illustrates the final accuracy results.

6.1 How Semi-Supervised Learning works

Semi-supervised learning uses supervised learning tasks and techniques to make use of unlabeled data for training. Generally, the amount of labeled data is very small, while the unlabeled data is much larger.

Figure 6.1 illustrates a visualization of semi-supervised input data. Grey point are the unlabled datas, where the black one and white one indicates different labels of unlabeled data.

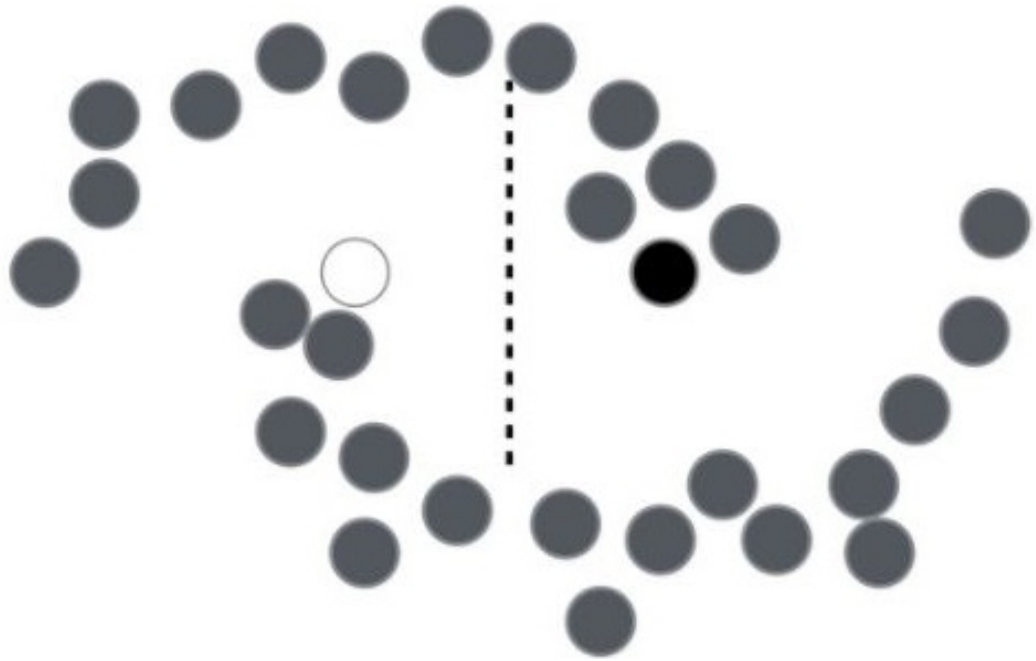


Fig. 6.1 Example of semi-supervised input data

It is really hard to make sense when having very small number of labeled and unlabeled data, but when all unlabeled ones are seen 6.1, it can be worked out to process.

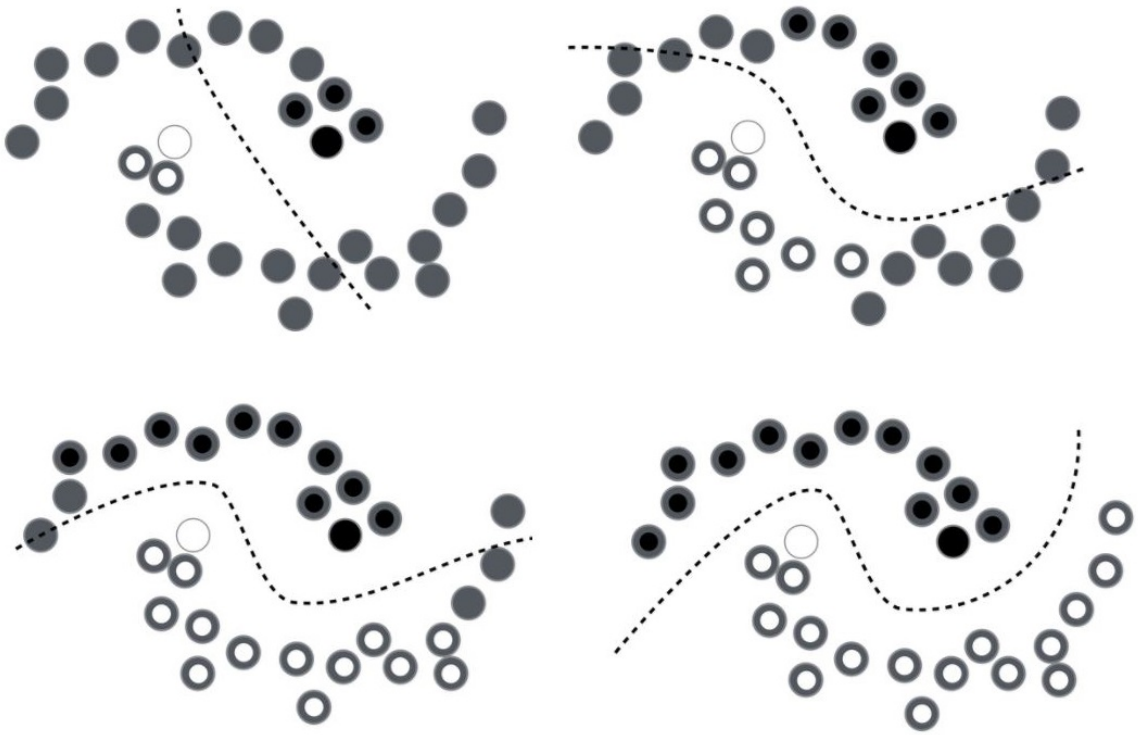


Fig. 6.2 Steps of classification based on smoothness assumption

Assuming, there is a structure underlying the distribution of data and labels are homogeneous in densely populated space i.e. With *smoothness assumption*[1], classification of the unlabeled data became much more easier to deal. Figure 6.2 shows steps of classification of unlabeled data based on smoothness assumption.

It has been found that the utilization of unlabeled data together with a small amount of labeled data can enhance accuracy extensively. In the most of the real world situation, collecting labeled data is very expensive while there is a lot of unlabeled data is on hand. Semi-supervised learning is best to be used in these kind of situations. Also, such learning system is very similar to how our brain works. For instance humans don't need to see all versions of chairs to understand if the new object is a chair or not. Seeing a small number of chair sample (*labeled data*) with all past experiences (*unlabeled data*) help us to categorize the objects. Hence, semi-supervised learning is a rational model for the human brain.

[1] According to smoothness assumption data points, which are close to each other tend to have the same label

6.2 Ladder Network

Ladder network unites both supervised and unsupervised learning in deep neural network. Generally, unsupervised learning is used for train the model in advance, a.k.a. as pre-training, before supervised training. Ladder network trains it simultaneously minimize the cost of supervised and unsupervised by back-propagation function, without using layer-wise pre-training.

6.2.1 Aspects of Ladder Network

- **Compatibility:** Ladder Network consists of feedforward networks (FFN), any supervised learning model can easily be added to existing structure. network.
- **Scalability:** Ladder network has unsupervised learning target in each layer, which make it suitable for deep neural networks.
- **Cost Efficiency:** Since the Ladder network is basically modified feedforward networks, structure is simple and easy to implement. Training function of ladder network is based on from a simple cost function. So it is quick to train and the convergence is fast with the help of *batch normalization* [28] [2].

6.2.2 Implementation of Ladder Network

This section has a simple introduction of implementation of the ladder network introduced in Rasmus et al. [42]:

1. Ladder network has a feed-forward model that used as a supervised learning encoder. The complete system has 2 encoder paths, one is *clean* the other is *corrupted*. The difference between the clean and corrupted one is, the corrupted encoder adds Gaussian noise at all layers.
2. Decoder added to invert the mapping on each layer of the encoder to support unsupervised learning. The decoder uses a denoising function to reconstruct the activation of each layer in corrupted encoder based on the activation of the clean encoder. Difference between reconstruction and the clean version is count as the denoising cost of that layer.

[2] Batch Normalization is a technique to provide any layer in a Neural Network with inputs that are zero mean/unit variance to improve the performance and stability

3. Difference between the output of corrupted encoder and output target is count as a *supervised cost* while summing of denoising cost of all layers scaled by *significance parameter* count as an *unsupervised cost*. Final cost is the sum of supervised and unsupervised cost.

4. It is possible to train both fully labeled and semi labeled using optimization techniques to minimize the cost.

Figure 6.3 illustrates the structure of 2 layered ladder network example in Rasmus et al. [42].

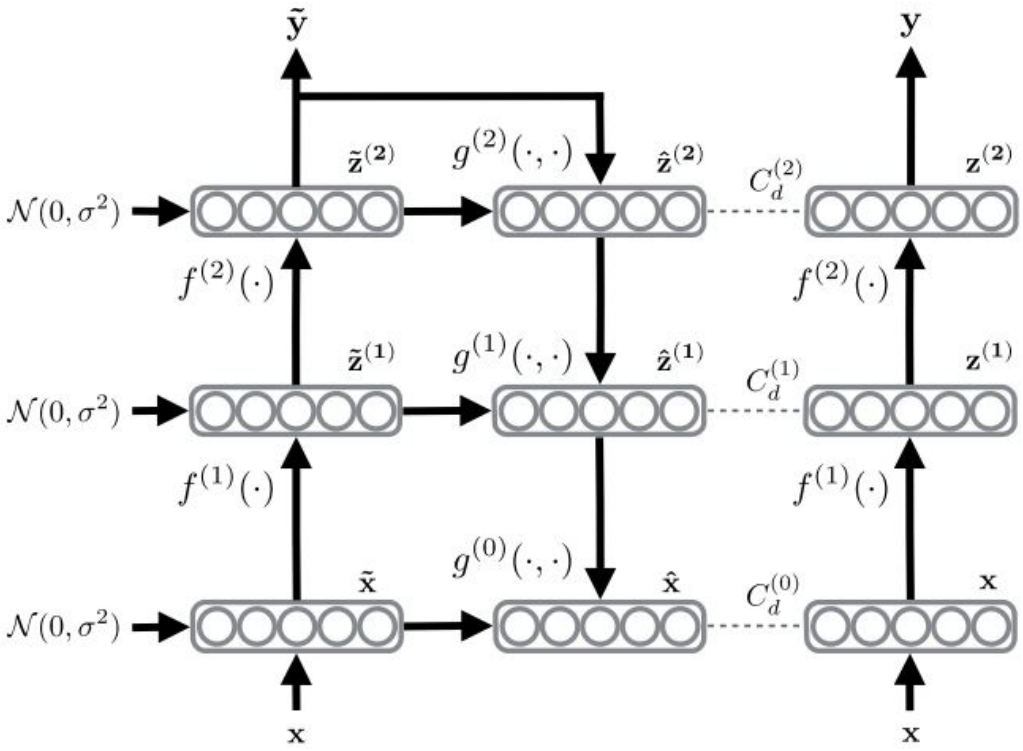


Fig. 6.3 Structure of 2 layered ladder network

Furthermore, batch normalization is applied to each pre-activation including the top most layer to improve convergence and to prevent the ladder network to end up with a futile solution. Encoder tends to output constant values that are easier to denoise which cause finding futile solutions. There is a direct connection between a layer and its decoded reconstruction.

The whole system is called Ladder network because the encoder-decoder architecture resembles a ladder.

Figure 6.4 illustrates the algorithm to calculate output and loss in ladder network.

Algorithm 1 Calculation of the output $\mathbf{y}$ and cost function C of the Ladder network

Require: $\mathbf{x}(n)$
  # Corrupted encoder and classifier
  $\tilde{\mathbf{h}}^{(0)} \leftarrow \tilde{\mathbf{z}}^{(0)} \leftarrow \mathbf{x}(n) + \texttt{noise}$
  for l = 1 **to** L **do**
    $\tilde{\mathbf{z}}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{W}^{(l)}\tilde{\mathbf{h}}^{(l-1)}) + \texttt{noise}$
    $\tilde{\mathbf{h}}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\tilde{\mathbf{z}}^{(l)} + \boldsymbol{\beta}^{(l)}))$
  end for
  $P(\tilde{\mathbf{y}} \mid \mathbf{x}) \leftarrow \tilde{\mathbf{h}}^{(L)}$
  # Clean encoder (for denoising targets)
  $\mathbf{h}^{(0)} \leftarrow \mathbf{z}^{(0)} \leftarrow \mathbf{x}(n)$
  for l = 1 **to** L **do**
    $\mathbf{z}_{\text{pre}}^{(l)} \leftarrow \mathbf{W}^{(l)}\mathbf{h}^{(l-1)}$
    $\boldsymbol{\mu}^{(l)} \leftarrow \texttt{batchmean}(\mathbf{z}_{\text{pre}}^{(l)})$
    $\boldsymbol{\sigma}^{(l)} \leftarrow \texttt{batchstd}(\mathbf{z}_{\text{pre}}^{(l)})$
    $\mathbf{z}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{z}_{\text{pre}}^{(l)})$
    $\mathbf{h}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\mathbf{z}^{(l)} + \boldsymbol{\beta}^{(l)}))$
  end for
  # Final classification:
  $P(\mathbf{y} \mid \mathbf{x}) \leftarrow \mathbf{h}^{(L)}$
  # Decoder and denoising
  for l = L **to** 0 **do**
    if l = L **then**
      $\mathbf{u}^{(L)} \leftarrow \texttt{batchnorm}(\tilde{\mathbf{h}}^{(L)})$
    else
      $\mathbf{u}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{V}^{(l+1)}\hat{\mathbf{z}}^{(l+1)})$
    end if
    $\forall i : \hat{z}_i^{(l)} \leftarrow g(\tilde{z}_i^{(l)}, u_i^{(l)})$ # Eq. (2)
    $\forall i : \hat{z}_{i,\text{BN}}^{(l)} \leftarrow \frac{\hat{z}_i^{(l)} - \mu_i^{(l)}}{\sigma_i^{(l)}}$
  end for
  # Cost function C for training:
  $\text{C} \leftarrow 0$
  if $t(n)$ **then**
    $\text{C} \leftarrow -\log P(\tilde{\mathbf{y}} = t(n) \mid \mathbf{x}(n))$
  end if
  $\text{C} \leftarrow \text{C} + \sum_{l=0}^{L} \lambda_l \left\| \mathbf{z}^{(l)} - \hat{\mathbf{z}}_{\text{BN}}^{(l)} \right\|^2$ # Eq. (3)

Fig. 6.4 Ladder Network algorithm presented in paper

6.3 Ladder Network with TCGA Data

Before using the ladder network model, FFN that is created in the Section 5.4 also gives a good result on TCGA kidney and breast data, but the imbalance of the input data makes it hard to reproduce the similar result after every training. In the repetitive trails, FFN model generally only gives a prediction for the class with the superior number while ignoring the other.

6.3.1 Structure of Ladder Network

As briefly mentioned in the Section 4.1.5 preprocessing is an important step to properly analyze the input data. To deal with the problem of the raw input data, and preparation data for the classification task, same steps as (Section 5.1, 5.2, 5.3) are followed. Same inputs and outputs as FFN is used for the ladder network as well.

Same 6 layer structure is adopted but the nodes in layers are modified. First and the last layer is modified with respect to the inputs and outputs of the system. Meaning, the first layer has the same number of nodes as the input features and the last layer has the same number of

nodes as the number of output class which is 2 in this case. The number of input features is changed depending on the selected feature selection method and the selected number of features. The layers between ,*hidden layers*, have exponentially increased nodes from output layer through input layer. Significance number which is mentioned in Section 6.2.2 in step 4, is selected [1000, 10 , 0.1,0.1,0.1,0.1,0.1] respectively to indicate the importance of the layer. Figure 6.5 illustrates the model that is used for classification of TCGA data.

```
=== Corrupted Encoder ===
Layer  1 :  200  ->  64
Layer  2 :  64  ->  32
Layer  3 :  32  ->  16
Layer  4 :  16  ->  8
Layer  5 :  8  ->  4
Layer  6 :  4  ->  2
=== Clean Encoder ===
Layer  1 :  200  ->  64
Layer  2 :  64  ->  32
Layer  3 :  32  ->  16
Layer  4 :  16  ->  8
Layer  5 :  8  ->  4
Layer  6 :  4  ->  2
=== Decoder ===
Layer  6 :  None  ->  2 , denoising cost:  0.1
Layer  5 :  2  ->  4 , denoising cost:  0.1
Layer  4 :  4  ->  8 , denoising cost:  0.1
Layer  3 :  8  ->  16 , denoising cost:  0.1
Layer  2 :  16  ->  32 , denoising cost:  0.1
Layer  1 :  32  ->  64 , denoising cost:  10.0
Layer  0 :  64  ->  200 , denoising cost:  1000.0
```

Fig. 6.5 Structure of model that is used for classification of TCGA data

The number of nodes in hidden layers are much smaller than the ones used in reference paper because of the same behavior with the FFN model through experiments. The more hidden nodes in the model the more complex the structure became, so the final accuracy falls because of over-fitting. So simpler is the better in this case.
The various number of labeled data is used in the experiments. As it can be guessed, if the

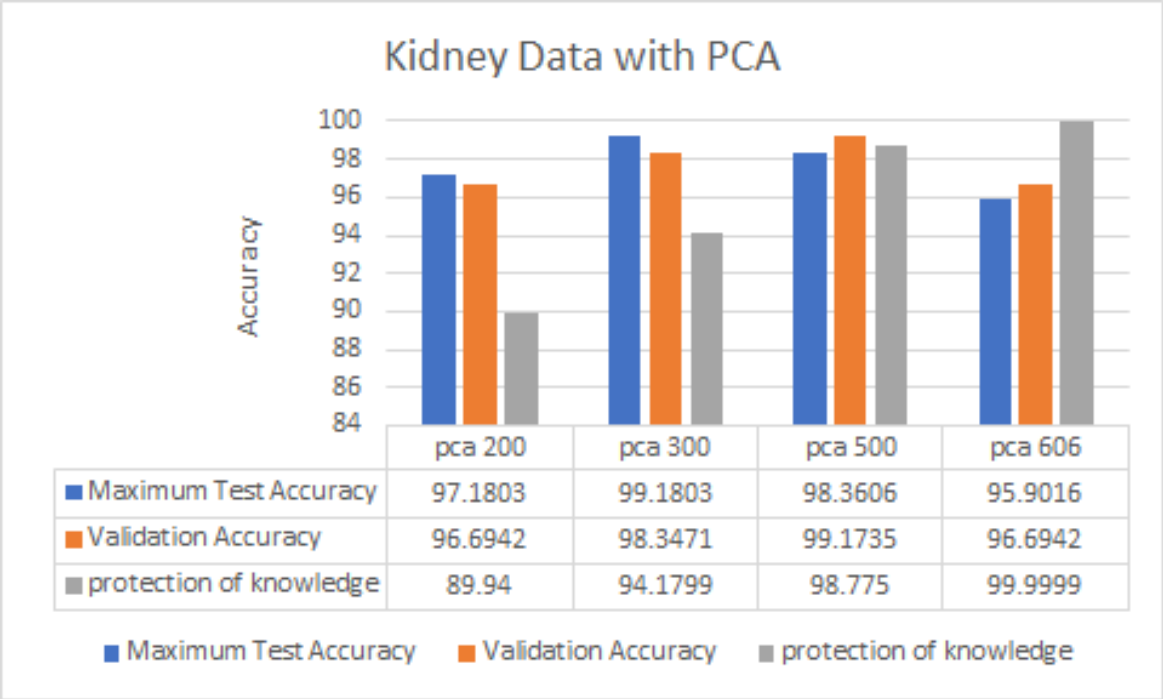
amount of the labeled sample is too small, the model can not improve the null hypothesis, such as 2, 10, 20. But after 50 labeled data, which is relatively higher amount which can be still be considered as a small amount of sample, the model work with very high accuracy. The ratio of the labeled data that feeds the supervised learning is 50%, 50% for both cancerous and healty samples.

The amount of the labeled data did not have a significant change in accuracy after 50. The final accuracy of 50 labeled data and 200 labeled data is very similar. This means that unsupervised learning works without bothering the supervised learning.

Batch size is chosen to be 60 as well as the number of labeled data in input feed to use in trials.

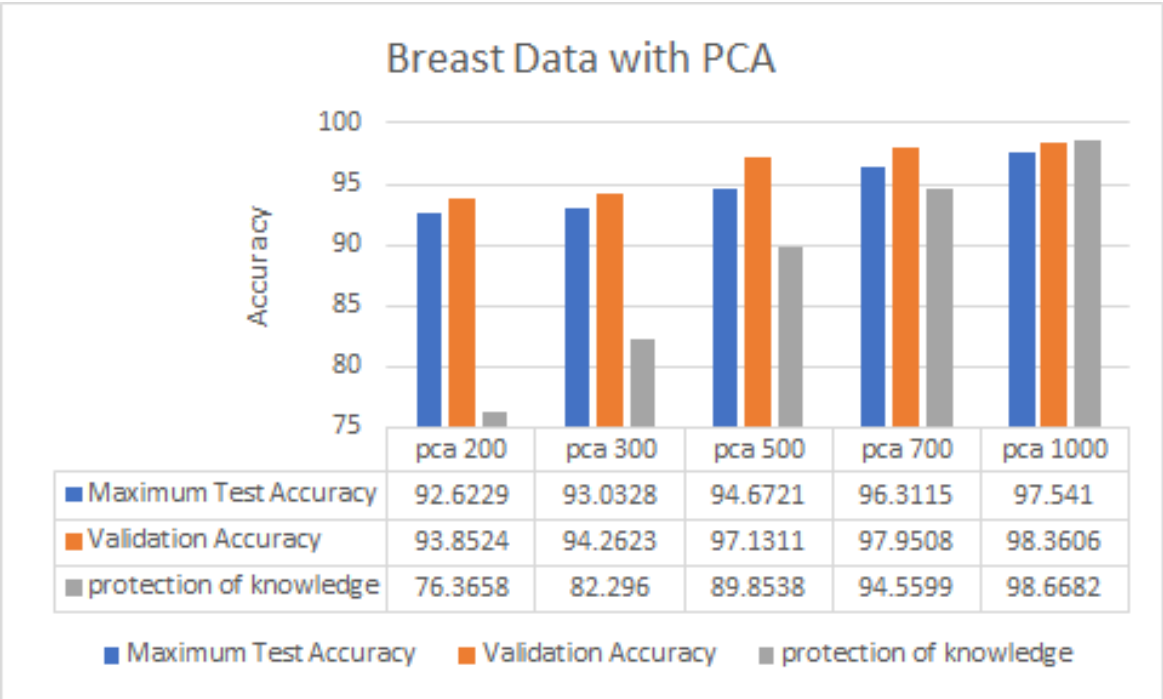
6.3.2 Results

Even though 60 is the selected number of labeled data for kidney and breast dataset from TCGA, smaller and higher amounts of labels also used for the experiment. Same feature selection methods with FFN is used to make the comparison easier, PCA with 200, 300, 500, 700 principal components, ANOVA with 200, 300, 500, 700 selected features and random forest with importance level higher than .001, .0005, .0001 selected for experiment to analyze the effects of the feature selection methods on the models accuracy. The graphics below show the result of the experiment on both data. More detailed information can gathered from Appendix B



	pca 200	pca 300	pca 500	pca 606
Maximum Test Accuracy	97.1803	99.1803	98.3606	95.9016
Validation Accuracy	96.6942	98.3471	99.1735	96.6942
protection of knowledge	89.94	94.1799	98.775	99.9999

Fig. 6.6 Accuracy in kidney data when PCA is used for feature selection method



	pca 200	pca 300	pca 500	pca 700	pca 1000
Maximum Test Accuracy	92.6229	93.0328	94.6721	96.3115	97.541
Validation Accuracy	93.8524	94.2623	97.1311	97.9508	98.3606
protection of knowledge	76.3658	82.296	89.8538	94.5599	98.6682

Fig. 6.7 Accuracy in breast data when PCA is used for feature selection method

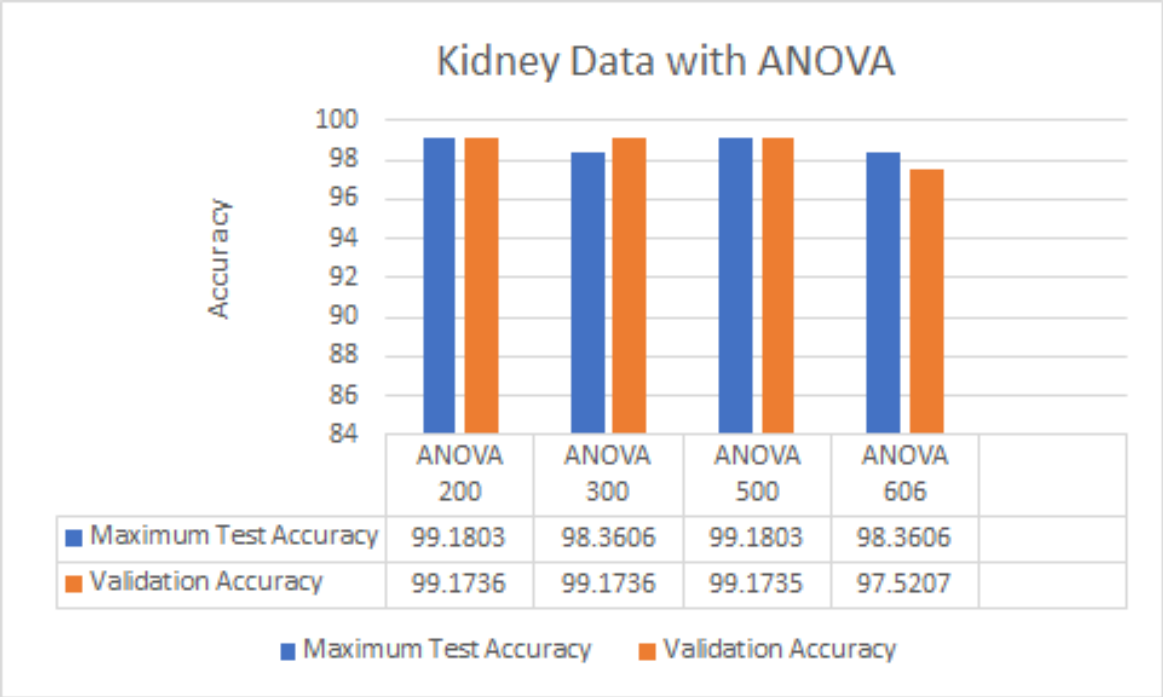


Fig. 6.8 Accuracy in kidney data when ANOVA is used for feature selection method

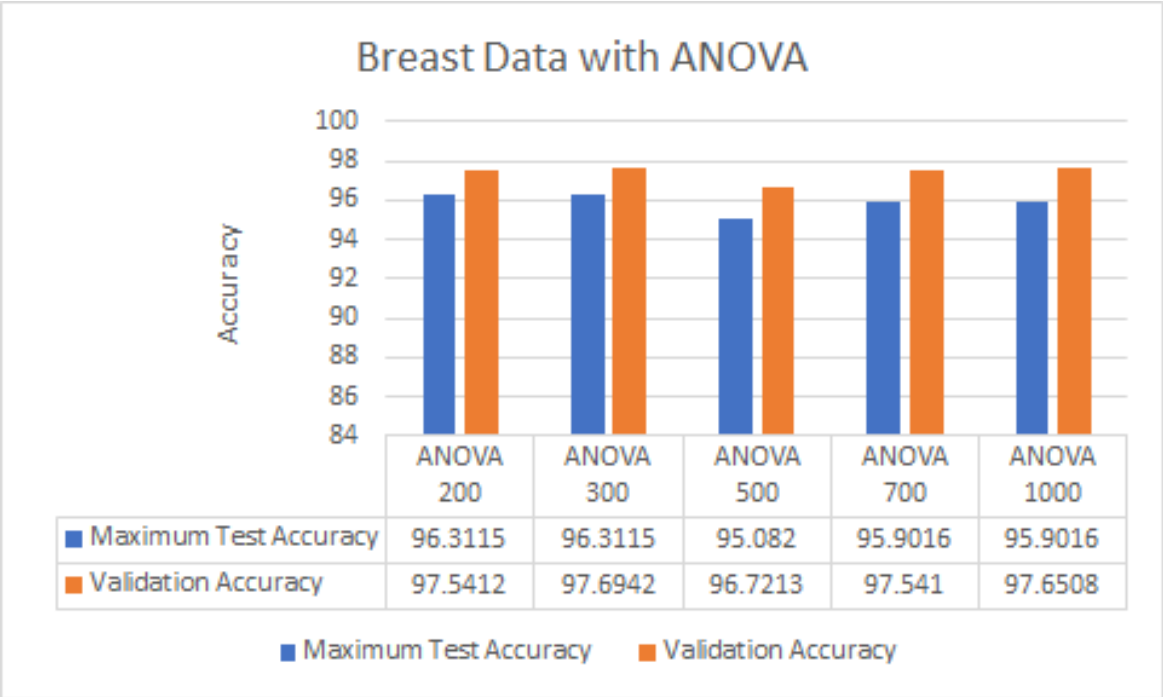


Fig. 6.9 Accuracy in breast data when ANOVA is used for feature selection method

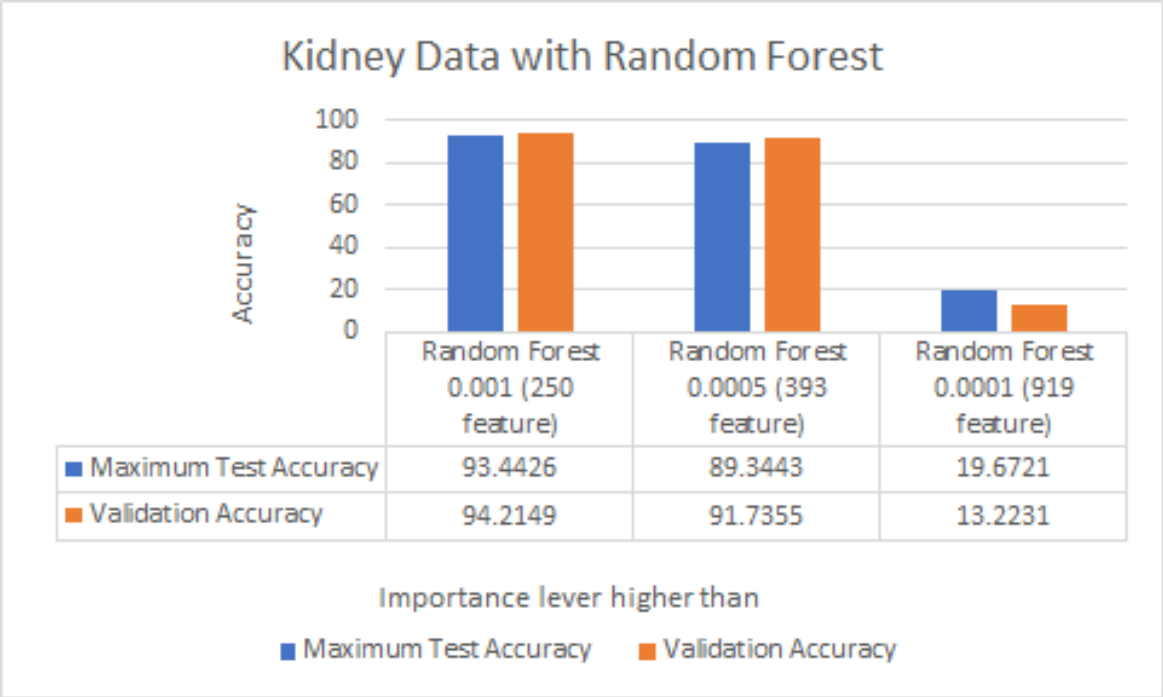


Fig. 6.10 Accuracy in kidney data when Random Forest is used for feature selection method

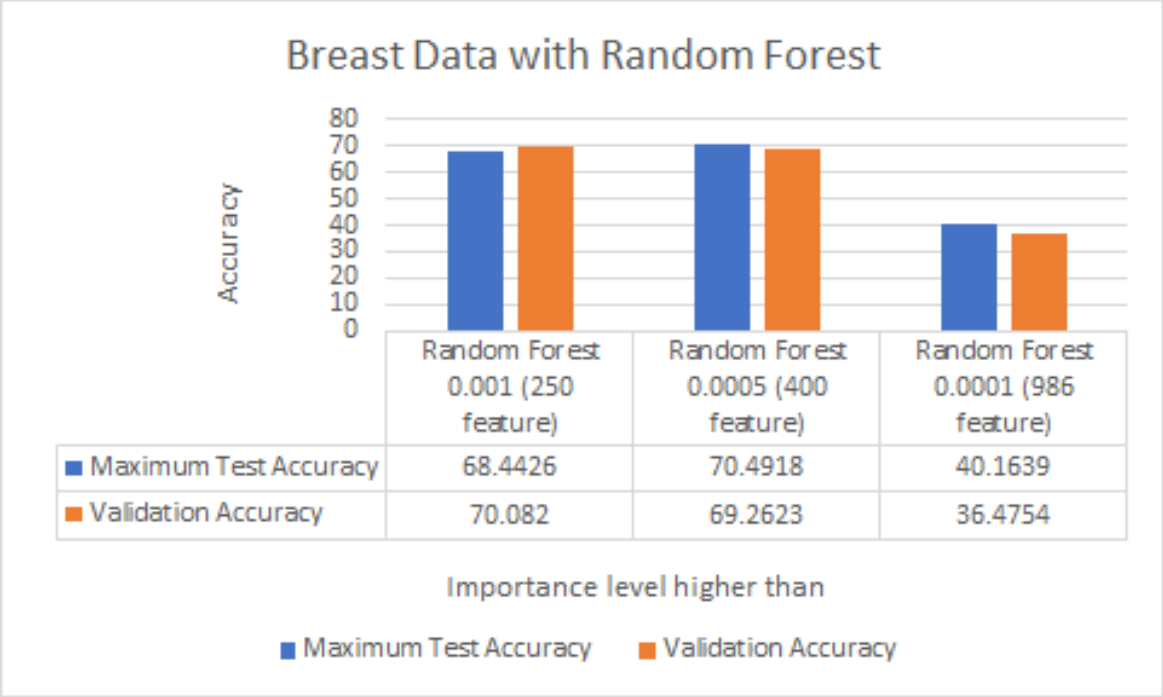


Fig. 6.11 Accuracy in breast data when Random Forest is used for feature selection method

The effects of the feature extraction are very similar with the FFN. Ladder network also fails to improve the null hypothesis when no feature extraction algorithm applied. However, ladder network works stable while having better accuracy result on most cases. It gives similar accuracy regardless of number of times it runs . To analyze the results, especially with the breast data, the difference between the PCA and ANOVA become clear. The final accuracy reaches the highest value even with only 200 features selected with ANOVA, while system needs 700 to 1000 principal components selected by PCA to reach the similar accuracy level. Features selected with random forest, on the other hand, fails even to reach the null hypothesis.

Chapter 7

Comparison

In this thesis, feedforward network model (Chapter 5) and ladder network model (Chapter 6) are created to do binary classification for TCGA breast and kidney data. In order to reach the highest accuracy possible, few feature selection methods with different amount of features applied. This chapter gives a comparison between created models, input data and selected feature selection methods.

TCGA kidney data has 606 samples and 20206 genes (features) in its raw form. We split the data with 60%,20%,20% for train, test and validation, respectively. As a final result, in training test, there is 363 samples in training dataset, in test and validation set, there is 122 sample.

TCGA breast data has 1218 sample with 20207 genes (features) in its raw form. Same ratio as kidney is used for train, test and validation split. Training set of breast data has 730 sample, test and validation set has 244 sample.

The input data feed through the both used structures in small batches, which follows the same ratio to overcome the problem with the imbalanced data. Thus, the null hypothesis, minimum expected accuracy is 79.8% for kidney data, and 84.5% for the breast data.

Both structures improves the null hypothesis with selected feature selection methods.

Feed-forward network as a choice is a good model to do experiments for the ones, who do not have much experience with deep learning. In structure, each deep learning step can be easily seen and modified for the preferred adaptations. It also reached acceptable accuracy rate with kidney data. However, the methods tried to stabilize the result whose, instability occurs most probably because of imbalanced input data, are failed and most of the time it ends up with highly biased prediction in favor of cancerous data.

Ladder network, on the other hand, gives a stable result with higher accuracy in all cases. The ladder network structure is complex when comparing with simple FFN, yet since as a base it uses FFN for encoder paths and training is based on back-propagation from a simple

cost function, the structure is simple and easier to implement than most of other deep learning structures. Imbalance in the input samples also does not affect the stability and final result in this model since the input feed of the supervised part is balanced.

Effects of the feature extracting algorithm changes through the selected structure and selected input data. For kidney data in FFN structure, best accuracy reached by the principal components selected by PCA. Result followed closely by the features selected by ANOVA and Random Forest gives the worst result, which only reaches 95% accuracy. FFN with breast data, on the other hand, best results are collected with features selected with Random Forests, followed closely by PCA and ANOVA. The performance of PCA and ANOVA is similar while the performance of random forest differs. It is possible to say that, random forest works better with complex datasets since breast data is more complex than kidney dataset. To protect the 94% of the variance in the breast dataset, 400 principal component is needed.

Moreover, the performance of the feature selection algorithms is more stable with the Ladder Network. Both in the breast and kidney dataset, best performance reached by the principal components by PCA, followed closely by ANOVA, while Random Forest falls too behind. However, if we focus on breast data, effects of the ANOVA is the outperform the PCA with a small number of features. FInal accuracy in breast dataset with ANOVA is around 97.5%, 98.1% without depending on the number of selected features. Yet, PCA only reaches the maximum accuracy with 1000 features involves. Performance after 1000 features starts to decrease for all algorithms. On the other hand, features selected by Random forest performs poorly, even fails to reach null hypothesis most of the time.

As a result, ladder network outperforms the FFN network, and best accuracy reached with PCA. Ladder network works accurately without being troubled by the 2 most important problem of input dataset; imbalance in the dataset, low amount of sample. Accuracy result may improve with different feature extracting methods, or selecting by hand with the one who have field information on the topic.Ladder network work fine with tow selected dataset, similar results are expecting with other cancer datasets in TCGA dataset. Further research of multiple label classification,instead of binary classification could be considered as future work.

Chapter 8

Conclusion

There is a lot of research on analyzing biological data and solutions related to these problems. Different machine learning methods have become more and more popular in this area. Deep learning stands out among rest, due to its remarkable performance. One weakness of deep learning that should be considered is its need of data. It really shines where there is high amount of data available to train. When there is not, it may lack of reflecting desired accuracy levels. Having said that, there are still new structures being developed to overcome this weakness.

In this thesis, two approaches has been introduced to solve binary classification problem of the TCGA mRNA sequencing data, and cancer data (breast and kidney), both of which trained with FFN and Ladder network structures. Besides of these data processing structures, three different feature extracting algorithms for pre-processing have been utilized: PCA, ANOVA, and random forest to observe the effects. After several experiments, it can be cleary seen that FFN reaches 99.2% accuracy rate with kidney dataset. However, despite of that accuracy level, it fails to achieve a stable result. On the other hand, Ladder network outperforms FFN not only in accuracy but also in stability. Therefore, one may selecting Ladder network over FFN should not be a suprizing result. As a side note, Appendix A can be given as a reference to see the instability between results of FFN.

To sum up, selecting appropriate data preprocessing and feature extraction methods, and defining a subsidiary model architecture are keys in binary classification and prediction tasks. To explain why these are the keys, it can be given as an example that the same ladder network acquires 2 different scores with 2 different selection methods. The best accuracy acquired is with Ladder network using PCA extraction algorithm in both datasets. Also, a very close result can be achieved in accuracy with ANOVA algorithm using much less selected features. This proves us utilizing different feature extraction algorithms may be preffered based on the purpose of experiment.

As a future work, this paper can be extended with including remaining datasets in TCGA representing other types of cancer which results in having multi-class classification. By having multi-class classification, one might have flexible approach to represent and work with multiple cancer data.

References

[1] (2016). Pygmql repository. ttps://github.com/DEIB-GECO/PyGMQL/. Accessed: 2017-08-08.

[2] (2016). Tcga cancer types. https://tcga-data.nci.nih.gov/docs/publications/tcga. Accessed: 2017-08-08.

[3] Abdel-Zaher, A. M. and Eldeib, A. M. (2016). Breast cancer classification using deep belief networks. *Expert Systems with Applications*, 46:139–144.

[4] Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. *Molecular systems biology*, 12(7):878.

[5] Bharathi, A. and Natarajan, A. (2010). Cancer classification of bioinformatics datausing anova. *International journal of computer theory and engineering*, 2(3):369.

[6] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[7] Ceri, S., Kaitoua, A., Masseroli, M., Pinoli, P., and Venco, F. (2016). Data Management for Heterogeneous Genomic Datasets. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 5963(c):1–14.

[8] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[9] Chial, H. (2008). Dna sequencing technologies key to the human genome project. *Nature Education*, 1(1):219.

[10] Consortium, . G. P. et al. (2012). An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56.

[11] Consortium, E. P. et al. (2007). Identification and analysis of functional elements in 1% of the human genome by the encode pilot project. *nature*, 447(7146):799.

[12] Cybenko, G. (1992). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 5(4):455–455.

[13] Danaee, P., Ghaeini, R., and Hendrix, D. A. (2017). A deep learning approach for cancer detection and relevant gene identification. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017*, pages 219–229. World Scientific.

[14] De Souto, M. C., Jaskowiak, P. A., and Costa, I. G. (2015). Impact of missing data imputation methods on gene expression clustering and classification. *BMC bioinformatics*, 16(1):64.

[15] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

[16] Díaz-Uriarte, R. and De Andres, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3.

[17] Eisenstein, M. (2015). Big data: The power of petabytes. *Nature*, 527(7576):S2–S4.

[18] França, L. T., Carrilho, E., and Kist, T. B. (2002). A review of dna sequencing techniques. *Quarterly Reviews of Biophysics*, 35(2):169–200.

[19] Fu, X., Fu, N., Guo, S., Yan, Z., Xu, Y., Hu, H., Menzel, C., Chen, W., Li, Y., Zeng, R., and Khaitovich, P. (2009). Estimating accuracy of rna-seq and microarrays with proteomics. *BMC Genomics*, 10(1):161.

[20] Giardine, B., Riemer, C., Hardison, R. C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., et al. (2005). Galaxy: a platform for interactive large-scale genome analysis. *Genome research*, 15(10):1451–1455.

[21] Golub, T. R., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. 286(October):531–538.

[22] Guan, X., Chance, M. R., and Barnholtz-Sloan, J. S. (2012). Splitting random forest (srf) for determining compact sets of genes that distinguish between cancer subtypes. *Journal of clinical bioinformatics*, 2(1):13.

[23] Haas, B. J. and Zody, M. C. (2010). Advancing rna-seq analysis. *Nature biotechnology*, 28(5):421–423.

[24] Hargreaves, A. D., Zhou, L., Christensen, J., Marlétaz, F., Liu, S., Li, F., Jansen, P. G., Spiga, E., Hansen, M. T., Pedersen, S. V. H., et al. (2017). Genome sequence of a diabetes-prone rodent reveals a mutation hotspot around the parahox gene cluster. *Proceedings of the National Academy of Sciences*, 114(29):7677–7682.

[25] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[26] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.

[27] Institute, N. H. G. R. (2017). Dna sequencing costs: Data.

[28] Ioffe, S. and Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*.

[29] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[30] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[31] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

[32] Lovell, P. V., Wirthlin, M., Wilhelm, L., Minx, P., Lazar, N. H., Carbone, L., Warren, W. C., and Mello, C. V. (2014). Conserved syntenic clusters of protein coding genes are missing in birds. *Genome biology*, 15(12):565.

[33] Mao, Y., Zhou, X., Pi, D., Sun, Y., and Wong, S. T. (2005). Multiclass cancer classification by using fuzzy support vector machine and binary decision tree with gene selection. *BioMed Research International*, 2005(2):160–171.

[34] Masseroli, M., Pinoli, P., Venco, F., Kaitoua, A., Jalili, V., Palluzzi, F., Muller, H., and Ceri, S. (2015). GenoMetric Query Language: A novel approach to large-scale genomic data management. *Bioinformatics*, 31(12):1881–1888.

[35] Mindru, F., Tuytelaars, T., Van Gool, L., and Moons, T. (2004). Moment invariants for recognition under changing viewpoint and illumination. *Computer Vision and Image Understanding*, 94(1):3–27.

[36] Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877.

[37] Moorthie, S., Hall, A., and Wright, C. F. (2012). Informatics and clinical genome sequencing: opening the black box. *Genetics in Medicine*, 15(3):165–171.

[38] Mortazavi, A., Williams, B. A., McCue, K., Schaeffer, L., and Wold, B. (2008). Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature methods*, 5(7):621–628.

[39] Network, C. G. A. et al. (2012). Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, 487(7407):330.

[40] Ozsolak, F. and Milos, P. M. (2011). Rna sequencing: advances, challenges and opportunities. *Nature reviews. Genetics*, 12(2):87.

[41] Present, I. (2000). Cramming more components onto integrated circuits. *Readings in computer architecture*, 56.

[42] Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554.

[43] Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, CORNELL AERONAUTICAL LAB INC BUFFALO NY.

[44] Schuster, S. C. (2008). Next-generation sequencing transforms today's biology. *Nature methods*, 5(1):16–18.

[45] Shendure, J. (2008). The beginning of the end for microarrays? *Nature methods*, 5(7):585–587.

[46] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[47] Singh, R., Lanchantin, J., Robins, G., and Qi, Y. (2016). Deepchrome: deep-learning for predicting gene expression from histone modifications. *Bioinformatics*, 32(17):i639–i648.

[sklearn] sklearn. StratifiedShuffleSplit.

[49] Statnikov, A., Wang, L., and Aliferis, C. F. (2008). A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC bioinformatics*, 9(1):319.

[50] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[51] Tarazona, S., García-Alcalde, F., Dopazo, J., Ferrer, A., and Conesa, A. (2011). Differential expression in rna-seq: a matter of depth. *Genome research*, 21(12):2213–2223.

[52] Tuncel, M. A. (2017). A statistical framework for the analysis of genomic data.

[53] Wang, L., Chu, F., and Xie, W. (2007). Accurate cancer classification using expressions of very few genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 4(1):40–53.

[54] Wang, Z., Gerstein, M., and Snyder, M. (2009). Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57–63.

Appendix A

Error Rates of the experiments with FFN

A.1 TCGA kidney data with three layered (1 hidden layer) FFN

Table A.1 Error rate for TCGA kidney data with 200 features with 1 hidden layer

Algorithm	Min	Mean	Median	Max
PCA 200	1.8	17.8	20.2	20.2
ANOVA 200	2.4	18.2	20.2	20.2

Table A.2 Error rate for TCGA kidney data with 300 features with 1 hidden layer

Algorithm	Min	Mean	Median	Max
PCA 300	0.8	16.4	20.2	20.2
ANOVA 300	1.5	17.1	20.2	20.2

Table A.3 Error rate for TCGA kidney data with 500 features with 1 hidden layer

Algorithm	Min	Mean	Median	Max
PCA 500	2.0	17.8	20.2	20.2
ANOVA 500	2.8	18.2	20.2	20.2

Table A.4 Error rate for TCGA kidney data 700 features with 1 hidden layer

Algorithm	Min	Mean	Median	Max
PCA 606	3.8	18.8	20.2	20.2
ANOVA 606	4.2	19.6	20.2	20.2

Table A.5 Error rate for TCGA kidney data with random forests with 1 hidden layer

Algorithm	Min	Mean	Median	Max
RF (higher 0.0010 (250))	16.4	19.6	20.2	20.2
RF (higher 0.0005(393))	10.1	14.7	20.2	20.2
RF (higher 0.0001(919))	18.0	26.4	20.2	79.8

A.2 TCGA breast data with five layered (3 hidden layer) FFN

Table A.6 Error rate for TCGA breast data with 200 features with 3 hidden layer

Algorithm	Min	Mean	Median	Max
PCA 200	8.2	11.9	15.5	15.5
ANOVA 200	9.1	12.2	15.5	15.5

Table A.7 Error rate for TCGA breast data with 300 features with 3 hidden layer

Algorithm	Min	Mean	Median	Max
PCA 300	7.8	11.7	15.5	15.5
ANOVA 300	8.5	11.9	15.5	15.5

Table A.8 Error rate for TCGA breast data with 500 features with 3 hidden layer

Algorithm	Min	Mean	Median	Max
PCA 500	5.7	10.1	15.5	15.5
ANOVA 500	6.3	10.0	15.5	15.5

Table A.9 Error rate for TCGA breast data 700 features with 3 hidden layer

Algorithm	Min	Mean	Median	Max
PCA 700	3.3	10.0	15.5	15.5
ANOVA 700	3.7	10.1	15.5	15.5

Table A.10 Error rate for TCGA breast data 1000 features with 3 hidden layer

Algorithm	Min	Mean	Median	Max
PCA 1000	3.6	11.0	15.5	15.5
ANOVA 1000	3.5	10.8	15.5	15.5

Table A.11 Error rate for TCGA breast data with random forests with 3 hidden layer

Algorithm	Min	Mean	Median	Max
RF (higher 0.0010 (250))	10.1	14.2	15.5	15.5
RF (higher 0.0005(400))	3.7	13.7	15.5	15.5
RF (higher 0.0001(989))	19.6	22.2	21.5	26.2

Appendix B

Error Rates of the experiments with Ladder Network

B.1 TCGA kidney data with Ladder Network

Table B.1 Error rate for TCGA kidney data with 200 features with ladder

Algorithm	Min	Mean	Median	Max
PCA 200	1.1	1.2	1.7	3.3
ANOVA 200	0.8	1.0	0.8	1.2

Table B.2 Error rate for TCGA kidney data with 300 features with ladder

Algorithm	Min	Mean	Median	Max
PCA 300	0.8	1.2	0.8	2.5
ANOVA 300	0.8	1.6	1.6	3.9

Table B.3 Error rate for TCGA kidney data with 500 features with ladder

Algorithm	Min	Mean	Median	Max
PCA 500	0.8	2.0	2.5	2.5
ANOVA 500	0.8	1.6	1.6	3.9

Table B.4 Error rate for TCGA kidney data with 606 features with ladder

Algorithm	Min	Mean	Median	Max
PCA 606	3.3	4.8	4.9	5.8
ANOVA 606	1.7	2.1	1.7	4.1

Table B.5 Error rate for TCGA kidney data with random forests with 1 hidden layer

Algorithm	Min	Mean	Median	Max
RF (higher 0.001 (250))	5.8	6.6	6.6	9.9
RF (higher 0.0005(393))	8.3	9.8	10.7	11.5
RF (higher 0.0005(919))	79.2	82.2	80.2	87.3

B.2 TCGA breast data with Ladder Network

Table B.6 Error rate for TCGA breast data with 200 features with ladder

Algorithm	Min	Mean	Median	Max
PCA 200	6.2	7.2	7.1	8.2
ANOVA 200	2.5	3.2	3.7	3.7

Table B.7 Error rate for TCGA breast data with 300 features with ladder

Algorithm	Min	Mean	Median	Max
PCA 300	5.7	6.1	6.2	7.1
ANOVA 300	2.4	2.5	3.7	3.7

Table B.8 Error rate for TCGA breast data with 500 features with ladder

Algorithm	Min	Mean	Median	Max
PCA 500	2.9	4.4	4.8	4.9
ANOVA 500	3.3	4.4	4.9	5.4

Table B.9 Error rate for TCGA breast data with 700 features with ladder

Algorithm	Min	Mean	Median	Max
PCA 700	2.1	2.5	2.7	4.1
ANOVA 700	2.5	4.4	4.9	4.9

Table B.10 Error rate for TCGA breast data with 1000 features with ladder

Algorithm	Min	Mean	Median	Max
PCA 1000	1.7	2.5	3.3	3.3
ANOVA 1000	2.1	2.7	2.9	4.1

Table B.11 Error rate for TCGA breast data with ladder network

Algorithm	Min	Mean	Median	Max
RF (higher 0.0010 (250))	74.5	84.5	84.1	87.3
RF (higher 0.0005(400))	27.5	29.0	29.1	30.8
RF (higher 0.0001(989))	59.2	61.5	60.2	63.5