POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Matematica (Mathematical Engineering)

Tesi di Laurea Magistrale

Novel Neural Techniques for Gene Expression Analysis in Cancer Prognosis

Relatore
prof. Elio Piccolo

Correlatori:
prof. Giansalvo Cirrincione
prof. Alberto Tonda

Pietro Barbiero
matricola: 252818
Gabriele Ciravegna
matricola: 234726

Anno accademico 2018 – 2019

To my sweet girlfriend,

who enlightened my life with love and joy,
who comforted me in times of trouble,
who shared my burdens making them lighter.

Summary

This manuscript summarises two years of analyses, experiments and developments in the machine learning field. During that period, authors have collaborated in devising novel ideas and applying them to real world problems.

The main application setting is related to the analysis of patient derived xenografts (PDXs) of metastatic colorectal cancer (mCRC). PDXs are obtained by propagating surgically derived tumor specimens in immunocompromised mice. Through this procedure, cancer cells remain viable ex-vivo and retain the typical characteristics of different tumors from different patients. Hence, they can effectively recapitulate the intra- and inter-tumor heterogeneity that is found in real patients. During the last decade, the Candiolo Cancer Institute (Italy, IRCC) has been assembling the largest collection of PDXs from mCRC available worldwide in an academic environment. Such resource has been widely characterized at the molecular level and has been annotated for response to therapies, including cetuximab, an anti-EGFR antibody approved for clinical use. The mCRC PDX samples analyzed in this manuscript were kindly provided by IRCC the in form of microarray data, i.e. a large table containing the gene expression levels of tumor cells. Indeed, the medical objectives of the analyzes described in this work concern, on the one hand, the extraction of gene expression patterns useful for the instruction of therapies and further clinical experiments, and, on the other hand, the creation of models capable to correctly classify unlabeled data according to the cancer response to drugs.

From a statistical and machine learning point of view, the main difficulty in dealing with such data is the so-called curse of dimensionality. Indeed, only few hundreds of PDXs (samples) were provided against tens of thousands of gene expressions (features). Preliminary analyses performed with state-of-the-art techniques perform poorly when dealing with this problem, reporting limited effectiveness and opaque models. Thus, the machine learning objectives were to improve the effectiveness of existing models and designing ad-hoc techniques to deal with high-dimensional data.

Analyses have been performed through both supervised and unsupervised methods. In particular, two different and delimited directions of work were carried out according to the typology of methods. This choice was made at the beginning of this work in order to also sub-divide the medical objectives to pursue. Roughly speaking,

in fact, the extraction of gene expression patterns is commonly achieved through unsupervised techniques, while the creation of a classifier model can be obtained only with supervised techniques. Nonetheless, meaningful insight into gene behaviours were provided also through supervised learning. Authors equally contributed in the development of all analyses presented by alternating on the two paths.

Chronologically, the research was carried out in two different stages: initially, the high dimensional problem was addressed through the development of feature selection algorithms. Moreover, during this phase, the dataset was analyzed with state-of-the-art techniques for data visualization, like parallel coordinates diagrams, as well as feature extraction and manifold analysis, like CCA. Results of these preliminary analyses were published in two different chapters as book chapter of the Springer series “Quantifying and Processing Biomedical and Behavioral Signals”: "Unsupervised gene identification in colorectal cancer" and "Supervised gene identification in colorectal cancer". In the second stage, the attention of this research, from the unsupervised point of view, focused on the development of a biclustering framework capable of extracting useful gene correlations. In order to do that, a completely novel neural clustering technique called GH-EXIN was devised. Further details about the main features of this technique will be given in the following. The results were presented in "Neural biclustering in gene expression analysis", in the proceedings of the CSCI conference, Las Vegas 2017. This biclustering framework was successfully tested also on an external dataset for face detection problem and the results were presented in "Assessing discriminating capability of geometrical descriptors for 3d face recognition by using the GH-EXIN neural network", in the proceedings of the 2018 WIRN conference in Vietri sul Mare. From the supervised point of view, instead, different states of the art classifiers were compared in order to create the best classifier model. Finally, the best performer was a shallow neural network, with hyper-parameters tuned through an evolutionary algorithm. At the same time, another feature selection algorithm based on shallow neural network weight analysis was developed and, compared with ANOVA, performs a more effective selection for instructing a successive neural network model. The works were proposed and accepted at the 2018 WIRN conference, in two conference articles entitled "Evolutionary optimization of neural network hyperparameters" and "Understanding cancer phenomenon at gene-expression level by using a shallow neural network chain".

The main innovation proposed in this manuscript is represented by GHEXIN, a novel neural based algorithm for hierarchical clustering. The proposed approach builds a divisive hierarchical tree in an incremental and self-organized way. Indeed, it is a top-down technique which divides data at deeper and deeper levels. It is data driven (self-organization), in the sense that the final tree is automatically estimated from data. Also, it does not require a predefined number of units and levels (incremental with pruning phase). With regard to state-of-the-art neural based algorithms (as GHNG and DGSOT), GH-EXIN shows remarkable innovations:

- It performs a semi-isotropic quantization of the input space, by exploiting both isotropic and anisotropic criteria. The isotropic criterion is based on the extent of the neuron neighborhood, like in most neural based approaches. The anisotropic criterion (unique feature of GH-EXIN) is topological, as it is modelled by the convex hull generated by the weight vector of the winner neuron and the weight vectors of its topological neighbours (i.e. those connected through edges).
- It exploits sophisticated data reallocation and outlier detection methods. Data reallocation is the mechanism by which data associated to pruned nodes (orphan data) are possibly reassigned to other neurons. In GH-EXIN all orphan data are labelled as potential outliers at the end of each epoch. For each potential outlier, GH-EXIN looks for a new winner among all leaf nodes. Only when the winner belongs to the same basic neural unit of the pruned node and the potential outlier is outside its hypersphere, is datum definitely marked as outlier and is not reassigned.
- It exploits a simultaneous vertical and horizontal growth in order to represent abstract characteristics of the observed phenomenon. At the end of the training process of a basic neural unit, the resulting graph is analyzed by searching for connected components. If more than one connected component is detected, the algorithm tries to abstract a representation from the observed phenomenon. Hence, each connected component, representing a cluster of data, is associated with a novel abstract neuron. The reference vectors of abstract neurons are placed in the centroids of the respective clusters. The tree structure is modified by inserting abstract neurons between the leaf nodes and the father node, resulting in a simultaneous vertical and horizontal growth.

The comparison with aforementioned techniques shows how GH-EXIN is typically more efficient, as it reaches similar performances in terms of peak-signal to noise ratio (PSNR) by using fewer neurons. Moreover, qualitative evaluation of the resulting topology shows how GH-EXIN is much more elegant in connecting neurons, providing superior manifold representations. Finally, the restricted number of user-dependent parameters makes the tuning process of GH-EXIN very easy. The GH-EXIN source code was fully developed by authors in MATLAB and is publicly available on BitBucket[1]. The application of the biclustering framework integrated with GH-EXIN on the biological dataset revealed some interesting gene correlation patterns. These results have been submitted to the attention of IRCC doctors, who are currently analyzing them for possible scientific implications.

The results above are promising and highlight the potential for future work. From the point of view of the biological advances, the outcomes of both the unsupervised

[1]Gabriele Ciravegna and Pietro Barbiero. GH-EXIN (version 1.0.1). https://bitbucket.org/machine_learning_research/ghexin/src/master/,2018

and the supervised path are promising yet opaque: while the models can be used effectively, the results are difficult to interpret from a human point of view. As for the unsupervised direction of work, the GH-EXIN neural network resulted to be effective and easy to use as aforementioned; results provided by the biclustering framework, instead, are rather difficult to interpret without a statistical knowledge, since biclusters need several additional tools to be correctly evaluated. As for the supervised direction, the major advances with respect to previous analyses are achieved by exploiting the shallow neural network model. Indeed, the simplicity of such model makes it easier to handle and interpret. On limiting the number of features used, however, the accuracy drops significantly.

Moreover, major limitations of our work directly derive from the analyzed data. On the one hand, the analyzed data represent an estimate of the amount of times each gene is transcribed in a tumor xenograft. However, gene replication does not always result in protein generation. Indeed, this kind of data may not represent cell behaviour correctly. On the other hand, the restricted amount of samples was the most serious issue, since machine learning reliability is directly related to the amount of data provided.

Hence, within the biological domain, future developments will involve the use of up-to-date data (e.g. representing proteins instead of gene expressions) and the integration with other sources of data, such as image samples. Besides, from a machine learning point of view, models easier to interpret may be developed in order to provide more reliable and human-understandable outcomes. Future research in this field will consist in devising new algorithms overcoming the intrinsic weaknesses of machine learning, above all understandability. To this purpose, novel algorithms which integrate classic symbolic artificial intelligence with machine learning techniques seem to be very promising.

Acknowledgements

We have to thank few people who personally contributed to the production of this work:

- *Our supervisors, Elio Piccolo, Giansalvo Cirrincione, and Alberto Tonda, who followed the development of this work with interest and passion far beyond institutional duties;*
- *Andrea Bertotti, from Istituto per la Ricerca e la Cura del Cancro (IRCC), who instructed us about the cancer phenomenon and provided us with the dataset;*
- *Giovanni Squillero, from Politecnico di Torino, who helped us in the development of the evolutionary algorithms;*
- *Vincenzo Randazzo, from Politecnico di Torino, who helped us in programming GH-EXIN.*

The following acknowledgments are related to Barbiero Pietro only.

I want to thank many people who have always supported me during my life. They have contributed to the achievements of this result too.

- *My sweet girlfriend, who enlightened my life with love and joy;*
- *The co-author Gabriele Ciravegna, dear friend and personal research fellow;*
- *My family, who always supported me in each phase of my life;*
- *My friends, who walked by my side during all these years;*

Thank you from the bottom of my heart.

Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Biological Background

1.1 Colorectal Cancer

Before starting any analysis, a brief but fundamental study of the matter has been necessary. As computer scientist and even more as data scientist, it is always important to spend some time trying to deeply understand the characteristics of the problem you are analyzing. Producing a deep statistical analysis without being able to understand the results is pointless most of the times. Nevertheless, in the case at hand, the problem is such complex that a medical response is still necessary. The following, indeed, is only a rough summary of main cancer features without claiming to be complete.

1.1.1 What Is Cancer?

Under the word cancer, a collection of related diseases is grouped. Common traits to these diseases are the abnormal growth of the cells and their capacity of spreading to surrounding tissues.

A list of different anomalies allows cancer cells to grow abnormally. Firstly, it is possible to notice that old and damaged cells don't die, differently from normal cells. This characteristic is caused by the capacity of tumor cells to ignore apoptosis signals. These signals are normally sent by the system to get rid of unneeded cells. In cancer cells, instead, cycle-regulating genes as RAS and p-53 result to be altered or often deactivated, causing apoptosis pathways not to be followed. Further signals sent by the immune system are also ignored, as those preventing cells from replicating when its DNA is altered. Furthermore, the immune system generally stops sending nutrients to unneeded cells causing target cells to starve. In cancer cells, however, gene

responsible to autophagy are also inhibited and cells do not get destroyed even though they are starving. Cancer cells are also able to influence surrounding micro-environment to avoid this phenomenon: blood vessels are created nearby tumoral cells in order to feed them of oxygen and nutrients, necessary elements for cell lives. These vessels are also used by the cells to throw away cellular waste products [1]. Secondly, but not least, new cells are formed even though they are not needed. These new cells keep dividing by mitosis which allows cancer to grow more and more.

Cancer spreading to other tissues, instead, may occur at different levels. At first, it is possible to notice that tumor has affected nearby tissues; afterwards, it may be possible to notice that tumoral cells are growing on distant organs. This phenomenon is called metastasis which from the Greek literally means "transposition" of a disease: in this case, the tumor. It is worth to notice that cells found inside a brain metastasis are colon tumoral cells, not brain tumoral cells. Original cells, in fact, spread through blood or through lymph system, attach to existing tissue of distant organs and start growing forming the metastasis, as shown in Fig. 1.1. Among many reasons, this occurs because cancer cells are less specialized than normal cells and, hence, they are less likely to be rejected by new tissues.

The capacity of spreading to surrounding tissues is the main difference between malignant and benign tumors. Another characteristic typical of malignant tumor is the capability of growing back after being extracted.

1.1.2 Why Does Cancer Arise?

Fundamentally, cancer is a genetic disease and its appearance is due to some gene behaviour changes. These changes are generally either inherited from our descendants or caused by environmental factor.

Going into further details, there are three main types of gene whose mutation is critical for cancer growth.

1. Proto-oncogenes: involved in cell growth or inhibition of apoptosis, these genes may turn into oncogenes after some genetic mutations and, therefore, promote cancer emergence.

2. DNA-repair genes: their scope is to repair damaged DNA after a mutation occurred, caused either by an environmental factor or a metabolic activity.

3. Tumor-suppressors: their role is to protect cell from alterations either by repressing genes that promote cell division when not needed or when

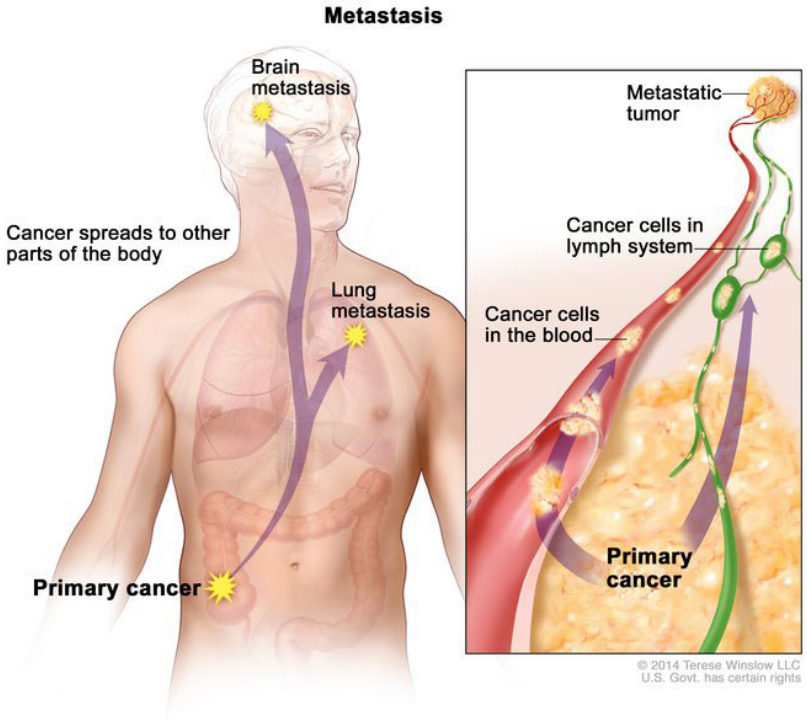


Figure 1.1: Cancer spreading from its original site to distant organs

DNA is damaged, or by starting apoptosis if DNA damage could not be fixed. Furthermore, they prevent tumor cells from dispersing (and creating metastasis) by blocking locomotion when DNA is irreparably damaged.

As first noticed in the *two-hit hypothesis* [2], cancer is the result of many mutations to cell's DNA: in particular, it is the result of both the mutation of proto-oncogenes and the deactivation of tumor-suppressor genes.

What is really important to notice here is that each cancer is a unique combination of genetic changes. Because of this inter-patient heterogeneity, research in these years has focused on personalized therapies to increase their efficiency [3]. Nevertheless, to achieve this goal further studies are needed, in particular on bio-marker discovery, which is the scope of this work. In fact, many gene functionalities and how different genes are co-regulated under specific circumstances is still unknown []. For personalized therapies, in particular, it is important to know a priori whether the patient will heal using a certain drug or not, given the genetic expression of the cancer tissue.

1.1.3 CRC

Colorectal cancer (CRC) is a particular type of cancer that develops in the large-intestine. It generally starts as an adenomatus polyp which may turn into an adenocarcinoma as shown in Fig. 1.2.

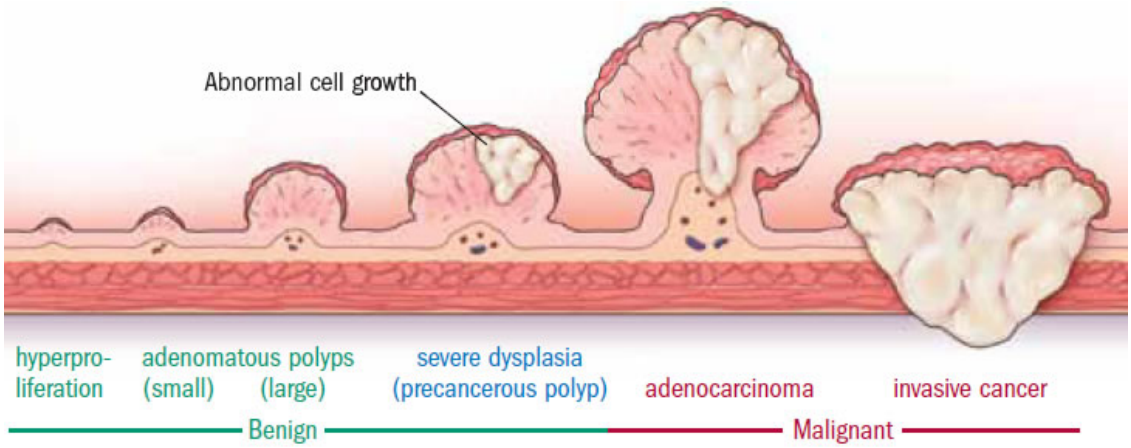


Figure 1.2: Colorectal cancer evolution

Recently, computer-aided diagnosis researches obtained good results in the classification of colorectal polyps through Convolutional Neural Network (CNN) system [4]. These systems are used in order to determine whether analyzed cells belong to a healthy tissue, or to an adenoma (which may be a signal of potential successive cancer), or to an adenocarcinoma (which is already cancer). The results are commonly used to create an attention map of possible cancer areas, which may drive doctors during a prognosis.

This type of cancer has as main causes the advanced age of the patients and lifestyle factors: among others, diet (in particular excessive consumption of red meat and alcohol), pollution, smoke, obesity and lack of physical activity play a key role.

Colorectal cancers are the third most common cancer globally as reported by the 2014 World Cancer Report [5]. There are about 1.4 million new cases and 694 thousand deaths each year because of colorectal cancer. It is more likely to appear in man than in woman and it is more present in the developed countries than backward countries.

1.2 Xenografts

Medical treatment of cancer is an extremely complex problem. Due to intra- and inter-tumor heterogeneity, the same drug may have different levels of effectiveness on patients with the same type of cancer. Therefore, personalized approaches are required to increase the reliability of prognostic predictions and the efficacy of therapies. Recently, new powerful tools have been developed for biomarker discovery and drug development in oncology, which rely on a technology called Patient-Derived Xenografts (PDXs). A Xenograft, in general, is a cell, tissue or organ that is transplanted from one species to another. PDXs, in this particular case, are surgically-extracted tumor tissue specimens form patient affected by colorectal cancer. These tissues are then transplanted into immunocompromised mice as shown in Fig. 1.3.

Through this, cancer cells remain viable ex-vivo and retain the typical characteristics of different tumors from different patients. Hence, they can effectively recapitulate the intra- and inter-tumor heterogeneity that is found in real patients. Based on this idea, the PDX technology has been employed to conduct large-scale analyses which, as in this work, try to identify reliable correlations between genetic or functional traits and sensitivity to anti-cancer drugs. In this context, metastatic colorectal cancers (mCRC) have been collected for the last ten years and have generated the largest PDX biobank available worldwide in an academic environment. This collection has been already characterized at the molecular level and has been exploited to identify clinically relevant biomarkers for prediction of therapeutic efficacy [6].

1.3 Gene Expression Analysis

Transcriptional data were obtained from mCRC PDXs through the Illumina microarray technology [7]. The microarray technology is based on thousands of DNA microscopic probes placed on a solid surface such as a piece of glass, plastic or a chip, forming an array. These arrays allow to measure the gene expressions on a tissue sample.

The word gene expression refers to the quantity of Messenger RNA (mRNA) produced at a certain moment in a certain cell. The RNA synthesis (transcription) is a transfer of the information from the DNA where it is stored into mRNA which can be transported and interpreted. Later, mRNA moves the information to the ribosomes to enable the production of protein (translation). Protein will finally respond to cellular needs for which transcription

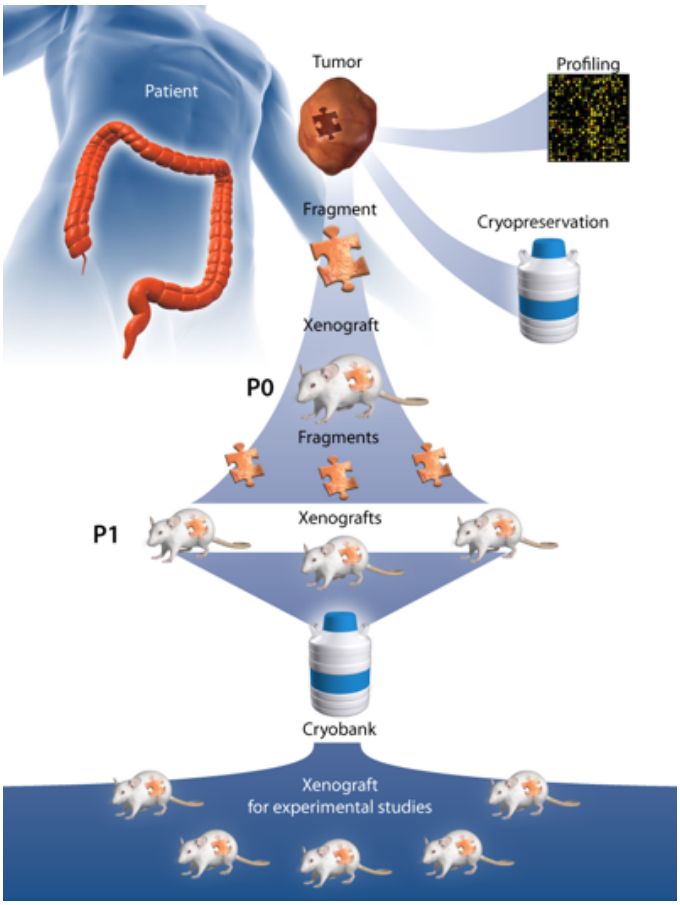


Figure 1.3: Patient-Derived Xenografts

started.

In theory, genes contain the information to produce all kind of mRNA. Nonetheless, in the biological samples, genes are transcribed into mRNA sequence in different quantities. This occurs for some reasons:

- Cell typology, different type of cells (brain cell, epithelial cells, liver cells) express different genes: among other factors this is what makes them different;

- Environmental factors such as the time of the day, whether the cell was proliferating or not and the presence of signals sent from other cells: according to cell needs, different gene are transcribed.

Gene expression analyzes often require measuring mRNA quantity in different conditions. High mRNA levels of a sequence under specific condition

may imply, indeed, a cellular need for the protein coded by the sequence. This kind of need may suggest a pathological condition when found on sick patients and not on healthy patients. As an example, if cancer cells express higher mRNA levels associated to a specific receptor, it is possible to infer that the receptor play a role in that type of cancer.

The focus of this work is about predicting drug sensitivity in PDXs and human patients affected by CRC. Hence, the different condition under which gene expression levels are evaluated are the different responses to drugs. In particular, this research tries to find out which genes discriminate whether a drug is capable to heal the patient or not. Further details will be given from part II.

Chapter 2

Machine Learning Background

In the last decade, Machine Learning proved to be the enabling technology for many innovations in different fields. Nowadays, it is the most important trend in computer science according to Gartner Hype Cycle[8]. Nonetheless, it is possible that the expectation created among investors, who are putting billions of dollars in AI research, and among customers will not be completely satisfied in the next future, as also machine learning has some intrinsic limits.

2.1 History

The term *artificial intelligence* (AI) was coined by John McCarthy during a conference at Dartmouth College in 1956. During his talk, he proposed to design an artificial machine able to learn and reason like human beings. In the same years, Allen Newell and Herbert Simon developed Logic Theorist, the first program able to apply basic reasoning functionalities. However, until 70's the advances in artificial intelligence were purely academic. Indeed, the proposed approaches were unable to tackle real world problems, due to computational power and memory limits, and reasoning deficiencies [9]. The first turning point in AI occurred when Feigenbaum introduced DENDRAL and MYCIN, the first expert systems, based on symbolic approaches. These algorithms were able to incorporate human knowledge and manipulate basic logic principles in order to provide reliable inferences in specific domains. The reliability guaranteed by expert systems allowed the spread of such algorithms for commercial use.

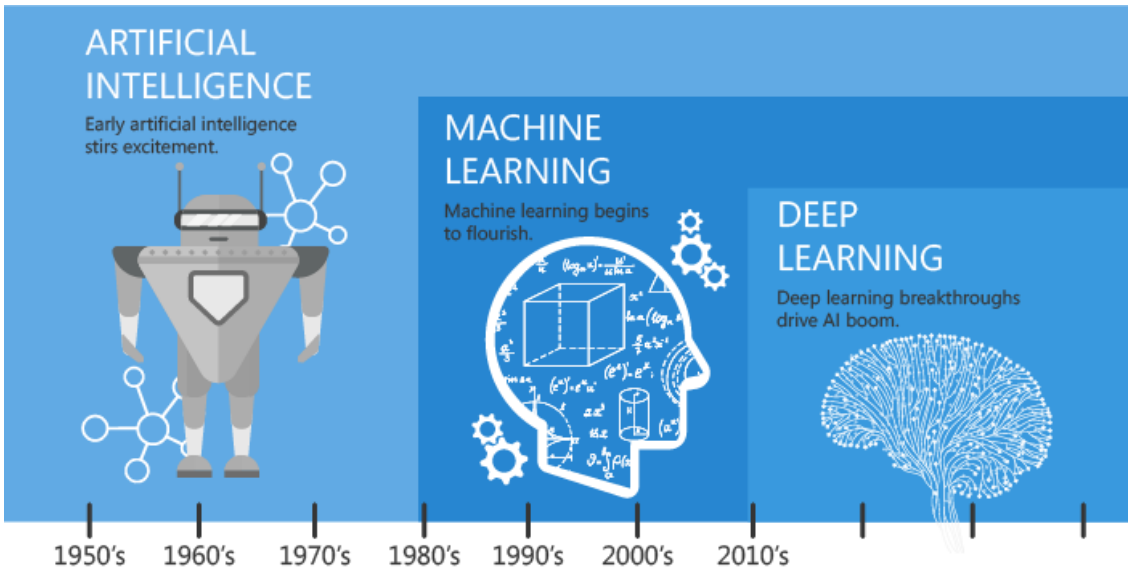


Figure 2.1: Artificial Intelligence progresses

The second turning point happened in 1986, thanks to the work of Rumelhart, Hinton, Williams and McClelland on the backpropagation algorithm. They showed how to apply efficiently this algorithm for training multi-layer perceptron neural networks, overcoming the deficiencies pointed out by Minsky [9]. With backpropagation the learning problem was shifted as an optimization problem, where the model fits its parameters directly on data, without being explicitly programmed. This change of perspective generated an AI sub-field picturesquely called machine learning.

Nevertheless, until few years ago it was not feasible to effectively employ complex (or deep in case of neural networks) models as they require large amount of both computational power and data to be optimized. Everything changed in 2012 when different works achieved important results using Deep Neural Networks in challenges such ImageNet in object recognition [10]. The technological progress provided scientists of ever more powerful processors, while the advent of Big Data and IoT at the beginning of 2010's started producing the quantity of data required by models to be effective. It is possible to state that the advances in the field of deep learning will be strictly correlated in the future to the ones of IoT and Big Data; the information extraction process always starts from data. As it will be possible to apply sensors on even more objects which in the past were not accessible (e.g. human organs), new applications of machine learning will be discovered.

2.2 Achievements

Machine learning is nowadays already changing our daily life in many different fields:

- Preventive diagnosis: machine learning is achieving good results in healthcare, in particular for heart diseases and cancer diagnosis. In some conditions, an early diagnosis can save a patient's life. Statistics state that 17.9 million people die every year because of a heart attack [11] and 9.6 million because of cancer [12]: together they account for about half of the global deaths. A similar idea is also applied in preventive maintenance for machines: it allows companies to save thousands of dollars, avoiding replacing parts unless it is necessary.

- Autonomous car: in the very next years we will be relieved of the chore of driving, resulting in a significant reduction of traffic collisions and enhanced mobility for the elderly, children and disabled. Nowadays, around 1.3 million people die every year in a car accident [13]: hypothetically, this number could be reduced to zero with autonomous cars.

- Security: face recognition algorithms are widely used in video surveillance to find wanted criminals. Thousands of cameras have been deployed in our cities, it would require hundreds of people to check them all. Furthermore, emotion recognition and other algorithms are used to preempt crimes from taking place by analyzing people unusual behaviours.

- Language translation: results provided by Google Translate are astonishing, in many languages it performs almost as well as a human translator.

- Virtual assistants: many people nowadays are used to speak to their smartphones to save appointment on the agenda or to check weather conditions. This has become possible thanks to advances in speech recognition.

- Fraud detection: machine learning is used in this field for analyzing mails, detecting fishing attempts and generally spam. Banks and other financial actors are also using it for analyzing bank transactions, in order to detect illegitimate ones and money laundering.

- Product recommendation: Amazon completely transformed online shopping by suggesting people items they may be interested in. Google AdWords did the same with personalized advertising: nowadays it is the only competitor for publishing commercials online.

Most of these achievements have been feasible due to deep learning. Nowadays, it is the most popular e probably powerful technique. Because of this reason, in the following it will be analyzed in further details trying to highlight what are its limits. Anyway, much of the below mentioned applies also for machine learning in general.

2.3 Deep Learning Limits and Weaknesses

Deep Learning advances in recent years have brought great excitement in AI world. Andrew Ng, one of the pioneers of Deep Learning and founder of Google Brain, asserted: “If a typical person can do a mental task with less than one second of thought, we can probably automate it using AI either now or in the near future” [14]. Is it true? Is current artificial intelligence (mainly based on deep learning) already able to carry out now or in the next future whatever human mental task? Other researchers expressed a different opinion about it: "Scaling up current deep learning techniques by stacking more layers and using more training data can only superficially palliate some of these issues. It will not solve the more fundamental problem that deep learning models are very limited in what they can represent, and that most of the programs that one may wish to learn cannot be expressed as a continuous geometric morphing of a data manifold” [15] As partial proof of it, an important project as the full-service chatbot M developed by Facebook already failed to achieve its initial goals because they were beyond deep learning current capacities and eventually has been closed.

2.3.1 Brief Deep Learning Theory

In order to better understand strengths and limits of deep learning, it is good to know how it works. Basically, it is a statistical technique that performs a mapping between an input space and an output space. Learning is performed in a supervised way with thousands of labelled data supplied to the algorithm to build a robust model. Internally, neural networks are made of nodes which exchange information. They are grouped into layers with an input layer which represents the input space, an output layer with as many nodes as

the elements of the output space are and several hidden layers: the more the number of hidden layers the deeper is the network. Nodes are linked to each other and a weight is associated to each connection. These networks are generally told to represents neural cortex although in a very simplistic way. Nodes can be thought as neurons and connections between nodes may stand for the synapses.

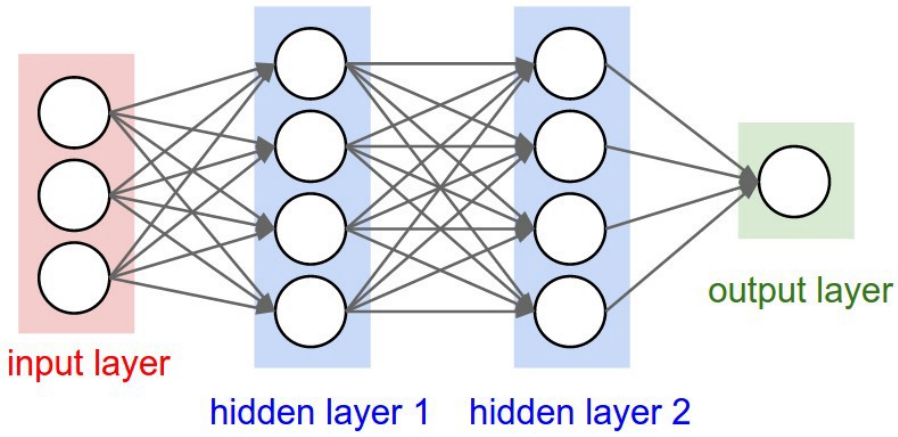


Figure 2.2: Classic Neural Network Representation

2.3.2 Limits

The incredible strength of deep learning is that "In principle, given infinite data, deep learning systems are powerful enough to represent any finite deterministic 'mapping' between any given set of inputs and a set of corresponding outputs" [16]. Nevertheless, this statement also reveals some of its limits:

- Generalisation: we will never be able to provide 'infinite data' to the systems. This implies deep neural networks will always be brittle, failing as fast as the context on which the model was trained change a little. An example may be the DeepMind's Atari game work [17]: the system learns in 240 minutes of training to play Breakout. Nonetheless, it has been proved [18] that the same previously trained system fails on a transfer test where a minor number of variations are made in the game such as inserting another wall in the middle. This implies that the system has not really learnt to break the wall. It has only acquired a sequence of moves to be performed to break that single wall in a very narrow context.
- Abstraction: deep learning system are only able to perform this specific

mapping operation. They are built without any abstract representation of the world. They have no idea of concept such as causality nor composition, that all human come to the world with, neither they have the possibility to learn it, as it is impossible to represent these concepts in terms of features.

- Understandability: deep learning is usually considered and treated as a black-box because the way in which this 'mapping' is produced is so complicated that is not explainable in form of rules. This introduces several limits to the use of this technique in critical applications as Medicine where the reason for which a decision needs to be taken is almost as important as the decision itself. Doctors cannot decide not to prescribe a saving-life drug to a patient only because a deep learning system suggested that it would not be effective on him, without really having the possibility to check why.

Hence, it seems possible that even with all conceivable computational power and data, new applications will be discovered, but deep learning effective capabilities may not grow any further.

Part II

Preliminary Data Analysis

Chapter 3

The High Dimensional Problem

The dataset stemming from the DNA microarrays is composed of the expression of 15396 genes in 203 Colorectal Cancer (CRC) murine tissues. For each tissue two additional quantities are available.

1. A discrete variable describing the cancer response to drugs, whose values are chosen as:
   - $+1$ (regressive cancer);
   - 0 (stable cancer);
   - -1 (worsening cancer).
2. A continuous variable representing the cancer response to drugs after three weeks, estimated as the difference in size of the tumor.

Data are preprocessed by the z-score technique in order to work on the same range. All analyses in this part have been done by considering the genes as variables. This is a very challenging problem because of analyzing very high dimensional data by means of a small training set. The only possible way to overcome this difficulty is the dimensionality reduction, even if it is data-driven too. As a consequence, the Principal Component Analysis (PCA in an under-determined framework) has been performed both for searching a first rough estimation of the intrinsic dimensionality of data and, above all, to test the non-linearity of the problem at hand. However, it results that at least 100 principal components are needed in order to explain the 90% of the data variance as shown in Fig. 3.1. This number has the same order of magnitude

of the training set size and is a consequence of the high dimensionality and, probably, of the fact that the manifold is nonlinear. Other tools are needed in order to check if the manifold is linear or not.

In the following chapters two different kind of manifold analysis are shown that approach the high dimensional problem in two different ways. In chapter 4 the high dimensional problem is approached through an unsupervised feature selection which make use only of gene expressions. In chapter 5, thanks to the discrete label assigned to each patient, the feature selection is performed in a supervised way. In both chapters, further analyses are then conducted with different techniques according to the path: unsupervised or supervised.

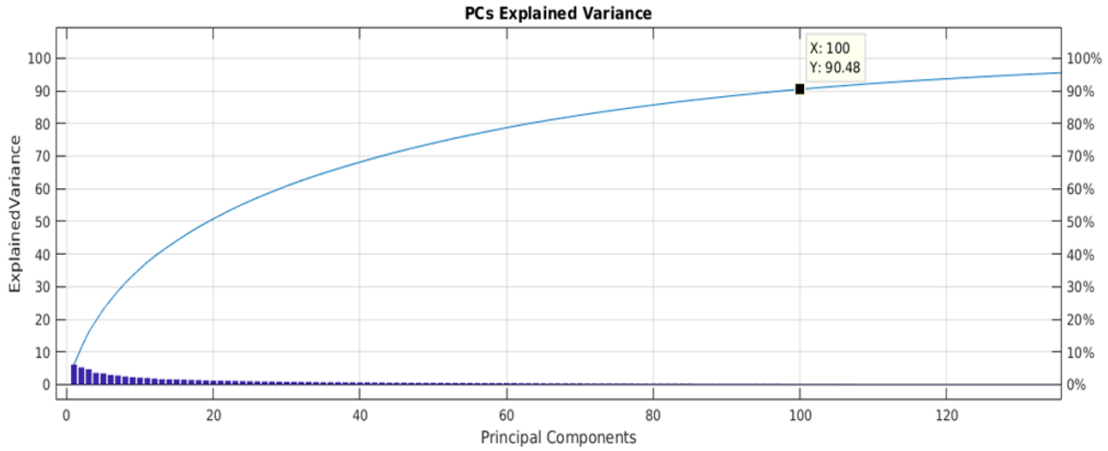


Figure 3.1: Pareto diagram of PC's explained variance

Chapter 4

Unsupervised Manifold Analysis

The unsupervised path makes use of a scoring algorithm of features based on their clustering ability to bypass the high dimensionality issue. Traditional methods of dimensionality reduction and projection are then used on subset features with high discriminant power in order to better analyze the data manifold. This chapter has been extracted from a work presented at the 2017 "Workshop Italiano sulle Reti Neurali" (WIRN2017) conference [19].

4.1 Unsupervised Feature Selection

Initially, tissues have been grouped considering all the available genes (features) with the Unweighted Pair Group Method with Arithmetic Mean (UPGMA), an agglomerative hierarchical clustering approach [20]. This bottom-up approach finds and merges the nearest pair of clusters r and s according to their mutual average distance:

$$d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} d(x_{ri}, x_{sj}) \tag{4.1}$$

The average distance algorithm has been chosen because it is robust against noise and outliers. Among different metrics, the Minkowski distance of order $p = 1$ has been found to have the highest cophenetic correlation coefficient (here 0.8392), therefore it is used in the following analyzes [21]. However, this approach has not provided satisfactory results; indeed, it was not able to find meaningful groups. For this reason, tissues have been processed using

a different procedure based on the Ward's minimum variance method [22]. This algorithm finds and merges the pair of clusters that leads to minimum increase in the total within-cluster variance after merging. The within-cluster variance increment due to the merging of r and s is proportional to the distances of the resulting cluster objects from the resulting cluster centroid:

$$d(r,s) = \sqrt{\frac{2n_r n_s}{(n_r + n_s)}||x_r - x_s||_2} \tag{4.2}$$

Differently from the first approach, the clustering algorithm is applied using one feature at a time, determining a one-dimensional clustering which yields the individual ability of discrimination of genes. This property is evaluated using the Calinski-Harabasz index (also called Variance Ratio Criterion, VRC, [23]):

$$VRC_k = \frac{SS_B(N-k)}{SS_W(k-1)} \tag{4.3}$$

where N is the number of samples, k is the number of clusters, SS_B is the between cluster variance and SS_W is the within cluster variance. Well defined clusters tend to have a high VRC. Therefore, genes are ranked according to this index. By defining a threshold in advance, several genes can be extracted. In other words, genes are selected according to their ability to discriminate tissues: this is estimated by checking the best possible separation (with regard to several choices of the number of clusters by means of the parameter $K \in [2,6]$) in terms of quality of the groupings by using an index (see Fig. 4.1).

4.2 Parallel Coordinate Plot

Parallel coordinates are a powerful way of visualizing high-dimensional data. This kind of data visualization was invented during the 19th century and sharpened by Wegman in 1990 [24].

A point in n-dimensional space is represented as a polyline with vertices on equally spaced parallel axes each of one representing a feature; the position of the vertex on the i-th axis corresponds to the i-th coordinate of the point. This plot is used in order to understand deeply the gene capacity of discrimination, by visualizing the distribution of the murine tissues (colored polylines) along all the dimensions (genes) represented as parallel vertical axes. In this figure blue lines represent worsening cancers, red lines

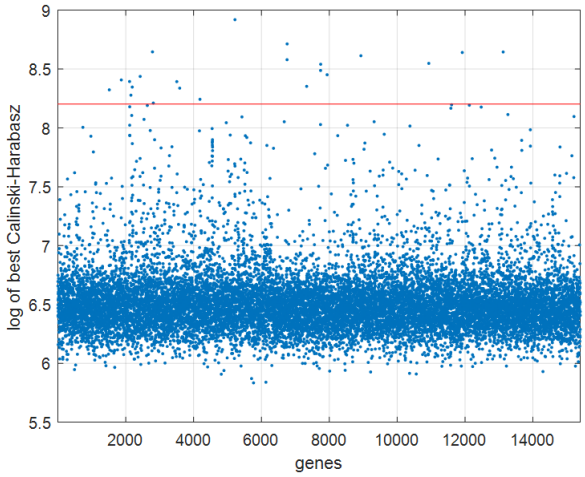


Figure 4.1: Cluster quality evaluation. The red line is the threshold.

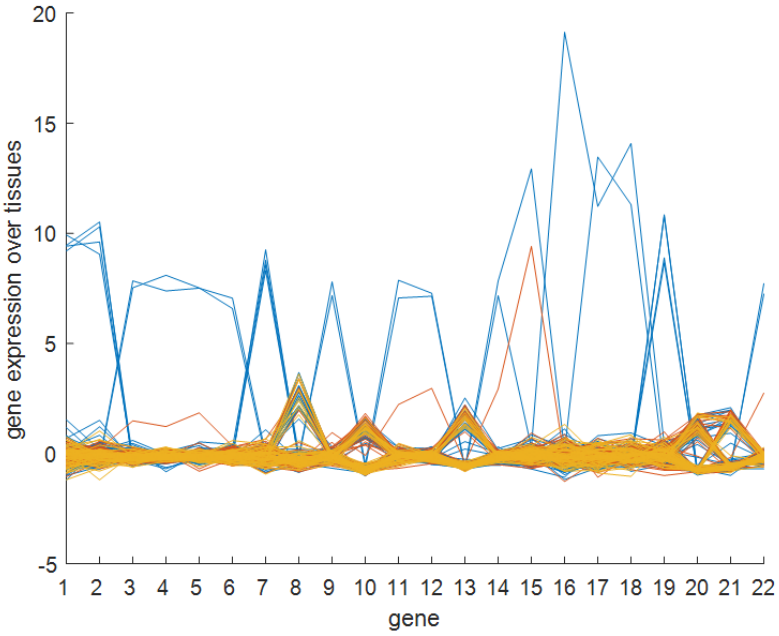


Figure 4.2: Parallel coordinate plot.

stable cancers and yellow lines regressive cancers, respectively. The intersections of the polylines with the vertical axes show there are some genes highly discriminating the three colors, which means that some mice have particular expression levels for some genes. Also, the colored grouping of polylines

show coherency, which means there is discrimination for tissues. Hence, these genes can be used as markers of CRC subtypes. This tool has been used as a visualization tool for the validation of the previous gene selection, based on the cluster quality evaluation. This technique has confirmed the selection of 22 genes (see Fig. 4.2). This allows the study of the data manifold representing the tissues in a lower dimensional space, just alleviating the problem of the curse of dimensionality.

4.3 Sub-Manifold Analysis

The reduced submatrix is composed of 203 rows, i.e. the tissue values (samples) and 22 columns, i.e. the selected genes (features). In order to check the intrinsic dimensionality and the linearity of the data manifold, the Principal Component Analysis (PCA, [25]) has been performed. The plot of the variance explained by the principal components shows an intrinsic dimensionality of about 5 (see Fig. 4.3), corresponding to 90% of variance explained. This result suggests that tissues belonging to the 22-dimensional space lay on a 5-dimensional hyperplane. The remaining 10% can be justified by either noise or small departure from linearity, that is nonlinearity only on a large scale, but not locally. In order to confirm this hypothesis, the Curvilinear Component Analysis (CCA, [26]) has been used. CCA is a neural technique for dimensionality reduction which projects points by preserving as many distances as possible in the input space. However, CCA is here used not for the exploitation of the projection, but for the information that can be derived from its dy-dx diagram (see Fig. 4.4). This plot represents distances between pairs of points in the input space (the dx value) and in the reduced (latent) space (the dy value) as a pair (dy, dx). If a distance is preserved in the projection, the corresponding pair is on the bisector (indicated in the figure). If the pair is under the bisector, it represents the projection as an unfolding of the input data manifold.

If the manifold is linear, all points tend to lie on or around (because of noise) of the bisector. Clusters of points on the bisector, but far from the origin, represent large inter-cluster distances, and, hence, reveal the presence of clusters. Fig. 4.4 shows this plot for a five-dimensional latent space (this choice is suggested by the previous PCA). The smaller grouping near the origin represents the intra-cluster distances and suggests the idea of one or several hyperplanes as data manifold. This is confirmed by the other groupings at larger distances. They represent the inter-distances and suggest

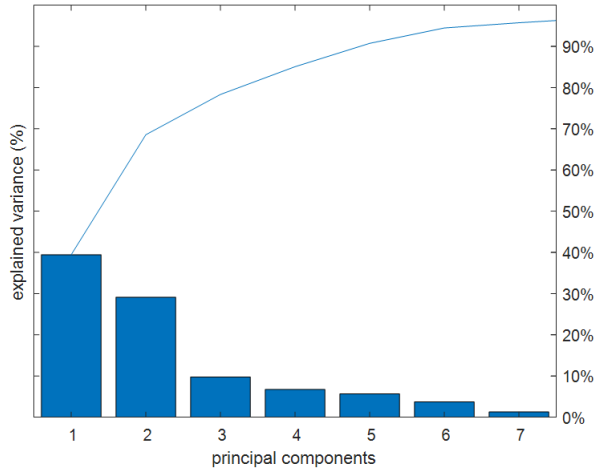


Figure 4.3: Variance explained by principal components.

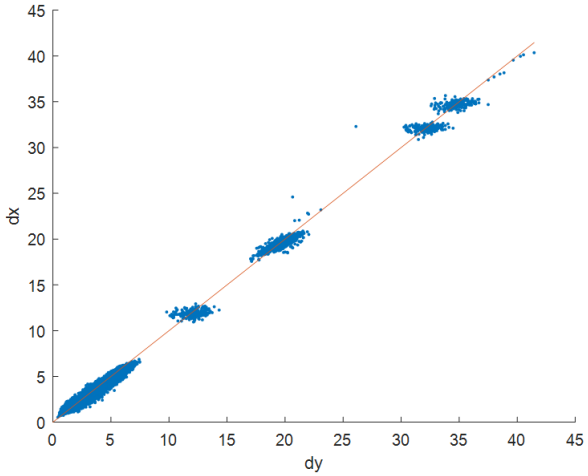


Figure 4.4: CCA dy-dx diagram.

the presence of at least four distinct clusters. The fact that, above all, the groupings of farthest distances have the biggest departure from the bisector, yields the idea of a curvature at large scale. Resuming, the data manifold in the space of the selected genes is composed of several well distinct nearly-flat submanifolds. This confirms the validity of our approach, in the sense that the extracted features discriminate well with regard to tissues.

Biplots [27] are now used in order to understand the reciprocal behavior (in statistical terms) between all tissues and the selected genes. They are a generalization of scatter plots. A biplot allows information on both samples

and variables of a data matrix to be displayed graphically. Samples are displayed as points while variables are displayed as vectors. Fig. 4.5 shows the biplot over the first three principal components. With regard to tissues (red points), there are only few data along the first and the second principal component. Instead, the third component has a good discriminant capacity over tissues. With regard to genes (blue vectors), most of them are strongly related to the first or the second principal component. However, five genes (in the figure, represented as 8, 10, 13, 20 and 21) stay along the third axis, thus explaining the variance of tissues along this direction. The biplot shows that a combination of these genes has a bimodal behavior along the third principal component. Fig. 4.6 shows a parallel coordinate plot of these five genes. This graph points out relationships between these genes and their bimodal behavior. Blue lines represent worsening cancers, red lines stable cancers and yellow lines regressive cancers, respectively. The most interesting gene is shown in the first vertical axis because it marks a coherent bundle of segments which represents a set of worsening cancers.

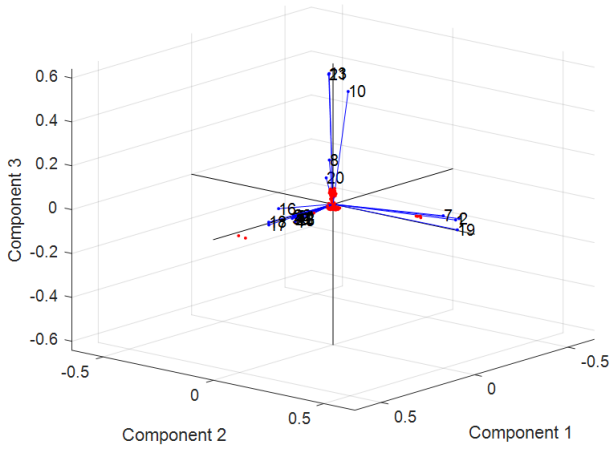


Figure 4.5: Biplot over the first three principal components.

4.4 Biological Feedback

The previous analysis shows that the selected genes, whose biological names are CRMP1, CSAG1, EIF1AY, PRAC1 and RPS4Y1 have a high discriminant power. From a biological point of view, some of these genes are strongly related with cancer. In particular:

- CRMP1 is supposed to be related to inhibition of metastasis [28];
- CSAG1 is supposed to be related to squamous cell carcinoma [28];
- PRAC1 is supposed to be related to human prostate and colorectal cancer [28].

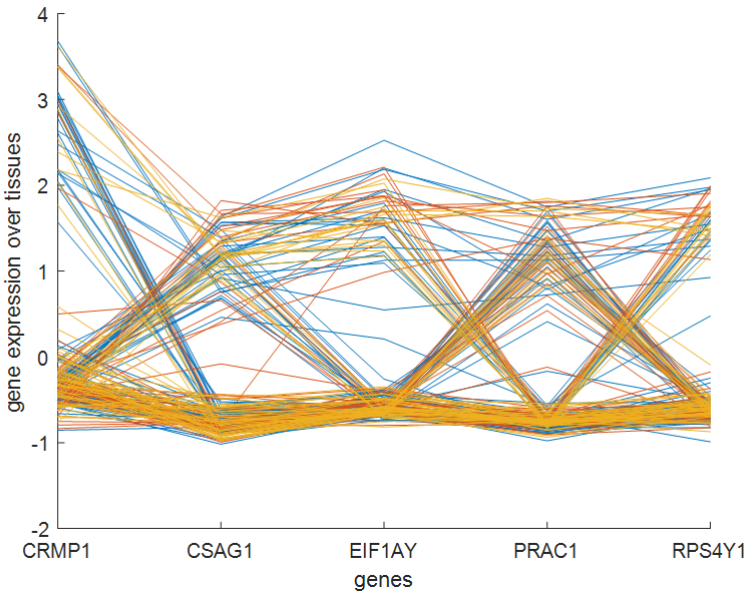


Figure 4.6: Parallel plot over the resulting gene selection.

Chapter 5

Supervised Manifold Analysis

As anticipated at the beginning of this part, in this chapter a supervised manifold analysis is performed. In addition, a simple model is built through a classic Multi Layer Perceptron (MLP): it correctly classifies unlabeled data according to the patient response to drug. Most of content exposed in this chapter was presented at the WIRN2017 conference [29].

5.1 Supervised Feature selection

As in the previous case, a manual feature selection based only on the parallel plot analysis is unfeasible due to the huge amount of features (15396) and to the difficulty to ascertain the colored bundle groupings. In order to circumvent this problem, a new algorithm for supervised feature selection, based on the Davies-Bouldin [30] clustering index, has been devised. Each gene has been evaluated in its capability of discriminating tissues with different response to drugs which is indeed the skill in grouping well-separated clusters, with high level of cohesion. Specifically, tissues have been divided, for each gene, into unidimensional clusters, exploiting only the associated label. Then, the resulting cluster quality is estimated. The Davies-Bouldin index is suitable for the case of study, because it considers both inter-cluster distances and intra-cluster distances, estimated according to the Euclidean distance. A quality threshold has been empirically selected in such a way that only the best genes are chosen. Hence, 19 genes have been retained as shown in Fig. 5.1. This way of selecting feature is unconventional because

it does not calculate directly the correlation between genes and the response to drugs, but it still selects the genes that will be more useful and reliable for a classification model based on those genes only.

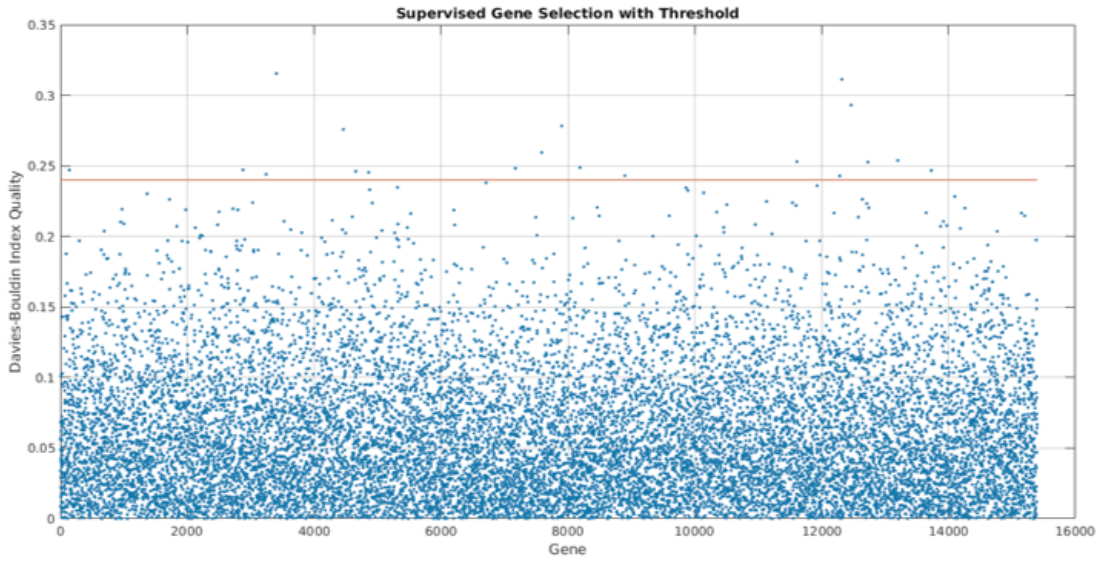


Figure 5.1: Cluster quality evaluation. The red line is the threshold.

5.2 Sub-Manifold Analysis

A subspace composed of the genes selected by the feature selection has been extracted from the original dataset, creating a matrix composed by 203 tissues and only 19 genes. In order to have a better insight of the data extracted, a PCA analysis has been performed for this reduced database as in the previous section. The inspection of the PCA explained variance suggests the intrinsic dimensionality of the manifold to be 11. This could imply data stay on a 11-dimensional hyperplane. A further insight of the manifold has been obtained through a CCA. Here CCA performs a projection from a 19-dimensional to a 11-dimensional space. As shown by the dy-dx diagram (see fig. 5.2), most pairs stay around the bisector, just confirming the hypothesis of 19-dimensional hyperplane.

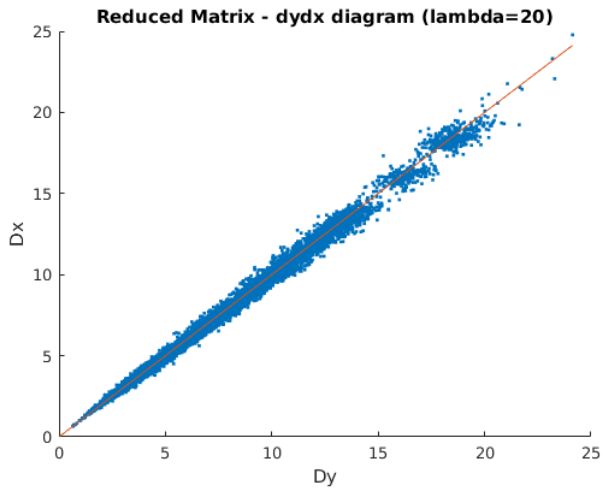


Figure 5.2: dy-dx diagram.

5.3 Lasso regression

Lasso regression [31] generally targets to improve the prediction accuracy and interpretability of the regression model by selecting only a subset of the available variables to use in the final model rather than using all of them. Lasso is able to achieve it by forcing the sum of the absolute value of the regression coefficients to be less than a fixed value, which forces certain coefficients to be set to zero, effectively choosing a simpler model that does not include those coefficients. It requires a regularization parameter λ which controls the trade-off between regression and constraint on the coefficients. Greater values of λ correspond to a lower number of variables inserted in the model. In the case of study, Lasso regression has been useful to confirm the intrinsic dimensionality of the reduced matrix previously established and, more importantly, to identify the 11 genes. This step has been possible because it is based on the linearity assumption of the reduced database, deduced from the previous manifold analysis. The response variable used previously cannot be exploited here, because it is discrete. Instead, the other variable associated to the tissues can be used (the tumor difference in size after 3 weeks). Fig. 5.3 shows gene coefficients that decrease until zero as λ increases. The value of λ suggested by Lasso (the dotted line in figure) is given by 0.05, because it is the one that guarantees the least Mean Square Error (MSE) as shown by Fig. 5.4. It is important to notice that the number of nonzero LASSO coefficients still present at $\lambda = 0.05$, is 11. This result confirms the previous assumption about the fact that the reduced manifold

is a 11-dimensional hyperplane.

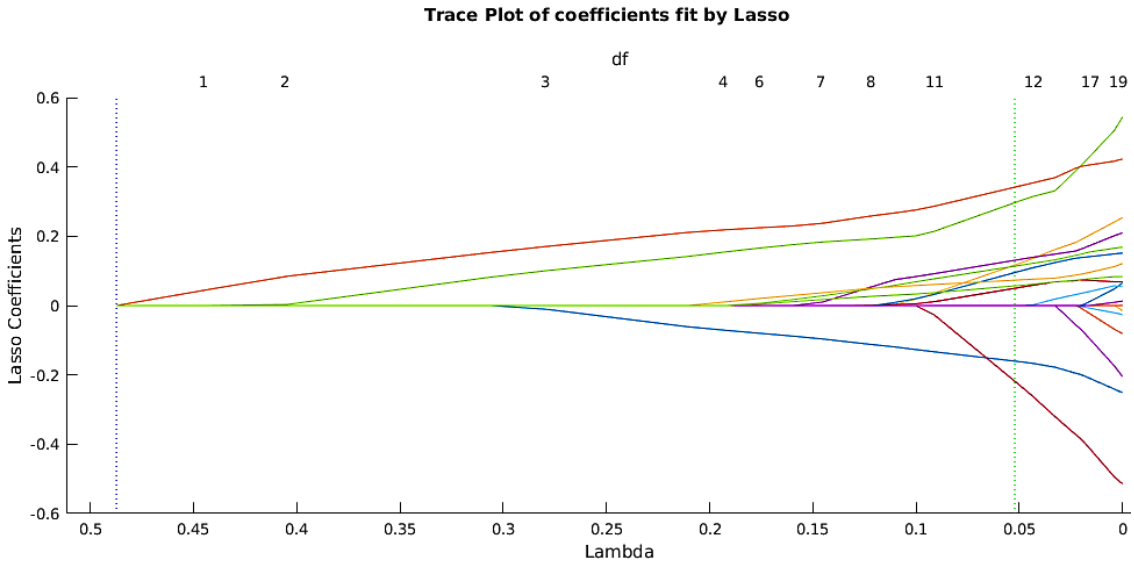


Figure 5.3: Lasso coefficients as a function of λ.

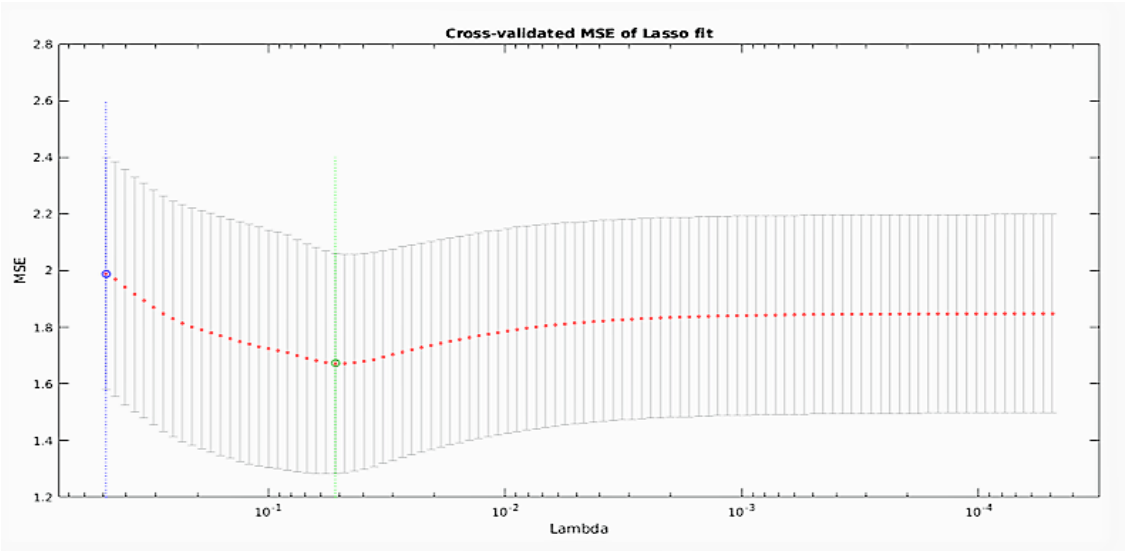


Figure 5.4: MSE of Lasso regression as a function of λ.

5.4 Biological Feedback

The 11 genes selected are the most meaningful for predicting the increment or decrement of the cancer volume size after three weeks. Their names are the following: LOC645233, FSCN1, ACSS2, SMAD9, MED1, TMEM118, LOC728505, SF3B4, LOC651316, SERPHIN1, GPR126. Some of these genes are already well known in the medical literature as correlated with cancer:

- FSCN1 is supposed to be related to cell motility [28];
- ACSS2 is supposed to be related to cancer cell survival [28];
- MED1 is supposed to be related to gene transcription [28]).

Correlation with cancer of the remaining genes has not been proved yet. Their presence in this work, however, suggests that they should be involved, at least in this particular context of the CRC response to drugs. In fact, it is important to observe that a gene expression may not be relevant for the presence of a tumor, but it may remain important for the survival of tumoral cells. Specifically, average expressions in patients of genes LOC645233 and ACSS2 seems to be in contrast with literature. Nevertheless, results regarding those genes have been published, since they are not an artifact of the analysis but they concern raw data. This analysis proposes a novel approach whose results may be considered as suggestions for further biological research.

5.5 Classification

At last, the expression of the selected 11 genes is used in order to train a classification model. Several models have been tested: the one that shows the best accuracy on the test set is the Support Vector Machine (SVM, [32]) with an accuracy of 78%. The model is tested through the hold-out validation with 25% of data randomly put in the test set. A further attempt to improve the accuracy of the model has been done through the use of a Multilayer Perceptron (MLP, [32]). It is composed of 11 inputs, 20 hyperbolic tangent hidden neurons and 1 output whose activation function is the logistic sigmoid. It is equipped with the cross-entropy error function and the backpropagation learning algorithm is used in order to evaluate the error derivatives for the BFGS training. For the purpose of this analysis two target classes have been selected: the first (1) corresponding to tissues with a regressive or stable

response and the second (0) for tissues where the disease has worsened. The robustness of the model is corroborated by both validation and test sets. The accuracy is shown in the Test Confusion Matrix (fig. 5.5) and is given by 80%. This result is not only important in itself, but can be considered as a figure of merit for the selected 11 genes: how accurate 11 genes over 15396 are in modeling the progression of the tumor.

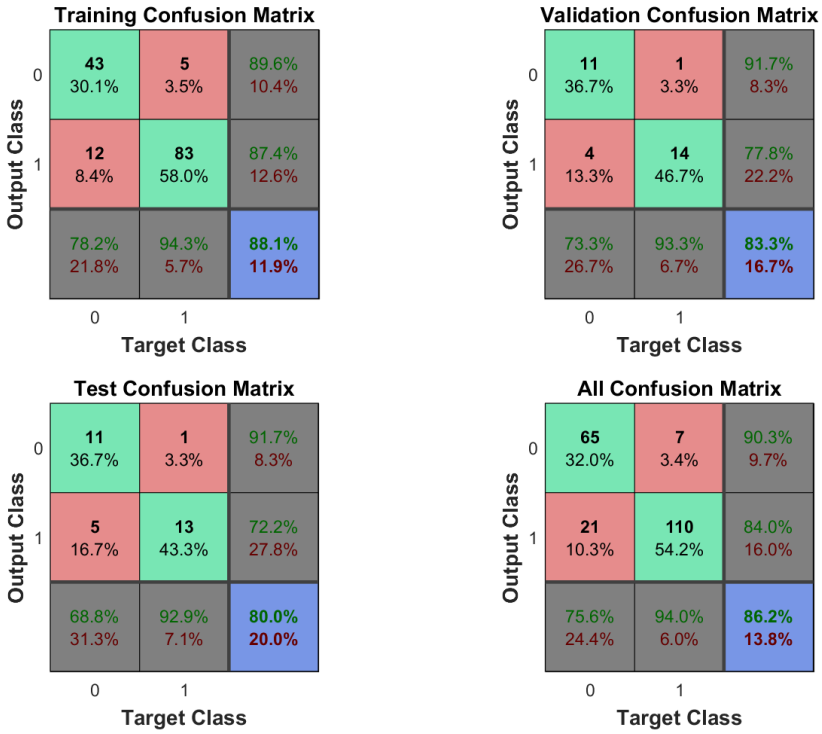


Figure 5.5: Confusion Matrices for the MLP classifier.

Part III

Advanced Neural Techniques

Chapter 6

Unsupervised Neural Techniques

6.1 Introduction

Following the unsupervised path previously introduced in chapter 4 of part II, further and deeper analysis have been conducted using several neural techniques. A further analysis in this direction has been required as class discovery is a key aspect in analyzing gene-expression data as reported in [33]

As already explained the high dimensional problem prevents a direct application of clustering algorithms in the feature space. Nonetheless, this is a common problem in gene expression analysis and in general while clustering high-dimensional data. As suggested in [34], this obstacle may be overcome through the use of Biclustering. This technique has been the key of the work: several advanced neural networks techniques performing clustering have been indeed tested in a biclustering framework. Eventually, a novel neural network, Growing-Hierarchical EXIN (GH-EXIN), devised ad hoc for the problem at hand has been successfully employed. GH-EXIN is based on a novel unsupervised neural approach called G-EXIN (Growing-EXIN [35]).

In Sec. 6.4, the biclustering technique is explained; in the following one, Sec. 6.3, two state-of-the-art techniques, used also in the context of biclustering, are reported; in chapter 7 the novel neural network, GH-EXIN, is introduced; a comparison between this technique and the previous ones is presented in Sec. 7.2; results of the neural framework applied on the current dataset is reported in Sec. 7.4.

6.2 Hierarchical Clustering Analysis

The hierarchical cluster analysis (HCA, [36]) is a multi-resolution clustering technique. It builds a tree of clusters with different levels of data interpretation. In data mining, for instance, HCA can yield a richer information than plain clustering. It is generally performed in two ways:

- Hierarchical Agglomerative Clustering (HAC), where each data in the beginning corresponds to a singleton cluster, and then pairs of clusters are merged until only one cluster containing all data is reached (bottom-up approach).
- Hierarchical Divisive Clustering (HDC), in which all data start in one cluster and splits are performed recursively until all clusters are singletons (top-down approach).

The top-down approach of HDC represents better the dataset because it starts taking into account all data. Instead, the bottom-up approach of HAC is, in this sense, more arbitrary in the initial steps, thus influencing the quality of the resulting tree. Also, HAC is intractable in case of large datasets. However, the way HDC splits the clusters is still an open problem. A promising approach is represented by the use of neural networks for clustering.

The neural algorithms for HDC can be classified according to both the way the neural tree is trained and the kind of basic neural network used for each node (basic neural unit). As a first taxonomy, there are two approaches: synchronous training (ST), where the training is performed on the whole tree, and asynchronous training (AT), where the training is performed node by node. The Dynamic Neural Tree Network (DNTN, [37]) adapts a dynamic hierarchy to data, as an output layer fed by the input: all growing nodes are fed by the same input and are trained simultaneously (ST). It requires a tolerance for determining the new neurons and a threshold for the child growth. It is not able to represent correctly the outliers. A variant of DNTN, the Competitive Evolutionary Neural Tree (CENT, [38]) claims it does not require user dependent parameters. Indeed, they become internal parameters which are dependent on data, but are empirical and not justified. CENT is based on the neuron activity (which is decreased in time for avoiding the poor initialization) and addresses the DNTN problem of outlier detection. Another ST approach uses the Self-Organizing Map (SOM), considered as a tree (TreeSOM, [39]). It is based on the interpretation of SOM as a tree

when a distance threshold is decreased in time [40]. TreeSOM addresses the problem of the sensitivity of the tree to the SOM topology and initialization by using the idea of consensus tree, which is a virtual tree averaging the trees resulting from different initial conditions. The best tree is the closest to the consensus tree.

Most neural approaches fall into the AT category. They can be classified according to the clustering technique used for each node. In [41], the k-means algorithm is exploited. It divides data in a fixed number of clusters (HDC), but uses an additional HAC for refinement. The Hierarchical Clustering Algorithm based on k-means with Constraints (HCAKC, [42]) uses an improved Silhouette for determining the k parameter. In [43] and [44], a preprocessing based on Principal Component Analysis and divide-and-conquer, respectively, is used for dealing with high-dimensional data. In [45], the Growing Hierarchical Tree SOM (GHTSOM) uses an elementary SOM given by three connected neurons as basic module (triangle). It requires two kinds of links: the train links for defining the neural triangles and the class links for clustering at each level of the tree. The use of triangles, which does not yield necessarily a Delaunay triangulation, limits heavily the performance of the network. If the basic neural unit is a Growing Cell Structure (GCS, [46]) the hierarchical version is called Hierarchical GCS (HiGCS, [47]). If, instead, the Growing Grid (GG, [48]), with decreasing learning rate and neighborhood range, is used, the algorithm is called Growing Hierarchical SOM (GH-SOM, [49]). The vertical and horizontal tree growths are controlled by using the mean quantization error, by means of two parameters whose setting is tricky, as discussed in [50]. In [50], a novel algorithm, called Growing Hierarchical Neural Gas (GHNG), is proposed, which exploits the Growing Neural Gas (GNG, [51]) as basic neural unit. In [50] it is proved it has a better performance than GHSOM. Another algorithm performing AT is the Dynamically Growing Self Organizing Tree (DGSOT, [52]) which is an enhanced dynamic version of the Self Organizing Tree Algorithm (SOTA, [53]), which builds a binary hierarchy.

In the following sections, the last two algorithms (GHNG and DGSOT) are discussed, along with the proposed AT approach (GH-EXIN), highlighting similarities, differences and novelties. More specifically, in Sec. 7.2 the GHNG and DGSOT algorithms are briefly summarized. In Sec. 7.1 the novel neural network is presented, together with considerations on the required user-dependent parameters and the analysis of the time complexity. Results and comparison of the three algorithms on the synthetic experiments are reported and discussed in Sec. 7.2. Real-world applications are presented

in Secs. 7.3 and 6.4, in a typical problem in video recognition and in two-way clustering, respectively.

6.3 State-of-the-art of Neural Clustering

6.3.1 GHNG

The Growing Hierarchical Neural Gas (GHNG, [54]) is a hierarchical self-organizing neural model, which learns a tree of Growing Neural Gas (GNGs) where each subgraph (i.e. each GNG) is the child of a processing unit (a.k.a. neuron) of the upper level. At each hierarchical level a GNG network is created by using the Voronoi set of the father neuron. Initially, a GNG network is composed of two neurons joined by a connection. At first, each one is initialized to a random sample from the Voronoi set of the father unit. When a new sample x_t is presented to the GNG network, the algorithm finds the nearest neuron w_q and the second nearest one w_s to the sample, and it increments the age of all the edges departing from w_q. The error variable associated to the winner e_q is incremented with the squared Euclidean distance between the winner itself w_q and the new sample x_t:

$$e_q(t+1) = e_q(t) + ||w_q(t) - x_t||^2 \tag{6.1}$$

Then, all the direct topological neighbors of w_q (i.e. neurons joined to w_q with an edge) are updated with step size ϵ_n, and w_q itself is updated with step size ϵ_b:

$$w_i(t+1) = (1-\epsilon)w_i(t) + \epsilon x_t \tag{6.2}$$

If the winner w_q and the second winner w_s are connected with an edge, the age of this edge is set to zero; otherwise, if they are not joined, an edge is created linking the two neurons. Finally, all the edges older than age a_{max} are removed, as well as all the isolated neurons (if any). At this point, if the current time step t is a multiple of a user-dependent parameter λ, then a backup copy of the GNG network is saved. Then the algorithm selects the neuron having the largest error w_r and, among its neighbors, the one having the largest error w_z. A new neuron w_k is then created halfway between w_r and w_z, decreasing the quantization error of the GNG graph. Otherwise, if

the time step t satisfies the relationship:

$$\text{mod } (t, 2\lambda) = \left\lfloor \frac{3}{2}\lambda \right\rfloor \tag{6.3}$$

a check is done in order to evaluate if the growth process has resulted in an improvement of the quantization error. Given the mean quantization of the last backup copy MSE_{old} and of the current GNG network MSE_{new}, the backup graph is restored if:

$$\frac{MQE_{old} - MQE_{new}}{MQE_{old}} < \tau \tag{6.4}$$

where $\tau \in [0,1]$ is a user-dependent parameter. If the above relationship is satisfied, then the graph enters the convergence phase. Thus, for high values of τ the quantization error improvement must be significant in order to continue the growth phase. Finally, the error variables are decreased by multiplying them by a user-dependent constant d. If the maximum number of time steps is reached the algorithm stops. Otherwise another sample is presented to the GNG graph. When the learning process ends, then the Voronoi set of each neuron is used to train another GNG network recursively (see Fig. 6.1). The vertical growth in a branch of the hierarchy stops when the deepest GNG enters the convergence phase having only two neurons. This leaf node is pruned because it is too small to represent any relevant distribution.

6.3.2 DGSOT

The Dinamically Growing Self-Organizing Tree (DGSOT) [52] is a self-organizing neural network. Similarly to all clustering algorithms presented in this work, it builds a hierarchical divisive tree.

It is an enhanced version of the Self Organizing Tree Algorithm (SOTA) [53]: basically, DGSOT adds to each vertical growth performed in SOTA also a horizontal growth which allows to better determine cluster partitioning at each level. SOTA, in fact, builds a binary tree: each cluster, if partitioned, is only split into two parts, but this is a very limiting approximation.

The type of tree built by DGSOT is slightly different from the one built by GHNG: in the latter, each node in the tree seems to represent the training of an associated neural network (GNG), which is composed of many neurons. In this case, instead, each node effectively represents a neuron. In the following,

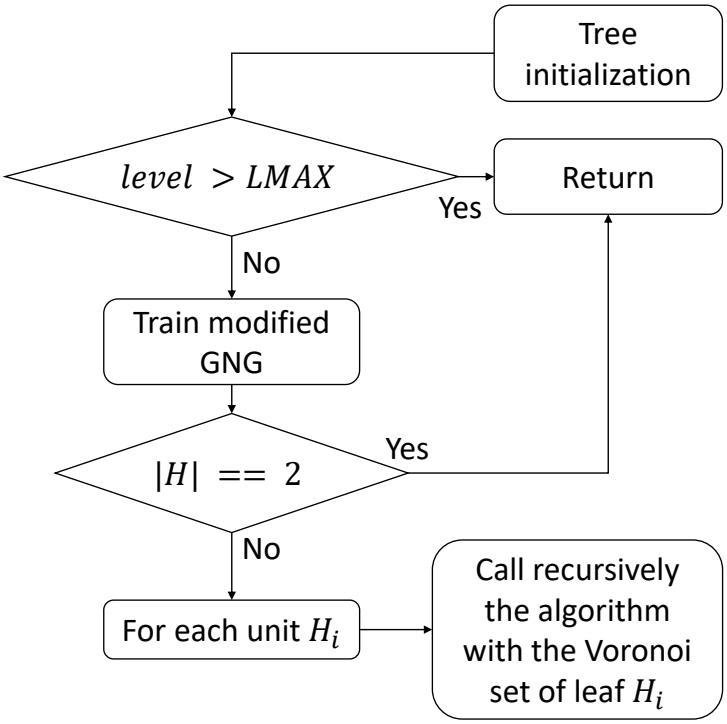


Figure 6.1: GHNG flowchart.

hence, the terms "neuron" and "node" will be used alternatively, as, at least in this context, they are interchangeable.

The tree initialization consists in the assignments of all data to the root node and in its positioning as the centroid of the dataset.

Then, while there exists at least a leaf whose heterogeneity is higher than a threshold T_R, a vertical growth is performed on this leaf. The heterogeneity of a node is defined as the average distance of the data to the neuron reference vector. Two descendent nodes of the current node are created and their reference vectors are initialized with father node's reference vector.

A learning process, in general, is the presentation of all data (epoch) for a few times to neurons for them to learn. The learning, basically, consists in the assignment of data to winner neurons and in the following reference vector adaptation. In this particular case, data previously assigned to the father node, are presented through the K-level up Distribution (KLD) mechanism. It selects potential winner neurons among all the leaves belonging to the sub-tree starting from the K-ancestor node, the ancestor node K-level above father node, where K is a parameter of the network. This mechanism allows

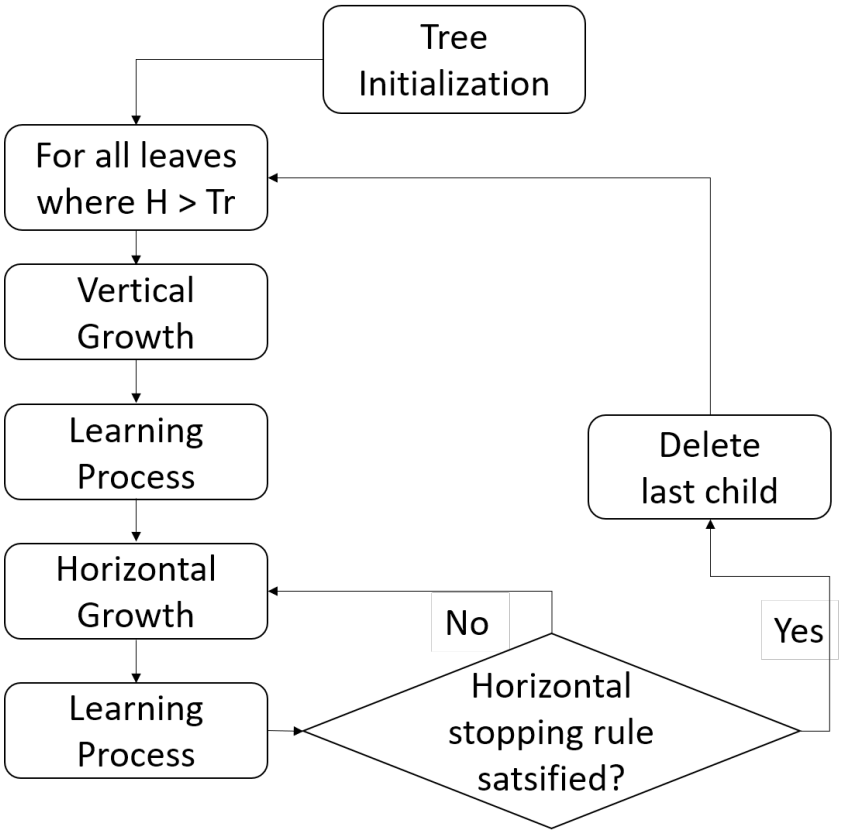


Figure 6.2: DGSOT flowchart

improperly clustered data in early hierarchical levels to be re-evaluated during later at lower layers. The winner, as commonly done, is chosen as the nearest node. Weight adaptation, instead, occurs not only for the winner but also for its neighbours, in a *winner-take-most* strategy. Neighborhood is defined as the union siblings and father node. For both winner node and neighbor nodes, reference vectors are updated according to the following function:

$$\Delta w_i = \phi(t) \cdot (x - w_i) \tag{6.5}$$

where w_i is the reference vector of the winner node, x is the datum and $\phi(t)$ is defined as:

$$\phi(t) = \alpha \cdot \eta(t) \tag{6.6}$$

where α is a user-dependent parameter which differs according to node taken into consideration, and, in general, is close to 1 for the winner, smaller for the siblings and close to 0 for the father node; $\eta(t)$, at last, is a function of

the time t which represent the number of times a neuron is selected as winner node: a good choice may be $\eta(t) = 1/t$

The learning process is repeated until the relative heterogeneity of all child nodes compared to the previous epoch is less than a user-dependent parameter T_E.

After the learning process a horizontal growth is performed. This kind of growth consists in the addition of a node to the current group of child nodes. It is always followed by a further learning process in which the current group of nodes finds the correct quantization associated to the current number of neurons. These two steps are repeated until the current number of neurons is found. After the learning process, in fact, a cluster validation test is performed in order to check the quantization error of the current neural network. In order to do that, DGSOT calculates the Cluster Separation (CS) as:

$$CS = \frac{E_{min}}{E_{max}} \tag{6.7}$$

Where E_{max} is the maximum distance between two of the current neurons, while E_{min} is the minimum distance. In case CS is above a given threshold T_c, user-dependent parameter, the process is repeated - i.e. a horizontal growth is performed again followed by a learning process. Otherwise, the last child is deleted and previous configuration is restored through another learning process.

6.4 Biclustering

Biclustering was introduced in the 60's, but has been properly defined only by Cheng and Church in 2000 and is also known as two-way clustering or manifold (subspace) clustering, [55]. As previously introduced, this technique has been chosen as it allows to perform grouping on a reduced dataset, overcoming the high dimensional problem. Nonetheless, this is not its only quality.

Basically, clustering can be applied to either the rows or the columns of the data matrix, separately. Biclustering, instead, performs clustering in both dimensions simultaneously. In this work it is achieved by alternating both row and column clustering on projected data derived from the previous steps. Compared to clustering, biclustering has several advantages since it groups items based on a subset of the features so that it does not only perform grouping but also discovers the context (subspace) in which the groups are

found. Furthermore, the projection of the biclusters into the features or the samples space allows to analyze the results as grouping of samples or features, respectively. In this work, biclusters are projected into the sample space in order to discover in which conditions - i.e. for which individuals - different genes coregulate.

In fact, common requirements in analyzing gene data are the grouping of genes according to their expression under multiple conditions (tissues) and the grouping of conditions based on the expression of a number of genes. These can be achieved by using clustering techniques. However, many activation patterns are common to a group of genes only under specific experimental conditions. Indeed, subsets of genes are coregulated and coexpressed only under certain experimental conditions, but behave almost independently under other conditions. Finding these local expression patterns is the goal of biclustering [34] and is the key for class discovery which in this specific case means uncover unknown genetic pathways.

Biclustering searches for biclusters with constant values, with constant values on rows or columns and with coherent values, respectively. It can be proved that the rank of the corresponding submatrices is less than or equal to three in the noiseless case. Hence, the numerical rank can be used as a figure of merit of the quality of the bicluster. The H_{cc} index, introduced by Cheng and Church [55], is used to control the quality of the bicluster as it also takes into account the noise in data. It is expressed as:

$$H_{cc} = \frac{\sum_i^{N_r} \sum_j^{N_c} r_{ij}^2}{N_r N_c} \tag{6.8}$$

where N_c represents the total number of columns of the matrix, N_r represents the total number of rows and $r_{i,j}$ is the residue, which is calculated as:

$$r_{ij} = a_{ij} - \frac{\sum_k^C a_{ik}}{C} - \frac{\sum_h^R a_{hj}}{R} + \frac{\sum_i^R \frac{\sum_j^C a_{ik}}{C}}{R} \tag{6.9}$$

The terms $a_{i,j}$ are the elements of the matrix (rows and columns represent faces and descriptors). C and R are the number of columns and of rows of the bicluster at hand, respectively. The second term is the average value of the ith row, the third term is the average value of the jth column, while the last one is the average value of the whole bicluster. This index decreases as the values in the bicluster tend to be constant, differing for a constant on the rows or a constant on the columns. It goes to zero for the trivial *1x1* bicluster. This fact implies additional controls on the biclusters in order to avoid this drawback.

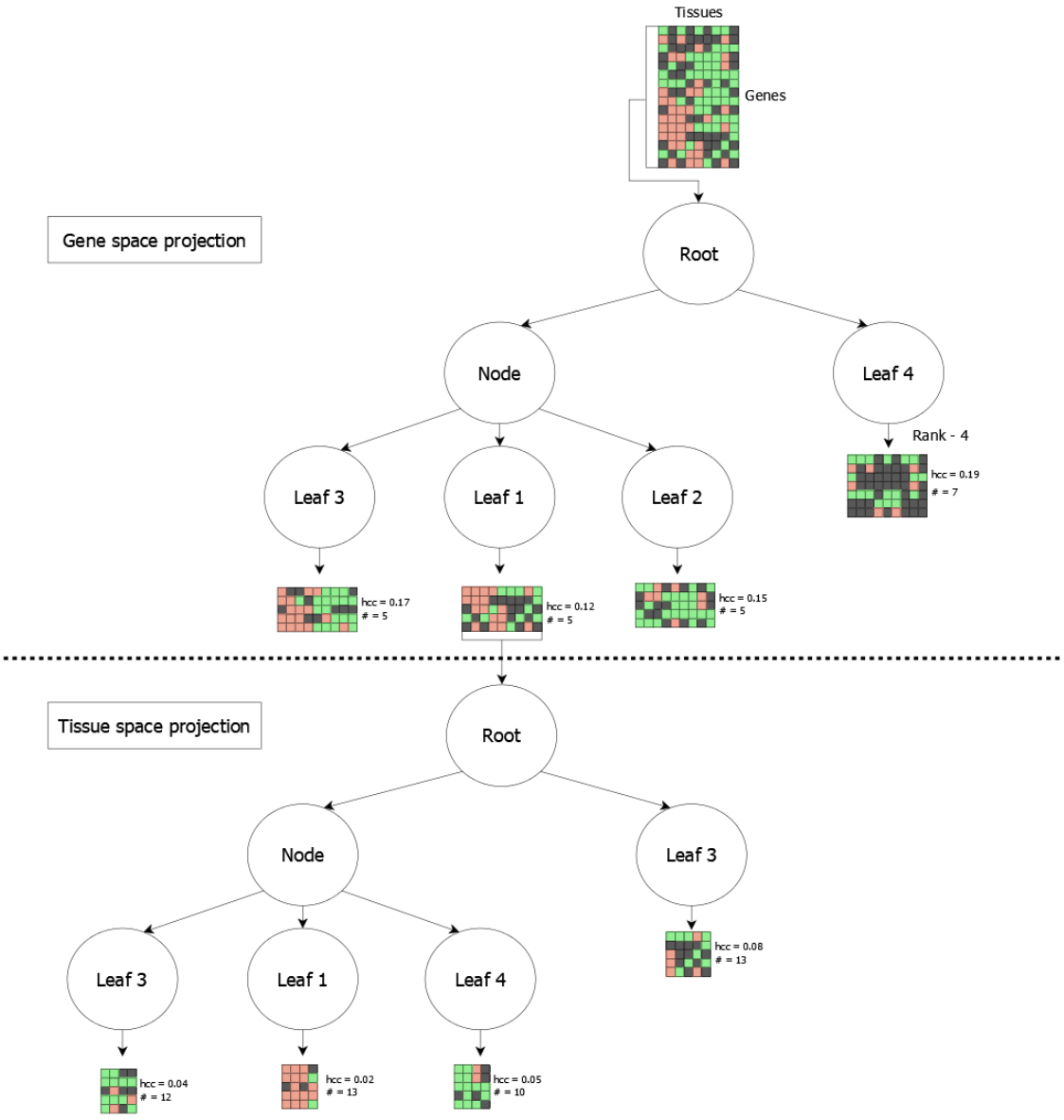


Figure 6.3: Neural Biclustering Framework

6.4.1 Neural Framework

In order to detect biclusters in the gene expression matrix, gene and tissue clustering are alternated. The preferable type of clustering in this case is a hierarchical divisive clustering: it easily allows to select the desired cluster resolution level at each iteration, by simply stopping the network when a certain height in the hierarchy has been reached. This is fundamental in biclustering because it allows to alternate the clustering in the two spaces

several times until the best biclusters are found. As shown in Fig. 6.3, indeed, a first a hierarchical clustering is achieved on genes in the tissue space, because it is the lowest dimensional space (in order to avoid the curse of dimensionality, which cannot be avoided if working on the gene space). Then another hierarchical clustering is performed on the tissues in the space of the genes associated to the best leaves produced in the first step (reduced gene space, as an orthogonal projection from the original space). The best leaves of the second step reduce the tissue space for the genes selected after the first step. This corresponds to another orthogonal projection. Resuming, each cluster (leave) decreases the dimensionality of the problem for the subsequent clustering, whose leaves yield a further dimensionality reduction. Considering that clustering implies a feature selection, this can be viewed as an orthogonal projection of the vectors. Indeed, selecting only some components results in setting the other components to zero. Considering that the basis is canonical, it corresponds to an orthogonal projection into the reduced subspace (cluster). These two steps, which pseudocode is illustrated in Alg. 1, are repeated (alternated projections) until bicluster candidates are identified.

The growth of the tree is controlled by the index H_{cc}, by simply modifying the stop criteria of the chosen algorithm. However, as seen before, the index tends to zero as the cardinality of the leaves decreases. In order to avoid trivial biclusters, before each clustering step, a check on the minimum number of data in the leaf (Cmin) is performed both in the tissue space and in the gene space (additional check). The choice of the quality index depends on the goal of the analysis. Other indices can be added (e.g. an index about the shape of the cluster) or replace H_{cc}. However, this choice remains basically heuristic and is an open problem.

Resuming, the parameters needed by the biclustering algorithm are the minimum cardinalities of leaves for both dimensions i.e. both in the number of genes grouped and in the number of tissues. These parameters, together with the bicluster quality index (H_{cc}), control the search and require a deep analysis, which, however is out of the scope of the work.

Algorithm 1 Biclustering Pseudocode

```
1: biclustering:
2: Clustering on genes
3: for all leaves do
4:     if leaf.cardinality ≤ Cmin1 then
5:         skip leaf
6:     else
7:         Clustering on the tissues of the leaf (projection)
8:         for all leaves do
9:             if projectedLeaf.cardinality ≤ Cmin2 then
10:                skip leaf
11:            else
12:                save projected leaf
13:                goto biclustering
14:            end if
15:        end for
16:    end if
17: end for
18: return
```

Chapter 7

GH-EXIN

7.1 The GH-EXIN Hierarchical Tree

The proposed approach builds a divisive hierarchical tree in an incremental and self-organized way. It is data driven (self-organization), in the sense that the final tree is automatically estimated. Also, it does not require a predefined number of units and levels (incremental with pruning phase). The resulting tree is neither binary nor balanced, because of its dependence on data. Both the GH-EXIN (Figs. 7.5 and 7.6), GHNG and DGSOT algorithms follow these criteria.

7.1.1 Tree Building

The hierarchical divisive clustering algorithms build a tree starting from a root node. By means of vertical and horizontal growths, successive splits are determined. These splits correspond to the transformation of the corresponding leaf into a node, called father neuron, whose sons are its associated leaves. For each father neuron a neural network (i.e. a basic neural unit) is trained on its corresponding Voronoi set, i.e. the set of data represented by the father neuron. The sons are then the neurons of the associated basic neural unit and determine a subdivision of the father Voronoi set. For each leaf the procedure is repeated.

Root Node

Both DGSOT and GH-EXIN associate the whole data set to a fictitious neuron (a.k.a. root node). The first basic neural unit is then trained on

the Voronoi set of this fictitious unit. Conversely, GHNG does not have any fictitious father, all nodes created at the first layer are orphans (they do not originate from a father neuron). It could be argued that GHNG builds a forest other than a single tree.

Vertical Growth

Vertical growth is the process in which a leaf becomes a node and a deeper layer is added. In these algorithms, it is always required the creation of a *seed*, i.e. a pair of neurons, which represents the starting structure of a new basic neural unit. This kind of growth is exploited as long as a higher resolution is needed. In order to evaluate whether a vertical growth is necessary, all algorithms check if the quantization error of the basic neural unit is below a user-defined threshold. DGSOT checks data heterogeneity, defined as the average distance of the data to the neuron reference vector. GHNG, instead, checks whether the number of levels in the hierarchy exceeds a user-dependent parameter MAX_Level. GH-EXIN simultaneously checks both data heterogeneity using a task-dependent index, say H_{max}, and the data cardinality (min_{card}), i.e. the size of the leaf Voronoi set. The H index is based on the quantization error and determines the quality of the clustering. Several choices are possible, in general depending on the application. In this paper the H_{cc} index [55] has been chosen. It is designed for biclustering problems, but, here, it is extended to two-way clustering. Its description is given in Sec. 6.4. Considering the characteristics of the biclusters that can be found, it can also be used for normal clustering, as seen in Sec. 7.2.

Horizontal Growth

Horizontal growth refers to the addition of further neurons to the initial seed, i.e. the creation of siblings. This method allows to expand a layer of the tree and build more complex hierarchical structures other than binary. This process is performed by all algorithms through the respective neuron creation mechanism later described (see Sec. 7.1.2).

7.1.2 sG-EXIN

The basic neural unit is intended as the neural network chosen for the clustering of the input data. All these methods use a basic neural network for the processing of each leaf. GH-EXIN is based on the stationary variant of G-EXIN [56], say sG-EXIN. GHNG uses a variant of GNG, while DGSOT,

deriving from SOTA, exploits as basic neural network an enhanced version of SOM. All these neural units do not employ any fixed topology (the induced topology is generated in the linking phase). Neural networks are composed of units called neurons, which are represented by weight (a.k.a. reference) vectors. As an abuse of language, the terms neuron and weight vector are used with the same meaning. With regard to training, both sG-EXIN and DG-SOT are based on the idea of *epoch*, which is the presentation, in a random order, of the complete training set. After each epoch controls are made for the horizontal growth by means of user-dependent parameters: H_{perc} (GH-EXIN), T_E (DGSOT). However, this is not the classical batch learning, which requires the weights to be updated after the presentation of the whole batch. For these two neural networks, instead, weights are updated or created at each iteration (data presentation). Also, GHNG learns at each iteration, but it is not clear if it requires epochs. Indeed, its algorithm does not use epochs; however, in the examples in [50], epochs are mentioned. There are some pros and cons with the choice of *epoch-learning*. Despite the fact that it lowers the training cost (GHNG is the fastest one), the use of epochs before controls implies the exploitation of the complete batch, which means all information is used for building a tree. It must be highlighted that the onset of the hierarchical tree is fundamentally a static problem, thus requiring the processing of the whole database for an accurate tree.

Neuron Creation

The basic neural units use incremental neural networks, i.e. they have a variable number of neurons (driven by data), achieved by the mechanism of neuron creation and pruning. Both DGSOT and GHNG decide in advance when to create a new neuron: DGSOT by considering the batches of the whole set of samples, at the end of an epoch, GHNG every λ (user-dependent parameter) iterations. On the contrary, for GH-EXIN, the creation of a neuron, is related to a novelty test: if the existing neurons are not able to describe the new data, say x_i, a new neuron is created. The test approach is less rigid because it is only driven by data. As discussed in the following, the choice of the test is not trivial.

Semi-Isotropic Region of Influence The novelty test requires, in general, a model of the region of influence of a neuron in the input space. All existing algorithms determine, in a way or another, a threshold value representing the radius of an hypersphere which models this region. An exhaustive

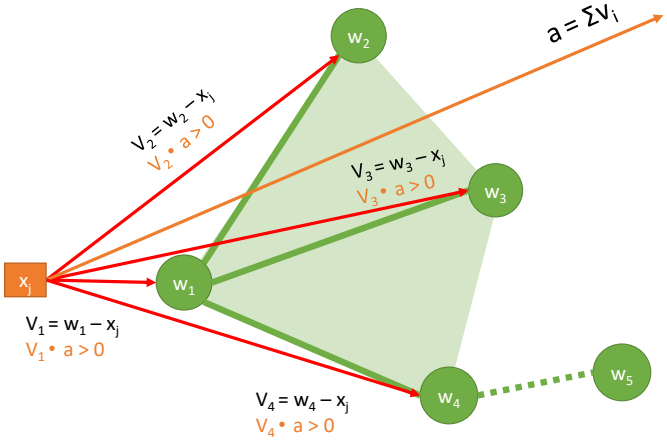
description can be found in [57]. This model is isotropic, in the sense that it does not take into account the orientation of the vector connecting the new data to the winner, but only its norm. As a drawback it does not consider the topology of the data manifold of the winner Voronoi set. The GH-EXIN proposed approach considers, at the same time, the shape (anisotropic criterion) and the extent (isotropic criterion) of the neuron neighborhood (determined by the linking phase, see Sec. 7.1.2). The anisotropic criterion consists of the analysis of the convex hull built on the neighbor neurons of the winner as explained in Fig. 7.1 and in more detail in [56]. In case x_i lies inside the convex hull, it is assigned to the winning neuron (w_γ), as shown in Fig. 7.2, and the weight adaptation is performed (see Sec. 7.1.2). Otherwise, the isotropic criterion is applied to check if the data is really novel w.r.t the existing units. The neuron isotropic threshold, say T_γ, is computed as the average of the Euclidean distances between the winning neuron (w_γ) and its N neighbors (w_i):

$$T_\gamma = \frac{1}{N} \sum_{i=1}^{N} ||w_\gamma - w_i||^2 \tag{7.1}$$

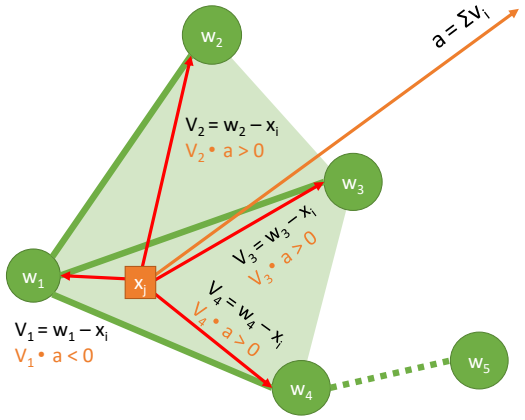
In case x_i is farther from the winner than T_γ, a new neuron is created on the data. Otherwise, it is assigned to the winner, and the weights are updated according to the mechanism described later (see Sec. 7.1.2).

The use of an anisotropic criterion is justified by the need of representing better the data manifold, which is not guaranteed by using only the isotropic threshold. Fig. 7.2 shows one of these cases. It is proved in [56] that this approach better models the border of the data manifold.
On the contrary, GHNG and DGSOT do not require a region of influence, but keep adding neurons as far as the quantization error (a.k.a. *cluster separation* in [52]) does not fall below a user-dependent threshold. GHNG computes the quantization error from data, by considering the average distance between the reference vectors and their Voronoi sets, while DGSOT from network topology, by considering the rate between the minimum and the maximum distance among neurons. As a consequence, GHNG and DGSOT are prone to the choice of a predetermined parameter, instead of exploiting the neighborhood topology, as in the case of GH-EXIN.

Lonely Neuron A neuron with no edges is named *lonely neuron*: in Fig. 7.2 the neuron at the bottom with no edges is a lonely neuron. Since DGSOT does not make use of edges, all its neurons are lonely. On the contrary, both GHNG and GH-EXIN exploit the concept of lonely neurons to determine



(a) If all the dot products between the vectors v_i from the new data x_j and the neuron w_i, and the vector a, sum of the v_i's, have equal sign, then the new data is outside the convex hull of the winner (w_1)



(b) If all the dot products, between the vectors v_i from the new data x_j and the neuron w_i, and the vector a, sum of the v_i's, have not the same sign, then the new data is inside the convex hull of the winner (w_1)

Figure 7.1: A new data x_j is presented to the sG-EXIN neural network composed of four connected neurons

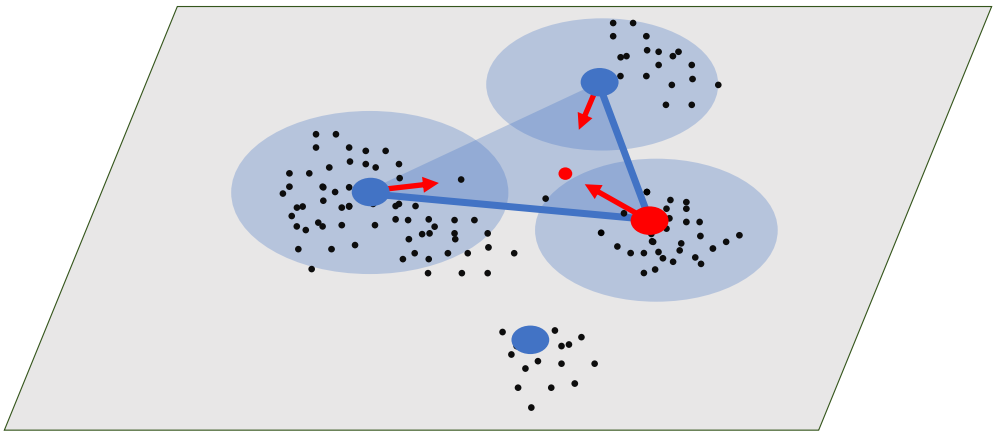


Figure 7.2: The anisotropic criterion: if a new data (small red dot) is presented and lies outside the hypershphere centered on the winner (big red dot), but within the convex hull created on its neighbours (blue connected nodes), it is also assigned to the winner. Furthermore, according to SCL both the winner and its neighbours move towards it.

leaf neuron pruning. In both algorithms, a neuron may become lonely in case all its edges are pruned. However, in GHNG neurons already have connections when created. In GH-EXIN, instead, new neurons are created lonely. Connections may be generated only in the next iterations.

Soft-Competitive Learning

The weight computation (training) is based on the Soft Competitive Learning (SCL) paradigm [58], which requires a winner-take-most strategy and the network topology, whose setting is explained in Sec. 7.1.2. The closest neuron to the new data is named (first) winner. The set of potential winners differs according to the clustering algorithm. In both GHNG and GH-EXIN, only neurons belonging to the same basic neural unit are competitive in the learning phase. In DGSOT, instead, all leaf neurons belonging to the subtree below the K-ancestor of the current node are competitive.
At each iteration, both the winner and its neighbors change their weights but in different ways as shown in Fig. 7.2. The winner w_γ and its direct

topological neighbors w_i are moved towards x_j by fractions $\alpha_\gamma(t)$ and $\alpha_i(t)$ (learning rates), respectively, of the vector connecting the weight vectors to the data. The update is given by:

$$\Delta w = \alpha(t) \cdot (w - x_j) \tag{7.2}$$

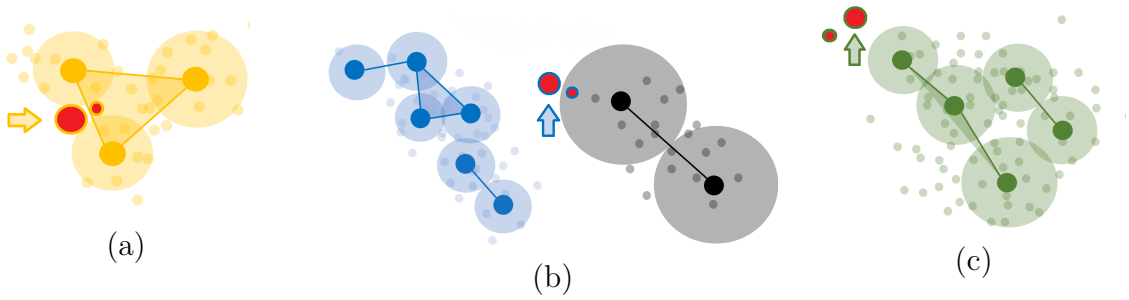
where $\alpha(t) = \alpha_0/t$, and α_0 is a user dependent parameter, higher for the winner and smaller for the neighbours, and t is the number of times a neuron wins (conscience). Regarding the weight adaptation, the difference among the three algorithms consists on the neighborhood determination. DGSOT considers as neighbors all neurons belonging to the same basic neural unit, plus the father neuron. On the contrary, the neighborhood of both GHNG and GH-EXIN is composed of only those neurons connected to the winner through an edge. Hence, GHNG and GH-EXIN exploit a more local information.

Edge Creation and Network Topology

Edges are exploited in order the determine the topology (neighborhood) of a network. Both GH-EXIN and GHNG use the Competitive Hebbian Learning (CHL) rule [58] for creating the neuron connections: each time a neuron wins (first winner), an edge is created, linking it to the second nearest neuron (second winner), if it does not exist yet. If there is already an edge, its age is set to zero. Furthermore, in both GHNG and GH-EXIN the same aging procedure is applied: the age of all links emanating from the winner is incremented by one. In case a link age is greater than the age_{max} scalar parameter, it is eliminated (pruned).

7.1.3 Neuron Pruning

Neuron pruning is the process through which neurons can be removed if redundant. DGSOT does not exploit any pruning technique, in the sense that the redundancy is only checked each time a neuron is added, but old neurons cannot be removed. Vice versa, GHNG and GH-EXIN remove all lonely neurons. GH-EXIN checks for lonely neurons at the end of each epoch (see Fig. 7.6), while GHNG checks at each iteration. Besides, GHNG may prune an entire set of neurons if its modified GNG enters the convergence phase. However, the onset of this phase is empirically determined.



(a) (b) (c)

Figure 7.3: GH-EXIN data reallocation: small dots represent data while big dots represent neurons. Big red dots, indicated by arrows (colored as the neurons of the same sG-EXIN), mark pruned neurons, while small red dots represent data belonging to them. Different colors represent different network units

Data Reallocation Data reallocation is the mechanism by which orphan data, i.e. associated to pruned neurons (their Voronoi sets) or lonely neurons (data coincident with the neuron) are possibly reassigned to other neurons. This mechanism is a novelty introduced in GH-EXIN (see Fig. 7.6), and is an improvement of the KLD method used in DGSOT. In fact, in GHNG, when neurons with no edges are removed, the associated data are not reallocated. In GH-EXIN, instead, all orphan data are labelled as potential outliers at the end of each epoch. For each potential outlier, GH-EXIN seeks a new winner among all leaf neurons. The data is reassigned in case it is inside the hypersphere (or the convex-hull) of another neuron within the same neural unit (Fig. 7.3(a)) or in case the nearest neuron belongs to another neural unit (Fig. 7.3(b)). If, instead, the winner belongs to the same neural unit of the pruned neuron, but the data is outside its hypersphere, the data is definitely marked as outlier and is not reassigned (Fig. 7.3(c)). This outlier identification can be useful in a lot of applications.
Resuming, only GH-EXIN and DGSOT may correct possible cluster errors made in the previous layers. Instead, this is not possible for GHNG: the advantage of the speed of its algorithm is counterbalanced by the fact that the errors always affect the final tree. As said before, the building of a tree is basically a static process. So, GHNG is less suited to hierarchical clustering than the other two techniques.

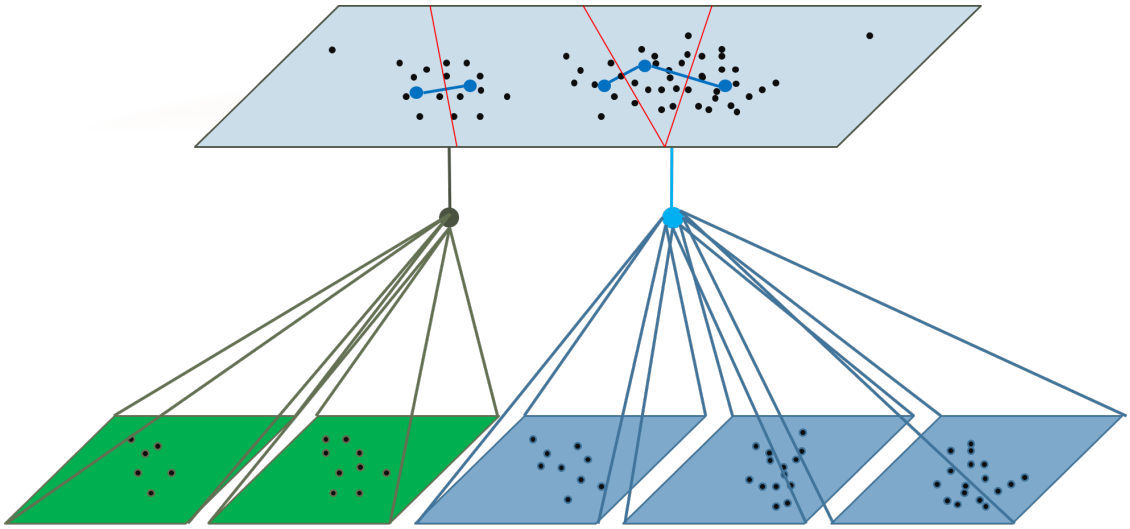


Figure 7.4: Connected Graph Test: the Voronoi regions of each neuron are represented with solid red lines; chains in the father sG-EXIN become sons

7.1.4 Connected Graph Test

Another remarkable novelty introduced by GH-EXIN consists of a possible double vertical growth. As shown in Fig. 7.5, at the end of the sG-EXIN training process, the resulting graph of the basic neural unit is analyzed by searching for connected components. If more than one connected component is detected, the algorithm tries to extract an abstract representation of data. Hence, each connected component, representing a cluster of data, is associated with a novel abstract neuron. The reference vectors of abstract neurons are placed in the centroids of the respective clusters. The tree structure is therefore modified by inserting a new layer between the leaf nodes and the father node, resulting in a double simultaneous vertical growth, as shown in Fig 7.4.
The proposed test is justified by the exploitation of the topology graph built by GH-EXIN. The estimated connected components are directly translated into the hierarchical tree through this additional vertical growth. In this sense, GH-EXIN does not only partition the data into nested Voronoi sets, but exploits its induced Delaunay triangulation, created by the CHL rule.

7.1.5 The GH-EXIN Algorithm

Resuming, for each node, an sG-EXIN neural network is trained on its corresponding Voronoi set (set of data represented by the father neuron). For

each leaf, the vertical growth is performed until either the H index of the leaf has fallen below H_{max} or the cardinality of the leaf is less the min_{card} (both are user-dependent parameters). For each epoch, the basic iteration starts at the presentation of a new data, say x_i. All neurons are ranked according to the Euclidean distances between x_i and their weight vectors. In case the data is considered new, i.e. it is both outside the convex polytope and the hypersphere of radius T_γ of the winner w_1 (novelty test), a new neuron x_{new} is created (left branch of Fig. 7.6). The initial weight vectors and neuron thresholds T_γ are given by heuristics: the novel neuron has its weight x_{new} equal to x_i, and its threshold is set equal to the w_1 threshold. No edge is created at this time: x_{new} is labelled as *lonely neuron*.

Otherwise, in case the data is not new (right branch of Fig. 7.6), the first winner w_1 and the second winner w_2 are linked by an edge (CHL), if it does not exist yet. If there is already an edge, its age is set to zero. Also, the age of all other links emanating from the winner is incremented by one; if a link age is greater than the age_{max} scalar parameter, it is eliminated (edge pruning). Reference vectors of w_1 and its direct neighbors are updated according to Eq. 7.2. Thresholds of the winner and of its neighbors are recomputed, as their position has been modified. This process is repeated for all data of the father Voronoi set. At the end of each epoch, if a neuron remains unconnected (no neighbors) or is still lonely, it is pruned, but the associated data are analyzed and possibly reassigned.

The training epochs, i.e. the horizontal growth, are stopped when the estimated H average value falls below a percentage (H_{perc}) of the H value of the father neuron.

This technique builds a vertical growth of the tree. The horizontal growth is generated by the neurons of each network. However, a simultaneous double vertical growth is possible, as specified in Sec. 7.1.4.
GH-EXIN has been developed in MATLAB. The code is freely available at [59].

7.1.6 Analysis of the User-Dependent Parameters

The user dependent parameters can be grouped according to their function in three classes: learning, hierarchy and design variables. The learning parameters handle the training of the basic neural unit. GH-EXIN uses sG-EXIN, which requires CHL and SCL. They are performed by using:

- the two learning rate constants, $\alpha_{\gamma 0}$ and α_{i0}, used to update reference vectors of the winner and its neighbours, respectively;

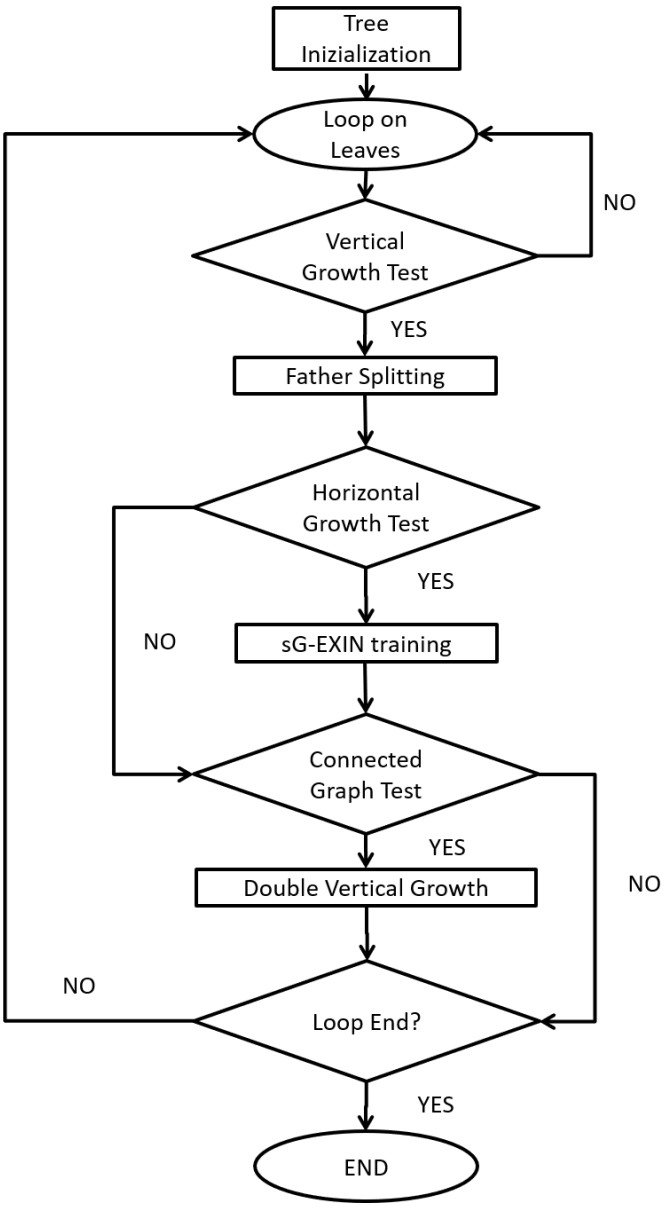


Figure 7.5: GH-EXIN flowchart

- the scalar age_{max} used for edge pruning: it has to be lowered if more edges (and neurons) have to be pruned, indirectly controlling the leaf cardinality.

Instead, GHNG requires three more parameters, λ, α, and D, which are related to the creation of a new neuron. In particular, deciding in advance when to insert it (it depends on λ) is a serious drawback. In the case of

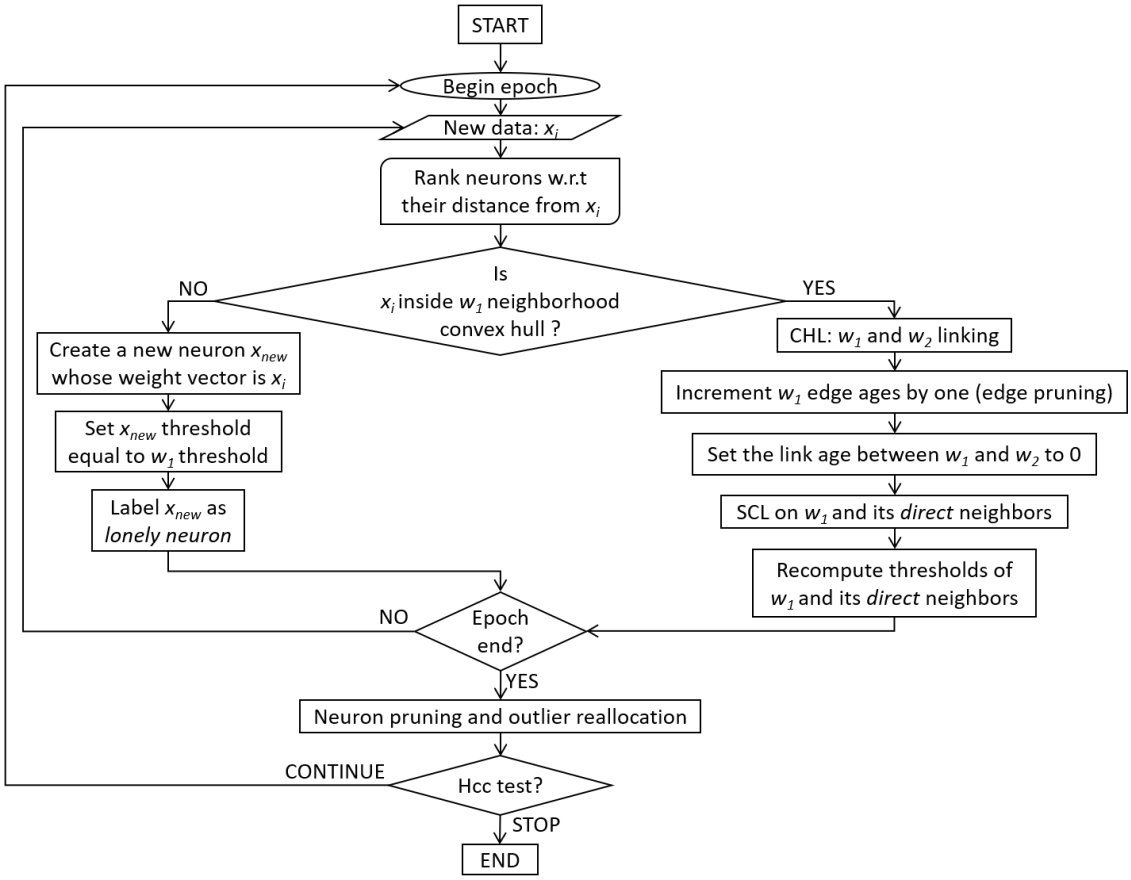


Figure 7.6: GH-EXIN: horizontal growth flowchart

DGSOT, there are three learning rates, for the winner, the parent and the siblings, respectively. Its horizontal growth can be compared to a neuron creation and is controlled by a threshold, T_E. The stop criterion for GH-EXIN depends on H_{perc} (other criteria can be used according to the application). GHNG controls the growth process by the parameter τ. It is then followed by a refinement step, terminated by a maximum number of epochs, decided in advance. DGSOT stops learning when the relative error of the entire three is less than the error threshold T_E.

The hierarchy parameters control the growth of the tree. GH-EXIN uses the H_{max}, which depends on the task. Instead, more rigidly, GHNG uses a MAX_Level threshold, which is not guided by the application at end, and can also be considered as a design variable. DGSOT controls the vertical growing using the threshold T_R for controlling the heterogeneity of any leaf.

The design parameters are not important for the hierarchical clustering

but help in deciding in advance a preferred tree depth. In this sense, they cannot be considered as relevant for the user dependent setting. GH-EXIN uses min_{card}, the minimum cardinality of leaves, used to avoid small clusters. The same holds true for GHNG, which seeks only for clusters with a cardinality more than three.

Both GH-EXIN and DGSOT have the possibility to reallocate data. However, this process is completely automatic in GH-EXIN, while DGSOT requires a parameter K.

Resuming the meaningful parameters to be set are 5 for GH-EXIN, 8 for GHNG and 7 for DGSOT. As a consequence, GH-EXIN is easier to be calibrated.

7.1.7 Analysis of Complexity

Define N as the number of data in the whole training set, d as the dimensionality of the input, J as the average number of epochs for the basic neural unit training, and k as the average number of neighbors for each neuron. Let be b as the average branching factor of the tree. Then, the height of the tree is $h = log_b M$, where M is the number of leaves in the hierarchy. For a full tree (each leaf node associated with one data), M is $O(N)$ where N is the number of data.

GH-EXIN Complexity

The computational cost of GH-EXIN can be estimated by considering the algorithm step-by-step (see Figs. 7.5 and 7.6).

sG-EXIN Iteration At each iteration in an epoch, see Fig. 7.6, an input x_i is fed to the network and the two closest neurons (weights), w_1 and w_2, are found according to their Euclidean distances from the data. Let m_i the number of neurons of sG-EXIN at the i-th iteration. Then, the distance estimation is $O(m_i d)$. If a simple min algorithm is employed, the second step (i.e. first and second minimum search) is performed in $O(m_i)$. In order to take into account an average of the cardinality of all the Voronoi sets in an horizontal growth, m_i can be safely replaced by the average branching factor b. Resuming, both steps have a complexity of $O(bd) + O(b) = O(b)$, because d is constant w.r.t. the evaluation of the algorithm complexity. Once the winner is determined, the neighborhood convex hull test is performed. It requires the identification of the convex hull of the winner, i.e. its neighbors.

For considering, in a global way, the vicinity of all neurons in the network, the average of the neighborhood cardinality, say k, is considered here. As explained in Fig. 7.1, to check if the input belongs to the winner convex hull, the vector a, sum of the difference vectors, v_i, between the winner neighborhood and the input x_i, needs to be computed; this requires $O(kd)$ operations. Then, the inner products between a and all the v_i's cost $O(kd)$ in the worst case (all comparison are needed). Resuming, the test costs $O(kd) = O(k)$, according to the previous considerations. After the novelty check, two scenarios can occur: either a new neuron is created or w_1 and its neighbors adapt their weights according to the SCL. It is easy to prove that the former costs $O(1)$ because it is just a sequence of atomic operations whose cost is, by definition, $O(1)$. The latter case, i.e. the SCL weight adaptation, is slightly more complex: the CHL linking is $O(1)$; the aging and pruning phase exactly needs $O(k)$ operations; the SCL adaptation requires $O(kd)$ because it performs a vector adjustment, k times (i.e. the size of the neighborhood), by means of adding a scaled difference vector to the weight, whose computational cost is, of course, $O(d)$; the threshold re-estimation implies to evaluate, for each of the k neurons moved by the SCL, the distances from its neighbors (i.e. $O(k^2d)$), assuming as negligible the search for the maximum neighbor distance. Resuming, it can cost either $O(1)$ in case of neuron creation or $O(k^2)$ in case of SCL weight adaptation.
In conclusion, a single sG-EXIN training iteration employs $O(b) + O(k^2)$ operations. However, it must be taken into account that the branching factor represents the number of neurons of a neural unit, and that GH-EXIN builds, by construction, a tree and not a fully connected graph. Hence, $b >> k$ and the overall cost becomes $O(b)$.

GH-EXIN Horizontal Growth The horizontal growth of GH-EXIN corresponds to the training of an sG-EXIN neural network (see Fig. 7.6), which means presenting all the father node Voronoi set to the sG-EXIN neural unit for several epochs. In other words, it implies to repeat the sG-EXIN iteration for an number of times equal to one epoch (i.e. the cardinality of the Voronoi set) and then, repeat this procedure for the necessary number of epochs. For considering, in a global way, all the horizontal growths (i.e. sG-EXIN training) of a single level of the hierarchy, the average number of epochs, J, is considered here. Furthermore, in the worst case all leaves become father nodes; then, the sG-EXIN networks are trained on input sets whose cardinality sums exactly to N. As a consequence, a GH-EXIN horizontal growth costs $O(JNb)$ (neuron pruning and outlier reallocation have a

negligible cost).

GH-EXIN Cost The overall complexity of GH-EXIN can be estimated by considering the cost of repeating a full horizontal growth (i.e. expansion of all the leaves) for all the levels of the hierarchy, that is, the height h of the tree. According to the previous considerations, $h = log_b M$, and $M = O(N)$ then the overall training is $O(b * J * N * log_b N)$. Note that both J and b are usually smaller compared to N; it implies that J and b can be considered as constants. The overall GH-EXIN complexity is then $O(N * log_b N)$.

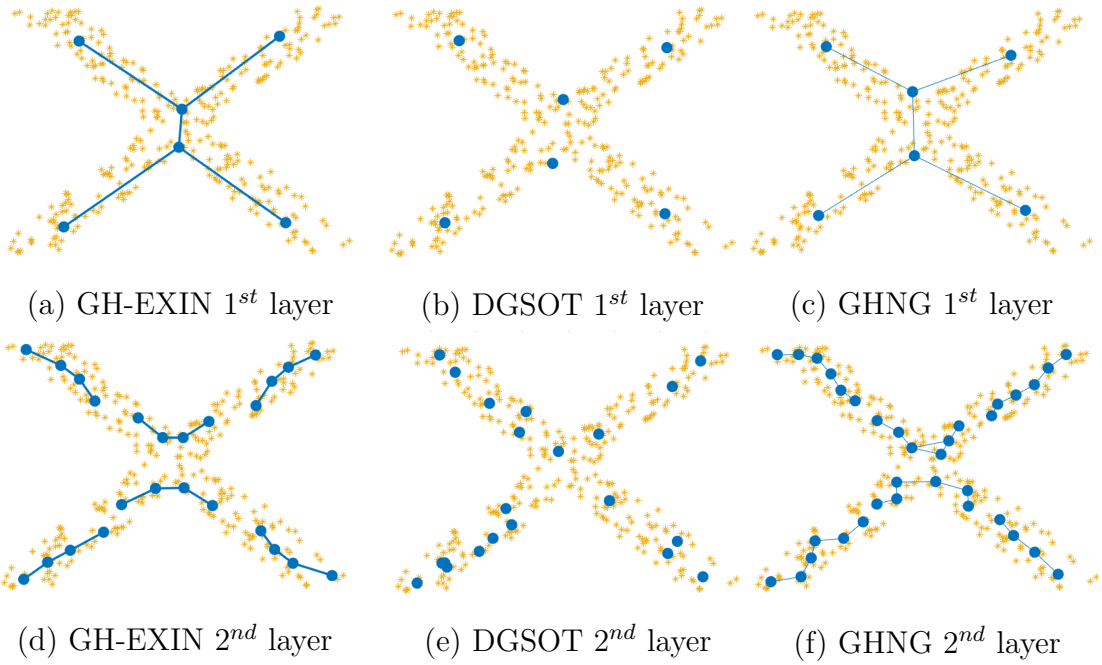
Complexity Comparison

As pointed out in [60], DGSOT has the same cost as GH-EXIN. However, the DGSOT analysis of the cost in [60] does not take into account the complexity of the horizontal growth. Indeed, both the approach in [60] based on the Minimum Spanning Tree and the approach in [52] based on the CS index, are very time consuming. Hence, the DGSOT complexity is probably underestimated.
On the contrary, GHNG is cheaper. Indeed, its cost is $O(N)$, according to our personal analysis, because there is no complexity estimation of this algorithm in the literature. However, a so simple technique prevents from building a really adaptive tree: the neuron creation does not depend on the data at hand, the tree height is predetermined (the number of levels is a hyperparameter) and the horizontal growth is only controlled by the leaf minimum cardinality. The latter is probably the worst problem, because it tends to flatten the hierarchical tree, in the sense that all the detected subgroups in a cluster are represented in the same level, even if they still contain nested levels.

7.2 Synthetic Experiments

GH-EXIN, GHNG and DGSOT are here tested on artificial datasets, for comparing their performances. At first, the same two planar datasets used in [50] have been chosen both for the analysis of their partitioning properties and for direct visual inspection. With regard to their ability in building a hierarchical tree, a third database has been expressly created.

The first two databases are randomly drawn from i) a planar uniform X-shape manifold and ii) a planar square-shaped manifold having a Beta distribution (higher density in the borders). The parameters of each network,



(a) GH-EXIN 1^{st} layer (b) DGSOT 1^{st} layer (c) GHNG 1^{st} layer

(d) GH-EXIN 2^{nd} layer (e) DGSOT 2^{nd} layer (f) GHNG 2^{nd} layer

Figure 7.7: First **(up)** and second **(down)** layers of GH-EXIN, GHNG and DGSOT on the X-shape distribution

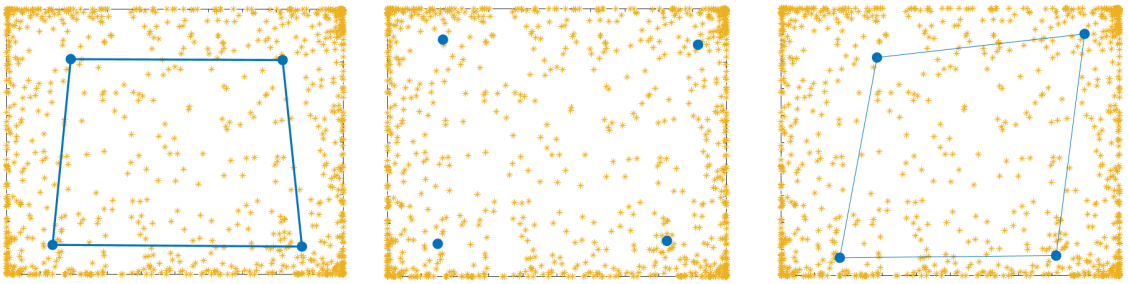
used in these experiments and in the next ones, are reported in Tabs. 7.1, 7.2 and 7.3.

For evaluating the quality of clustering, some internal indexes are used: the peak-signal to noise ratio ($PSNR$) index [50], the Davies–Bouldin index (DB) [61] and the global Silhouette value (S) [62].

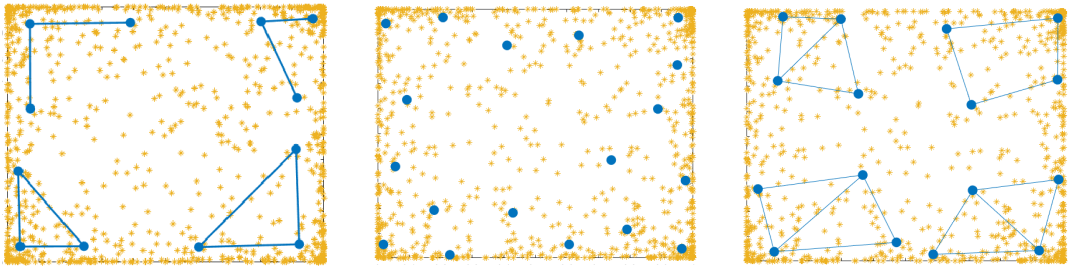
The $PSNR$ index is defined as follows (in decibel, higher is better):

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{7.3}$$

where MAX_I^2 is the squared Euclidean norm of the vector which joins the two most distant points in the input distribution support and MSE is the Mean Squared Error computed as the sum of the Euclidean distances between the reference vector of each leaf neuron and its associated data. $PSNR$ takes into consideration only the intra-cluster compactness, while ignoring the inter-cluster separation. For this reason, it is not a very accurate index of the clustering quality. However, it is here introduced because of its use in [50].



(a) GH-EXIN 1^{st} layer (b) DGSOT 1^{st} layer (c) GHNG 1^{st} layer



(d) GH-EXIN 2^{nd} layer (e) DGSOT 2^{nd} layer (f) GHNG 2^{nd} layer

Figure 7.8: First **(up)** and second **(down)** layers of GH-EXIN, GHNG and DGSOT on the square distribution

The DB index, instead, takes into consideration both aspects, and is defined as follows:

$$DB = \frac{1}{N} \sum_{i=0}^{N} \max_{j \neq i} \frac{RMSE_i + RMSE_j}{D_{i,j}} \tag{7.4}$$

where $RMSE_i$ is the Root Mean Squared Error for the ith cluster, $D_{i,j}$ is the Euclidean distance between the centroids of the ith and jth clusters and N is the number of clusters. The lower the value, the better is DB.

Also the S index takes into account both the intercluster and the intracluster distances and is computed as follows:

$$S = \frac{1}{C} \sum_{i=1}^{C} \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{7.5}$$

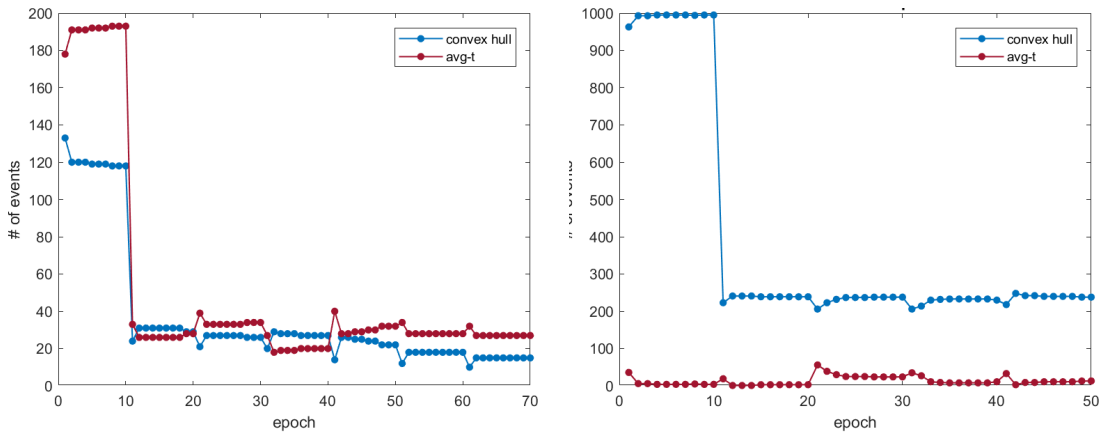
where $a(i)$ is the average distance of the ith point from the points in the same cluster, while $b(i)$ is the minimum among the average distances of the ith point from the points in the other clusters and C is the cardinality of the current dataset. The Silhouette index, in general, is defined for each point in the dataset. Hence, the average value is considered. While DB aims at

identifying sets of clusters that are compact and well separated, the S index is more suitable for estimating if, on average, samples are correctly assigned to the nearest neighbouring cluster.

On the X-shape distribution (Fig. 7.7), at the first layer, the three algorithms have learnt approximately well the manifold. With regard to the second level of the hierarchy, GH-EXIN uses less neurons than GHNG for covering the manifold. Furthermore, the proposed approach represents the symmetry of the manifold using symmetric branches composed of the same number of neurons. On the contrary, GHNG neurons and edges do not respect the symmetry (branches have not the same number of neurons). Also there are some higher neuron densities in branches. More specifically, DGSOT yields the worst results in terms of symmetry (e.g. the overlap of three neurons in the SW branch of the manifold), because of the absence of connections. For instance, the SW branch and the NE branch have eight and three neurons, respectively.

Bar plots in Fig.7.15 report some statistics about the algorithms and the quality of the clustering. Results have been validated by running all the algorithms 10 times for each data distribution and the bar plots report the mean value and the standard error mean. With regard to the X-letter experiment, GH-EXIN uses less neurons on average and takes a few more seconds to end. In this experiment, the quality of GH-EXIN and GHNG clusterings is similar for all indexes, except for DB, which is slightly worse for GH-EXIN. They both overcome DGSOT results (S and DB). Nonetheless, GHNG and DGSOT are less stable than GH-EXIN in terms of number of neurons created. While this does not seem to affect GHNG performance on average, DGSOT DB index, instead, changes too much at each run.

On the square dataset (Fig. 7.8), with regard to the first layer, GH-EXIN and DGSOT distribute the four neurons in a more symmetrical way with respect to GHNG. However, the GH-EXIN distribution represents better the rectangle manifold (isosceles trapezoid). Concerning the second layer, GH-EXIN uses less neurons than GHNG and DGSOT for covering the manifold. Further, the proposed approach represents the symmetry of the manifold, correctly placing nodes along the sides of the squares, according to the point distribution, while mostly ignoring the emptier central part. On the contrary, GHNG and above all DGSOT place many neurons also in the central part and do not respect the symmetry of the dataset. From a quantitative point of view, GH-EXIN is the best algorithm for all indexes but $PSNR$, while using far fewer neurons. Nevertheless, also in this case it takes longer to terminate than DGSOT and, above all, GHNG. On this dataset, algorithms



(a) X-shape distribution (b) Square distribution

Figure 7.9: Number of times the two novelty test approaches are called during the training phase (see Paragraph 7.1.2 and Fig. 7.2). The blue line represents the number of calls to the convex-hull technique (see Fig. 7.1), while the red line refers to the isotropic threshold criterion (see Equation 7.2). Each node training lasts ten epochs. The first ten epochs refer to the root node, while the following ones to the second layer nodes

seems to be quite stable on average.

With regard to the GH-EXIN novelty test, Figs. 7.9(a) and 7.9(b) show the importance of the convex-hull mechanisms with regard to the isotropic threshold for explaining the X-shape and the square distribution, respectively. The convex-hull criterion is extensively exploited for the square distribution because of the importance of the border.

Resuming, in both experiments GH-EXIN yields the best clustering, as confirmed by the visualization, and in terms of the S and DB indexes. It requires fewer neurons, but is more time consuming.

7.2.1 Hierarchical Synthetic Experiment

The previous experiments highlight the performance in quantization of the three algorithms for each level. However, these techniques have been conceived not for partitional, but for hierarchical clustering. The proposed experiment checks for the quality of the estimated tree, by using, as a ground truth, a predefined hierarchical clustering. At this aim, a dataset composed of two Gaussian mixture models has been devised: the first model is made of three Gaussians, the second one of four Gaussians, as shown in Fig. 7.10.

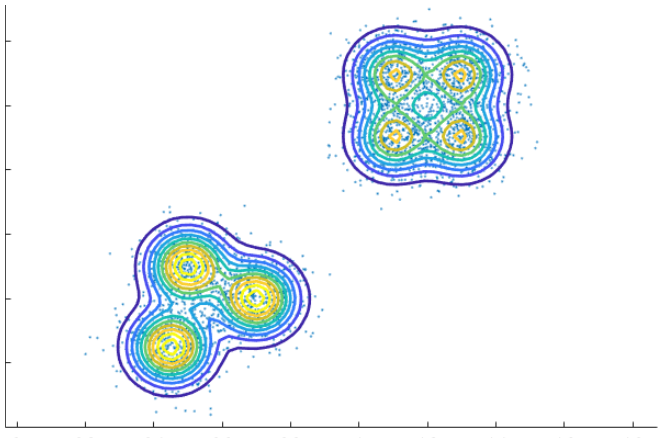
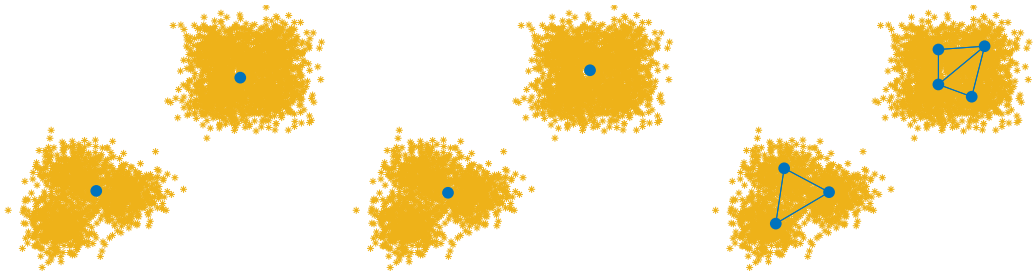
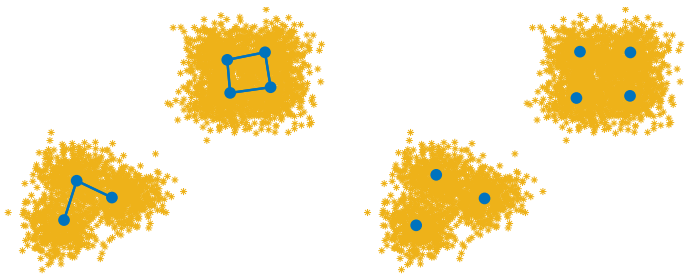


Figure 7.10: Two mixtures of Gaussians: data and contours



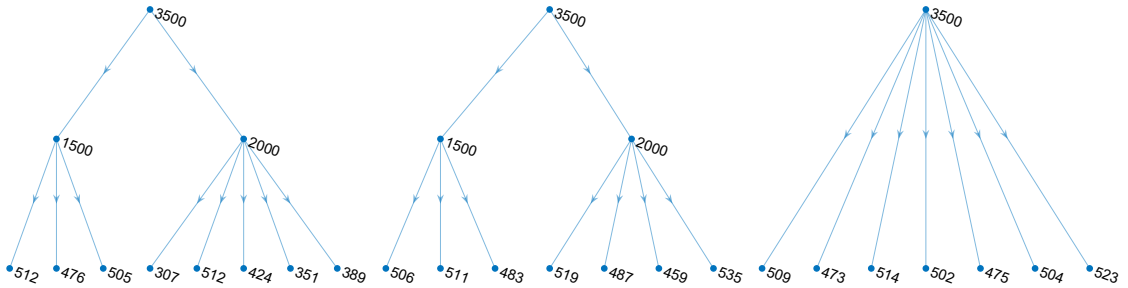
(a) GH-EXIN 1^{st} layer (b) DGSOT 1^{st} layer (c) GHNG 1^{st} layer



(d) GH-EXIN 2^{nd} layer (e) DGSOT 2^{nd} layer

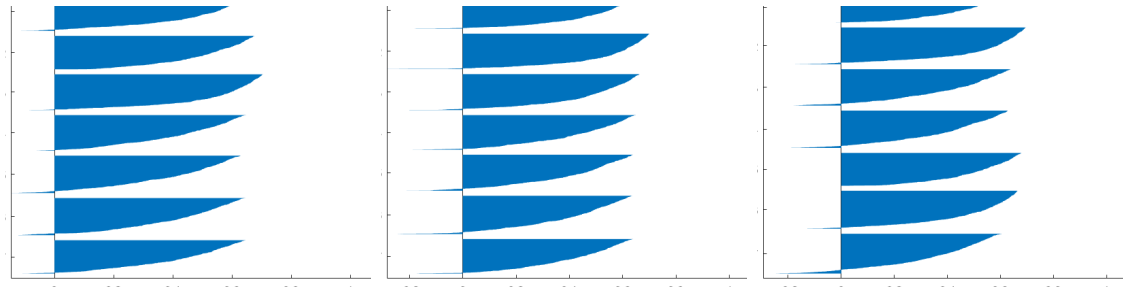
Figure 7.11: First **(up)** layers of GH-EXIN, DGSOT and GHNG and second **(down)** layers of GH-EXIN and DGSOT on the Gaussian distribution; GHNG second layer is not reported as it already covers all Gaussian distribution at the first level

The results, visualized in Fig.7.11 whose trees are shown in Fig. 7.12, clearly show that only GH-EXIN and DGSOT build the correct hierarchy: two nodes in the first layer (level), which represent the two clusters, and as



(a) GH-EXIN tree (b) DGSOT tree (c) GHNG tree

Figure 7.12: Tree structures (labelled by the cluster cardinality) of the three algorithms for the Mixture of Gaussians dataset



(a) GH-EXIN Silhouette (b) DGSOT Silhouette (c) GHNG Silhouette

Figure 7.13: Silhouette scores for the three algorithms on the Mixture of Gaussian Dataset

many leaves as Gaussians in the second layer, which represent the mixtures. Neurons are also positioned correctly w.r.t. the centers of the Gaussians. On the contrary, GHNG spreads all the information in the first level. In this sense, it only partitions, but does not reveal the hierarchy. The quality indexes, as illustrated in Fig. 7.15, only refer to the final partition. They show a slightly better PSNR and S for GH-EXIN, but a similar DB. This assessment is supported also by the analysis of the Silhouette plots reported on Fig. 7.13, which shows the S values for each neuron: the S values for GH-EXIN are mostly positive and the other values are only slightly negative, unlike the other two methods. All the algorithms require a similar number of neurons, but GHNG is much faster.

Resuming, GHNG is a faster algorithm, but is better for partitional clustering, even if it has been conceived for finding hierarchies in the data. It opens the question if its performance simply results from its modified GNG module.

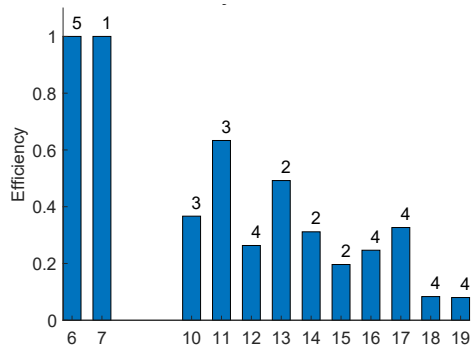
7.3 Hierarchical Clustering for Video Sequences

The first real experiment is the same proposed in [50], where GHNG is compared to GHSOM, GNG and SOM (GHNG is comparable to GNG and better than GHSOM and SOM). It has been devised for checking the quality of the hierarchical clustering. A database [63] composed of five video sequences, each representing exclusively either one of four people (two men, classes 2 and 4, and two women, classes 1 and 5) or one container (class 3), is used, with the goal of grouping frames of the same class. There are 1432 input frames of dimensionality 25344 (176×144 pixels), each with 3 channels (RGB). At first the color images are converted to grayscale images by eliminating the hue and saturation information while retaining the luminance. Because of the high dimensionality of data, the inputs are linearly projected to dimension 8 by using the Principal Component Analysis (PCA), performed by the eigenface method [64]. The projection retains 83% of the original data variance.

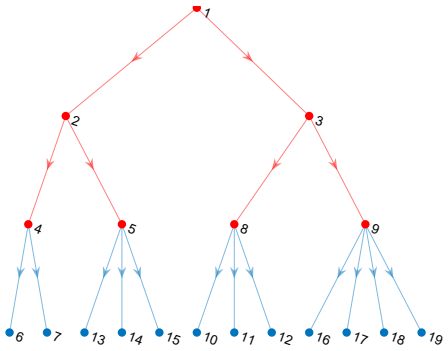
Fig.7.14 shows the hierarchical trees (labeled by the numbers of the nodes and leaves) and the associated best leaf efficiencies. The efficiency of a class in a cluster is defined as the percentage (w.r.t. the whole database) of elements of the class in the cluster. The best efficiency reported for each leaf in Fig.7.14 is the maximum of these values (the number on the top of the bar corresponds to the class) and represents an external qualitative index of the clustering. It has been observed that all leaves of the GH-EXIN and DGSOT trees have a 100% purity, while a few GHNG leaves do not share this property, where the purity is defined as the percentage of elements in a cluster belonging to the most common class. The following conclusions about the experiment can be drawn.

- GH-EXIN nodes 2 and 3 have been built by the double simultaneous vertical growth (red edges in Fig.7.14(b)) and reflect the fact two chains have been found in the global dataset. The node 2 Voronoi set only contains data from classes 1, 2 and 5. The node 3 Voronoi set is only composed of data from classes 3 and 4. Hence, the first level of the GH EXIN tree perfectly divides in two clusters.

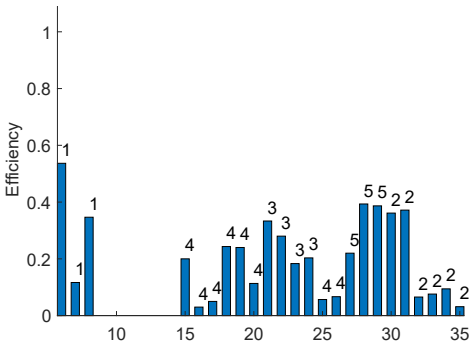
- DGSOT, which has not this double vertical mechanism, has a first level which is comparable with the second level of GH-EXIN. DGSOT finds exactly all data of class 1 in node 2, all data of class 3 and 65% data of class 4 in node 3, all data of class 2 and 5 in node 4 and 35% data of class 4 in node 5. Resuming, node 2 is 100% pure and efficient, but node 5 is 100% pure and only 35% efficient.



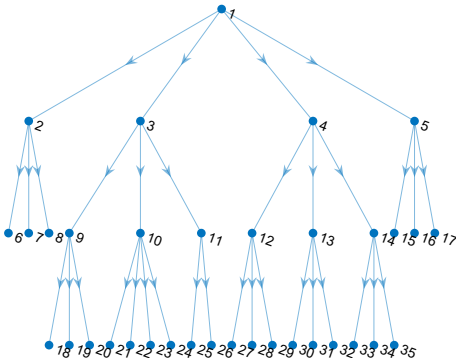
(a) GH-EXIN leaves efficiency



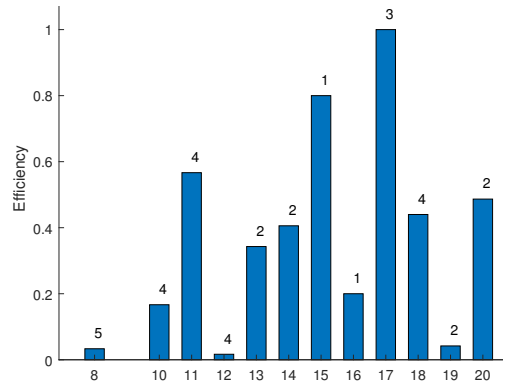
(b) GH-EXIN tree



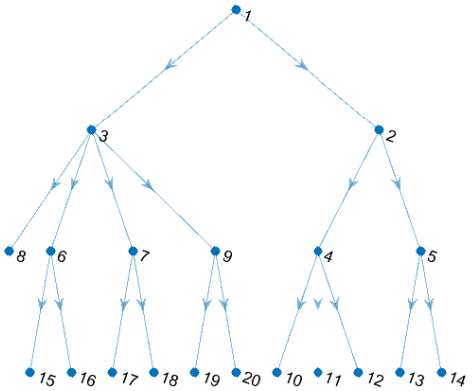
(c) DGSOT leaves efficiency



(d) DGSOT tree



(e) GHNG leaves efficiency



(f) GHNG tree

Figure 7.14: Leaf efficiency (left) and tree structure (right) of the three algorithms for the videos dataset. Regarding the efficiency bar plot, the row represents the number of the corresponding leaf on which the efficiency has been computed. Each bar is labelled on top by the class of the group of data with the highest efficiency within the leaf.

- The GH-EXIN second level finds exactly all data of class 2 in node 5, all data of class 3 and 25% data of class 4 in node 8, all data of class 1 and 5 in node 4 and 75% data of class 4 in node 9. W.r.t. DGSOT, node 9 is 100% pure but 75% efficient for the same class of DGSOT node 5. Also, node 4 perfectly identifies the two women classes.

- The GH-EXIN third level and the DGSOT second level neurons have Voronoi sets composed of a unique class. However, 65% class 4 data for DGSOT and 25% class 4 data for GH-EXIN are nested in another cluster (together with class 3).

- The GHNG tree first level does not compute a correct clustering. Instead, node 2 retains a portion of class 2 and 4 data, with the consequence that these classes will be grouped in the next levels by clusters in different branches. The absence of a reallocation tool in the algorithm prevents from correcting this problem.

- The GHNG tree second level has more nodes than the same level for the other two algorithms. Class 2 is shared, nearly fifty-fifty, by node 9 and node 5, belonging to different branches. The same can be repeated for node 4 and 7 w.r.t. class 4. Class 1 is perfectly retrieved in node 6, while class 3 is only retrieved at the third level. The worst result is yielded in the third level by node 8, which only retains less than 0.1% class 5 data, thus preventing a correct clustering of class 5 in the tree. Indeed, the node 8 Voronoi set is empty (this is allowed by the GNG algorithm[1]). However, this value of efficiency derives from the recall phase, in which it is possible that an *empty* neuron wins because it has moved in the Voronoi set of another neuron. The same considerations can be repeated for the empty neuron 12. On the contrary, this problem is avoided in GH-EXIN because of the reallocation technique.

With regard to the results in Fig.7.15, GH-EXIN requires fewer neurons, before being automatically stopped. GHNG, as usual, is by far the fastest. The quality indexes, which do not take into account the hierarchy, but only

[1]GNG creates a neuron in the middle between the father and the mother neurons. Its position does not depend on the presence of a data. It is linked to its parent neurons. If it never wins, but one of its two neighbors wins, it is possible that it changes position (SCL) and approaches data of another cluster, which will not be presented anymore to the network. Hence, it remains empty, but may win in the recall phase.

the quality of the final partitioning, show comparable PSNR and a far better S index for GH-EXIN, despite the better DB for DGSOT. Notice the very high value of DB for GHNG. It has been found experimentally that it is related to the presence of empty neurons.

Summing up the previous observations, GH-EXIN and DGSOT find an optimal hierarchical clustering. GH-EXIN is also able to detect the difference between male and female faces. On the contrary, GHNG yields a very poor hierarchy. Probably, this explains why, in [50], the authors of GHNG do not show the entire tree, but only the results of some nodes, with the associated leaves.

Table 7.1: GH-EXIN hyperparameters

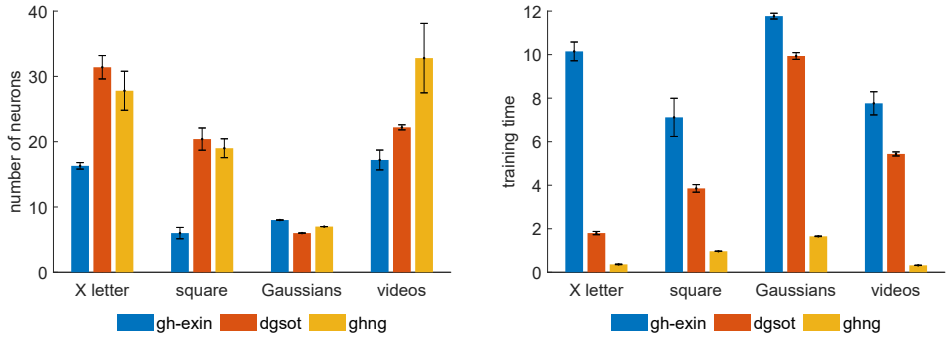
	H_{max}	H_{perc}	$\alpha_{\gamma 0}$	α_{i0}	age_{max}	min_{card}
X-shape	0.00002	0.9	0.1	0.01	5	10
Square	0.00002	0.9	0.35	0.001	10	30
Gaussians	0.001	0.9	0.5	0.05	5	300
Videos	0.8	0.9	0.8	0.1	20	10

Table 7.2: DGSOT hyperparameters

	α	σ_0	T_R	T_E	ϵ_{AD}	ϵ_{ET}	K
X-shape	0.2	1	0.3	10	0.046	0.03	1
Square	0.1	1	0.001	10	0.09	0.03	0
Gaussians	0.2	1	200	2	0.2	0.05	1
Videos	0.2	1	250	2	0.2	0.05	1

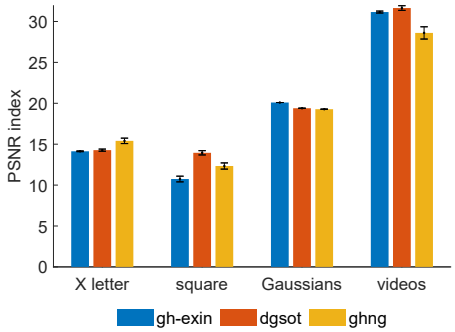
Table 7.3: GHNG hyperparameters

	MAX_{LEVEL}	τ	λ	ϵ_B	ϵ_N	α	A_{max}	D
X-shape	2	0.25	100	0.1	0.01	0.5	50	0.995
Square	2	0.3	100	0.35	0.01	0.5	50	0.995
Gaussians	2	0.1	100	0.4	0.01	0.5	14	0.995
Videos	3	0.2	100	0.001	0.001	0.5	50	0.995

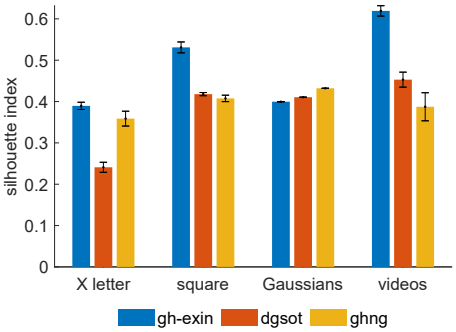


(a) Number of neurons

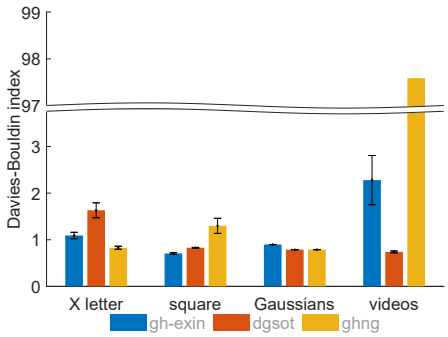
(b) Training time



(c) PSNR index



(d) Silhouette index



(e) Davies-Bouldin index. Notice that the s.e.m. error bar of GHNG on the videos dataset is not reported as it would have impaired the visualization of the other bars

Figure 7.15: Bar plots with the standard error of the mean (s.e.m.) bars showing the mean of some relevant statistics for each dataset and for each neural network

7.4 Gene Analysis

7.4.1 Neural Biclustering with GH-EXIN

In the previous sections, we presented the GH-EXIN neural network and we compare its pros and cons with state-of-the-art hierarchical neural techniques. In this section we exploit GH-EXIN in order to analyze CRC microarray data. The previously introduced high-dimensional issue in handling microarray data may be bypassed by unsupervised techniques thanks to biclustering. Indeed, by considering genes as samples and cellular tissues as features, the dimensionality of the input space is dramatically reduced. Successively, each batch of genes is further analyzed by clustering cellular tissues in a lower dimensional space. We will refer to *neural biclustering* as the successive exploitation of GH-EXIN in biclustering microarray data. Each time GH-EXIN is exploited, a tree is built, either in the gene or in the tissue space, for the gene clustering in the higher-level leaves. The validity of the leaves is tested and possibly GH-EXIN is called again in the corresponding projected space of each leaf. This procedure is recursively repeated until the cardinality of the tissues or the cardinality of the genes of a leaf is under the minimum threshold. At this point the leaf is saved and the algorithm continues by processing the other leaves. The order in which leaves are processed depend on their ranking, based on their Hcc value. Low values of Hcc associated to an acceptable cardinality do not imply a final bicluster has been detected, above all for the presence of high noise in data. An additional analysis is required, which depends on several considerations.

Here, the final leaves are studied from two different points of view: parallel coordinates and singular value decomposition (SVD), for the analysis of the numerical rank of the submatrices associated to the biclusters.

7.4.2 Validating Techniques

Fig. 7.16 shows this kind of plot by visualizing genes as samples (colored polylines) and murine tissues as features (parallel vertical axes) on a leaf of GH-EXIN in the gene space, whose characteristics are shown in the top line of the figure. Blue polylines represent all genes available in the dataset, while red polylines stand for genes collected in the 19th gene cluster. The red grouping of polylines show coherency, which confirms the quality of gene clustering. A similar validation analysis is used after the GH-EXIN clustering in the tissue space which is run after projecting the Voronoi set of the 19th

gene leaf (cluster).

Figs. 7.17 and 7.18 show two parallel coordinate plots in which vertical axes (here visualized as the corresponding abscissas in the coordinate axis) represent the 41 genes belonging to cluster 19, while polylines stand for murine tissues. In particular, blue polylines represent all the tissues and red ones the tissues grouped in the bicluster. The difference between the two images consists in a different setup of a parameter of the algorithm, Cmin2, which regulates the maximum number of tissues accepted in a bicluster. In the first case a higher value of the parameter is set, in order to find a bigger bicluster. However, both pictures show an excellent bicluster coherency revealing the goodness of GH-EXIN as a tool for biclustering. The biclusters shown in both figures are coherent additive values biclusters in which the values vary both according to the rows (the axes in this case) and according to the columns (the polylines). This can be inferred from the pictures, because a difference is present between two gene expressions on different polylines but along the same axes, but this difference remains stable along the polyline. The same is also valid between two gene expressions on different axes of the same polylines.

This visualization tool can be considered as a first validation of the quality of the leaves. A second validation can be performed by analyzing the singular values of the resulting bicluster matrices. According to the theory, in case of

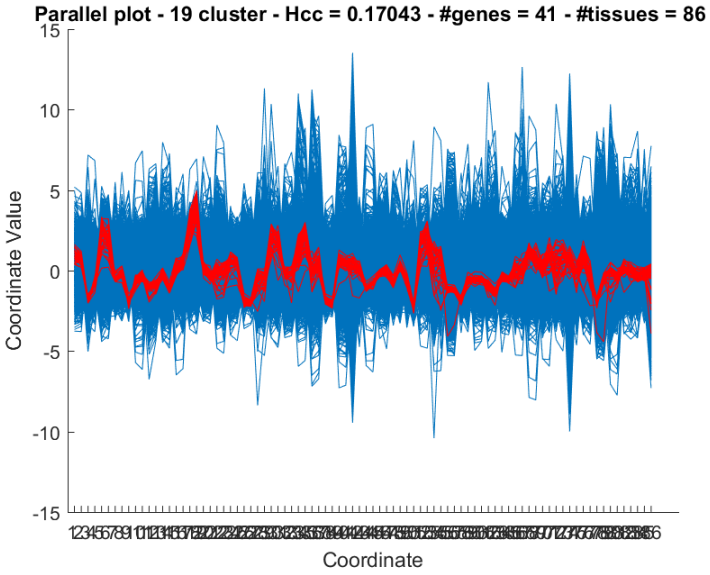


Figure 7.16: Parallel coordinates of a cluster of gene

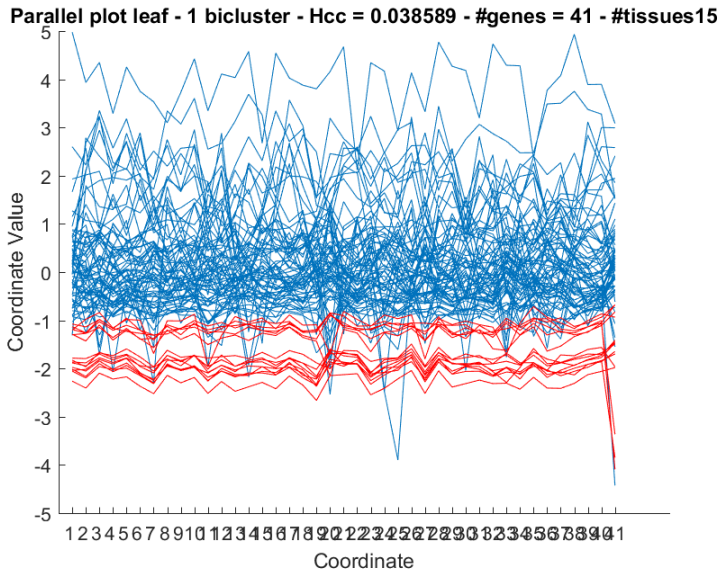


Figure 7.17: Parallel coordinates of a bicluster.

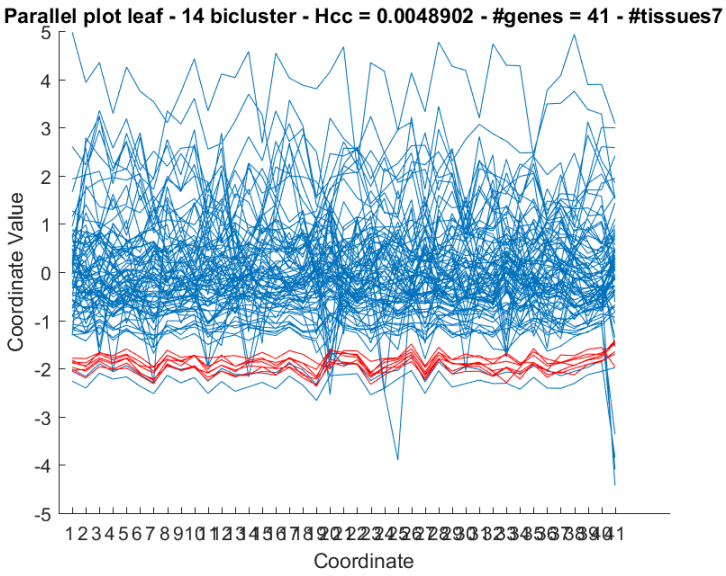


Figure 7.18: Parallel coordinates of a smaller bicluster.

noiseless data, biclusters with constant values and with constant values on rows or columns have rank one, while biclusters with coherent values have rank three. The difficulty raises in case of noise, because there are no more zero singular values. Indeed, the size of the last values increases with the

level of noise. It then becomes a problem in numerical rank estimation. The SVD of the matrix of the bicluster in fig. 7.17 has the first two singular values (42.5 and 4.5) well separated from the other ones (the third one is equal to 1.5), considering also that the matrix has been scaled in the preprocessing stage). This result represents the sum of two biclusters of rank one, certainly, considering the associated parallel plot, two constant row biclusters. Indeed, fig. 7.17 shows two clusters (coherent polylines, whose thickness depends on noise level). Hence, it can be deduced that a further clustering (and projection) is needed in order to have a single bicluster. Instead, the SVD of the matrix of the bicluster in fig. 7.18 (see fig. 7.19) has only the first singular value (32.5) well separated from the remaining ones (the second one is equal to 1.1). As also confirmed in fig. 7.18, it represents a constant row bicluster. This result does not require a further analysis.

Leaf node	**Cardinality**		**Hcc**
tissue class	*genes*	*tissues*	
1	54	8	0.027
1	58	6	0.059
2	8	8	0.027
2	8	6	0.029
2	71	5	0.031
2	5	5	0.039
3	7	8	0.025
3	7	13	0.025
3	19	6	0.026
3	19	5	0.026
3	41	13	0.028
3	19	8	0.029

Table 7.4: Best leaves in terms of biclustering quality (H_{cc} index).

7.4.3 Experimental Findings

In order to better analyze genetic expressions common for different patients, the dataset has been divided into three parts (classes). This division follows the murine tissues response to anti-cancer drugs. At the end, three datasets have been derived, one for the mice which started recovering after three weeks of treatments, a second one for the mice which had a stable situation and at

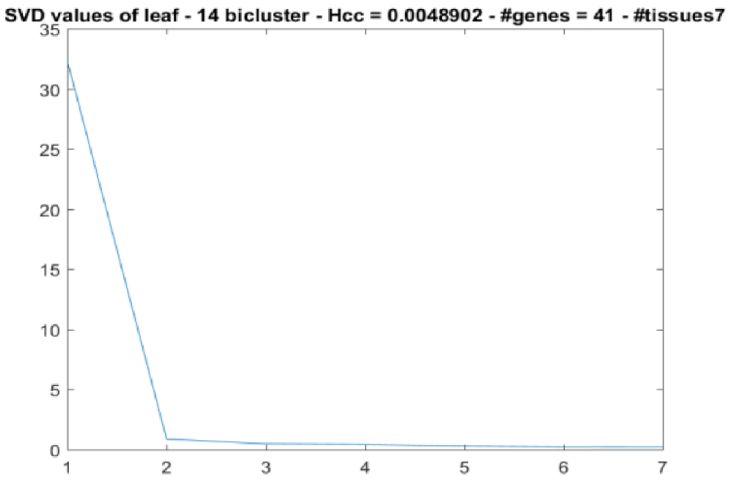


Figure 7.19: Singular values for leaf 14.

last one also for the case in which drugs had no effect and the cancer kept growing. This type of division has been maintained also in the summary Table 7.4, where it has been reported the information about the cardinality of the biclusters, both in the tissue and in the gene space and the value of the Hcc index. In the table there are only the best biclusters for each class, ranked according to the class and the Hcc index. As last step, as a biological feedback, the scientific relevance of the selected genes has been taken in account. Among all the biclusters found, the one that grouped the most interesting genes in the cancer field has been the one that also had the lowest Hcc index value. Indeed, the 7 genes present in the bicluster are the following:

- "CSAG1", "CSAG3", "CSAG3A", which belong to the same CSAG family. These genes are well known in literature as associated with chondrosarcomas, but they are also present in normal tissues. Furthermore, CSAG3 and CSAG3A are gene coding the "Chondrosarcoma-associated gene 2/3 protein" which is a "drug-resistance related protein, its expression is associated with the chemotherapy resistant and neoplastic phenotype. May also be linked to the malignant phenotype" [65].
- "MAGEA2", "MAGEA3", "MAGEA12", "MAGEA6", which belong to the same MAGEA family. These genes are melanoma antigens which “Reduce p53/TP53 transactivation function” and also "Represses p73/TP73 activity" [66]. Both p53 and p73 are tumor suppressor proteins which

regulate cell cycle and induct apoptosis.

The relevant issue is that these gene families are not only important by themselves, but this analysis suggests that, at least in the observed condition, they may also coregulate each other. It is also important to notice that this bicluster phenomenon has been observed within the tissues belonging to the third class, the one where tissues unable to respond to drugs are present.

Chapter 8

Supervised Neural Techniques

8.1 Introduction

In the following chapters, the supervised path is enriched with advanced and deeper analyzes. In the first chapter a powerful combination of evolutionary algorithm and neural techniques are exploited in order to enhance class prediction models in the context of cancer prognosis. Although powerful and accurate when tested on unseen samples, the neuroevolved model is opaque i.e. it is difficult to understand and interpret from a human point of view. This issue motivates section 8.3, where a very simple neural-based model has been used to investigate the underlying phenomenon more transparently. The analysis of both sections were presented in two different works at the 2018 WIRN conference [67] [68].

8.2 Neuroevolution

Evolutionary Algorithms (EA) are powerful metaheuristic procedures able to explore efficiently the search space of complex (NP-hard or NP-complete) problems finding good approximate solutions. The hyper-parameter optimization of a neural network is a complex problem because there are no polynomial-time algorithms able to solve it. In the following, we exploit EA in order to find satisfying approximate solutions to address this problem.
One of the most widely spread EA is the Genetic Algorithm (GA) [69][70]. GAs are metaheuristics inspired by natural selection processes. Broadly

speaking, GAs involve the evolution of a population of candidate solutions towards better ones. A predefined fitness function evaluates the individual goodness.

8.2.1 Mathematical Model of a Shallow Neural Network

The neural network architecture we used in the following experiments is known as *Adaline* [71][72]. It has 20023 inputs corresponding to the input features (genes) and one output neuron equipped with a linear output function. The network does not have hidden layers. During forward propagation, the network computes the dot product between the weight vector w and the i^{th} sample $x^{(i)}$ plus the bias b. This corresponds to a weighted sum of the inputs with bias correction (as in a linear regression model):

$$z^{(i)} = w^T x^{(i)} + b \tag{8.1}$$

$$\hat{y}^{(i)} = f(z^{(i)}) = z^{(i)} \tag{8.2}$$

where w is the weight vector, b the bias, f the activation function and $\hat{y}^{(i)}$ the network output.

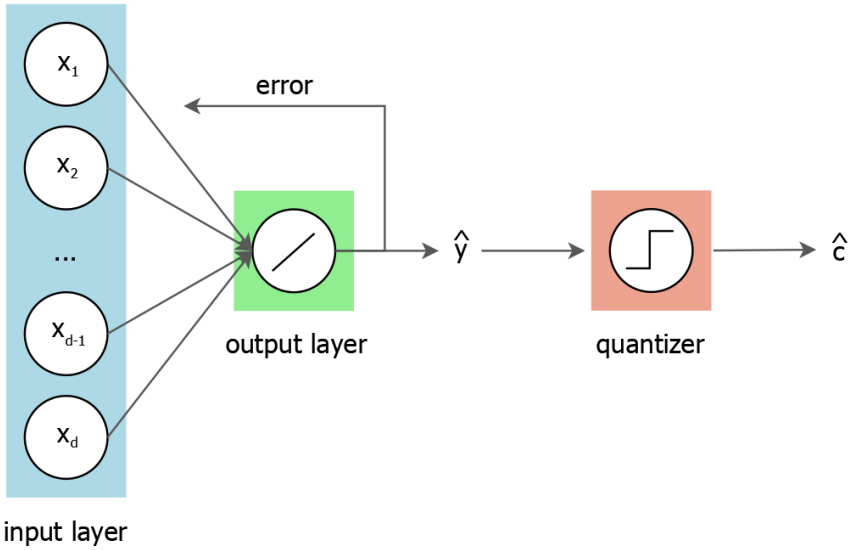


Figure 8.1: Shallow neural network architecture.

Objective Function

The squared error function evaluates the performance of the algorithm on an individual sample:

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = (y^{(i)} - \hat{y}^{(i)})^2 \tag{8.3}$$

where $y^{(i)}$ is 1 if the i^{th} sample belongs to class 1 and 0 if it belongs to class 0. In order to evaluate the global performance of the classifier, we use a cost function with L2 regularization of the weights. L2 regularization is a technique that applies to objective functions in ill-posed optimization problems [73][74]. In our case, the proposed neural model is ill-posed, since the solution is not unique and it changes continuously according to initial conditions and randomness in the cross-validation procedure. Appending a term to the cost function that penalizes large weights leads to a reduction of the search space, and the problem becomes less sensitive to initial conditions:

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} ||w||_w^2 \tag{8.4}$$

where λ is the regularization parameter and $||w||_w^2$ is the L2 norm of the weight vector. For big values of λ the regularization is stronger, increasing the penalization related to weights. As a result, the weights which are not useful for the purpose of minimizing the MSE (i.e. the first part of the objective function) are shrunk towards zero. On the contrary, for low values of λ, the regularization effect is weaker[1]. In order to provide a quantitative measure of the network performance, we transform the regression outcomes into class labels by using a Heaviside step function:

$$\hat{c}^{(i)} = \frac{d}{d\hat{y}} \max\{0, \hat{y}^{(i)}\} \tag{8.5}$$

and we compute the accuracy as if it were a classification task.

Parameter Optimization

Since the cost function measures the errors in the current predictions, the problem of the learning process is equivalent to the minimization of the cost function. Whereas the training samples are fixed, the cost function depends

[1]The described shallow neural network model is equivalent to a linear regression model with an L2 regularization of the parameters also known as Ridge Regression [74].

only on the network's parameters (weights and bias). So, the cost function minimization is equivalent to the optimization of the network parameters. For the following analyses, we use the Adaptive momentum estimation optimizer (Adam). Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [75]. It is a variant of the classical gradient descent algorithm, designed to combine the advantages of two popular methods: AdaGrad and RMSProp. According to [75] Adam's advantages are that its step-sizes are approximately bounded by the learning rate, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing. In the context of feed-forward neural networks, the objective function to be minimized is the cost function $J_t(\theta)$, where t denotes the t^{th} epoch and θ is a label for w and b. The authors identify with g_t the gradient, i.e. the vector of partial derivatives of J_t, w.r.t w and b evaluated at epoch t (8.6). This estimate is then used to update two exponential moving averages of the gradient (m_t, (8.7)) and the squared gradient (v_t, (8.8)). The two hyper-parameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these moving averages. High values for β_1, β_2 reduce the time-window size of the moving averages, resulting in low inertial effects and greater oscillations. On the contrary, low values of β_1, β_2 increase the time-window size, providing a stronger smoothing effect. The first moving average m_t is an estimate of the 1^{st} order moment (the mean) of the gradient. The second one instead is an estimate of the 2^{nd} order moment (the uncentered variance) of the gradient. Since these moving averages are initialized as vectors of zeros, the moment estimates are biased towards zero during the initial time-steps (especially when the decay rates are small, i.e. the βs are close to 1). This issue can be alleviated by the bias correction shown in (8.9) and (8.10). The ratio of the two moving averages corresponds to a standardization of the first order moment of the gradient. The network parameters are finally updated by using the classical formula of gradient descent in (8.11). The term ϵ (typically 10^{-8}) ensures that the denominator is always non-zero, avoiding numerical

issues.

$$g_t = \nabla_\theta J_t(\theta_{t-1}) \tag{8.6}$$
$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{8.7}$$
$$v_t = \beta_2 \cdot m_{t-1} + (1 - \beta_2) \cdot g_t \odot g_t \tag{8.8}$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{8.9}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{8.10}$$
$$\theta = \theta - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{8.11}$$

The initial conditions are: $m_0 = 0$, $v_0 = 0$ and $t = 0$. Typical values for βs are $\beta_1 \approx 0.9$ and $\beta_2 = 0.999$. Overall, Adam is a very efficient algorithm, requiring very few computations and memory space, which is crucial in our case, given the size of the data set.

8.2.2 Genetic Algorithm

Individuals

As previously outlined, the objective of this work is the optimization of a neural network model. In particular, the Adaline model presented in the previous section can be optimized tuning its hyper-parameters. Since each set of hyper-parameters uniquely identifies a neural network, then each neural network can be represented by its hyper-parameters. For this reason, an ordered list of hyper-parameters is an efficient representation of an Adaline model. Having assigned a value to each hyper-parameter from its domain, then the list is called candidate solution or individual. The set of optimized hyper-parameters is composed of:

- the learning rate α;
- the learning decay rate r;
- the number of epochs T;
- the regularization parameter λ.

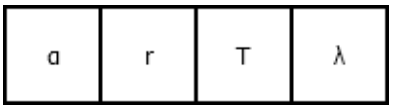


Figure 8.2: The graphical representation of an individual. It is represented as an ordered list of "genetic material". Each "gene" stands for a neural network hyper-parameter.

Generator

The generator is an EA method devoted to the initialization of new individuals. One of the most common generators is a random generator. In this case each hyper-parameter is sampled by a uniform distribution within a user defined range. The ranges chosen are:

- $\alpha \in [10^{-1}, 10^{-7}]$
- $r \in [10^{-1}, 10^{-7}]$
- $T \in [10, 300]$
- $\lambda \in [10^{-1}, 10^{-8}]$

The uniform distribution guarantees that the initial individuals are sufficiently different from each other. This biodiversity will help the EA search since there is more genetic material available for exchanges.

Evaluator

In order to optimize individuals generation by generation, the next population should be better that the previous one. Therefore, after the generation process, each individual is evaluated in order to estimate the goodness of its genetic material. To do this, an Adaline is built for each candidate solution, i.e. it is set up by using the corresponding hyper-parameters. The learning process is validated through a 10-fold cross validation. At the end of the training process, the average validation accuracy is considered as an estimate of the individual fitness.

Selector

After the evaluation process, the GA selects a random set of individuals from the population and selects a subset of them through a fitness-based criterion.

It ranks the randomly selected individuals in ascending order according to their fitness and it picks out some of the best ones. These small sets of individuals are then used to produce the next generation.

Variator

In order to modify and (hopefully) improve the current population, the selected solutions should be slightly modified. At first, they go through a crossover (or recombination) process in which pairs of individuals mix their genetic material to produce two new child solutions.

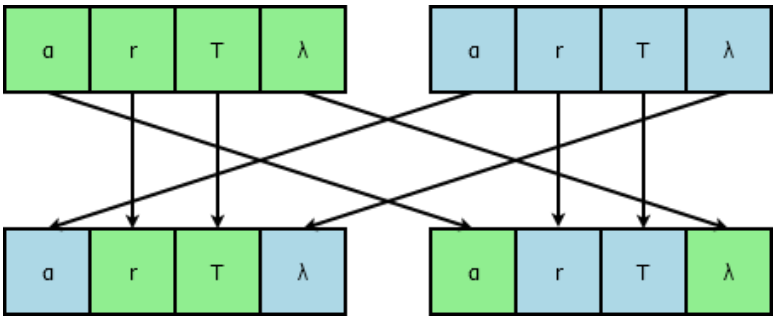


Figure 8.3: During the crossover process, two individuals mix their genetic material in order to produce two child solutions.

Secondly, the child solutions randomly mutate one of their components (hyperparameters).

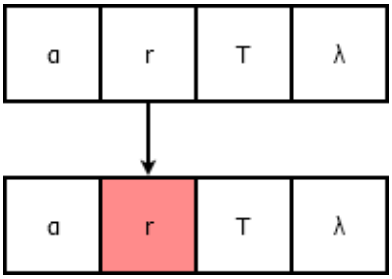


Figure 8.4: The figure shows the mutation process. The individual randomly mutates one of its "genes".

Replacer

After having generated new candidate solutions, the replacer method selects the best half of old population individuals and the best half of offspring in

order to select the candidate solution of the next generation.

Terminator

For the purpose of this work, the entire process is repeated for a certain amount of generations exploring the hyper-parameter space and (hopefully) providing better and better configurations.

Algorithm 2 Genetic Algorithm

```
1: Input: DNA microarray and cancer growth targets
2: Generate random individuals
3: for each generation do
4:     for each individual do
5:         Evaluate the individual through cross-validation
6:     end for
7:     Select next generation parents
8:     Breed random parents
9:     Mutate child individuals
10:    Replace old individuals with the new generated ones
11:    go to next generation
12: end for
13: Output: best individual
```

8.2.3 Comparison of the Results

The GA set up includes at least the choice of the population size, the maximum amount of generations and the mutation rate. In this work the following choices are made:

- population size: 100
- number of generations: 100
- mutation rate: 20%

In the end the GA provides the list of the candidate solutions of the last (and hopefully the best) generation. The best individual is picked up and the corresponding neural network is evaluated by using an unseen test set. The experiment is repeated 10 times in a 10-fold cross-validation setting. The outcomes are then compared with state-of-the-arts algorithms [76] in Fig. 8.5.

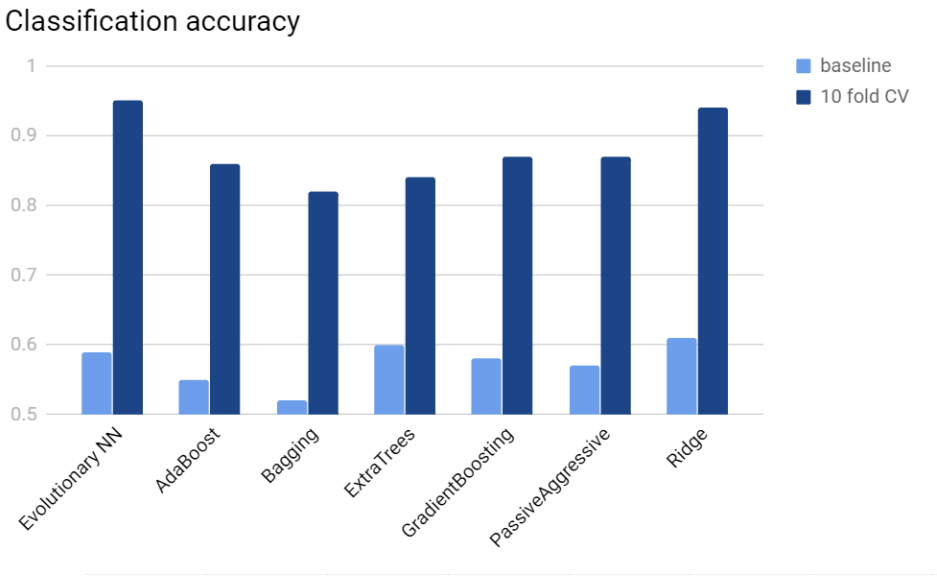


Figure 8.5: The figure shows the accuracy over a 10-fold cross-validation of several classifiers. The baseline accuracy refers to the performance of the classifier for random generated labels.

The proposed algorithm outperformed state-of-the-art techniques. However, linear based techniques, such as Ridge, show great performances, very close to the evolutionary approach. Nonetheless, although powerful and accurate, these algorithms are excessively complex (they are composed of thousands of parameters), resulting in opaque models, difficult to analyze and interpret from a human point of view. This consideration lead us towards a simpler and more transparent approach described in the following section.

8.3 Transparent Neural Based Model for Feature Selection

In this section, we propose a supervised feature selection process based on the recurrent exploitation of the Adaline model described in the previous section. In order to assess the goodness of the proposed approach, we perform a series of cross-validated training of the Adaline model. In particular, at each iteration the neural network is trained 30 times, each of which using a 10-fold cross validation with random folds. The neural network hyperparameters are heuristically fixed to:

$$\lambda = \frac{1}{\#\text{samples}} \tag{8.12}$$

$$\alpha = \frac{1}{\lambda + L + 1} \tag{8.13}$$

according to [77][78], where L is the maximum sum of the squares over all samples. Since the objective function to minimize contains the L2 norm of the weights (see equation 8.4), weights who are not fundamental for the classification task are shrunk towards zero by the optimizer [74]. Fig. 8.6 and 8.7 show respectively the histogram and the notched box plot of the weights after the training process in the first iteration. It is important to notice that most of the weights are set to zero or are very close to zero. This means that their contribution to the weighted sum in equation (8.1) is almost negligible. Exploiting this result, for each fold we take note of the input features (i.e. the genes) which correspond to weights having an absolute value w_j after a training process:

$$|w_j| > 2\sigma_w \tag{8.14}$$

where σ_w is the variance of the weight distribution (see fig. 8.7). At the end of the 30 iterations, we found that some input features are chosen more frequently than others.

Biologically speaking, this result suggests that the information contained in the DNA-microarray related to these genes may be relevant in understanding the cancer resistance to drugs. In order to investigate more deeply the biological phenomenon, we repeat the same experiments, modifying the database by keeping only the most frequently selected features, i.e. those which are selected at least half of the times after the 300 training processes. So, iteration by iteration we gradually reduce the number of input features used to train the neural network. Fig. 8.8 shows for each iteration the

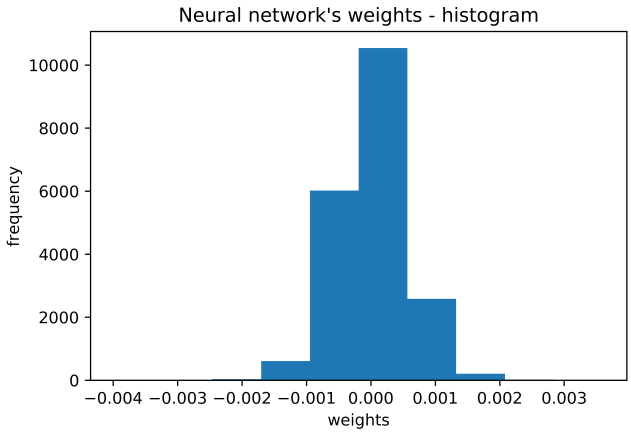


Figure 8.6: An example of histogram of the neural network weights after the first training iteration.

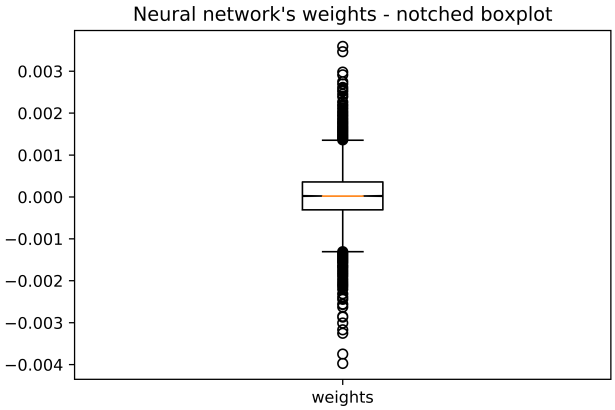


Figure 8.7: An example of notched box plot of the neural network weights after the first training iteration.

number of features used to train the neural network and the corresponding 10-fold cross validation accuracy. The blue bars correspond to the feature selection technique described above, while the violet ones to the ANOVA-F statistic method[2]. Notice that, initially, by using the original data set, the

[2]The corresponding standard deviation is always in the order of few percentage decimals and it is not directly displayed since it is not relevant for the purpose of the discussion. However, you can reproduce the experiment by using our code if you need more precision.

cross-validation accuracy is around 0.7. Such result may have two main explanations. First, the classes are not perfectly balanced, since 66% of samples belong to class 0. Secondly, the high dimensionality of the data may generate a slight overfitting. However, by reducing the input features using the method previously, the cross-validation accuracy raises above 0.9, decreasing progressively as the number of features are further diminished.

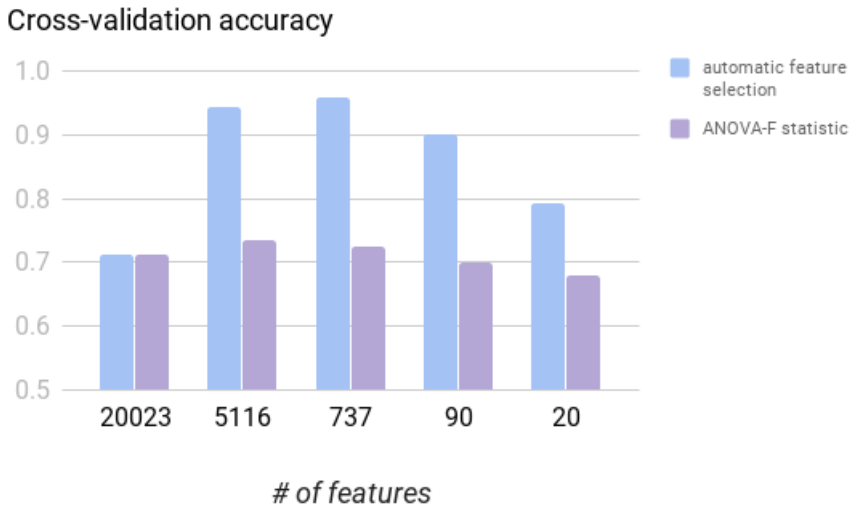


Figure 8.8: Histogram displaying the 10-fold cross validation accuracy at each iteration of the experiment.

It is important to notice that the neural network used as classifier is linear, i.e. geometrically speaking it delimits the input space with a hyperplane in order to classify data. This means that the proposed approach provides better results if the underlying phenomenon represented by the input data set is also linear. The results in fig. 8.8 show how the shallow neural network classifier delivers better results in the 737-dimensional space identified by the proposed feature extraction technique, than in the original 20023-dimensional data set. This may suggest that the underlying biological phenomenon at the DNA-microarray level is more linear in the reduced space than in the original one. Practically speaking, a linear problem is much easier to understand and tackle because the superposition principle holds i.e. the net response caused by two or more stimuli is the sum of the responses that would have been caused by each stimulus individually. Therefore, from a biological point of view, these results may suggest that the above experiment generates subspaces of the input features where the cancer resistance to treatments can be

studied more easily. In particular, in the 737-dimensional space the biological phenomenon is easier in the sense that it is more linear than in the original space; while in the 90 or 20-dimensional spaces it is easier because, while the classification accuracy decreases, the limited number of genes involved can be more thoroughly analyzed by human experts.

Part IV

Conclusion and Future Developments

Chapter 9

Critical Analysis

In this work we presented a summary of the analyzes performed on the largest CRC xenograft data set available in the academic world. On one hand, these analyzes are relevant because the outcomes could be used to instruct further biological and clinical research. On the other hand, the issues encountered in approaching high-dimensional data has led to the development of novel techniques, which may be exploited in different fields other than biology.

The major machine learning novelty is represented by the creation of GH-EXIN, a new neural-based technique for hierarchical clustering. The comparison with DGOST and GHNG techniques shows how GH-EXIN is typically more efficient, as it reaches similar performances in terms of peak-signal to noise ratio (PSNR) by using fewer neurons. Moreover, qualitative evaluation of the resulting topology shows how GH-EXIN is much more elegant in connecting neurons, providing superior manifold representations. Finally, the restricted number of user-dependent parameters makes the tuning process of GH-EXIN very easy.

The application of the biclustering framework integrated with GH-EXIN on the biological dataset revealed some interesting gene correlation patterns. These results have been submitted to the attention of IRCC doctors, who are currently analyzing them for possible scientific implications.

Other minor novelties have been introduced within both the unsupervised path and the supervised one. They always enhanced effectiveness levels compared to current state-of-art techniques.

The results above are promising and highlight the potential for future work. From the point of view of the biological advances, the outcomes of both the unsupervised and the supervised path are promising yet opaque: while the models can be used effectively, the results are difficult to interpret

from a human point of view. As for the unsupervised direction of work, the GH-EXIN neural network resulted to be effective and easy to use as aforementioned; results provided by the biclustering framework, instead, are rather difficult to interpret without a statistical knowledge, since biclusters need several additional tools to be correctly evaluated. As for the supervised direction, the major advances with respect to previous analyses are achieved by exploiting the shallow neural network model. Indeed, the simplicity of such model makes it easier to handle and interpret. On limiting the number of features used, however, the accuracy drops significantly.

Moreover, major limitations of our work directly derive from the analyzed data. On the one hand, the analyzed data represent an estimate of the amount of times each gene is transcribed in a tumor xenograft. However, gene replication does not always result in protein generation. Indeed, this kind of data may not represent cell behaviour correctly. On the other hand, the restricted amount of samples was the most serious issue, since machine learning reliability is directly related to the amount of data provided.

Hence, within the biological domain, future developments will involve the use of up-to-date data (e.g. representing proteins instead of gene expressions) and the integration with other sources of data, such as image samples. Besides, from a machine learning point of view, models easier to interpret may be developed in order to provide more reliable and human-understandable outcomes. Future research in this field will consist in devising new algorithms overcoming the intrinsic weaknesses of machine learning, above all understandability. To this purpose, novel algorithms which integrate classic symbolic artificial intelligence with machine learning techniques seem to be very promising. Further details are given in the next chapter. Both authors will carry on these researches during the doctorate.

Chapter 10

Future Works: Towards Artificial General Intelligence

Strictly concerning machine learning, future works will consist first of an analysis of current state of art of Machine Learning (ML), in order to clearly understand whether it is possible to enrich current techniques capabilities.

A second part of the future researches, instead, will consist in devising new AI algorithms that may go towards an Artificial General Intelligence (AGI). This part will be considered either in case machine learning issues result to be unsolvable, or not, as it seems to be promising per se. A good starting point "may be to integrate deep learning, which excels at perceptual classification, with symbolic systems, which excel at inference and abstraction. One might think such a potential merger on analogy to the brain; perceptual input systems, like primary sensory cortex, seem to do something like what deep learning does, but there are other areas, like Broca's area and prefrontal cortex, that seem to operate at much higher level of abstraction" [16]

Interestingly, symbolic systems and Machine Learning in computer science somehow correspond in philosophy to deductive and inductive reasoning respectively. In fact, deductive reasoning is a process that tries to reach a certain conclusion by applying general rules, narrowing the space of possible conclusions until only one is left. Classic logic and expert systems are strongly based on these principles. Inductive reasoning, instead, is the process in which starting from observations a possible conclusion is derived. The conclusion anyway cannot be considered as undoubtedly truth. All the ML

techniques are based on these assumptions.

Summing up, a long-standing controversy exists in literature regarding the role of induction and deduction in reasoning. Nevertheless, none would state that human reasoning is based on only one of them and neither we should suppose it in AI: combining several approaches is the only way to reach AGI.

At the same time, future researches will also study developmental psychology to understand how reasoning processes take place in human brain in order to get useful hints about what are the best ways of replicating them. It is also possible that completely new paradigm, with little in common with existing ones, can be derived from these observations as happened with neural networks.

Bibliography

[1] National Cancer Institute . What Is Cancer? . https://www.cancer.gov/about-cancer/understanding/what-is-cancer , Feb 2015 . Accessed on 2018-10-06.

[2] Alfred G. Knudson. Mutation and cancer: Statistical study of retinoblastoma. *Proc Natl Acad Sci U S A*, 68(4):820–823, Apr 1971. 5279523[pmid].

[3] J. S. de Bono and Alan Ashworth. Translating cancer research into targeted therapeutics. *Nature*, 467:543 EP –, Sep 2010. Perspective.

[4] Y. Komeda, H. Handa, T. Watanabe, T. Nomura, M. Kitahashi, T. Sakurai, A. Okamoto, T. Minami, M. Kono, T. Arizumi, M. Takenaka, S. Hagiwara, S. Matsui, N. Nishida, H. Kashida, and M. Kudo. Computer-aided diagnosis based on convolutional neural network system for colorectal polyp classification: Preliminary experience. *Oncology*, 93(suppl 1)(Suppl. 1):30–34, 2017.

[5] Shelley McGuire. World cancer report 2014. geneva, switzerland: World health organization, international agency for research on cancer, who press, 2015. *Adv Nutr*, 7(2):418–419, Mar 2016. 012211[PII].

[6] Manuel Hidalgo, Frederic Amant, Andrew V. Biankin, Eva Budinská, Annette T. Byrne, Carlos Caldas, Robert B. Clarke, Steven de Jong, Jos Jonkers, Gunhild Mari Mælandsmo, Sergio Roman-Roman, Joan Seoane, Livio Trusolino, and Alberto Villanueva. Patient derived xenograft models: An emerging platform for translational cancer research. *Cancer Discov*, 4(9):998–1013, Sep 2014. 25185190[pmid].

[7] Illumina . BeadArray Microarray Technology . https://emea.illumina.com/science/technology/beadarray-technology.html , July 2017 . Accessed on 2018-10-08.

[8] Gartner . Top Trends in the Gartner Hype Cycle for Emerging Technologies . www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/

, July 2017 . Accessed on 2018-10-08.

[9] Marvin Minsky and Seymour Papert. *Perceptrons: an introduction to computational geometry*. 1969.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[11] WHO: World Health Organization . Cardiovascular diseases (CVDs) . http://www.who.int/cardiovascular_diseases/en/ , July 2017 . Accessed on 2018-08-08.

[12] WHO: World Health Organization . Cancer . http://www.who.int/cancer/en/ , July 2017 . Accessed on 2018-08-08.

[13] WHO: World Health Organization . Road traffic deaths. . www.who.int/gho/road_safety/mortality , July 2017 . Accessed on 2018-08-08.

[14] Andrew Ng . What Artificial Intelligence Can and Can't Do Right Now . https://hbr.org/2016/11/what-artificial-intelligence-can-and-cant-do-right-now , November 2016 . Accessed on 2018-10-10.

[15] Francois Chollet. *Deep Learning with Python*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2017.

[16] Gary Marcus. Deep learning: A critical appraisal. *CoRR*, abs/1801.00631, 2018.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –, Feb 2015.

[18] Ken Kansky, Tom Silver, David A. Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, D. Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *CoRR*, abs/1706.04317, 2017.

[19] Pietro Barbiero, Andrea Bertotti, Gabriele Ciravegna, Giansalvo Cirrincione, Eros Pasero, and Elio Piccolo. Unsupervised Gene Identification in Colorectal Cancer. In *Quantifying and Processing Biomedical and*

Behavioral Signals, pages 219–227. Springer International Publishing, 2018.

[20] Sokal R. R. and Michener C. D. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 28:1409–1438, 1958.

[21] Sokal R. R. and Michener C. D. The comparison of dendrograms by objective methods. *Taxon*, 1962.

[22] Ward Joe H. JR. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 1963.

[23] Calinski T. and Harabasz J. A dendrite method for cluster analysis. *Communications in Statistics*, 1974.

[24] Wegman Edward J. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 1990.

[25] Jolliffe I. T. *Principal Component Analysis*. Springer Series in Statistics, 2002.

[26] Demartines P. and Hérault J. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 1997.

[27] Gower J. C. and Hand D. J. *Biplots*. Chapman and Hall, 1996.

[28] Usa national center for biotechnology information.

[29] Pietro Barbiero, Andrea Bertotti, Gabriele Ciravegna, Giansalvo Cirrincione, Eros Pasero, and Elio Piccolo. Supervised gene identification in colorectal cancer. In *Quantifying and Processing Biomedical and Behavioral Signals*, pages 243–251. Springer International Publishing, 2018.

[30] Davies D. L. and Bouldin D. W. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1979.

[31] Tibishirani R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 1996.

[32] Bishop C. M. *Pattern Recognition and Machine Learning*. Springer, 2016.

[33] James J Chen. Key aspects of analyzing microarray gene-expression data. *Pharmacogenomics*, 8(5):473–482, 2007. PMID: 17465711.

[34] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, Jan 2004.

[35] Cirrincione G. Ciravegna G. Pasero E. Randazzo, V. Inonstationary topological learning with bridges and convex polytopes: the g-exin neural network. In *IEEE - Proceedings of IJCNN 2018 International Joint Conference on Neural Networks*, 2018.

[36] B.S. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, 2011.

[37] T Li, Y.Y. Tang, S.C. Suen, L.Y. Fang, and A.J. Jennings. A Structurally Adaptive Neural Tree for Recognition of Large Character Set. *Proceedings of the 11th IAPR International Joint Conference on Pattern Recognition*, 2:187 – 190, 01 1992.

[38] R. G. Adams, K. Butchart, and N. Davey. Hierarchical Classification with a Competitive Evolutionary Neural Tree. *Neural Networks* , 12(3):541 – 551, 1999.

[39] Elena Samsonova, Joost Kok, and Ad Ijzerman. TreeSOM: Cluster Analysis in the Self-Organizing Map. *Neural networks : the official journal of the International Neural Network Society*, 19:935–49, 07 2006.

[40] Johan Himberg. A SOM Based Cluster Visualization and its Application for False Coloring. *IEEE Int. Joint Conf. on Neural Networks*, 3:587 – 592 vol.3, 02 2000.

[41] M Venkat Reddy, Makara Vivekananda, and R U V N Satish. Divisive Hierarchical Clustering with K-means and Agglomerative Hierarchical Clustering. *International Journal of Computer Science Trends and Technology (IJCST)*, 5, 10 2017.

[42] GuoYan Hang, DongMei Zhang, Jiadong Ren, and ChangZhen Hu. A Hierarchical Clustering Algorithm Based on K-Means with Constraints. *Innovative Computing ,Information and Control, International Conference on*, 0:1479–1482, 12 2009.

[43] George Aloysius. Efficient High Dimension Data Clustering using Constraint-Partitioning K-Means Algorithm. *International Arab Journal of Information Technology*, 10, 11 2013.

[44] Madjid Khalilian, Norwati Mustapha, Nasir Suliman, and Ali Mamat. A Novel K-Means Based Clustering Algorithm for High Dimensional Data Sets. In *International MultiConference of Engineers and Computer Scientists*, pages 17–19, 2010.

[45] Alberto Forti and Gian Luca Foresti. Growing Hierarchical Tree SOM: An Unsupervised Neural Network with Dynamic Topology. *Neural networks*, 19(10):1568–1580, 2006.

[46] Bernd Fritzke. Growing cell structures–a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7:1441–1460, 1994.

[47] Vanco Burzevski and Chilukuri K. Mohan. Hierarchical Growing Cell Structures. In *IEEE int. conference on neural networks*, pages 207–218, 1996.

[48] Bernd Fritzke. Growing Grid - A Self-Organizing Network with Constant Neighborhood Range and Adaptation Strength. *Neural Processing Letters*, 2(5):9–13, Sep 1995.

[49] A. Rauber, D. Merkl, and M. Dittenbach. The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, Nov 2002.

[50] Esteban J Palomo and Ezequiel López-rubio. The Growing Hierarchical Neural Gas Self-Organizing Neural Network. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–10, 2016.

[51] Bernd Fritzke. A Growing Neural Gas Network Learns Topologies. In *Advances in neural information processing systems*, pages 625–632, 1995.

[52] Latifur Khan and Feng Luo. Hierarchical Clustering for Complex Data. *International Journal on Artificial Intelligence Tools*, 14:791–810, 2005.

[53] Joaquin Dopazo and José María Carazo. Phylogenetic Reconstruction Using an Unsupervised Growing Neural Network that Adopts the Topology of a Phylogenetic Tree. *Journal of Molecular Evolution*, 44(2):226–233, 1997.

[54] Esteban J Palomo and Ezequiel López-rubio. The Growing Hierarchical Neural Gas Self-Organizing Neural Network. pages 1–10, 2016.

[55] Yizong Cheng and George M. Church. Biclustering of Expression Data. *Proceedings. International Conference on Intelligent Systems for Molecular Biology*, 8:93–103, 2000.

[56] Vincenzo Randazzo, Giansalvo Cirrincione, Gabriele Ciravegna, and Eros Pasero. Nonstationary Topological Learning with Bridges and Convex Polytopes: the G-EXIN Neural Network. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, jul 2018.

[57] Mohamed-Rafik Bouguelia, Yolande Belaid, and Abdel Belaid. Online Unsupervised Neural-Gas Learning Method for Infinite Data Streams. In *Pattern Recognition Applications and Methods*, pages 57–70. Springer, 2015.

[58] Giansalvo Cirrincione, Vincenzo Randazzo, and Eros Pasero. The Growing Curvilinear Component Analysis (GCCA) neural network. *Neural Networks*, 103:108–117, 2018.

[59] Gabriele Ciravegna and Pietro Barbiero. Gh-exin (version 1.0.1). https://bitbucket.org/machine_learning_research/ghexin/src/master/, 2018.

[60] Farokh Bastani I-Ling Yen Feng Luo, Latifur Khan and Jizhong Zhou. A dynamically growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles. *Bioinformatics*, 20:2605–2617, 2004.

[61] David L. Davies and Donald W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1979.

[62] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 1987.

[63] M. Reisslein, L. J. Karam, P. Seeling, F. H. Fitzek, and T. K. Madsen . YUV Video Sequences . http://trace.eas.asu.edu/yuv/index.html , December 2010 . Accessed on 2019-06-07.

[64] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591. IEEE, 1991.

[65] http://www.genecards.org/cgi-bin/carddisp.pl?gene=CSAG3. Accessed: 2017-11-20.

[66] http://www.genecards.org/cgi-bin/carddisp.pl?gene=MAGEA2. Accessed: 2017-11-20.

[67] Barbiero P., Bertotti A., Ciravegna G., Cirrincione G., Piccolo E., and Tonda A. Dna microarray classification: Evolutionary optimization of neural network hyperparameters. In *Italian Workshop on Neural Networks (WIRN 2018)*, 06 2018.

[68] Barbiero P., Bertotti A., Ciravegna G., Cirrincione G., Piccolo E., and Tonda A. Understanding cancer phenomenon at gene-expression level by using a shallow neural network chain. In *Italian Workshop on Neural Networks (WIRN 2018)*, 06 2018.

[69] Zbigniew Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs, 1996.

[70] Aaron Garrett. inspyred (version 1.0.1) inspired intelligence. https://github.com/aarongarrett/inspyred, 2012.

[71] Widrow B. and Lehr M. A. Artificial neural networks of the perceptron, madaline, and backpropagation family. *Neurobionics*, 1993.

[72] François Chollet et al. Keras. https://keras.io, 2015.

[73] Ng A. Y. Feature selection, l_1 vs. l_2 regularization, and rotational invariance. In *International Conference on Machine Learning*, 2004.

[74] Hastie T., Tibshirani R., and Friedman J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics, 2009.

[75] Kingma D. P. and Ba J. Adam: A method for stochastic optimization. *International Conference for Learning Representations.* 2017.

[76] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion,

O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[77] Schmidt M., Le Roux N., and Bach F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2013.

[78] Defazio A., Bach F., and Lacoste-Julien S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in Neural Information Processing Systems*, 2014.