Università degli Studi di Firenze

Facoltà di: **Economia**

Scuola di Dottorato in: **Statistica Applicata - Ciclo XXIII (SECS-S/01)**

# Statistical models for high dimensional data: selection and assessment

**Author:**
Davide De March

**Tutor:**
Prof. Andrea Giommi

**Coordinator:** Prof. Fabio Corradi

Anno Accademico 2010/2011

*To Fanny*

# Preface

In recent year, a mixture of expertise and the new technologies leads to the availability of massive amount of data. Statisticians must face the problem of high dimensionality, reshaping the classical statistical thinking and data analysis (Fan & Li 2006).
In many real-world problems the number of covariates is very large and often statisticians have to tackle the challenge of treating data in which the number of variables $p$ is much larger than the number of observations $n$ (i.e when $n \ll p$). Such high dimensional settings with their many new scientific problems create great opportunities and significant challenges for the development of new techniques in statistics and machine learnings.
From a classical statistical point of view, many algorithms for solving the problem of dimensional reduction and feature extraction have been conceived in order to obtain parsimonious models that are desirable as they provide simple and interpretable relations among scientific variables in addition to reducing forecasting errors. But in high-dimensional systems, we work with large size problems (from on the order of 50-100 up to thousands of variables) and the space of all possible subset of variables is of the order of $2^p$. Treating exhaustively all the possible subsets of models is not realistic because the study of all the submodels is a NP-hard problem with computational time increasing exponentially with the dimensionality, (Saeys et al. 2007).
Moreover, high dimensional real problems often involve costly experimentations and new techniques are needed to reduce the number of the experimental trials though guaranteeing satisfactory results. The expensive experimental and computational costs make traditional statistical procedures infeasible for high-dimensional data analysis, and this motivates the writing of this thesis: we want to approach the optimization of high dimensional problems with

variable selection and model assessment procedures combined with global stochastic optimization algorithms and local search methods.
This research is part of an international project dedicated to “Designing Informative Combinatorial Experiments” (DICE) for living technology, granted by the *Fondazione di Venezia* (`http://www.fondazionevenezia.org/`).
“DICE” project, coordinator prof. Irene Poli, is developed at the European Center for Living Technology (ECLT, `http://www.ecltech.org/`), an international research center, where methodology and experimentation are combined to solve high combinatorial optimization problems. The “DICE” project aims at designing evolutionary combinatorial experiments in high dimensional and high throughput settings in order to search new biological entities, such as new artificial proteins. The cooperation between biologists of the laboratory of ECLT and statisticians allows to combine biological expertise with new statistical methods to model and optimize complex biological problems.
The research is a continuum of previous projects and in particular of the EU, IST-FET Integrated Project “Programmable Artificial Cell Evolution”, coordinator prof. John McCaskill, where important results are achieved in tackling high dimensional biological problems (Baragona et al. 2011, De March et al. 2008, D. De March 2009, Forlin et al. 2008, D. Slanzi 2009).
The thesis presents a new optimization algorithm (the Evolutionary Neural Network Design) that merges the strengths of statistical model and variable selection with stochastic optimization strategies. The method is then applied in order to discover new “synthetic” proteins, showing its advantages in a biological experimentation.

# Contents

# List of Figures

# List of Tables

# Introduction

In recent years, technological innovation have had deep impact on society and on scientific research. Last decades have shown an impressive revolutionized way of approaching complex problems, and a strong develop of a variety of methods to collect massive amounts of information about phenomena of interest.

Real-world problems often are characterized by a growing in size, in dimensions and in complexity: (i) size, for the huge number of data provided by the great technological advances; (ii) dimensions, for the very large number of variables that investigators consider in developing research; (iii) complexity, for the high level of connectivity that characterizes these data sets. At the same time, applications have emerged in which the number of experimental units is comparatively small but the underlying dimension is massive; this is the case when collecting data on image processing, microarrays, text-mining, astronomy, military and atmospheric science(Johnstone & Titterington 2009).

It is more and more common to face situations when: (i) the number of collected variables can greatly exceed the number of observations and (ii) strong non-linearity behaviors are present and (iii) phenomena of interest, which can be characterized by particular combinations of the variables, are rare occurrences when compared to the overall size of the allowed search space. Rarely all these issues are observed together and only few approaches in literature show that it is possible to model such kind of data. Moreover, the previous problems are particularly challenging when the variables at hand are factors taking possibly many categorical attributes. Under these challenging circumstances, the existing multivariate statistical procedures alone do not suffice to ensure satisfactory solutions, even under strong assumptions on the marginal

variables, such as the assumption of independence.
Older methods, such as stepwise regression, all-subsets regression and ridge regression fall short in such scientific and applicable contexts, so that, as a response to these new challenges, statisticians and computer scientists devoted remarkable effort to develop predictive procedures as well as practical applications in this direction. One of most active statistical area that tries to give responses to these challenges, is related to the penalized regression models family; from the seminal paper, presenting the $\ell_1$ penalized regression model, better known as LASSO model (Tibshirani 1996), subsequent or competitive approaches have grown in importance determining a very active area in variable and model selection (LARS model, Efron et al. (2004), Elastic Net, Zou & Hastie (2005), among others).
Until now, however, no prevalent paradigm has been established for the efficient treatment of so many variables when data are scarce. In addition, within this framework, little progress has been accomplished on rare event prediction in a combinatorially large search space. Many of the proposed statistical procedures in literature seem to be accompanied by an overly optimistic view of their performance in terms of prediction accuracy, as well as practical applicability to very large problems.
The thesis proposes a new general approach able to take into account the aforementioned situations in high dimensional problem, representing a bottleneck in the cutting-edges researches. The algorithm we present, is suited to optimization problems in high dimensionality when categorical variables are observed and when the response of the system is complex and non-linear. The state of the art is to collect more and more data both in number of observations and variables. This paradigm of collecting huge data sets creates an overall problem in modeling and variable selection that suggest to avoid collecting huge data sets:

1. irrelevant variables may add extra noise which deteriorates the accuracy of the model;
2. too many variables can cloud meaningful relationship between the very important one;
3. the increasing number of parameters of the model causes consequently unreliable estimations due to the lack of observation $n$ in comparison

with the number of variables $p$;

4. a strong risk of abusing of redundant information in the model;
5. the accuracy of the predictions are underperformed due to the overfitting.

This thesis moves toward another direction: it presents a new procedure able to capture only important information creating very small data sets throughout the combination of methods for information extraction, global and local search algorithms and non linear models. The proposed procedure takes the name of Evolutionary Neural Network Design (ENN-Design) and it combines strategies as adaptive paradigm for data collection and regularization by information entanglement measures. In order to achieved this new competitive procedure, the proposed approach embodies the most pleasant features of three very different fields:

- the evolutionary optimization assures about the global convergence of the method,
- the non-linear models suffice to guarantee good fitting and prediction capability in very complex systems,
- the regularization procedure improves the performance of raw estimation by combining external information (usually by constraining the region of potential solutions).

Moreover, progressive iterative mechanisms can be exploited to select potentially useful subsets of data in subsequent rounds of experiments. Regularization approach via information entanglement allows for improving the knowledge about structures of interactions among variable attributes using only partial information about such interaction structures. Therefore, the "only-informative" data gathering shifts the problem of variable and model selection in the domain of the experimental design. The decision of deriving this new method for data gathering is related to real experimental requests: the maximal reduction of experimental trials to be tested. This reduction implies consequently great advantages in term of reduction of experimental time and cost.

This thesis has an important experimental component that is used to evaluate the ENN-Design approach: the discovery on new "synthetic" proteins.

The collaboration with a biological laboratory allow us to test the performance of the proposed approach in a real experimentation, where the high dimensionality problem plays a relevant role. Proteins are the fundaments of everyday life activities, and about $10^{13}$ natural proteins are known. Since the number of all the possible proteins is potentially infinite (in fact, the number of all possible combination of 20 amino acids in a protein with length $l$ is $20^l$), we are interested to know if and why the evolution has selected only a very small number of proteins. The Darwinian process is generally accepted as an optimal selective process and this should demonstrate that the natural protein are "the best" protein for our life activities. But many questions arise around the problem of the origin of life: are there other proteins (not natural) that might be able to have some good functionalities? Did a random protein become an existing one after the evolutionary process?
The thesis aims to present a new method, based on the evolutive paradigm and on model and variable selections, to optimize the hugh experimental space of "synthetic" proteins, starting from random sequences of amino acids, in order to reproduce the Darwinian paradigm of evolution. The interest of the research is to determine if there are other evolutive paths that can lead to new "synthetic" proteins able to enhance catalysis.
Some biological research teams are studying procedure to derive new proteins able to reproduce natural functionalities and two important branches have emerged:

- the rational protein design,
- the irrational protein design.

The former approach is an expert-driven application where biologists pick natural proteins and change some bonds among amino acids to create new proteins. Remarkably results are obtained with this methods even if its strongest advantage is also its main drawback: the so identified new proteins are only small variations of existing ones so that the search in the space of all the possible proteins is very limited. The latter approach is based on the idea of "try as much as you can" by no means of what is the reasoning behind the amino acids selection. The great advantage of this approach is that is a very cheap biological method, and some interesting results are obtained in this area too. On the other hand, biologists have to face the problem that there's no knowledge extraction during the selective procedure of the new proteins with a great lack of information about the bonds among amino acids.

The approach proposed in this thesis tries to overcome the two previous limitations, adopting a methods that investigate only very informative points, assuming only few biological information. We adopt an iterative strategy to search new biological entities, starting with very few random experimental trials and evolving them towards more informative regions of the experimental space. The evolutive path is driven by a new contrived algorithm that combined variable and model selection procedures.

The principal problem encountered in this experimentation is that we are treating an enormous amount of possible combinations of amino acids in a not known but complex experimental space in order to construct new proteins which have some good functionalities. Moreover, the real experimentation are costly and time consuming and only a very small number of experimental trials can be performed. The high dimensionality of the settings and the scarce number of possible observations is not trivial to handle, and the modeling procedure is made more difficult by the fact that we are working with categorical data. All these issues suggest to shift from the classical multivariate approach, and all its derivations, toward a better shaped algorithm, able to simultaneously address them.

The thesis is organized as follow: Chapter 1 describes the recent developments in gathering huge data sets. Technological improvement and new specific needs have brought the availability of massive data sets and the Chapter presents some examples where this trend is evident. Moreover, we present some high dimensional data analysis problems, currently statisticians have to face. Chapter 2 presents classical statistical procedures to treat variable selection and model assessment (Khalili & Chen 2007); an overview of filter, wrapper and embodied variables selection are presented and we focus in particular on the cutting-edge methods for high dimensionality such as LASSO (Tibshirani 1996) and Elastic Net (Zou & Hastie 2005). Chapter 3 deals with stochastic optimization approaches; we show the advantages of this approaches in optimizing high dimensional systems and we mainly focus on evolutionary algorithms (Eiben & Smith 2008, Forlin et al. 2008) and neural networks, (Fouskakis & Draper 2008, De March et al. 2009, 2008). Chapter 4 presents the new approach in order to handle the optimization of high dimensional problems: the Evolutionary Neural Network Design. We introduce the theoretical aspects of the proposed method and we compare it via some Monte Carlo simulations with benchmark methods. Eventually, we apply the ENN-Design to a real experimentation for discovering *"synthetic proteins"*.

# Chapter 1

# High Dimensional Data

This Chapter will be an overview of the High Dimensional (HD) problems that currently mathematicians, statisticians and data miners are trying to address. The approaches from these fields are often different from each other in the way of tackling high dimensional data. However, there is one main point that reconcile these scientific communities: something has to be done to reshape the classical approaches to better analyze HD data.
Donoho (2000), in a talk in memory of John Wilder Tukey, during the American Mathematical Society conference at the Statistics Department of Stanford University, wisely directed the attention of the audience to the new century challenges in high dimensional systems. The milestone book for statisticians and data miners by Hastie, Tibshirani and Friedman, the Element of Statistical Learning (Hastie et al. 2003), devotes many pages to this topic. In this Chapter we will mostly refer to many parts of those contributions.
The discovery of “synthetic” proteins problem, considered in this thesis, represents the very nature of the statistical challenges of high dimensional problems. In this real experimentation we can focus our attention on areas where active development of learning techniques demonstrates promising performance, but where signicant gaps in the theoretical foundations are still present. Filling those gaps will help to explain and improve upon this performance. The problems addressed in this thesis, are high dimensional categorical data, sparse and scarce data, supervised learning and variable selection and model prediction. The combination of these issues is highly biased (and therefore has high risk), but these challenging areas can benefit

from a combination of the statistics and computer science perspectives on learning from data.

## 1 The increasing interest in data collection

The current century is surely the *century of data*. Our society invests massively in the collection and processing of data of all kinds, on scales unimaginable until recently. Hyperspectral imagery, internet portals, financial tick-by-tick data, and DNA microarrays are just a few of the better-known sources, feeding data in torrential streams into scientific and business databases worldwide. In traditional statistical methodology, we assumed large number of observations and small numbers of well-chosen variables. But the meaning of "large" and "small" is time-variant; through the years the meaning of "large" has changed. In the early 1900's three could be considered as a large number of variable, in the 1980's 100 was large. Now the concept of large is becoming more complex.
The trend today is towards more observations but even more larger number of variables. We are seeing examples where the data collected on individual observation are curves, or spectra, or images, or even movies, so that a single observation has dimensions in the thousands or billions, while there are only tens or hundreds of observations available for study. Classical methods cannot cope with this kind of explosive growth of dimensionality of the observation matrix. Therefore high dimensional data analysis will be a very signicant activity in the future, and completely new methods of high dimensional data analysis will be developed.
In this context, many interesting researches are developed and two of the most influential principles in the high dimensionality have been originally discovered and cultivated by mathematicians: the blessings of dimensionality and the curse of dimensionality. The curse of dimensionality refers to the apparent intractability of systematically searching through a high dimensional space, the apparent intractability of accurately approximating a general high dimensional function, the apparent intractability of integrating a high dimensional function. The blessings of dimensionality are less widely noted and known, but they include the concentration of measure phenomenon, which means

that certain random fluctuations are very well controlled in high dimensions and the success of asymptotic methods, used widely in mathematical statistics and statistical physics, which suggest that statements about very high dimensional settings may be made where moderate dimensions would be too complicated.

## 1.1 Recent trends

Over the last few decades, data, data management, and data processing have become ubiquitous factors in modern life and work. Huge investments have been made in various data gathering and data processing mechanisms. The information technology industry is the fastest growing and most lucrative segment of the world economy, and much of the growth occurs in the development, management, and warehousing of streams of data for scientific, medical, engineering, and commercial purposes. Some recent examples include:

- Biotech Data: the fantastic progress made in the last years in gathering data about the human genome have spread statistical concepts toward biological fields. This is actually just the opening round in a long series of developments. The genome is only indirectly related to protein function and protein function are only indirectly related to overall cell function. Over time, the focus is likely to switch from genomics to proteomics and beyond. In the process more and more massive databases will be compiled.
- Financial Data: over the last decade, high frequency financial data have become available; in the early to mid 1990s data on individual currency trades, became available, tracking individual transactions. After the recent economic crisis, statistical models for long and high dimension streams of data are required to better predict trembling situations.
- Satellite Imagery: providers of satellite imagery collect vast databases of images. There $N$ is in the order of millions. Projects are in place to compile databases to resolve the entire surface of the earth to 1 meter accuracy (and already Google Earth app is a very interesting example). Applications of such imagery include natural resource discovery and agriculture.

- Hyper-spectral Imagery: it is now becoming common, both in airborne photographic imagery and satellite imagery to use hyperspectral cameras which record, instead of three color bands RGB, thousands of different spectral bands. Such imagery is presumably able to reveal subtle information about chemical composition and is potentially helpful in determining crop identity, spread of diseases in crops, in understanding the effects of droughts and pests, and so on. In the future, we can expect hyperspectral cameras to be useful in food inspection, medical examination, visual retrieval and so on.
- Consumer Financial Data: many transactions are made on the web; browsing, searching, purchasing are being recorded, correlated, compiled into databases, and sold and resold, as advertisers scramble to correlate consumer actions with pockets of demand for various goods and services.

The existing trends are likely to accelerate in the future, as each year new sensors and sources of data are invented. A very important additional trend is that society will more and more think of itself as a data-driven society, a consumer of data. Data industry, firms devoted to the creation and management of data (for example biotech or infotech companies), can be as valuable as firms creating physical tangible objects. Moreover, consumers are becoming data processors. For example, a few years ago, a $1024 \times 1024$ image was considered quite a substantial object for handling on a modern computer, and only computer scientists were really working with digital imagery. Now, consumer cameras costing a few hundreds of dollars, generate 10 times more precise images routinely. Consumers are becoming familiar with the process of capturing images, downloading them onto their home computers, processing them with various software tools, creating custom imagery. Such consumer acceptance will drive massive investment and technological development in data gathering and data analysis.

## 1.2 The high dimensional data analysis

Previous examples showed that we are in the era of massive automatic data collection, systematically obtaining many measurements, not knowing which ones will be relevant to the phenomenon of interest. This is a big break from the original assumptions behind many the tools being used in high dimensional data analysis today. For many of those tools, it was assumed that we are dealing with a few well-chosen variables, for example, using scientific knowledge to measure just the right variables in advance. But in the most recent applications, we do not know which variables to measure in advance anymore. The number of variables $p = p_n$ grows with $n$ in the asymptotic analysis, possibly very fast, so that $p_n \gg n \quad \text{for } n \to \infty$. Crucially, one has to assume in this setting that the data have "sparse structure", meaning that most of the variables are irrelevant for accurate prediction. The task is hence to filter-out the relevant subset of variables. While high dimensionality of a data set is evident from the start, it is usually not easy to verify structural sparseness. Often, sparseness is an assumption one has to make in the high dimensional case, as it is almost impossible to analyze non-sparse high dimensional data. In the words of Friedman et al. (2004), this is termed the "bet on sparsity":

> "Use a method that does well in sparse problems, since no procedure does well in dense problems".

This "post-classical world" is different in many ways from the *classical world.* The basic methodology which was used in the *classical world* no longer is applicable with good results. The theory underlying previous approaches to data analysis was based on the assumption of $p < n$, and $n \to \infty$. Many of the standard methods results concerned properties of observations which were multivariate normal, and used extensively tools from linear algebra and from group theory to develop some exact distributional results. These results commonly fail if $p \gg n$. Even worse, they envision an asymptotic situation in which $n \to \infty$ with $p$ fixed, and that also seems contradicted by reality, where we might even have $p \to \infty$ with $n$ remaining fixed. The $p \gg n$ case is not anomalous in real problems, on the contrary is becoming a state of the art. For many types of event we can think of, we have the potential of a very large number of measurable quantifying that event, and a relatively few instances of that event. In this scenario, classical assumptions fall short when analyzing huge Data and two new principles have been theorized: The course and the blessing of Dimensionality

### The Course of Dimensionality

According to Bellman (1961):

> "In view of all that we have said in the foregoing sections, the many obstacles we appear to have surmounted. What casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from earliest days?"

His interpretation: if our goal is to optimize a function over a continuous product domain of a few dozen variables by exhaustively searching a discrete search space defined by a crude discretization, we could easily be faced with the problem of making tens of trillions of evaluations of the function. Bellman argued that this curse precluded the use of exhaustive enumeration strategies, and argued in favor of his method of dynamic programming. From a statistical point of view, suppose we have a data set with $p$ variables, and we suppose that the first one is dependent on the others, and we call it $Y$, through a model of the form:

$$Y_i = f(x_{i,1}, \ldots, x_{i,p}) + \epsilon_i. \tag{1.1}$$

Suppose that $f$ is unknown, as we are not willing to specify a specific model for $f$ , such as a linear model. Instead, we are willing to assume merely that $f$ is a Lipschitz function of these variables and that $\epsilon_i$ variables are in fact i.i.d. Gaussian with mean 0 and variance 1. How does the accuracy of estimation depend on $n$? The very slow rate of convergence in high dimensions is the ugly head of the curse of dimensionality.

### The Blessing of Dimensionality

The plague of the course of dimensionality sometimes hides a proactive research of very small set of data which concentrates all the important feature of data distributions. This concentration of information around very small number of $q$ variables, where $q \subset p$, transform the "course of dimensionality" into the "blessing of dimensionality". There are many factors that can make a problem easier:

- only few of the inputs may be relevant,

- the input data may lie on a low-dimensional subset of the input space,
- special noise conditions may be valid,
- the expansion of the target function may be sparse in a predefined basis,
- the target function may be smooth.

We call these factors regularities of the problem. Some efforts are definitely devoted in the statistical learning to projected high dimensional problem on possibly much lower dimensional subspaces with the preservation of properties of high dimensional objects identifying the set of optimal variables (for example based on concentration of measure phenomena). New statistical models are trying to face the high dimensional problem, introducing in their estimate procedure relevant statistical aspects like:

- the prediction accuracy: the most common method in estimating coefficients is the least squares estimates, that often have low bias but large variance. Prediction accuracy can sometimes be improved by shrinking or setting some coefficients to zero. By doing so we sacrifice a little bit of bias to reduce the variance of the predicted values, and hence may improve the overall prediction accuracy.
- the easy interpretation: with a large number of predictors, we often would like to determine a smaller subset that exhibit the strongest effects. In order to get the "big picture", we are willing to sacrifice some of the small details.

One possible goal of this thesis is to develop a new method for function approximation in high dimensional spaces that facilitate fast and robust learning, and to use them to devise algorithms whose computational complexities do not grow rapidly with the dimension of the search space.

## 1.3 The drawbacks of standard statistical approaches for high dimensional data

Among the most familiar theoretical results in classical statistics are the "Laws of Large Numbers" and the "Central Limit Theorems". The former says that the sample mean of a random sample of size $n$ from a population has as a limit, in a well-defined sense, the population mean, as $n$ tends to $\infty$. The corresponding central limit theorem shows that the limiting distribution of the sample mean about the population mean (when scaled up by $\sqrt{n}$ ) is of the normal or Gaussian type. In statistics, such results are useful in deriving asymptotic properties of estimators of parameters, but their validity relies on there being, in theory at least, many observations per parameter (Johnstone & Titterington 2009).

Statisticians and data miners devote many efforts to reduce dimensionality of the problems in oder to relate to classical statistical theorems. Automatic model-building algorithms are familiar, and sometimes notorious, in the linear model literature. Variable selection and feature extraction are fundamental to dimensional reduction and knowledge discovery from massive data. Many variable selection procedures have been proposed in statistics aiming at creating parsimonious models that provide simple and interpretable relations among variables and that reduce forecasting errors.

But traditional statistical variable selection such as $C_p$ (Mallows 1973), AIC (Akaike 1973), BIC (Schwarz 1978) involves a combinatorial optimization problem, which is NP-hard, with computational time increasing exponentially with the dimension (Fan & Li 2006). Even the more raffinate techniques of model selection, when the search space of all possible models is too large, fail dramatically in determining models able to generalize.

Stepwise regression, the best subset selection, ridge regression, considered very attractive approaches in the 1990's, show very deep lacks in terms of performance and accuracy in high dimensionality. The stepwise selection, in all of its derivation (forward, backward or both), has to face the degree of freedom problem. In fact, these techniques are able to identify only models with complexity at most equal to the number of observation $n$, generating model with very scarce prediction accuracy.

The best subset selection minimizes the Residual Sum of Square (RSS)[1] subject to the number of non-zero coefficients equals some $q$, with $q \leq p$; the best subset selection produces a sparse model, that is very variable because of its inherent discreteness. Ridge regression (Hoerl & Kennard 1970) minimizes

[1]Given a model, the RSS is the sum of squares of the residuals, calculated as the difference between the observed values and the fitted values

RSS subject to a bound on the $\ell_2$ norm of the coefficients. By doing so, ridge regression shrinks the coefficients continuously toward zero. It achieves its better prediction performance through a bias-variance trade-off. However, ridge regression always keeps all the predictors in the model, so it cannot produce a parsimonious model.
Moreover, all the previous classical method have expensive computational costs that makes them infeasible procedure for high dimensional data analysis; furthermore, they are usually applied in continuous linear regression cases and become more and more unstable when the problems are non-linear or have a combinatorial explosion due to categorical variables. These procedure will be presented in Chapter 2, highlighting their limits in high dimensionality. The real challenge in high dimensional variable and model selection is to find out automatic procedure able to establish models with high accuracy in prediction but with the simpler structure as possible, in computational times constraints. To develop such kind of iterative procedure, a big deal is the trade-off between variance and bias.

#### Trade-off between bias and variance

If the dimension $p$ of the covariate matrix $X$ is large, as stated before, it is possible to get better predictions by shrinking some variables. Models with many covariates have low bias but high variance; models with few covariates have high bias but low variance. The best predictions come from balancing these two extremes. This is called the bias-variance tradeoff. The problem of deciding which variables to include in the model to achieve a good trade-off refers to the problem of **variable selection** and **model assessment**.
The variance-bias trade-off is absolutely universal and shows up under different guises in any kind of data modeling. This trade-off is the basis of many data mining approach to define the best models, and represent a comparative measure, useful when many different algorithms are developed. For example, suppose the data arise from a model $y_i = f(x_i) + \epsilon$, with $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma^2$ and that a good approximation of $f(x_i)$ is a generic model $\hat{f}(x_i)$. For simplicity here we assume that the values of $x_i$ in the sample are fixed in advance (non-random). The expected prediction error (PE) at a new

point, $x_0$, also known as generalization error, can be decomposed in:

$$\begin{aligned} PE(x_0) &= \sigma^2 + MSE(\hat{f}(x_0)) \\ &= \sigma^2 + (E[\hat{f}(x_0)] - f(x_0))^2 + E[(\hat{f}(x_0) - E[f(x_0)]]^2 \\ &= \sigma^2 + Bias^2(\hat{f}(x_0)) + Var((\hat{f}(x_0)). \end{aligned} \tag{1.2}$$

There are three terms in this expression. The first term $\sigma^2$ is the irreducible error, the variance of the new test target, and is beyond our control, even if we know the true $f(x_0)$. The second and third terms are under our control, and make up the *Mean Squared Error* (MSE) of $\hat{f}(x_0)$ in estimating $f(x_0)$. It is broken down into a bias component and a variance component. The *bias* term is the squared difference between the true mean $f(x_0)$ and the expected value of the estimate, $E[(\hat{f}(x_0)]$, where the expectation averages the randomness in the data, used to fit the model (training data). This term will most likely decrease with the complexity of the model. The *variance* term is simply the variance of the hypotheses, determined by how the prediction varies around its average prediction and usually increase with the complexity of the model. So, as the complexity varies, there is a bias-variance tradeoff.

### New directions in high dimensional data analysis

The statistical problems in high dimensional data analysis, presented in this introductory overview, paved the way to a very large discussion in the scientific community about the best approach to model and variable selection. Variable selection, in fact, can efficiently reduce the space of all possible variables $p$ by removing irrelevant, noisy and redundant features. The smaller the variables space, the easier it is to find correct models. *Irrelevant variables* can be removed without affecting learning performance (John et al. 1994). *Redundant varibles* are variables that do not contribute to give information about the system under study (Yu & Liu 2004). *Noisy varaibles* produce not desirable effects on the model accuracy and generalization (Lashkia & Anthony 2004). In data mining fields variable selection is commonly divided into three classes, which are filter, wrapper, and embedded approaches. In the filter approach, the input selection procedure is independent from the fitting of the final prediction model; in the wrapper approach, input variable subsets are ranked according to some estimate of generalization capability of the model; in the embedded approach, input selection is incorporated as a part of the fitting process. These classes are presented extensively in the Chapter 2, coupling for each of them the most important advantages and drawbacks. Additionally some of the newest statistical approaches will be presented, with particular attention of the class of $\ell_q$-norm penalized model.

# Chapter 2

# Variable Selection and Model Assessment

This Chapter presents a survey of variable selection and model assessment strategies. Terms like *variable selection, feature selection,* or *input selection*, terms deriving from computational and statistical data analysis fields, will be used indifferently. The input selection methods can be divided into three classes, which are filter, wrapper, and embedded approaches (Blum & Langley 1997, Guyon & Elisseeff 2003, Kohavi & John 1997). In the filter approach, the input selection procedure is independent from the fitting of the final prediction model. In the first phase, a subset of input variables is identified according to some measure and only the best are taken into account in the phase of modeling. This second phase, where the final prediction model is fitted, uses only the selected input variables. The filter approach is computationally tractable, since the input selection process is fast and the final prediction model is fitted only once. On the other hand, the subset of input variables that is found to be optimal in the first phase may not be optimal in the second phase.

In the wrapper approach, input variable subsets are ranked according to some estimate of generalization capability of the model; Cross-validation error (Stone 1974) and Bootstrap (Efron 1983) are often used in the estimate of the parameters. The variable selection process is supported by a general-purpose search algorithm that proposes promising combinations. Heuristic methods like greedy search, forward selection and backward elimination, step-

wise selection are the most typical search algorithms. The performance of the final prediction model is optimized directly, which is advantageous over the two-phase filter approach. However, the wrapper approach is computationally more expensive, because the potentially time-consuming model fitting has to be performed several times.
In the embedded approach, input selection is incorporated as a part of the fitting process. It is highly specific to the model structure compared with the more universal wrapper approach, where the model is treated as a black box. Certain penalization techniques offer another more direct category of embedded methods. The model fitting is penalized in such a way that the parameters associated with some input variables become zero during the estimation process. Estimated generalization capability (the predictive error is one of the most used measure) is often used to determine the stopping condition for the amount of penalization in direct methods. The embedded approach is usually computationally much lighter than the wrapper approach.
This Chapter presents an overview of statistical methods related to the three variables selection classes with a particular attention to the embedded methods in which the penalized regression has very large interests in the statistical learning and in current applications. All the three categories of features selection can be applied both in *supervised learning* and in *unsupervised learning.* In *supervised learning*, the goal is to predict the value of one or more output measures, $Y_d$, based on a number of input measures, $\mathbf{X}$; in *unsupervised learning*, there is no output measures, and the goal is to describe the associations and patterns among a set of input measures. This thesis refers only to supervised learning algorithm, since the structure of data that will be analyzed in Chapter 4 contain both input variables and output variable. Most of the variable selection and model assessment methods make strong assumptions on the problem, for example they often assume that the model generating the data (called also true model) belongs to the family of linear multiple regression models.
Usually the emphasis in the variable model selection depends on the goal of the regression analysis. In general, one can distinguish between the following purpose, according to Kardaun (2005):

1. estimation of the regression parameters $\beta = \{\beta_0, \beta_1, \ldots, \beta_p\}$;

2. estimation of the variance of the random errors $\sigma^2$;
3. estimation of standard deviations of the estimated regression parameters;
4. testing of hypotheses about the regression parameters;
5. prediction of the response variable (interpolation and extrapolation);
6. outlier detection;
7. determination of the goodness-of-fit.

In this thesis we are more interested in the model prediction accuracy and for the problem that will be analyzed some typical assumption of the linear regression models cannot be done. The new method that is presented in Chapter 4 of this thesis has taken inspiration on the above mentioned methods and substantially ensembles inspiring techniques both of the filter and of the wrapper approaches. It is then presented as a new statistical procedure to determine the best model in terms of prediction when data are qualitative and scarce and when the dimensionality and the combinatorial complexity is particularly high.

## 1 Filters methods

Many variable selection algorithms, that are considered as filters methods, include variable ranking as a principal or auxiliary selection mechanism because of its simplicity, scalability, and good empirical success. Several papers present variable ranking as a baseline method (see, e.g., Bekkerman et al. (2003), Caruana et al. (2003), Forman (2003)). Variable ranking is not necessarily used to build predictors. The ranking criteria hereafter are defined for individual variables and independently of the context of others. Variable ranking makes use of a scoring function $S(X_j)$ computed from the values $X_j$ and $Y$. By convention, we assume that a high score is indicative of a valuable variable and that we sort variables in decreasing order of $S(X_j)$. To use variable ranking to build predictors, nested subsets incorporating progressively more and more variables of decreasing relevance are defined. Following the classication of Kohavi & John (1997), variable ranking is a filter method: it is a preprocessing step, independent of the choice of the predictor and the model. Still, under certain independence or orthogonality assumptions, it may be optimal with respect to a given predictor. Even when variable ranking is not optimal, it may be preferable to other variable subset selection methods because of its computational and statistical scalability: computationally, it is efficient since it requires only the computation of $p$ scores and sorting the scores; statistically, it is robust against overfitting because it introduces bias but it may have considerably less variance (Hastie et al. 2003). We introduce some additional notation: if the input matrix $\mathbf{X} = \{x_{11}, \ldots, x_{ij}, \ldots, x_{np}\}$ can be interpreted as the realization of a random vector drawn from an underlying unknown distribution, we denote by $X_j$ the random variable corresponding to the j-*th* component of $\mathbf{X}$. Similarly, $Y$ will be the random variable of which the outcome $y_i$ is a realization. We further denote by $X_j$ the $n$ dimensional vector containing all the realizations of the j-*th* variable, and by $y_i = \{y_1, \ldots, y_n\}$ the $n$ dimensional vector containing all the target values. The most common measure for ranking variables in supervised learning are **correlation criterion** e **information criterion**.

## 1.1 Correlation criterion

Let us consider first the prediction of a continuous outcome $Y$. The Pearson correlation coefficient is defined as:

$$\rho(X_j, Y) = \frac{cov(X_j, Y)}{\sqrt{var(X_j) var(Y)}}, \tag{2.1}$$

where *cov* designates the covariance and *var* the variance. The estimate of $\rho(X_j, Y)$ is given by

$$r(X_j, Y) = \frac{\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{2.2}$$

where the bar notation stands for an average over the index $j$. This coefficient is also the cosine between vectors $X_j$ and $Y$, after they have been centered (their mean subtracted). Although the $r(X_j, Y)$ is derived from $\rho(X_j, Y)$ it may be used without assuming that the input values are realizations of a random variable. In linear regression, the coefcient of determination, which is the square of $r(X_j, Y)$, represents the fraction of the total variance around the mean value $\bar{y}$ that is explained by the linear relation between $X_j$ and $Y$.

Therefore, using $r^2(X_j, Y)$ as a variable ranking criterion enforces a ranking according to goodness of linear fit of individual variables.
Correlation criteria such as $r(X_j, Y)$ can only detect linear dependencies between input variables and output variables. A simple way of lifting this restriction is to make a non-linear fit of the output with single variables and rank according to the goodness of fit. Because of the risk of overfitting, one can alternatively consider using non-linear preprocessing (e.g., squaring, taking the square root, the log, the inverse, etc.) and then using a simple correlation coefficient. Other measures in this category are basically variations of the above formula, such as *least square regression error* and *maximal information compression index* (Mitra et al. 2002).

## 1.2 Mutual information criterion

Several approaches to the variable selection problem using information theoretic criteria have been proposed. Many of them rely on empirical estimates of the mutual information between each variable and the target an it is calculated as:

$$I(X_j, Y) = \int_{X_j} \int_{Y} p(X_j, Y) log \frac{p(X_j, Y)}{p(X_j)p(Y)} dx\, dy, \tag{2.3}$$

where $p(X_j)$ and $p(Y)$ are the probability densities of $X_j$ and $Y$, and $p(X_j, Y)$ is the joint density. The criterion $I(X_j, Y)$ is a measure of dependency between the density of variable $X_j$ and the density of the output $Y$. The difficulty is that the densities $p(X_j)$, $p(Y)$ and $p(X_j, Y)$ are often all unknown and are hard to estimate from data.
In discrete or nominal variables cases the mutual information criterion is the density probabilities estimated as the empirical frequency counts. Mutual information becomes:

$$I(X_j, Y) = \sum_{X_j} \sum_{Y} P(\mathbf{X} = X_j, Y = y) log \frac{p(\mathbf{X} = X_j, Y = y)}{p(\mathbf{X} = X_j)p(Y = y)}. \tag{2.4}$$

Several variations on mutual information have been proposed to suit various need, but for the sake of this thesis we introduce only these two filters criterion.

## 2 Wrapper methods

Differently to the filtering approach, in the wrapper methods input variable subsets are ranked according to some estimate of generalization capability of the model. The wrapper methodology offers a simple and powerful way to address the problem of variable selection, regardless of the chosen learning machine. In fact, the learning machine is considered a perfect black box and the method lends itself to the use of off-the-shelf machine learning software packages. In its most general formulation, the wrapper methodology consists in using the prediction performance of a given learning machine to assess the relative usefulness of subsets of variables. In practice, one needs to define:

i) how to search the space of all possible variable subsets;

ii) how to assess the prediction performance of a learning machine to guide the search and halt it;

iii) which predictors to use.

An exhaustive search can conceivably be performed, if the number of variables is not too large. But, the enumeration of all the subspaces is a NP-hard problem and the search becomes quickly computationally intractable. A wide range of search strategies (Kohavi & John 1997) can be used, including greedy strategy, best-first, branch-and-bound, simulated annealing, genetic algorithms and their performances are usually evaluated using a validation set, by cross-validation or by bootstrap. Many computer algorithms have been designed to automate the process of finding relevant subsets of variables and good models in terms of fitting and prediction accuracy without using the "brute force" method of fitting all subsets. It is important to note that there are in fact two separate goals that we might have in mind:

- *Model and variable selection*: estimating the performance of different models in order to choose the best one.
- *Model assessment*: having chosen a final model, estimating its prediction error (generalization error) on new data.

## 2.1 Model and variable selection: the search in the space of all possible variable subsets

In order to find the (nominally) best fitting subset of explanatory variables, theoretically the only way is to compare all possible subsets. In practice this concept is often not realistic, since there are $2^p - 1$ different, non-empty subsets of a given set of all candidate variables $X_j$ with $j = 1, \dots, p$, which may be inconveniently large even in a computerised environment. Furthermore, the danger of overfitting the data is prominent in such an approach. Moreover, to evaluate search algorithms many goodness-of-fit criteria have been defined to compare various subsets. Among them the most popular are:

- *Adjusted* $R^2$: it is a suitable measure to compare models with different number of parameters. It adjust the coefficient of determination $R^2$ penalizing models with higher number of variables:

$$R^2_{adj} = 1 - \frac{MS_{Res}}{MS_{Tot}} = 1 - \frac{(1-R^2)(n-1)}{n-(p+1)}, \tag{2.5}$$

  which uses, in contrast to $R^2$, respectively the mean sum of squares of the residuals[1] and the total sum of square[2] of the model. $R^2_{adj}$ may decrease if variables, entering the model do not add significantly to the model fit.

- Mallows $C_p$ statistics: it addresses the issue of overfitting, in which model selection statistics such as the residual sum of squares always get smaller as more variables are added to a model. This criterion is

$$C_p = \frac{RSS_q}{s^2} + M(k+1) - n, \tag{2.6}$$

  where $s^2$ is an estimator of $\sigma^2$ from the model containing all $p$ variables plus intercept, and $M$ is a suitable constant (larger is the value of $M$, smaller is the number of selected variables). If a subset containing $q$ variables is approximately correct, then both $s^2 = \frac{RSS_q}{(n-q-1)}$ and $s^2 = \frac{RSS_p}{(n-p-1)}$ are reasonable estimators for $\sigma^2$ where $RSS_q$ and $RSS_p$ represent the residuals sum of squared of the model respectively estimated on a subset of $q$ variables and on the complete model.

[1]The $MS_{Res}$ is calculated as $\frac{RSS}{n}$

[2]The $MS_{Tot}$ is calculated as $\frac{\sum_{i=1}^{n}(y_i-\bar{y})^2}{n}$

- Akaike Information Criterion (AIC): it proposes to choose a model that minimizes the Kullback-Leibler (KL) divergence of the fitted model from the true model. In linear regression, it considers the maximum likelihood estimator (MLE) $\hat{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_p)$ of the parameter vector $\theta$ and showed that, up to an additive constant, the estimated KL divergence can be asymptotically expanded as:

$$AIC = -log(L(\hat{\theta})) + \lambda dim(\hat{\theta}) = -log(L(\hat{\theta})) + \lambda \sum_{j=1}^{p} I(\hat{\theta}_j = 0), \quad (2.7)$$

  where $log(L(\theta))$ is the log-likelihood of the function, $dim(\theta)$ represent the number of non-zero parameters in the model and $\lambda$ is fixed equal to one. The $\lambda$ parameters is a regularization parameter that will be broadly presented in the embedded methods.
- Bayesian Information Criterion (BIC): it is applicable in settings where the fitting is carried out by maximization of a log-likelihood, like AIC. The generic form of BIC is the same of Equation 2.7, but the parameter $\lambda$ assume the value $\frac{log(n)}{2}$. BIC tends to penalize complex models more heavily, giving preference to simpler models.
- The minimum description length (MDL): this approach gives a selection criterion formally identical to the BIC approach, but motivated from an optimal coding viewpoint.

Heuristic algorithms are usually adopted in the search of the best model. The stepwise regression is widely used and it approaches the variable and model selection through the above mentioned goodness-of-fit criteria as stopping rules; it can be classified in three different classes:

- *Forward Selection*: this search algorithm starts with one variable. Defined which criterion to use, in each step of the forward selection the variable that causes the largest decrease of the criterion value (or increase for the $R^2_{adj}$) is added into the model. In the first step, the variable that has the highest score for the selected criteria respect to the response variable is added; then, in following steps the variable entering the model has the highest partial score respect to the response variable, given all variables that are already included in the model. The sequential procedure lasts until the stopping rule is achieved, that usually correspond to a very low improvement of the model performance, given by the new variable to include in the model.

- *Backward Selection*: this method starts with all candidate variables included in the model. Defined which criteria to use, in each step of the forward selection the variable whose elimination determines the largest decrease of the criterion value (or increase for the $R^2_{adj}$) is deleted from the full model.

It should be noted that through addition or elimination of a variable the "importance" of other variables can change. Therefore usually is suggested to use a combination of *forward* and *backward selection*:

- Stepwise Selection: is an improvement of the forward selection technique, and it has been proposed as a technique that combines advantages of forward and backward selection. At any point in the search, a single predictor may be added or deleted. Commonly, the starting subset is the empty set. Stepwise selection evaluates more subsets than the other two techniques, tending to produce better subsets. The drawback of this methods is that to find better subsets the computational speed is reduced.

All these techniques can cause model misspecification and large variability that can prevent the discovery of the optimal model. To overcome some limitations of the previous methods when the search space of all possible subspace has combinatorial explosion, new kind of search algorithms were developed. These algorithms are better suited to optimization problems in high dimensional systems and where traditional numerical methods cannot be applied at all due to the discreteness (Winker 2000): these algorithms, called "Meta-heurisitics" can better perform in situation where:

- the number of all possible models is vast,
- the number of important interactions among variables is huge,
- very high multicolinearity exists,
- the models are referred to categorical variables with many factors for each of them,
- strong non-linearities are present in the unknown but "true" model.

These algorithms performs "meta-heuristic search" in the whole space of possible models. Often the term meta-heuristic is linked to algorithms mimicking some behavior found in nature, e.g. the principle of evolution through selection and mutation (genetic algorithms), the annealing process of melted iron (simulated annealing), the self organization of ant colonies (ant colony optimization) or the fly of birds swarm (Particle swarm optimization) (Gilli & Winker 2008).

Among these algorithms, Genetic Algorithms and Simulated annealing, are the most used in model selection. An overview of some of these and other meta-heuristics will be provided in Chapter 3.

## 2.2 Model assessment: the prediction performance of a learning machine to guide the search and halt it

Previous automatic procedures for selecting a model are directed to find the best possible fit of the dataset at hand. The next step toward a reliable model is the model validation procedure. This can be carried out by checking the model with another, independent dataset. Validation is important, above all if one applies automated model selection, because of the danger of overtting and selection bias on one hand and omission bias on the other.

Overfitting occurs if the response variable can be described by a submodel with $q$ predictors, but the data have been fitted with $p \gg q$ variables. Selection bias occurs if the response variable seems to be explainable by $q$ regressors, but in fact these $q$ regressors have been heavily selected from a considerably larger set of $p$ regressors. Omission bias occurs if important regression variables are not included in the model. Ideally, both types of bias should be avoided.

To avoid these biases, if we are in a data-rich situation, the best approach for both problems is to randomly divide the dataset into three parts as in Figure 2.1: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model. In most of the recent applications, however, the number of available data is scarce and the validation procedure used only the

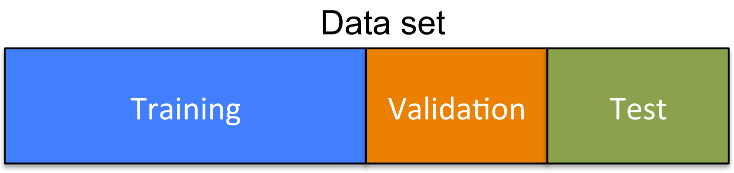


Figure 2.1: Data-split into training, validation and test set.

training and test set partition. *cross-validation* and *bootstrap* are currently the widest applied procedure in the assessment of the models.

### Cross-validation

As noticed in the early 30s by Larson (1931), training an algorithm and evaluating its statistical performances on the same data yields an overoptimistic result. Cross-validation was raised to fix this issue, starting from the remark that testing the output of the algorithm on new data would yield a good estimate of its performance (Mosteller & Tukey 1968, Stone 1974). In most real applications, only a limited amount of data is available, which leads to the idea of splitting the data into 2 parts: one bigger part of the data (the training set) is used for training the algorithm or fitting the model, and the remaining data (the test set) are used for evaluating the performance of the algorithm or testing the model. The test set plays the role of "new data" where to evaluate the prediction accuracy of the models. A single data split yields a *validation* estimate of the prediction accuracy, and averaging over several splits yields a *cross-validation* estimate. Several splitting procedure are developed and for a complete description we refer to Arlot & Celisse (2010). For the sake of this thesis, we present the most applied, *k-fold cross-validation* and *leave-one-out cross-validiation*.

- *k-fold cross-validation*: since data are often scarce, the validation estimate cannot be usually a good estimate. To finesse the problem, *k-fold cross-validation* uses part of the available data to fit the model, and a different part to test it. In particular, the algorithm splits the data into $k$ roughly equal-sized parts; in Figure 2.2, for example, data are divided into $k = 4$ parts. By choosing $k$, the initial set of experimental

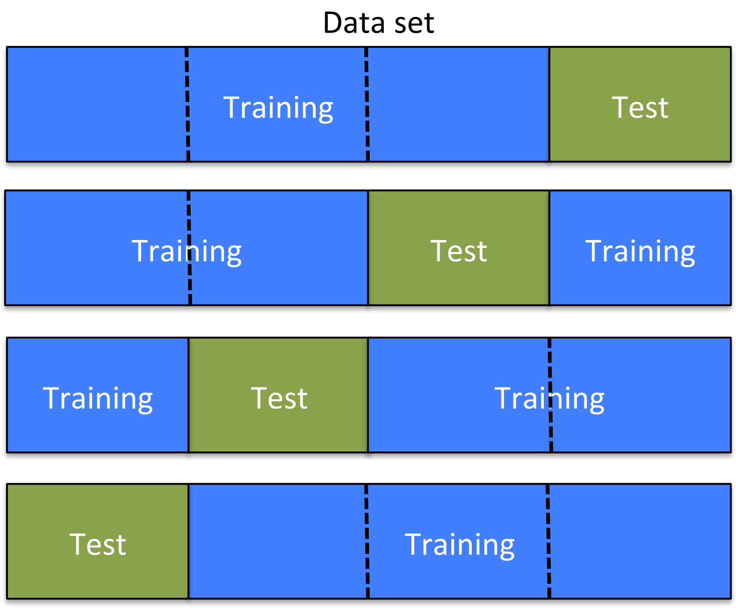


Figure 2.2: k-fold cross-validation with k=4.

points is partitioned k-folds, and each partition is used in turn exactly once as a test set. Then, the procedure fits the model on $k-1$ parts of the data, and calculate the Root Prediction Error (RPE)

$$RPE_k = \sqrt{\frac{\sum_{t=1}^{m^*}(Y_t - \hat{Y}_t)^2}{m^*}}, \tag{2.8}$$

where $m^*$ represents the number of observation of the $k$-th part of the data, for each turn of the procedure. Then the cross-validation estimate of prediction error is then the average of the $k$ root prediction errors.

$$CVRPE = \frac{1}{k}\sum_{s=1}^{k} RPE_s, \tag{2.9}$$

providing a good estimate of the generalization of the model.

- *leave-one-out cross-validation* involves using a single observation (instead of $k$ in the *k-fold cross-validation*) from the original sample as the test set, and the remaining observations as the training data (as shown in Figure 2.3). This is repeated such that each observation in the sample is used once as the test data. Formulas 2.8 and 2.9 are still

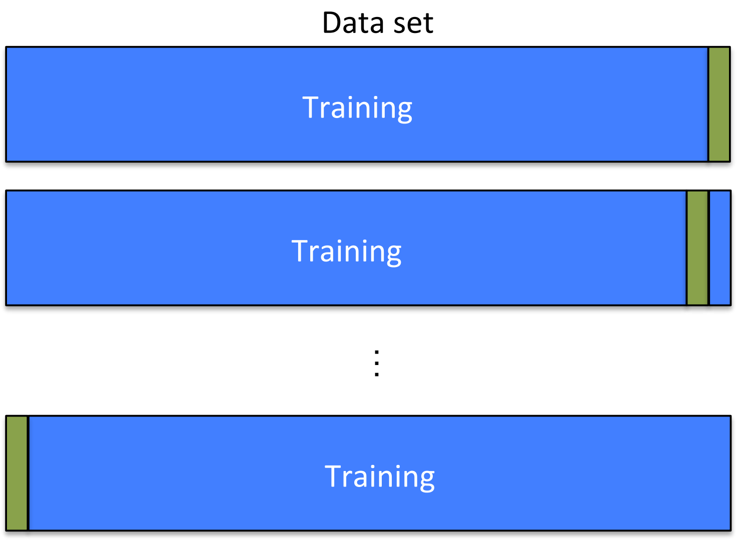


Figure 2.3: Leave-one-out cross-validation.

valid in this type of cross-validation, noting that in this case $k = n$ and $m^* = 1$. Leave-one-out cross-validation is usually very expensive from a computational point of view because of the large number of times the training process is repeated.

Even in the choice of the kind of cross-validation, the trade-off between variance and bias, presented in the previous Chapter, plays an important role. In fact, the *leave-one-out cross-validation* estimate is approximately unbiased for the prediction error, but can have high variance because the $n$ training sets are very similar one another. On the other hand, the $k$*-fold* cross-validation (usually $k = \{5, 10\}$) has lower variance but the bias could be very high, depending on how the performance of the learning method varies with the size of the training set.

### Bootstrap

The bootstrap is another general tool for measuring statistical models accuracy and it is used typically for estimating the expected prediction error. Bootstrap techniques (also called *resampling computation techniques*) have introduced new advances in modeling and model evaluation. Using resampling methods to construct a series of new samples which are based

on the original data set, allows to estimate the stability of the parameters. This technique was introduced by Efron (1979) and represents a simulation technique based on the empirical distribution of the observed sample. Let $\mathbf{X} = \{x_1, \ldots, x_n\}$ be an $n$ sample, with an unknown distribution function $F$, depending on an unknown real parameter $\theta$. The problem consists in estimating this parameter $\theta$ by a statistic $\hat{\theta} = s(x)$ function of the sample data and in evaluating the estimate accuracy, although the distribution $F$ is unknown. In order to evaluate this accuracy, $B$ random samples are built from the initial sample $\mathbf{X}$, by re-sampling. These samples are called bootstrapped samples and denoted by $\mathbf{X}^{*b}$
A bootstrapped sample $\mathbf{X}^{*b} = \{x_1^{*b}, \ldots, x_m^{*b}\}$ is built by a random drawing (with repetitions) in the initial sample $\mathbf{X}$: the distribution function of a bootstrapped sample $\mathbf{X}^{*b}$ is $\hat{F}$, i.e. the empirical distribution of $\mathbf{X}$ . A bootstrap replicate of the estimator $\hat{\theta} = s(x)$ will be $\hat{\theta}^{*b} = s(\mathbf{X}^{*b})$
This algorithm is often presented as a model selection techniques with many derivation from the original idea as presented in Efron & Tibshirani (1994), Shao (1996), Efron & Tibshirani (1997), Kallel et al. (2002).
More in detail, let $\mathbf{X}$ be a data set of size $n$

$$\mathbf{X} = (\{x_1, y_1\}, \ldots, \{x_i, y_i\}) \text{ with } i = 1, \ldots, n, \tag{2.10}$$

where $x_i$ is the $i$-th value of a $p$-vector of explanatory variables and $y_i$ is the response of the system. From the original data set $\mathbf{X}$, the algorithm generates $B$ bootstrapped data set $\mathbf{X}^{*b}$, from an uniform drawings of $n$ data points in $\mathbf{X}$, with replacement. For any generated data sest $\mathbf{X}^{*b}$ an estimator of the model parameters vector $\theta$, called $\hat{\theta}^{*b}$, is found. So the bootstrap procedure provides $B$ replications $\hat{\theta}^{*b}$ for the model. Then the initial data set $\mathbf{X}$ is used as the test set, and evaluate, for each $b = 1, \ldots, B$, the residuals estimate and consequently the mean square error. Even the bootstrap methods refer to the bias-variance decomposition and, in fact, this procedure is used to estimate the residual variance of the model, estimated from the bootstrapped samples, and the average bias of the model estimate.
All the methods presented before had lot of success in many applicable fields, above all when the collected data are in $n \gg p$ condition and under strict assumption like linearity, omoschedasticity and incorrelation (typical of the

multiple linear regression models); substantial innovations are developed with bootstrap and cross-validation to be more flexible to be applied without constraints due to the model type, but as a consequence, the required computational time is increased enormously. In high dimensional settings many efforts are devoted to create more efficient techniques, able to estimate sufficiently good models in acceptable computational time. Cross-validation and bootstrap represent two examples of these efforts but these methods have not always proven to be adequate to address the model assessment problem in high dimensionality.

# 3 Embedded Methods

Embedded methods differ from other feature selection methods in the way feature selection and learning algorithm interact. Filter methods do not incorporate learning. Wrapper methods use a learning machine to measure the quality of subsets of features without incorporating knowledge about the specific structure of the classification or regression function. They can therefore be combined with any learning machine. In contrast to filter and wrapper approaches, in embedded methods the learning part and the feature selection part can not be separated; the structure of the class of functions under consideration plays a crucial role (Lal et al. 2006). Some embedded methods guide their search by estimating changes in the objective function value incurred by making moves in variable subset space. Combined with greedy search strategies (backward elimination or forward selection) they yield nested subsets of variables (Guyon & Elisseeff 2003).
The basic idea of the embedded methods is to contemporary define the best model in term of goodness-of-fit and generalization capability, putting inside the training process some variable selection procedure that allow to delete not-important features. Among the embedded methods, the $\ell_q$-norm penalized regression models represents a vast research area, but many other classical statistical algorithms have been modified to embed the variable selection directly into the learning algorithms. Some examples are CART (Breiman et al. 1984) with the pruning rules inside the learning process, Random Forest (Breiman 2001) with ranking methods for variable selection, Support

Vector Machine (Boser et al. 1992) with penalties or kernel methods variable selection.

## 3.1 $\ell_q$-norm penalized regression models

Formula 2.7 presents a very simplified penalized methods, where the regularization parameter $\lambda$ was set equal to one. A more general formulation of Formula 2.7 can be the follow:

$$-log(L(\theta)) + \lambda\|\theta\|_0, \tag{2.11}$$

where the $\|\theta\|_0$, called $\ell_0$-norm of $\theta$, counts the number of non-vanishing components in $\theta$ and $\lambda \geq 0$ is a regularization parameter. Given $\|\theta\|_0 = q$, the solution to Formula 2.11 is the subset with the largest maximum likelihood among all subsets of size $q$. The model size is then chosen to maximize Formula 2.11 among the subsets of sizes $q$ with $1 \leq q \leq p$. $\ell_0$ regularization arises naturally in many classical model selection methods. It gives a nice interpretation of best subset selection and admits nice sampling properties. However, the computation is infeasible in high dimensional statistical problems, therefore, other penalty functions should be used, providing a more general formulation:

$$-\frac{1}{2n}log(L(\theta)) + \lambda\sum_{j=1}^{p} p_j|\theta_j|, \tag{2.12}$$

where $log(L(\theta))$ is the log-likelihood function and $\lambda\sum_{j=1}^{p} p_j|\cdot|$ is a penalty function indexed by the regularization parameter $\lambda \geq 0$. The dependence of the penalty function on $j$ allows to incorporate prior information. For instance, we may wish to keep certain important predictors in the model and choose not to penalize their coefcients. For simplicity, in this thesis, we assume that the penalty functions for all coefficients are the same, denoted by $p(|\cdot|)$ and $\lambda p(\beta)$ as $p_\lambda$. By maximizing the penalized likelihood in 2.12, variables are selected and simultaneously their associated regression coefficients are estimated. Those variables whose regression coefficients are estimated as zero are automatically deleted. A natural generalization of penalized $\ell_0$-regression is the penalized $\ell_q$-regression, called bridge regression in Frank & Friedman (1993), in which $p_\lambda|\theta_j| = \lambda|\theta|^q$ for $0 < q \leq 2$.

According to the initial aim of $\ell_q$-penalized methods, the linear regression case is presented. Many other improvement are done in the last years to generalize the $\ell_q$ penalty for logistic regression (Meier et al. 2008), generalized linear models (Roth & Fischer 2008) and non linear models (Friedman et al. 2010). In the regression form, the penalized likelihood 2.12 is equivalent, up to an affine transformation of the log-likelihood, to the penalized least squares (PLS):

$$PLS(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\beta)^2 + \lambda\sum_{j=1}^{p} p_j(\beta). \tag{2.13}$$

According to Fan & Li (2001) penalty functions have to obey three properties:

- *Sparsity*: the resulting estimator should automatically set small estimated coefcients to zero to accomplish variable selection.
- *Unbiasedness*: the resulting estimator should have low bias, especially when the true coefficient $\beta$ is large.
- *Continuity*: the resulting estimator should be continuous to reduce instability in model prediction.

The penalty function is becoming a vast field of research and many different kind of penalties are introduced in statistical model and variable selection, including SCAD (Fan & Li 2001), adaptive LASSO (Zou & Hui 2006), relaxed LASSO (Meinshausen 2007) among others.

### Ridge Regression

One of the most used penalty method is the ridge regression. Ridge regression shrinks the regression coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squares when the penalty function uses $q = 2$ so that the penalty becomes $\lambda\sum_{j=1}^{p}\beta^2$. The estimate of the coefficient are then calculated as:

$$\hat{\beta}^{ridge} = \arg\min_{\beta}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\beta)^2 + \lambda\sum_{j=1}^{p}\beta^2. \tag{2.14}$$

Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of $\lambda$, the greater the amount of shrinkage. The coefficients are shrunk toward zero (and each other). An equivalent way to write the ridge problem is

$$\begin{aligned}\hat{\beta}^{ridge} &= \arg\min_{\beta} \sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\beta)^2 \\ &\quad \text{subject to } \sum_{j=1}^{p}\beta^2 \leq t.\end{aligned} \tag{2.15}$$

In matrix formula the ridge regression can be seen as:

$$RSS^{ridge}(\lambda) = (y - \mathbf{X}\beta)^T(y - \mathbf{X}\beta) + \lambda\beta^T\beta, \tag{2.16}$$

whose solution is:

$$\hat{\beta}^{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T Y. \tag{2.17}$$

As a continuous shrinkage method, ridge regression achieves its better prediction performance, minimizing the bias-variance trade-off presented in Formula 1.2. However, ridge regression cannot produce a parsimonious model, for it always keeps all the predictors in the model. In contrast, the use of an $\ell_1$ penalty does reduce terms to zero. This yields LASSO.

## LASSO

The LASSO is a shrinkage method like ridge, with subtle but important differences. The LASSO estimate is defined by

$$\begin{aligned}\hat{\beta}^{LASSO} &= \arg\min_{\beta} \sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\beta)^2 \\ &\quad \text{subject to } \sum_{j=1}^{p}|\beta| \leq t.\end{aligned} \tag{2.18}$$

This latter constraint makes the solutions nonlinear in the $y_i$, and there is no closed form expression as in ridge regression. In fact, the regression coefficients are estimated as

$$\hat{\beta}^{LASSO} = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T Y - \frac{\lambda}{2}\mathbf{w}), \tag{2.19}$$

where the elements $w_j$ of $\mathbf{w}$ are either $\pm 1$, depending on the sign of the corresponding regression coefficient $\beta_j$. This is a least squares problem with two

inequality constraints (there are $2^p$ possible sign patterns for the coefficients) and can be solved using complicated quadratic programming methods.
The LASSO has many desirable features that have made it a popular regression algorithm. It is, at the same time, a shrinkage estimator of $\beta^{OLS}$ (OLS coefficients are shrunk towards the origin) and a variable selection technique (Izenman 2008), performing a kind of continuous subset selection (Hastie et al. 2003). The value of $t$ controls both the amount of shrinkage and the number of nonzero coefficients. In particular, a smaller value of $t$ produce a smaller subset of nonzero coefficients. The entire LASSO sequence of paths can be generated by a slight modication of the LAR algorithm (called LARS), which is a procedure that efficiently combine LASSO, Forward-Stagewise and LAR algorithms (Efron et al. 2004). Although the LASSO has shown several successes in many situations, it has some limitations. Let consider the following three scenarios:

(a) in the $p \gg n$ case, the LASSO selects at most $n$ variables before it saturates, because of the nature of the convex optimization problem. This seems to be a limiting feature for a variable selection method. Moreover, the LASSO is not well defined unless the bound on the $\ell_1$-norm of the coefficients is smaller than a certain value.
(b) If there is a group of variables among which the pairwise correlations are very high, then the LASSO tends to select only one variable from the group and does not care which one is selected.
(c) For $n > p$ situations, if there are high correlations between predictors, it has been empirically observed that the prediction performance of the LASSO is dominated by ridge regression.

Scenarios (a) and (b) make the LASSO an inappropriate variable selection method in many real situations (Zou & Hastie 2005). Moreover it is known that the convex $\ell_q$ penalty with $q > 1$ does not satisfy the sparsity condition, whereas the convex $\ell_1$ penalty does not satisfy the unbiasedness condition, and the concave $\ell_q$ penalty with $0 \leq q < 1$ does not satisfy the continuity condition. In other words, none of the $\ell_q$ penalties satisfies simultaneously all the three conditions (presented in Section 3.1), considered fundamental in variable and model selection. A very competitive approach is then created, to assolve of the three properties: the Elastic Net procedure

**Elastic Net**

Similar to the LASSO, the Elastic Net simultaneously does automatic variable selection and continuous shrinkage, and it can select groups of correlated variables. It is like a stretchable fishing net that retains "all the big fish" and it outperforms the LASSO in terms of prediction accuracy (Zou & Hastie 2005). The regression model penilized with the Elastic Net is:

$$\begin{aligned}\hat{\beta}^{enet} &= \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \beta)^2 \\ &\text{subject to } \sum_{j=1}^{p} |\beta| \leq t_1 \text{ and } \sum_{j=1}^{p} |\beta|^2 \leq t_2.\end{aligned} \tag{2.20}$$

The penalty function is a convex combination of the LASSO and ridge penalty. The first term encourages highly correlated features to be averaged, while the second term encourages a sparse solution in the coefficients of these averaged features. For the Elastic Net the regression coefficients are estimated as

$$\hat{\beta}^{enet} = (\mathbf{X}^T\mathbf{X} + \lambda_2 I)^{-1}(\mathbf{X}^T Y - \frac{\lambda_1}{2} w), \tag{2.21}$$

where it is immediate to see the similarities with ridge regression and LASSO models.

Sometimes, the estimate of the model is a secondary task, respect to the generalization aspect. If the main objective of modeling is to have minimal Prediction Error (PE), many other algorithms can be better shaped. Moreover in many real applications, we have very little information about the system, so general assumption, requested by methods presented in this Chapter, cannot be defined. Therefore, other algorithms are developed with more general, robust and powerful search mechanism (Back et al. 1997). They usually possess other characteristics that are desirable for problems involving above all intractably large and highly complex search spaces. They are often suited to provide faster convergence in optimization problems when large number of variables and complex system response functions are present. These methods are called meta-heuristics

# Chapter 3

# Evolutionary Algorithms and Neural Network models

The basic concept of heuristic search as an aid to problem solving was first introduced by Polya (1971). A heuristic is a technique (consisting of a rule or a set of rules) which seeks (and hopefully finds) good solutions at a reasonable computational cost. A heuristic is approximate in the sense that it provides (hopefully) a good solution for relatively little effort, but it does not guarantee optimality (Voß 2001). Many definition of heuristic are presented in literature, one the most interesting is given by Gilli & Winker (2008)

> "Here, we follow a slightly more general denition of heuristic based on the properties of an algorithm (Winker & Maringer 2007). First, a heuristic should be able to provide high quality (stochastic) approximations to the global optimum at least when the amount of computational resources spent on a single run of the algorithm or on repeated runs is increased. Second, a well behaved heuristic should be robust to changes in problem characteristics, i.e. should not fit only a single problem instance, but the whole class. Also, it should not be too sensitive with regard to tuning the parameters of the algorithm or changing some constraints on the search space. In fact, these requirements lead to the third one, namely that a heuristic should be easily implemented to many problem instances, including new ones. Finally, despite of its name, a heuristic might be stochastic, but should

not contain subjective elements".

Given the above denition of heuristics, one of their major advantages consists in the fact that their application does not rely on a set of strong assumptions about the optimization problem. In fact, for the implementation of most of the algorithms discussed hereafter, it is sufficient to be able to evaluate the objective function for a given element of the search space. It is not necessary to assume some global property of the objective function, nor it is necessary to be able to calculate derivatives. In particular, several heuristics also allow to tackle discrete optimization problems or are even tailor made for this class of problems. On the other side, heuristics do not produce high-quality (or even exact) solutions with certainty, but rather stochastic approximations. However, when traditional methods fail, heuristics might still work in providing satisfying approximations.
Widely used heuristic are:

- *constructive methods or greedy heuristics*: simple heuristics available for any kind of combinatorial optimization problem; a greedy heuristic is an iterative methods and it usually starts with a given feasible or infeasible solution. In each iteration there is a number of alternative choices (*moves*) that can be made to transform the solution. From these alternatives which consist in fixing (or changing) one or more variables, a greedy choice is made, i.e., the best alternative according to a given evaluation measure is chosen until no such transformations are possible any longer.
- *Local Search:* whose basic principle is that solutions are successively changed by performing moves which alter solutions locally. Valid transformations are defined by neighborhoods which give for a solution all neighboring solutions that can be reached by one move. Moves must be evaluated by some *heuristic measure* to guide the search. As the solution quality of local optima may be unsatisfactory, we need mechanisms which guide the search to overcome local optimality. A simple strategy called *iterated local search* is to iterate/restart the local search process after a local optimum has been obtained, which requires some perturbation scheme to generate a new initial solution (e.g., perform-

ing some random moves). Of course, more structured ways to overcome local optimality might be advantageous.

Based on the definition of heuristics, we derive the definition of a meta-heuristics. The description that better fit with the underlying idea of meta-heuristic is presented in Voßet al. (1999):

> "A meta-heuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a population of solutions at each iteration . The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method. The family of meta-heuristics includes, but is not limited to, adaptive memory procedures, tabu search, ant systems, greedy randomized adaptive search, variable neighborhood search, evolutionary algorithms, scatter search, neural networks, simulated annealing, and their hybrids."

Among them, brief overview of *Tabu Search* and *Simulated Annealing* is presented since they achieved important results in real applications and they are still widely applied. Then this thesis will focus on meta-heuristics that are labelled as Evolutionary Algorithms (EAs) and Neural Networks.
**Tabu Search:** it was proposed initially by Glover (1977) as a combinatorial optimization problem solver. As with other combinatorial approaches, Tabu Search (TS) carries out a number of transitions in the search space aiming to find the optimal solutions or a range of near-optimal solutions. The name tabu is related to the fact that in order to avoid revisiting certain areas of the search space that have already been explored, the algorithm turns these areas tabu (or forbidden). It means that for a certain period of time (the tabu tenure), the search will not consider the examination of alternatives containing features that characterize the solution points belonging to the area declared tabu (Lee & El-Sharkawi 2008). A simple TS usually implements two forms of memory:

- A frequency-based memory, which maintains information about how often a search point has been visited (or how often a move has been made) during a specified time interval.

- A recency-based memory, which maintains information about how recently a search point has been visited (or how recently a move has been made). Recency is based on the iteration at which the event occurred.

If, for example, the frequency count of a search point exceeds a given threshold, then that point is classified as tabu for the next cycle of iterations. Positions specified in the tabu list are excluded from the neighborhood of candidate positions that can be visited from the current position. Positions remain in the tabu list for a specified time period (Engelbrecht 2007).

**Simulated Annealing:** this refinement of the local search, proposed by Kirkpatrick et al. (1983), is based on an analogy between combinatorial optimization and the annealing process of solids. In fact, as temperature reduces, the mobility of molecules reduces, with the tendency that molecules may align themselves in a crystalline structure. The aligned structure is the minimum energy state for the system. To ensure that this alignment is obtained, cooling must occur at a sufficiently slow rate. If the substance is cooled at a too rapid rate, an amorphous state may be reached. In the context of combinatorial optimization, the minimum of an objective function $\phi(\cdot)$ represents the minimum energy of the system. Simulated annealing (SA) uses a random search strategy, which not only accepts new positions that decrease the objective function (assuming a minimization problem), but also accepts positions that increase objective function values. The latter are accepted probabilistically based on a parameter $T_k$ depending on the time $k$. If $P_{ij}$ is the probability of moving from point $x_i$ to $x_j$, then $P_{ij}$ is calculated using

$$P_{ij} = \begin{cases} 1 & \text{if } \phi(x_j) < \phi(x_i) \\ e^{-\frac{\phi(x_j)-\phi(x_i)}{T_k}} & \text{if } \phi(x_j) > \phi(x_i). \end{cases} \tag{3.1}$$

Initially, when T is large, larger deterioration in the cost function is allowed; as the temperature decreases, the simulated annealing algorithm becomes greedier, and only smaller deteriorations are accepted; and at the end, when $T \rightarrow 0$, no deteriorations are accepted any longer.

# 1 Evolutionary Algorithms

Evolutionary algorithms originate in Darwin's theory of evolution, which explains the creation of species based on the evolution of life on Earth. Darwin introduced three fundamental components of evolution: replication, variation, and natural selection. Replication is the formation of a new organism from a previous one, but replicating would only produce identical copies of organisms, thereby stalling evolution. However, during the replication process there occur a series of errors called variations that allow a change in individuals. One manner of variation is sexual reproduction. In addition to replication and variation, evolution needs natural selection, which happens when individuals of a same species compete for the scarce resources of their environment and the possibility of reproducing. Such competition allows for the fittest individuals to survive and the weakest to die (Darwin 1872).
The common underlying idea behind all the EAs is the same: given a population of individual, the environmental pressure causes natural selection (survival of the fittest) and this causes a rise in the fitness of the population. Given an objective function $\phi(\cdot)$ to be optimize (minimization or maximization are specular treated), the algorithmic procedure randomly create solutions, i.e. elements of the function's domain, and apply the objective function as an abstract fitness measure.
Based on this fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation to them. Recombination is an operator applied to two or more selected candidates (called parents) and results one or more new candidates (the children). Mutation is also applied to one candidate, modifying a very small part of it, and creating one new candidate. Executing recombination and mutation leads to a set of new points (the offspring) that compete, based on their fitness, with the old ones for a place in the next generation. This process is then reiterated until a candidate with sufficient quality is found or a previously set computational limit is reached. In this brief description, extract by Eiben & Smith (2008) it is possible to figure out two important aspects of EAs:

- the variation operators (recombination and mutation) that create the necessary diversity and thereby facilitate novelty;
- the selection pressure that acts as a force pushing quality.

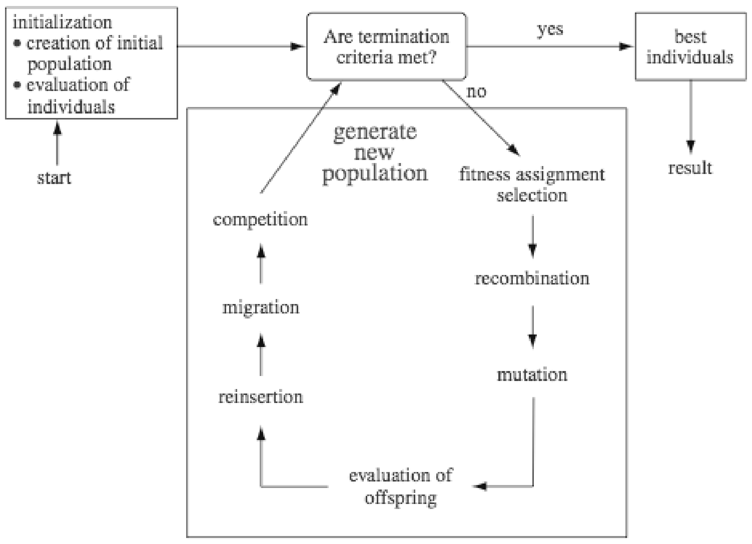


Figure 3.1: General structure of Evolutionary Algorithms.

The different ways in which the EA are implemented result in many paradigms and methods (i.e. *Genetic Algorithms, Genetic Programming, Evolutionary Strategies, Differential Evolutions, Swarm Intelligence,* among others), that are widely compared in many books and articles both in theoretical properties (Bäck 1996, Rudolph 1996, Fouskakis & Draper 2002) and in practical applications (Whitley 2001, Fogel & Corne 2002). What always emerge in all the Evolutionary Algorithms are the presence of some common components:

- they are population based, i.e. they process a whole colleciton of candidate solutions simultaniously;
- they mostly use recombination to mix information of more candidate solutions into new one;
- they are stochastic.

In order to search global optimum or very good local optima candidate, Evolutionary Algorithms generally have to define some important components (Engelbrecht 2007):

- representation (definition of individuals);
- evaluation function (usually called fitness function);
- population;

- parent selection mechanism;
- variation operator (i.e recombination and mutation);
- survivor selection mechanism (replacement).

We refer to Eiben & Smith (2008) for the description of the components but they will be emphasized in the Section 1.1 where all of them will be described for the genetic algorithm structure. EAs have several nice features that made them very popular in real problems optimization.
In contrast to many other optimization techniques, an important advantage of evolutionary algorithms is that they can cope with multi-modal functions and multi-objectives problems. Additional advantages are that their representation is independent of the complexity of the system, in contrast with other numerical techniques, which might be applicable for only continuous values or other constrained sets. Moreover, they offer a framework such that it is comparably easy to incorporate prior knowledge about the problem. Incorporating such information focuses the evolutionary search, yielding a more efficient exploration of the space of possible solutions. They are very flexible and can be combined with more traditional optimization techniques. This may be as simple as the use of a gradient minimization used after primary search with an evolutionary algorithm, or it may involve simultaneous application of other algorithms. As the least feature, that make them feasibly in many fields, is that they are robust to dynamic changes in problem, adapting solutions to changing circumstance.
The evolutionary algorithms, however, presents some drawbacks as:

- it is not guaranteed to reach the optimal solution within finite time (even if there is in convergence);
- they work under weak theoretical basis;
- they may need parameter tuning;
- they are computationally expensive.

The Genetic Algorithms (GAs) are possibly the most widespread variant of EAs. They were conceived by Holland (1975), and revolutionized the developments of optimization computer aided techniques.

## 1.1 Genetic Algorithm

Before the introduction of *genetic algorithms* by Holland (1975), other scientists with different backgrounds were also involved in developing similar ideas (Fraser 1962, Rechenberg 1973). The common thread in these ideas was the use of mutation and selection, the concepts at the core of the neo-Darwinian theory of evolution. But unlike the earlier evolutionary algorithms, which focused only on the mutation as a straightforward developments of hill-climbing methods, Holland's GA had an extra ingredient: the idea of recombination. As basic idea let us assume a discrete search space $\Omega$ and an objective function

$$\phi : \Omega \rightarrow \mathbb{R}, \tag{3.2}$$

and we want to optimize this function, let us say maximize it

$$\arg\max_{\mathbf{x}\in\Omega} \phi, \tag{3.3}$$

where $\mathbf{x}$ is a possible candidate vector of the search space $\Omega$ and $\phi$ is the objective function. The original motivation for the GA approach was a biological analogy. As in the selective breeding of plants or animals, for example, offspring are sought that have certain desirable characteristics that are determined at the genetic level by the way the parents' chromosomes combine. Similarly, In the case of GAs, the population of candidates $\mathbf{x}$ is encoded as a string of gene, and these candidates are often referred to in the GA literature as *chromosomes*.
The recombination of strings is carried out using simple analogies of genetic crossover and mutation, and the search is guided by the results of evaluating the objective function $\phi$ for each element of the population. Based on this evaluation, candidates that have higher fitness (i.e., represent better solutions) can be identified, and these are given more opportunity to breed.The procedure is then iterated until a convergence toward the optimal solution, optimizing $\phi$.
Initially genetic algorithms were supposed to be codified as string of bits, but recent development has shown that binary-coded genetic algorithm has some disadvantages, such as lower operating speed, precocious convergence (Yu-Fen & Xiao-Juan 2009). It does not suffice to abandon the original idea

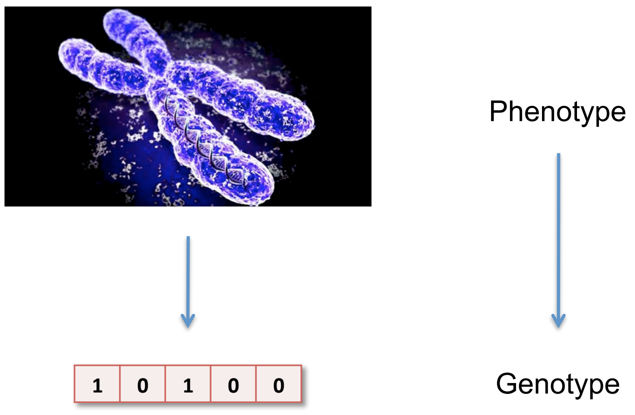


Figure 3.2: Binary code of the genetic algorithms chromosome.

of binary-code, but it allow researcher to create more flexible tools, without a deep understanding of binary code conversion. Genetic Algorithms usually use all the components of the Evolutionary Algorithms, presented in Section 1.

### Representation

In order to apply a GA to a given problem, the first decision one has to make is how to represent a candidate solutions ( the *phenotype*) as the kind of *genotype* the problem needs (Lee & El-Sharkawi 2008). The issue of selecting an appropriate representation (i.e. transform the phenotype in genotype) is crucial for the algorithm. The symbol alphabet used is often binary (see Figure 3.2 as an example of the binary code representation), though other representations have been used, including character-based and real-valued encodings (Kelly et al. 1994) or Gray code (Gray 1953). Even if the binary code is still very applied, in some problems like complex applications, it is suggested to use non-binary alphabets. Integer or continuous valued genes are typically used in large-scale function optimization problems (and in this case there is no distinction between genotype and phenotype). Another advantage of non-binary representations, particularly the real-valued one, is the easy definition of problem-specific operators. Therefore, the very hard encoding problem still remains in the hands of the designer. In order to achieve good performance for large tasks, GAs must be matched to the search problem at hand. The only way to succeed is by using domain-specic knowledge to select an appropriate representation.

### Evaluation function

Each candidate is evaluated and assigned a fitness value $f(\cdot)$ after the creation of an initial population as:

$$f(\mathbf{x}) = h(\phi). \tag{3.4}$$

It is useful to distinguish between the objective function and the fitness function used by a GA. The objective function provides a measure of performance with respect to a particular set of gene values, independently of any other candidate. The fitness function transforms that measure of performance into an allocation of reproductive opportunities (i.e., the fitness of a candidate is defined with respect to other members of the current population).
After decoding the chromosomes (i.e., applying the genotype to phenotype transformation), each element of the population is assigned a fitness value. The phenotype is used as input to the fitness function. Then, the fitness values are employed to relatively weight the candidates in the population. The specication of an appropriate fitness function is crucial for the correct operation of a GA (Radcliffe & Surry 1995).
At the beginning of the iterative search, the fitness function values for the population members are usually randomly distributed and widespread over the problem domain. As the search evolves, particular values for each gene begin to dominate. The fitness variance decreases as the population converges. This variation in fitness range during the evolutionary process often leads to the problems of premature convergence and slow finishing.
Related with the fitness function is the problem of *exploration* and *exploitation*: exploration is used to investigate new and unknown areas in the search space and exploitation to make use of knowledge found at points previously visited to help find better points (Beasley et al. 1993).
When GA finds genes from a few comparatively highly fit (but not optimal) individuals that may rapidly come to dominate the population, it converges on a local maximum or stagnates somewhere in the search space. Similarly a slow convergence means that the average fitness is high, but the difference

between the best and the average individuals is very small. Therefore, there is insufficient variance in the fitness function values to localize the optimal solutions. These problems show an huge exploitation around good solutions and a lack of exploration of the search space.
On the other hand, an extreme search of the search space, due to an excessive randomness, causes a very high variance of the results with a lack of concentration around the optimal solution with, as a consequence, a massive exploration and inadequate exploitation around the best candidates. These two requirements are contradictory and a good search algorithm must find a trade-off between them. GAs try to combine both exploration and exploitation in the selection mechanism and in the variation operator (better known as recombination operators).

### Population

As the other EAs, genetic algorithms are stochastic, population-based search algorithms. Each GA therefore maintains a population of candidate solutions, also called *generation*. Therefore, in order to have the GA started, it is necessary to create the initial population of solutions. This is typically addressed by randomly generating the desired number of solutions. The goal of random selection is to ensure that the initial population is a uniform representation of the entire search space. If regions of the search space are not covered by the initial population, chances are that those parts will be neglected by the search process.
The size of the initial population has consequences in terms of computational complexity and exploration abilities. Large numbers of individuals increase diversity, thereby improving the exploration abilities of the population. However, the more the individuals, the higher the computational complexity per generation. While the execution time per generation increases, it may be the case that fewer generations are needed to locate an acceptable solution. A small population, on the other hand will represent a small part of the search space. While the time complexity per generation is low, the GA may need more generations to converge than for a large population (Engelbrecht 2007).
Moreover, it is generally accepted that randomization is a good way of defining the candidates, but many other ideas are proposed, like the use of sta-

tistical model to define the “best” initial points (i.e. with a latin hypercube design of experiments), or with other meta-heuristics for seeding the population with good solutions (Reeves 1995). These methods usually suffer of premature convergence (Levine 1997).

### Parent selection mechanism

Selection determines which individuals are chosen for mating (recombination) and how many offspring each selected individual produces. Each candidate solution in the generation receives a reproduction probability depending on the fitness function $f(\cdot)$. The most popular techniques are fitness-proportionate method. In these methods, the probability of selecting an individual for breeding is proportional to its fitness:

$$p(\mathbf{x}_i(k)) = \frac{f_\gamma(\mathbf{x}_i(k))}{\sum_{i=1}^{n(k)} f_\gamma(\mathbf{x}_i(k))}, \tag{3.5}$$

where $k$ represent la k-*th* generation, $i$ is the i-*th* candidate, and $f_\gamma(\mathbf{x}_i(k))$ is the scaled fitness of the i-*th* candidate at the k-*th* generation. When $f_\gamma(\mathbf{x}_i(k)) = f(\mathbf{x}_i(k))$ the scaling function is the fitness itself. This selection method is the simplest and it is called *Roulette-wheel*, but more complex scaling function usually are applied.
The scaling function $f_\gamma(\cdot)$ play a pivotal role in the selection procedure; in fact, fitness-proportionate selection faces problems when the fitness values of individuals are very similar among them or when the numbers of candidates is high. In this case, $p(\mathbf{x}_i(k))$ would be approximately $\left[\sum_{i=1}^{n(k)} f_\gamma(\mathbf{x}_i(k))\right]^{-1}$ for each $i = 1, \ldots, n$, and hence selection would be essentially random. Some improvement are therefore implemented to avoid random selection and they can be summarize as follow:

- *linear scaling*: where $f'(\cdot) = af(\cdot) + b$ with $a$ and $b$ real number;
- *exponential scaling*: where $f'(\cdot) = f(\cdot)^\alpha$ with $\alpha$ a real number, the higher $\alpha$, the harder the selection of better experiments;
- *sigma truncation*: in which $f'(\cdot) = max(0, f(\cdot) - \bar{f}(\cdot) - c \cdot \sigma(k))$ where $\bar{f}(\cdot)$ is the average fitness of the population at generation $k$, $c$ is a positive constant and $\sigma$ represent the fitness standard deviation at generation $k$.

Other type of selection pressures are developed like *stochastic universal selection*, *tournament selection*, *ranking selection*, *Boltzmann selection* and more complicated solutions like *dynamical selection*, but in this thesis they are just mentioned and we refer to other sources for a wider explanation (Beasley et al. 1993, Glover & Kochenberger 2003, Engelbrecht 2007, Lee & El-Sharkawi 2008).
One particular case of selection is the *elitism*, where the "best" individual/s of the k-*th* generation is/are cloned in the following generation in order to reduce diversity, augmenting exploitation.

### Variation operator

Variation operators allow to generate offspring from selected parents by applying crossover and/or mutation. Crossover is the process of creating one or more new individuals through the combination of genetic material randomly selected from two or more parents. If selection focuses only on the fittest individuals, the selection pressure may cause premature convergence due to reduced diversity of the new populations. Mutation is the process of randomly changing the values of genes in a chromosome. The main objective of mutation is to introduce new genetic material into the population, thereby increasing genetic diversity. Mutation should be applied with care not to distort the good genetic material in highly fit individuals.

**Crossover operator** (also called *recombination operator*) is applied to many pairs of individuals selected for mating and usually the probability of crossover being applied is typically between 0.6 and 1.0 (Beasley et al. 1993). The typical forms of crossover are:

- *Single Point Crossover*: it takes two parents and cuts their chromosome strings at some randomly chosen position to produce two "head" and two "tail" segments. The tail segments are then swapped over to produce two new full length chromosomes (see Figure 3.3). The two children inherit some genes from each parents.
- *Multi Points Crossover:* it operates on two parents, but as the name suggests, two or more points are selected at random rather than a single point and the sequence of components between the points is

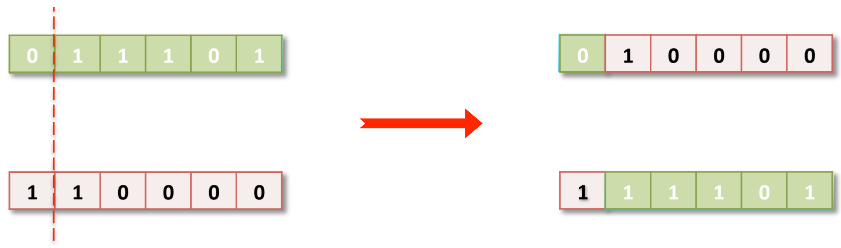


Figure 3.3: Single point crossover.

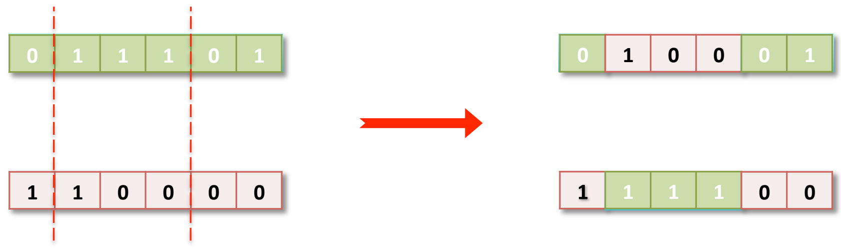


Figure 3.4: 2 points crossover.

exchanged. Figure 3.4 presents a 2 points crossover. In this more complicated crossover chromosome are better represented, rather than by linear strings, as loops formed by joining the ends together; in fact, to exchange a segment from one loop with that from another loop requires the selection of as least two cut points.

- Uniform crossover: it is radically different to the other crossover. Each gene in the offspring is created by copying the corresponding gene from one or the other parent, chosen according to a randomly generated crossover mask. Where there is a 1 in the crossover mask, the gene is copied from the first parent, and where there is a 0 the gene is copied from the second parent (as shown in Figure 3.5).

There is no paradigm about the best crossover mechanism, but generally the uniform crossover has shown better performance in reaching better or optimal solutions. A very comprehensive comparison of the three methods is presented in Fogel (2005).

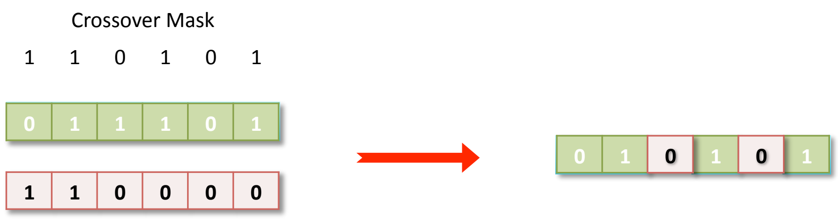


Figure 3.5: Uniform crossover.

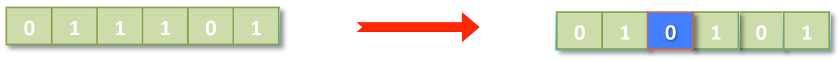


Figure 3.6: Uniform random replacement mutation.

**Mutation** is a genetic operator that alters one or more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool, incrementing the exploration of the search space. With these new gene values, the genetic algorithm may be able to reach better solutions than was previously possible. Mutation is an important part of the genetic search as it helps to prevent the population from stagnating at any local optima. Mutation occurs during evolution according to a user-definable mutation probability. This probability should usually be set fairly low (0.01 is a good first choice). If it is set too high, the search will turn into a primitive random search.
The simplest mutation form is the *uniform random replacement*. In this case mutation points are randomly selected from one parent (or alternatively from the offspring after recombination) and replaced with another possible values for that gene (in binary code it is just flipping from 0 to 1 and viceversa) as shown in Figure 3.6.
Even if many different algorithms have been developed for the mutation operator like *Non-Uniform mutation* or *Gaussian mutation*, with increasing performance in some applications, the simplest mutation performs well in many situation and it does not need a very complicated tuning of other parameters. The same reasoning can be done for every genetic algorithms component, where more and more complex ideas are presented in various

papers. But, unfortunately, more innovative procedures tend to need more complicate solutions, involving more parameters with an increase of computational costs and tuning time.

### Survivor selection mechanism (replacement)

The role of replacement is keeping the population size constant. To do so, some individuals from the population have to be substituted by some of the individuals created during reproduction. This can be done in several ways:

- *replacement-of-the-worst* : the population is sorted according to fitness and the new individuals replace the worst ones from the population.
- *Random replacement*: the individuals to be replaced are selected at random.
- *Tournament replacement*: a subset of $\alpha$ individuals is selected at random from the generation, and the worst one is selected for replacement. This case is a generalization of the *random replecement* if $\alpha = 1$.
- *Direct replacement*: the offspring completely replace their parents.

Some variants can be considered like the use of *elitism*, presented before, or *innovation* that is the opposite of *elitism* and introduce new children chosen randomly directly form the search space.

## 2 Neural Network models

As the genetic algorithm, neural networks are adaptive, learn, can deal with highly nonlinear problems and noisy data and they are robust, weak random search methods. They do not need gradient information or smooth functions. Initially they were thought to be a quasi-brain systems, able to reproducing the physical connection and behaviors of human brain neurons. A brief description of the biological inspiration can be very useful to understand the ratio behind neural network models.
The brain consists of a very large number of neurons, about $10^{11}$, in average. Biological neurons (Figure 3.7) have three principal components: the dendrites, the cell body (soma) and the axon. A neuron's dendritic tree is connected to about a thousand neighbouring neurons. When one of those

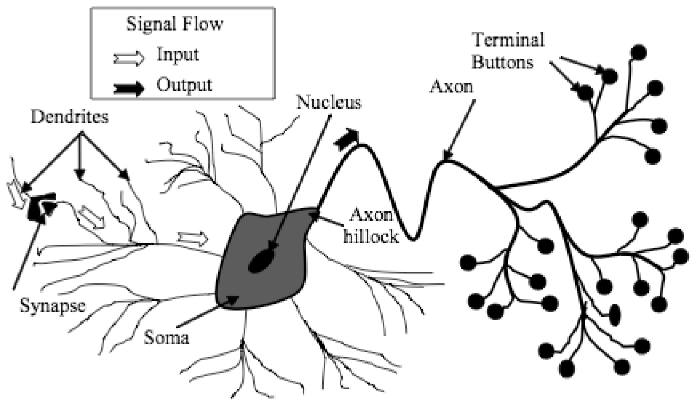


Figure 3.7: A typical biological brain neuron

neurons *fires*, a positive or negative charge is received by one of the dendrites. The strengths of all the received charges are added together through the processes of spatial and temporal summation. Spatial summation occurs when several weak signals are converted into a single large one, while temporal summation converts a rapid series of weak pulses from one source into one large signal. The aggregate input is then passed to the cell body or soma. If the aggregate input is greater than the axon hillock's threshold value, then the neuron fires, and an output signal is transmitted down the axon. The strength of the output is constant, regardless of whether the input was just above the threshold, or a hundred times as great. The output strength is unaffected by the many divisions in the axon; it reaches each terminal button with the same intensity it had at the axon hillock. This continuum of input-output neuronal signal transmission allow the brain to feed itself by environmental stimuli, and adapt to changes, in other word learn by experience.
In the '40s this exceptional adaptability of our brain was inspiration to McCulloch & Pitts (1943) who, first, introduced the idea of artificial neuron. The initial idea was very simple (see Figure 3.8): the output of the neuron is a nonlinear transformation, due to the activation function $f(\cdot)$, of the summation of inputs $X_j$, weighted by the parameters $w_j$. The parameters settings in the former artificial neurons were very basic, in fact, McCulloch and Pitts defined:

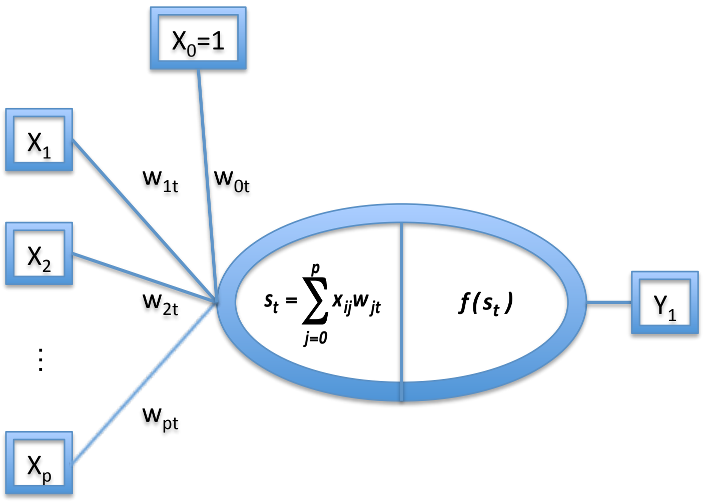


Figure 3.8: The McCulloch and and Pitts' Artificial Neuron

- all the inputs were binary $X_j \in \{0, 1\}$ with $j = 1, \dots, p$,
- there was one single binary output,
- the bias neuron $x_0$ associated to the weight $w_{0t}$ does not appear,
- the activation function $f(\cdot)$ was a *threshold function*:

$$f(s_t) = \begin{cases} 1 & s_t(\mathbf{x}, \mathbf{w}) = \sum_{i=0}^{n} x_{ij} w_{jt} > \theta \\ 0 & s_t(\mathbf{x}, \mathbf{w}) = \sum_{i=0}^{n} x_{ij} w_{jt} < \theta, \end{cases} \tag{3.6}$$

  where $\theta$ is the activation threshold; without loss in generality, Expression 3.6 can use an activation threshold equal to 0 with the introduction of the bias neuron.

Only with the introduction of "adaptive weights" (Rosenblatt 1962), the artificial neuron (now called perceptron) had found a widespread application. The Rosenblatt's seminal idea was to build up a very simple neural network model, formed by only the input layer and the output layer, but the parameters of the model (the weights) can change their values in a learning procedure. The adaptation of weights allow the model to better fit with the observed output.
More specifically, let $\mathbf{X} = \{X_1, \dots, X_j, \dots, X_p\}$ be a matrix of $p$ observed variables and $W(k) = \{w_{1t}(k), \dots, w_{jt}(k)\}$ be the weights matrix associated

to the input vector at the iteration $k$. By Formula 3.6, the perceptron estimates the output as $\hat{y} = f(s_t)$. Rosemblatt introduced the idea of using the estimate error in the perceptron $\epsilon = y - \hat{y}$ to modify the weights in order to minimize this error. The weights update is performed using the following equation:

$$w_{jt}(k+1) = w_{jt}(k) + \alpha\,\epsilon\,x_{ij}\ \forall j = 1, \ldots, p, \tag{3.7}$$

where $\alpha$ is a constant, chosen so that the $\epsilon$ asymptotically tends to 0.
The $\Delta_{ij} = \alpha\,\epsilon\,x_{ij}$, in Formula 3.7, represents the *delta rule*, i.e. the change the network weights have to be done in order to reduce the error $\epsilon$.
The learning procedure is iterated until a user-specified error threshold or a predetermined number of iterations have been completed. The limitation of the perceptron is due to the very simple topology of the network and above all to the use of threshold activation function. In order to modify weights of the neural network architecture, continues and differentiable activation functions have been introduced (and presented in Section 2.1). The Rosemblatt's idea is the seed of the modern neural network complex architectures, where various components are changed (learning rule, number of layers, activation functions among others) even if Minsky & Papert (1969) wrote a very critic book in which they described the limitations of single layer perceptrons. The impact that the book had was tremendous and caused a lot of neural network researchers to loose their interest. The book showed mathematically that single layer perceptrons could not do some basic pattern recognition operations like determining the parity of a shape or determining whether a shape is connected or not. What they did not realised, until the 80's, is that given the appropriate training, multilayer perceptrons can do these operations.

## 2.1 Activation function

Although theoretically any differential function can be used in the neural network model, usually the identity and sigmoid functions are the most used. The choice of the activation function is strictly connected to the choice of the training algorithms (present in Section 2.2). The threshold function, in formula 3.6 was the first idea that mimic what a biological neuron do, but networks with threshold function are difficult to train because the error function is stepwise constant, hence the gradient either does not exist or is zero, making it impossible to use back-propagation or more efficient gradient-based training methods. Even for training methods that do not use gradients, such as simulated annealing and genetic algorithms, sigmoid and linear activation functions are easier to handle than the threshold one. Some common activation functions are:

- *Linear function*: The linear function produces a linearly modulated output depending for a constant $C$.

$$f(s_t) = C\, s_t(\mathbf{x}, \mathbf{w}), \tag{3.8}$$

  where $C$ is the slope of the function.

- *Ramp function*: The ramp function is a combination of the linear and the threshold functions

$$f(s_t) = \begin{cases} \gamma & \text{if } s_t(\mathbf{x}, \mathbf{w}) > \theta \\ s_t(\mathbf{x}, \mathbf{w}) & \text{if } -\theta < s_t(\mathbf{x}, \mathbf{w}) < \theta \\ -\gamma & \text{if } s_t(\mathbf{x}, \mathbf{w}) < -\theta. \end{cases} \tag{3.9}$$

- *Sigmoidal function*: the sigmoid function is a continuous version of the ramp function, which maps the net into the codomain $[0, 1]$.

$$f(s_t) = \frac{1}{1 + e^{\lambda s_t(\mathbf{x}, \mathbf{w})}}, \tag{3.10}$$

  where parameter $\lambda$ controls the steepness of the function and it is usually set to 1.

- *Hyperbolic tangent function*: is a sort of sigmoidal function that map the network into the codomain $[-1, 1]$

$$f(s_t) = \frac{e^{\lambda s_t(\mathbf{x}, \mathbf{w})} - e^{-\lambda s_t(\mathbf{x}, \mathbf{w})}}{e^{\lambda s_t(\mathbf{x}, \mathbf{w})} + e^{-\lambda s_t(\mathbf{x}, \mathbf{w})}}. \tag{3.11}$$

After defining the activation function, the neural network models should be trained in order to estimate the weights $W$. Learning algorithms are suited to cope with this issue.

## 2.2 Lerning algorithms

The neural network learns the best values for the $W$ by the observed data. Learning consists of adjusting weights until a certain criterion (or several criteria) is (are) satisfied.
Learning algorithms can be applied both in supervised and unsupervised problems, but in this thesis we will refer only to the former case. The learning algorithms can be considered as an iterative optimization procedure that define the "best" parameters of the network which minimize one loss function $L(\cdot)$, also called *empirical or generalization error.* The loss function usually approximates a distance measure between the observed output and the estimated output by the neural network model. The most common loss function is the *Mean Square Error* (presented in formula 1.2), that measures the average squared error between the network's output, $f(s_t)$, and the target value $Y$.
Many learning algorithm are developed to detect the best parameters of the neural network, but usually the "gradient descent methods" are computational affordable and with good solutions in many examples. The learning techniques require two ingredients at each iteration of the step:

- the computation of the gradient of the cost function,
- the updating of the parameters as a function of that gradient, in order to get closer to a minimum of the cost function.

Until the idea of back propagation (Rumelhart et al. 1986), that revolutionized the application of neural networks, the learning algorithms represented a bottle-neck due to their slow computational time.
Back-propagation consists of an iterative procedure based on the *delta rule*; the mathematical computation of the delta rules will be presented in the next Section. The gradient descent methods compute at each generation of the procedure can be summarize in the following way:

**Algorithm 1** Stochastic Gradient Descent Learning Algorithm

```
Input: Initialize weights $W$, $\alpha$, and the number of maximum iterations $k$
Output: $\hat{W}$ which minimize the $L(\cdot)$
1.. while stopping condition(s) not true do
2..     Let $\epsilon = 0$
3..     for for each $X_j$ do
4..         Calculate $f(s_t)$
5..         Calculate the empirical error $y_i - f(s_t)$
6..         Adjust weights $w_{jt}$ with the delta rule in formula 3.7
7..         Compute the mean square error of the model as $\sum_{i=1}^{n}(y_i - f(s_t))^2$
8..     end for
9..     $t = t + 1$
10.. end while
```

The learning algorithms are performed until a stopping criteria is met. Stopping criteria usually includes:

- maximum number of epochs ($k$) has been exceeded,
- the mean squared error (MSE) is small enough (below the $\alpha$ threshold).

When the number of parameters grows, the gradient methods, due to the slow convergence and their problems to stuck in local optima, are surpassed by conjugate gradient optimization. These methods trade off the simplicity of gradient descent and the fast quadratic convergence of Newton's methods; in fact, in the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. Several conjugate gradient learning algorithms have been developed (in Battiti (1992) many examples), most of which are based on the assumption that the error function of all weights in the region of the solution can be accurately approximated by

$$\epsilon_T(X_j, \mathbf{w}) = \frac{1}{2}\mathbf{w}^t H \mathbf{w}, \tag{3.12}$$

where $H$ is the Hessian matrix. These learning algorithms take also the name of second-order method because of the use of the second derivative matrix (The Hessian matrix).

Since the dimension of the Hessian matrix is the total number of weights

in the network, the calculation of conjugate directions on the error surface becomes computationally infeasible in high parametrized networks. Computationally feasible conjugate gradient algorithms compute conjugate gradient directions without explicitly computing the Hessian matrix, and perform weights update along these directions. Common algorithms, called quasi-Newton's methods, are Levenberg-Marquardt (Levenberg 1944, Marquardt 1963) and BFGS (Broyden 1970, Fletcher 1970, Goldfarb 1970, Shanno 1970); they update an approximate Hessian matrix at each iteration of the algorithm. The update is computed as a function of the gradient.
In the definition of the neural network models, the last important step in order to define a good model is the choice of the architecture.

## 2.3 Feed-forward neural network

The perceptron convergence theorem states that, if a linear separation exists, the perceptron error-correction scheme will find it (Rosenblatt 1962). Obviously this theorem let the application of neural network model to a very close class of models, that is the linear separable set of binary input and the perceptron does not converge when the system is not linear. The introduction of multi-layer neural network, often called feed-forward neural network, allows a generalization of the model which becomes a general function approximator. The feed-forward neural network introduced to the perceptron some new layers between the input variables $X$ and the observed response $Y$. These layers are called *hidden layers* and each of that can assume different number of neurons and different activation functions. The most common model is a *single hidden layer neural network* (see Figure 3.9) with a sigmoidal function between the input and the hidden layer and a linear function between the hidden layer and the outputs.
More specifically, let $X = \{x_{i1}, \ldots, x_{ip}\}$ be the $p$ vectors of $i$ observations, with $i = 1, \ldots, n$, and $Y = \{y_{i1}, \ldots, y_{id}\}$ be the $d$ vectors of $i$ observed outputs and $r$ be the number of neurons in the hidden layers. Then, let $W_{jt} = \{w_{11}, \ldots, w_{pr}\}$ and $W_{tq} = \{w_{11}, \ldots, w_{rd}\}$ be respectively the weights matrixes between the input layer and the hidden layer and between the hidden layer and the output layer and $f_1(s_t)$ and $f_2(s_q)$ be respectively the sigmoidal function and the linear function, the estimated output can be cal-

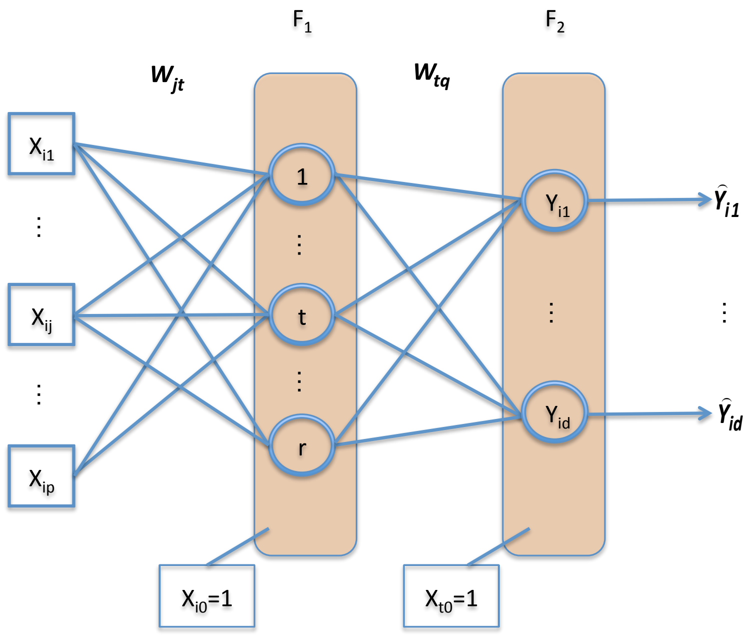


Figure 3.9: Feed Forward Neural Network with single hidden layer.

culated as:

$$\hat{y}_{id} = f_2(f_1(s_t)). \tag{3.13}$$

Substituting in Formula 3.13 the activation function in Expressions 3.8 and 3.10 we obtain:

$$\begin{aligned} \hat{y}_{id} &= \sum_{q=0}^{d} \lambda W_{tq} f_1(s_t) \\ &= \sum_{q=0}^{d} \lambda W_{tq} \frac{1}{1 + e^{\sum_{j=0}^{p} \lambda x_{ij} W_{jt}}}. \end{aligned} \tag{3.14}$$

This simple architecture is wildly used because it can approximate any function with a finite number of discontinuities, arbitrarily well, given sufficient neurons in the hidden layer (Hagan et al. 1995). Although the theoretical results are of great importance because they demonstrate the powerful capabilities of feed-forward neural networks, they don't give an indication of how to choose the number of hidden units needed per hidden layer. In addition, even if for some problems one hidden layer may be enough theoretically, in

practice more than one hidden layers should be utilized to solve the problem faster and more efciently.
However, in certain problems, a large number of hidden nodes may be required in order to achieve the desired accuracy. Thus, a network with two hidden layers and much fewer nodes overall, should be able to solve the same problem more efficiently. Hence, choosing an appropriate network size for a given problem is still something very complicated. The selection of the topology becomes really important in the definition of the generalization capability of the model because too complicated networks often suffer of overfitting problem. After the selection of the network topology, it is necessary to train it with some learning algorithm in order to identify the “best” estimate of the parameters which minimize the mean squared error or other loss functions.

## 2.4 Back-Propagation algorithm

Defined one gradient descent algorithm, the most used algorithms to compute efficiently the gradient of the cost function $L(\cdot)$ of the network is the *Back-propagation.* We consider a feed-forward neural network with one single hidden layer with $r$ neurons, and a single output neuron (the extension to neural networks with several output neurons is straightforward). After choosing the initial weights of the network randomly, the back-propagation algorithm is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

- *Feed-forward step*: a propagation phase, where the inputs $X$ feed the network, and the potentials and outputs of all neurons are computed with Formula 3.14. The Mean square error is then calculated as:

$$\epsilon = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2. \tag{3.15}$$

- *A backpropagation to the output layer*: the back-propagation path, from the q-*th* output of the network up to the t-*th* unit of the hidden layer, is shown:

$$\begin{aligned} \frac{\partial \epsilon}{\partial W_{rd}} &= \frac{1}{2}\frac{\partial(\hat{y}_i - y_i)^2}{\partial W_{rd}} \\ &= -(\hat{y}_i - y_i)\frac{\partial \hat{y}_i}{\partial W_{rd}}, \end{aligned} \tag{3.16}$$

where

$$\begin{aligned}\frac{\partial \hat{y}_i}{\partial W_{rd}} &= \frac{\partial f_2(s_q)}{\partial W_{rd}}\\ &= \frac{\partial f_2(s_q)}{\partial s_q}\frac{\partial s_q}{\partial W_{rd}}\\ &= f_2'(s_q)\frac{\partial \sum_{q=0}^{d} W_{tq}s_t}{\partial W_{rd}}\\ &= f_2'(s_q)s_t.\end{aligned} \tag{3.17}$$

From Formulas 3.16 and 3.17 we obtain:

$$\frac{\partial \epsilon}{\partial W_{rd}} = (\hat{y}_i - y_i)f_2'(s_q)f_1(s_t), \tag{3.18}$$

and we define

$$\delta_q = (\hat{y}_i - y_i)f_2'(s_q). \tag{3.19}$$

Substituting $\delta_q$ into Equation 3.18, we obtain the delta rule in formula 3.7 for the weights between the output and the hidden layer.

- *A backpropagation to the hidden layer*: similarly the partial derivative of $\frac{\partial \epsilon}{W_{jt}}$ between the hidden and the input layer should be computed as:

$$\begin{aligned}\frac{\partial \epsilon}{\partial W_{jt}} &= \frac{1}{2}\frac{(\partial \hat{y}_i - y_i)^2}{\partial W_{jt}}\\ &= -(\hat{y}_i - y_i)\frac{\hat{y}_i}{\partial W_{jt}},\end{aligned} \tag{3.20}$$

where

$$\begin{aligned}\frac{\partial \hat{y}_i}{\partial W_{jt}} &= \frac{\partial f_1(s_t)}{\partial W_{jt}}\\ &= \frac{\partial f_1(s_t)}{\partial s_t}\frac{\partial s_t}{\partial W_{jt}}\\ &= f_1'(s_t)\frac{\partial s_t}{\partial W_{jt}},\end{aligned} \tag{3.21}$$

and

$$\begin{aligned}\frac{\partial f_1(s_t)}{\partial W_{jt}} &= \frac{\partial \sum_{j=0}^{p} W_{jt}f_1(s_t)}{\partial W_{jt}}\\ &= \sum_{j=0}^{p} W_{jt}\frac{\partial \hat{y}_t}{\partial W_{jt}}\\ &= W_{jt}\frac{\partial \hat{y}_t}{\partial W_{jt}}.\end{aligned} \tag{3.22}$$

Extended to all the units in the hidden layer we obtain:

$$\begin{aligned}\frac{\partial \hat{y}_t}{\partial W_{jt}} &= \frac{\partial f_1(s_t)}{\partial W_{jt}} \\ &= \frac{\partial f_1(s_t)}{\partial s_t}\frac{\partial s_t}{\partial W_{jt}} \\ &= f_1'(s_t)\frac{\partial \sum_{j=0}^{p} W_{jt}x_{ij}}{\partial W_{jt}} \\ &= f_1'(s_t)x_{ij}.\end{aligned} \tag{3.23}$$

From Expression 3.20,3.21, 3.22 and 3.23 we obtain:

$$\frac{\partial \epsilon}{\partial W_{jt}} = \sum_{t=0}^{d}[(\hat{y}_t - y_t)f_2'(s_q)W_{tq}]f_1'(s_t)x_{ij}, \tag{3.24}$$

where substituting Formula 3.19 $(\hat{y}_t - y_t)f_2'(s_t) = \delta_d$ we obtain:

$$\frac{\partial \epsilon}{\partial W_{jt}} = \sum_{t=0}^{d}[\delta_d W_{tq}]f_1'(s_t)x_{ij}. \tag{3.25}$$

Eventually we then define the *delta rule* for the entire network as:

$$\delta_t = \sum_{t=0}^{d}[\delta_d W_{tq}]f_1'(s_t)x_{ij}. \tag{3.26}$$

- *Weights update*: after computing all partial derivatives the network weights are updated in the negative gradient direction. A learning constant $\eta$ defines the step length of the correction. The corrections for the weights are given by

  $$\Delta W_{rd} = \eta\delta_d s_t, \tag{3.27}$$

  and

  $$\Delta W_{jt} = \eta\delta_t x_{ij}, \tag{3.28}$$

  respectively for the weights between hidden and output layer, and between input and hidden layer

Despite its prevalent use, backpropagation can lead to entrapment in local minima, as the techniques based on gradient descent, making neural networks

incapable of sufficient performance and generalization. Some generalization techniques are applied in order to fit good model on the data, but also capable to predict correctly new data. In order to achieve a good network in terms of generalization accuracy, it is important to define models with a balanced trade-off of bias and variance. Thus, a very complex model, with a large number of adjustable parameters, may have a very low bias, i.e. may have the ability of fitting the data whatever the noise present, but it is apt to have a very large variance, depending strongly on the specific realization of the noise present in the training set. Conversely, a very simple model, with a small number of adjustable parameters, may be insensitive to the noise present in the training data, but turn out to be unable to approximate the regression function (Dreyfuss 2005).
Generalization techniques are strongly suggested and some of them are already presented in Chapter 2 such as cross-validation and bootstrapping, and more classical techniques like *weight decay* are often used.

# Chapter 4

# The Evolutionary Neural Network Design

This thesis proposes a new approach to address high dimensional variable selection and model assessment: the Evolutionary Neural Network Design (ENN-Design) approach. The ENN-Design method combines the strongest features of some methods, presented in previous Chapters, in order to optimize a high dimensional biological system. In particular, we develop a new approach which embodies a model selection procedure to identify neural network models with high prediction accuracy and a filter variable selection procedure to identify subset of important variables. These approaches are combined together and used to evolve a population of experimental points to achieve optimal solutions in a very high dimensional problem. The idea has been developed to address the biological problem of Protein Engineering and Design (PED).

This research is part of an international project dedicated to "Designing Informative Combinatorial Experiments" for living technology. The "DICE" project aims at designing evolutionary combinatorial experiments in the high dimensional and high throughput setting that characterizes the search of new biological entities, such as new artificial proteins.

PED can be seen as a walk through a multi-dimensional experimental space to find mutants with improved or novel properties. The exhaustive exploration of the experimental space is unattainable and beyond current technical reach, due to the inherent combinatorial nature of proteins, the non-linear interactions among variables and the complexity of the fitness landscape (Zhao 2007). The biological requirements in the experimentation were particularly strict due to the technical constraints in the way the testing in the laboratory is carried out. Therefore, the main goal is to find new solutions to a high dimensional problem in the presence of scarce data.

## 1 The biological problem

Synthetic biology aims at designing novel biological components (i.e. proteins, metabolic and regulatory networks) for useful purposes: medical applications (Hong et al. 2010), industrial productions (Clomburg & Gonzalez 2010) and environmental applications (Danino et al. 2010). Proteins are ubiquitous in nature and they are responsible for the major part of biological tasks. A protein $x_i$ is a sequence of monomers, called amino-acids, covalently joined together to form a complex string, called polypeptide. Each protein may differ in length, amino acid composition and sequence and is characterized by a well defined three-dimensional structure which in turn defines the proteins function. One of the most intriguing feature of proteins is catalysis (i.e. capability to enhance chemical reactions within a cell). For this reason, significant effort has been made to engineer novel or improved version of existing proteins to perform this specific task. Despite a number of successful applications reported in literature, engineering natural proteins has been a challenging task for biochemistry. Extant proteins are the results of a long evolutionary history and they have evolved within the constrains of cellular environment and ecological niche. Thus, engineering natural proteins need to compe with these constrains which significantly impair protein plasticity.
In the "DICE" project, the research is developed addressing the challenge of protein engineering starting from non-natural random sequences of amino acids (i.e. protein sequences with no significant homology level to extant ones). The idea is to recreate a sort of hypothetical "primordial soup" (theorized by Haldane (1928), Oparin (1924) and experimentally tested by Miller (1953)) where random sequences of amino acids were deprived of any evolutionary history and constrains. In this way, these sequences can be seen as the starting point of an evolutive path that brought to the extant proteins. From

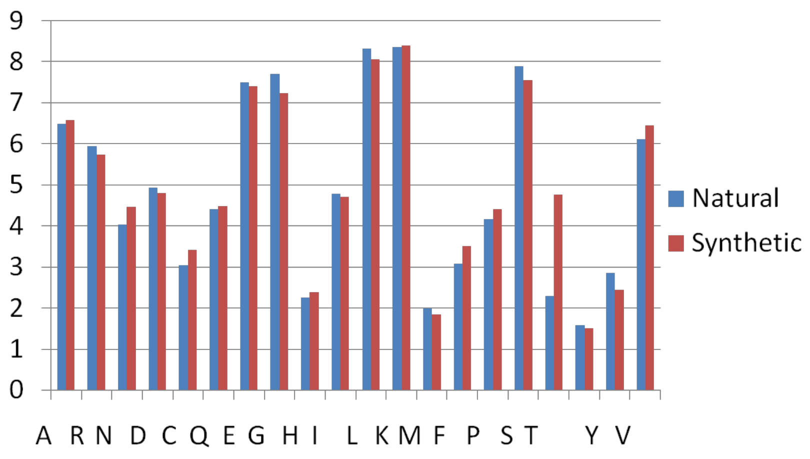
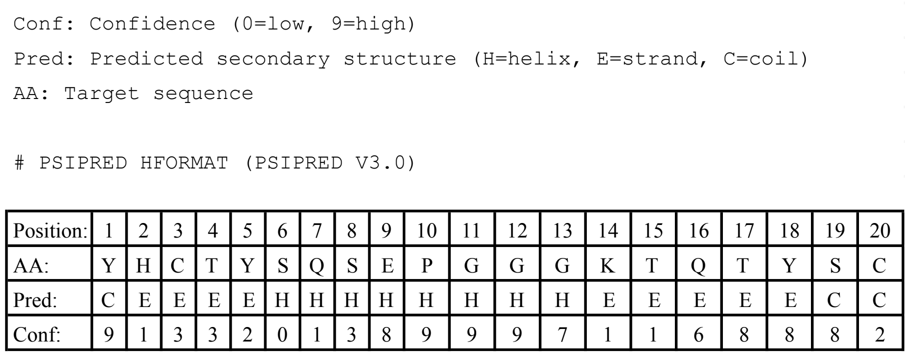


Figure 4.1: Distribution of amino-acids in the non-natural random domains, compared to natural frequencies.

a biological point of view, these sequences are more versatile and prone to be engineered. Within this framework, we design a library of 95 protein random domains $d_k = \{d_1, \ldots, d_{95}\}$ (each one composed of 50 amino acids) with no signicant homology to extant proteins. The amino acid frequency used to generate random domains reflects the composition of natural proteins, as presented in Figure 4.1

Random domains are combinatorially assembled to generate full-length random proteins of 200 amino acids (each protein has 4 domains) to be subsequently screened for assessing its catalytic function. Thus each individual protein can be considered as a string composed of 4 domains selected among the 95 polymers. Accordingly, all possible permutations with repetition of 95 elements in 4 positions are $95^4 \simeq 8.1 \times 10^7$ which represents the cardinality $(N)$ of the problem, i.e. the number of possible different full-length synthetic proteins to be screened.

The output/response $Y = f(x_i)$ of the protein is obtained as a measure of similarity between the i-*th* synthetic protein $x_i$ and the catalytic natural protein, the Serine esterase (cutinase) of Fusarium Solani (in Figure 4.3), calculated by the bioinformatic tool PSI-BLAST (`http://toolkit.tuebingen.mpg.de/psi_blast` created by Altschul et al. (1997)), that is able to identifying biologically relevant sequence similarities. In order to obtain a response the first step is to calculate the secondary structure of the domains using PSIPRED software (Mcguffin et al. 2000) a tool of PSI-BLAST. PSIPRED predicts whether a given domains adopts an helix, coiled-coil or beta-sheet conformation. The typical output of the PSIPRED algorithm is presented in Figure 4.2.



```
Conf: Confidence (0=low, 9=high)
Pred: Predicted secondary structure (H=helix, E=strand, C=coil)
AA: Target sequence

# PSIPRED HFORMAT (PSIPRED V3.0)
```

| Position: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AA: | Y | H | C | T | Y | S | Q | S | E | P | G | G | G | K | T | Q | T | Y | S | C |
| Pred: | C | E | E | E | E | H | H | H | H | H | H | H | H | E | E | E | E | E | C | C |
| Conf: | 9 | 1 | 3 | 3 | 2 | 0 | 1 | 3 | 8 | 9 | 9 | 9 | 7 | 1 | 1 | 6 | 8 | 8 | 8 | 2 |

Figure 4.2: PSIPRED output of a test protein.

Each letter in Figure 4.2 represents a biological structure of the protein where the letter $C$ is a coiled-coil conformation, $E$ is a beta-sheet conformation and $H$ is a helix conformation.
The second step is to use secondary structure profile of domains calculated using PSIPRED to calculate the similarity among different domains according to the method proposed by De Lucrezia et al. (2009). Briefly, the similarity of domain is calculated using secondary structure prediction by aligning all domains in a pair-wise fashion and calculating similarity score by a two-step procedure as follows:

1 *Position-related score calculation:*
$s_{i,j}$ = **IF** $(pred_{i,j} = pred_{i,k})$ $confi_{i,j}$ **ELSE** $\{0\}$
where $s_{i,j}$ is the position-score of the i-*th* amino acid of the j-*th* domain, $pred_{i,j}$ is the secondary prediction of the i-*th* amino acid of the j-*th* domain whereas $(pred_{i,k})$ is the secondary prediction of the i-*th* amino acid at the same position in the k-*th* domain. When the IF statement is satisfied the output value is the correspondent confidence value $confi_{i,j}$ of the i-*th* amino acid of the j-*th* domain, otherwise the output value is zero.

Figure 4.3: The confocal microscope image of the serine esterase (cutinase) of Fusarium Solani.

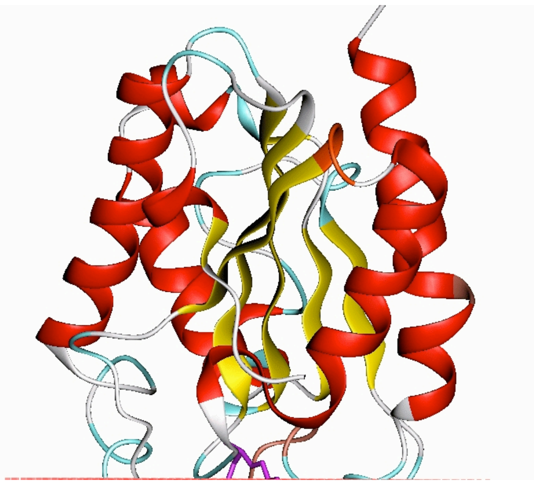


Figure 4.4: The 3D-folded serine esterase (cutinase) of Fusarium Solani estimated by Chimera.

2 *Global score calculation:*
$S_j = \sum_{i=1}^{50} s_{i,j}$
which is a summation of individual position-related score over the entire domain length.

In order to evaluate the shape of the predicted synthetic proteins by the PSIPRED tool, the biologists identify the similarities between the natural protein and the artificial one also comparing the 3-D structure. The Rosetta software (`http://boinc.bakerlab.org/rosetta/`) is applied to determine the 3-dimensional shapes of proteins and “Chimera” software (`http://www.cgl.ucsf.edu/chimera/`) in order to identify similarities in the folded “synthetic” proteins with the natural one (the natural protein 3-D structure is presented in Figure 4.4).
The main challenge is to develop effective methodologies to identify the best domain in the proper position to construct functional proteins without the need to screen a large number of points (i.e. sequences) in the experimental space (i.e. combinatorial protein space). In order to be experimentally tested, candidate proteins should respect the following biological restrictions:

i) the number of cysteine residues should be at most 9 and different from 5 and 7, since proteins with these features are hard to express and purify.

ii) The percentage of *coil* should not be larger than 70% otherwise proteins will hardly fold into stable tertiary structure.

## 2 The statistical problem

This challenging research have to face most of the statistical problems presented in previous Chapters. Real experimental constraints and complete lack of information about the biological process pushed us to develop a new method which combines evolutionary algorithms, statistical modeling and variable selection procedure, in order to find the global optimal solution ( i.e. the protein that is able to enhance chemical reaction), or some good solutions that can be used as starting point for biologist in order to engineer synthetic catalytic proteins. The statistical problems encountered in this research are:

- The stochastic representation of the variables of the biological problems: the 95 synthetic domains, created by the biologists, are independent and there is no evidence about a rank among them. From a statistical point of view, we can consider each of them as a simple label (unordered categorical data). Moreover, it is extremely important the order of the domains in the $q = \{1, \dots, 4\}$ positions. More specifically, let $\Omega = \{z_1, \dots, z_i, \dots, z_N\}$ be the matrix of the whole experimental space where $N \simeq 8.1 \times 10^7$. Each protein is then formed by 4 domains $z_i = \{d_1, d_2, d_3, d_4\}$ where $d_k \in \{1, \dots, 95\}$ and consequently $\Omega$ is represented as:

| Exp | Pos1 | Pos2 | Pos3 | Pos4 | Y |
|---|---|---|---|---|---|
| $z_1$ | 1 | 1 | 1 | 1 | $y_1$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $z_i$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $y_i$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $z_{8.1*10^7}$ | 95 | 95 | 95 | 95 | $y_{8.1*10^7}$ |

Table 4.1: The experimental space $\Omega$.

The statistical representation adopted in this thesis, is a transformation of the original search space into a set of binary variables (presence or absence of the k-*th* domains in the q-*th* position). The transformed search space becomes the follow:

| Exp | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $\cdots$ | $X_j$ | $\cdots$ | $X_{377}$ | $X_{378}$ | $X_{379}$ | $X_{380}$ | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | 1 | 1 | 1 | 1 | $\cdots$ | 0 | $\cdots$ | 0 | 0 | 0 | 0 | $y_1$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $x_i$ | | | | | | $x_{ij}$ | | | | | | $y_i$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $x_{8.1*10^7}$ | 0 | 0 | 0 | 0 | $\cdots$ | 0 | $\cdots$ | 1 | 1 | 1 | 1 | $y_{8.1*10^7}$ |

Table 4.2: Binary variables representation of the experimental space.

The binary transformation converts the $z_i$ experiment into a vector $x_i$ of zeros with length $p = 95 \times 4$ and when the k-*th* domain appears in the q-*th* position of the $z_i$ protein, the variable $x_i$ in position $[(k*q)-(q-j)]$ assumes the value 1. The biological problem is then coded by $p = 380$ binary variables with an increase of the problem dimensionality.

- Sparsity and scarcity of data in high-dimensional setting: this issue is related to the number of possible experimental trials that can be tested in laboratory compared with the number of possible solutions of the search space $\Omega$. Due to the specific technology adopted in this experimentation, the biologists can initially test only $m = 96$ proteins among about $8.1 \times 10^7$ possible candidates. This means that in the first set of trials we have $m \ll p$. The main problem of the scarcity of data in the first generation concerns the information about the domains. In fact, without any *a priori* information about the experimental space and about relations among the domains, only 96 experiments do not suffice to give information about all the $p = 380$ variables. Some variable selection procedures are indeed necessary to detect the most informative possible singles (95 for each position), $95^2$ possible couples and $95^3$ possible triplets of domains.
- Model selection and assessment: the main problem is to define a good statistical model able to identify which are the most important variables and simultaneously shrink the less informative ones. In continuos settings, LASSO and Elastic Net, presented in Chapter 2, perform very well if some assumption like linearity of the parameters or independence

among marginal distributions are made. But their performances have not been proven to be effective in very high combinatorial problems with $n \ll p$ and with categorical variables in complex surfaces. The necessity of shifting toward more robust non-linear models like neural networks can be very useful. Moreover, due to the scarcity of data, a very complex task is to define models with high prediction accuracy in order to estimate precisely the response surface of the search space.

- Computational complexity: in order to determine the most interesting region of a high combinatorial problem, it is necessary to estimate the response surface of the whole experimental space and to identify most relevant subspaces. In such a setting, estimating about $8.1 \times 10^7$ experimental points becomes intractable in terms of computational cost. Therefore, we need to identify only small regions of $\Omega$ that are supposed to be the most informative about the biological system.
- Small number of experimental trials: according to the biologists, the procedure can be competitive with classical biological techniques if the algorithm is able to find the optimal or good proteins in at most 5 generations. Within this experimental constraint, new and more efficient procedures that shift the research into the problem of optimization of the design of experiments should be identified.

## 3 The proposed solution: Evolutionary Neural Network Design

In order to derive a procedure that can tackle the above mentioned problems, the Evolutionary Neural Network Design (ENN-Design) merges the strengths of the filter variable selection presented in Chapter 2 with the evolutionary techniques presented in Chapter 3. The aim of this new approach is to evolve a population of solutions to identify small subsets of good solutions from a huge combinatorial experimental space using only few but "the most informative" observations. These subsamples of the whole experimental space should represent areas with large probability to find the global optimum of sufficient good solutions. The choice of the new points in the experimental space is driven by neural network models that, at each generation, change their topology in order to increase the accuracy in predicting the surface of the search space. The optimization procedure in this approach has three steps:

1. The model selection procedure related to the definition of the best neural network model in terms of generalization capability given the observed data (lines 4-11 of Alg 2).
2. The variable selection process to identify the most informative variables (line 12-13 of 2).
3. The identification of the experimental points to form successive generations where the higher system response values are more likely to be present (lines 17-26 of Alg 2).

The iteration of such steps drives the search of the subsets of experimental points with highest response. The iterative procedure of the ENN-Design allows to evolve the initial set of points (that it is considered as the "first generation"), composed by a set of random experimental points, towards more relevant regions of the experimental space. The proposed algorithm is suited to tackle problems where many discrete variables, scarcity of data and high dimensionality are present. The use of filter variable selection and model assessment in the evolutionary procedure allows to shift the initial population of trials, generation by generation, toward the global optimum or where good solutions are more likely to be. Then, at each generation of the algorithm, a new set of $m$ individuals made by optimal or quasi-optimal experimental points is created and added to the previous generations forming a more enriched set of experimental points and a more precise neural network model in terms of adaptation and prediction is estimated.
Formally, at the beginning of the ENN-Design algorithm, a very small population of $m = 96$ experimental points is defined (Alg.2 line 1), picked randomly from the experimental space $\Omega$. Denoting $z_i = \{d_1, \ldots, d_q\}$ with $q = \{1, \ldots, 4\}$ and $d_k \in \{1, \ldots, 95\}$ one possible candidate protein of the experimental space, $m$ experiments are selected, with $m \ll N$, representing the initial population of the algorithm and their results are recorded (Alg.2 line 2).

At each iteration of the ENN-Design algorithm, the model selection procedure is applied to the collected data. A family of neural networks is built up, where a sigmoidal activation function maps the input layer to the hidden layer and a linear activation function maps the hidden layer to the output layer (Bishop 1996). Each network is characterized by the same number of nodes in the input layer, representing the $p = 380$ binary variables (obtained according to the transformation presented in Table 4.2), but it differs in the number of neurons in the hidden layer, from 2 to 20, (Alg.2 lines 4-11).
For every topological configuration, 20 different initial random weights matrixes are created, forming the pot of candidate models, among which the best network in prediction is selected. The learning of the networks is based on a back-propagation algorithm (Rumelhart et al. 1986), which runs the Levenberg-Marquardt (LM) quasi-Newton learning algorithm up to convergence. The choice of LM algorithm is due to its robustness and fast convergence compared with other methods, (Gopalakrishnan 2010, Demuth et al. 2009), when the complexity of the topology in terms of number of input variables and number of connections is very high. We train all the networks on the training set of the collected data (80% of the collected experimental points) and evaluate the generalization accuracy on the test set, which is formed by the remaining 20% of the data. The best model in prediction is the one which minimizes the root predictive error (RPE) on the test set:

$$RPE = \sqrt{\frac{\sum_{t=1}^{n^*}(y_t - \hat{y}_t)^2}{n^*}}, \tag{4.1}$$

where $n^*$ represents the 20% of the collected data. The network with the lowest RPE, is considered as the best predictive model given the observed data, and it will be used to determine the new points, that will form the next generation.
Since the evaluation of the whole experimental space is computationally demanding, a procedure to localize the variables that can be used to identify more informative subsets of the search space is then derived. Therefore, at each generation, the collected data are used also to identify the most relevant variables. In fact, according to the measured response $Y$, it is possible to determine the empirical marginal and conditional distribution for each q-*th*

position. The marginal distribution of each position $Pos_q$ is:

$$\hat{P}(Pos_q = d_k) = \frac{\sum_{i=1}^{n} I(Pos_q = d_k)}{n} \tag{4.2}$$

where the $I(\cdot)$ denotes the indicator function and assume 1 when the k-*th* domain is present in the q-*th* position of the i-*th* protein and 0 elsewhere. Since the aim of this research is to maximize the system response, we can estimate the distribution of the domains for each position conditional to the response being larger than a threshold $\gamma$ .

$$\hat{P}(Pos_q = d_k | Y \geq \gamma) = \frac{\sum_{i=1}^{n} I(f(x_i) \geq \gamma\, ,\, Pos_q = d_k)}{\sum_{i=1}^{n} I(f(x_i) \geq \gamma)}, \tag{4.3}$$

where $\sum_{i=1}^{n} I(f(x_i) \geq \gamma)$ counts the number of experiments over the threshold. Hereafter we adopt $\hat{P}(Pos_q)$ and $\hat{P}(Pos_q|Y)$ to indicate respectively the empirical marginal and conditional distribution for each position.
In Tables 4.6 and 4.7, respectively, the conditional and the marginal distributions of the first generation of the real experimentation are presented and it is possible to observe that, at most, only $(1-\gamma)\%$ of the domains appear in Table 4.6.
From Equations 4.2 and 4.3 we can calculate the Shannon's Entropy (Shannon 1948) and use it to rank all the domains.

$$H(Pos_q) = -\hat{P}(Pos_q)log(\hat{P}(Pos_q)) - (1-\hat{P}(Pos_q))log(1-\hat{P}(Pos_q)), \tag{4.4}$$

and consequently the gain of information derived by the domains that appeared only in good experiments.

$$\begin{aligned} H(Pos_q|Y \geq \gamma) = &- \hat{P}(Pos_q|Y)log(\hat{P}(Pos_q|Y)) - \\ &-(1-\hat{P}(Pos_q|Y))log(1-\hat{P}(Pos_q|Y)). \end{aligned} \tag{4.5}$$

By Equation 4.5, it is possible to identify the domains for each position which embody the highest quantity of information conditional to the higher values of the system response. We then derive a sampling probability of these domains as $\frac{H(Pos_q|Y)}{\sum_{k=1}^{95} H(Pos_q|Y)}$ and we use it to sample 10 domains for each position proportional to this probability distribution. The procedure, therefore, generates all the possible combinations of these selected domains, forming the subset of $10^4$ of the experimental space (called $S_{best}$) where the optimal solution is more likely to be present.

Similarly we can calculate the sampling probability for all the domains of each position by Equation 4.4. We use it to select proportionally other 20 domains for each position in order to give every single protein of $\Omega$ the chance to be selected for the next generation of experimental points. Therefore, all the possible combinations of these domains are produced, forming the subset of $20^4$ of the experimental space (called $S_{other}$) where the optimal solution is less likely to be present. This idea derives form the evolutionary algorithms paradigm of exploration and exploitation; in fact, the subset $S_{best}$ is formed by points that exploit areas of the search space around the best solution achieved in that moment and $S_{other}$ is composed by new candidates where few information are still available, exploring new and less known areas of the search space.
The subset of candidate solutions to enter the second generation, $S = \{S_{best} \cup S_{other}\}$, is then formed. Due to the small number of experiments in the first generation, many domains for each position are not present (in Table 4.7 they are represented by the value 0). The algorithm, therefore, generates a set of new experiments, $m_{new}$, combining only the never-appeard domains and they become part of the second generation. In this way, we may obtain within two generations at least one observation for each domains in each positions.
The third step of the optimization procedure is the selection of the points that will enter the next generation. We adopt the neural network model with the lowest value of RPE (obtained in the first step of the optimization procedure) to predict all the responses of $S$. The so predicted response surface of $S$ is used to identify experimental points with the highest predicted outputs. If we are creating the second generation, $m - m_{new}$ of the predicted points with higher response values are added to $m_{new}$ to form the set of $m$ new points; in the following generations the best neural network is used to detect all the $m$ new points (Alg.2 line 23-27).
The neural network model in the ENN-Design plays a fundamental role in detecting the best points. In fact, it drives the selection of points, identifying among the subsets $S$ of possible candidate solutions which ones are predicted to be with higher response. In this way, it guarantees that if some points of $S_{other}$ enter the next generation, these are with higher predicted response values and consequently more likely to be good experimental points.

The new generation is then evaluated by the biologists and added to the previous ones, forming a reacher and supposed to be more informative set of points. Initially, we expect to obtain more exploration of the search space, due to the new random experimental points and to scarce accuracy of the model but the procedure is built up to balance the exploration with an exploitation around the best solutions due to the identification of good domains with the entropy measure. On the enlarged set of data, a new neural network model is then learnt and the entropy measures are calculated. With the increase of observations, the neural network becomes more accurate in predicting points. Therefore, iterating the algorithms for some generations (in the real experimentation we have at most 5 generation) we expect that the procedure might identify more and more good solutions and eventually the global optimal solution, reducing the variance within the generations.
The computation is performed coupling MATLAB® (MATLAB 2008) for the neural network model (using the package *nnet* (Demuth et al. 2009)), Python (van Rossum 1995) for the computational demanding variable transformations and software R (R Development Core Team 2010) for statistical analysis. We can summarize the ENN-Design approach in the pseudo-code presented in the algorithm 2:

**Algorithm 2** Evolutionary Neural Network Design Algorithm

```
Input: High dimensional experimental space $\Omega$
Output: The Optimal solutions or a pot of very good solutions via ENN-Design
1.. $X_n \leftarrow$ First Random Population of $m$ points
2.. Measure the response $Y_n \, \forall \, X_n$
3.. for $l$ in $1, \ldots, \#$ of generations do
4..    for $r$ in $1, \ldots, \#$ of topologies do
5..       divide the population in training and test set
6..       train the $Net_r$ on the training set with Backpropagation
7..       calculate $RPE_r$ Formula (4.1) on the test set
8..       if $RPE_r \leq$ all $RPE_{1,\ldots,r-1}$ then
9..          BestNet $\leftarrow Net_r$
10..      end if
11..   end for
12..   calculate $\hat{P}(Pos_q)$ Formula (4.2) and $\hat{P}(Pos_q|Y)$ Formula 4.3;
13..   calculate $H(Pos_q)$ Formula (4.4) and $H(Pos_q|Y \geq \gamma)$ Formula 4.5;
14..   if any $\hat{P}(Pos_q) == 0$ then
15..      create $m_{new}$ as all the possible combinations of the not yet appeared domains
          (the ones with $P(Pos_q) == 0$)
16..   end if
17..   sample 10 domains for each position proportionally to $H(Pos_q|Y \geq \gamma)$
18..   create all the possible combinations of the domains obtained in line 17 ($S_{best}$)
19..   sample 20 domains for each position proportionally to $H(Pos_q)$
20..   create all the possible combinations of the domains obtained in line 19 ($S_{other}$)
21..   form the subset of candidate solution for the next generation $S = \{S_{best} \cup S_{other}\}$

22..   predict $S$ with the BestNet;
23..   if any $\hat{P}(Pos_q) == 0$ then
24..      add the best predicted experimental points to $m_{new}$ forming $X_n^*$
25..   else
          define the new population $X_n^*$ of $m$ points formed by the best predicted points
          and not yet tested;
26..   end if
27..   $X_n \leftarrow$ concatenate$\{X_n, X_n^*\}$
28.. end for
```

## 4 The simulation study

In order to evaluate the properties of the ENN-Design approach in terms of convergence toward the optimal solution and of modeling high dimensional systems, we carry out a study based on Monte Carlo simulations. The algorithms compared in this section are run for 10 generations of 96 experimental points each and replicated in 10 Monte Carlo simulations. In each simulation, the first generation of points is set to be the same for both the approaches in order to fairly compare them. Specifically, we compare the ENN-Design approach with an *ad hoc* genetic algorithm that showed, in pre-test simulations, good performances in reaching optimal solutions in high dimensional problems when the number of possible generations is very limited. The parameters of the genetic algorithm we adopt in these simulations are set as follow:

- The mutation rate is equal to 0.05.
- The innovation is due to the selection of one new complete random experiment from the not yet tested search space at each generation.
- The use of a single-point crossover.
- The selection pressure of the parents is a weighted dynamic selection probability where the experiments are selected as follow:

  $$p(x_i) = W_i \times \frac{f(\mathbf{x}_i)}{\sum_{i=1}^{m} f(\mathbf{x}_i)}, \tag{4.6}$$

  where the weights are assumed to be:

  $$W_i = \begin{cases} 0.6 & \text{if } f(x_i) \geq \gamma \\ 0.3 & \text{if } 1-\gamma \leq f(x_i) < \gamma \\ 0.1 & \text{if } f(x_i) < 1-\gamma. \end{cases} \tag{4.7}$$

  In this way we give more chance to the best parents to be selected respect to the most common proportional selection presented in Formula 3.5.

These simulative studies highlight the performance in finding the optimum or some good solutions in different scenarios that, according to the biologists, can represent some possible biological surface landscapes. Moreover, the model estimated in the ENN-Design approach is compared with LASSO and Elastic Net models in order to present the performance of the model driving the optimization of the proposed algorithm. We compare the three model in terms of their generalization accuracy in predicting complex high

| Pos1 | Pos2 | Pos3 | Pos4 |
|---|---|---|---|
| 6 (31.556) | 12 (36.823) | 5 (34.438) | 13 (31.94) |
| 8 (31.463) | 18 (38.843) | 18 (39.406) | 19 (36.991) |

Table 4.3: Non zero domains in the sparse regression model simulation.

dimensional systems. The simulative study starts with simpler situations, where the number of possible domains for each of the $q = 4$ positions are assumed to be 20, generating an experimental space $\Omega = 20^4$ of possible candidates. Then, the simulations are extended to more complex cases to highlight the robustness of the ENN-Design approach in different situations. Eventually, problems simulating the real setting with 95 possible domains for each of the 4 positions are then performed.

## 4.1 Sparse regression model

One possible biological scenario is that only very few possible domains in each q-*th* position of the protein sequence largely influence the response of the system and the others are close to 0. Such a scenario closely represents an experimental protein fitness landscape where most of the protein sequences do not posses any function (zero fitness) whereas rare functional proteins are tightly clustered together (Aita & Husimi 2001). We simulate the response of the system using the following model:

$$y = \sum_{i=1}^{k} \sum_{j=1}^{q} \beta_{ij} x_{ij} \tag{4.8}$$

where $k = 20$, $p = 80$ is the binary transformation of $q = 4$ positions and the $x_{ij}$ is equal to 1 when the i-*th* domain is in the j-*th* position, otherwise it is 0. In this case the elements that influence the response are randomly selected with uniform probability and their coefficients are drawn from a normal distribution $N(35, 10)$. Table 4.3 presents the non zero domains and their coefficient values are in the brackets. The aim of this simulation is to maximize the response of the system and Function 4.8 is optimized for $x = (6, 18, 18, 19)$ having a response value equal to 146.703.

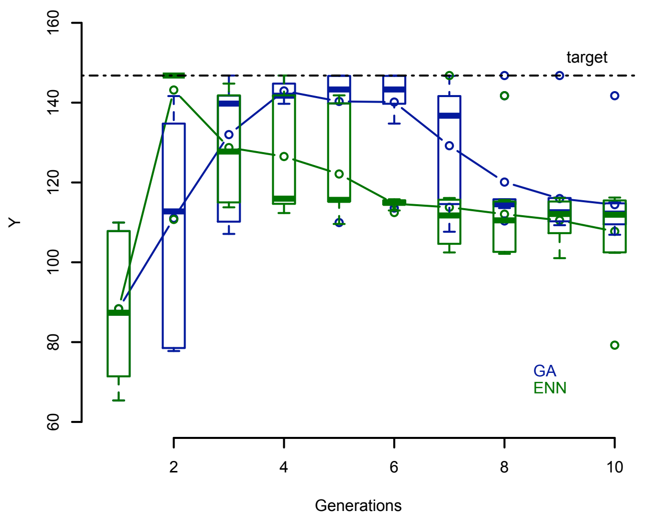


Figure 4.5: Sparse regression model: behavior of the best solutions achieved in each of the 10 Generations in 10 Monte Carlo simulations.

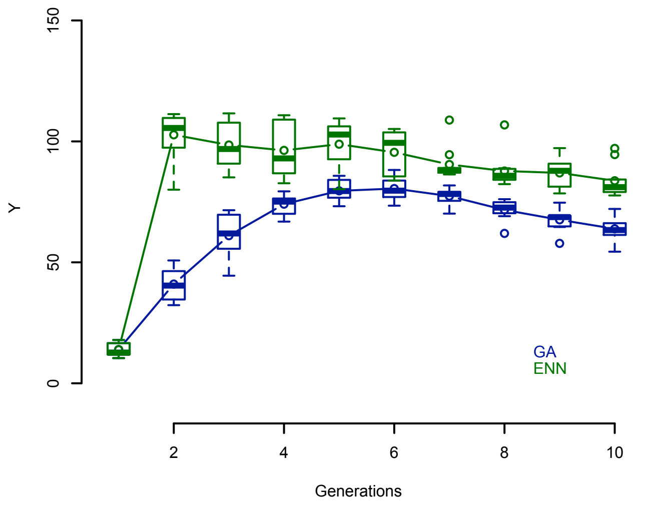


Figure 4.6: Sparse regression model: behavior of the average responses in ten 10 generations in 10 Monte Carlo simulations.

Figure 4.5 shows the comparison of the ENN-Design with the genetic algorithm in terms of the behavior of the best solutions achieved in each generation in 10 Monte Carlo simulations. The boxplot represents important statistics of the distribution of the best solutions. In particular, the bottom and top of the box are the $1^{st}$ and $3^{rd}$ quartile of the distribution of the system response and the band near the middle of the box represent the median; the bottom and upper whiskers are calculated as the lowest datum still within $1.5 \times (3^{rd} - 1^{st})$ of the lower quartile, and the highest datum still within $1.5 \times (3^{rd} - 1^{st})$ of the upper quartile. Extreme values, laying out of the whiskers, are called *outliers*. In Figure 4.5 the dash dotted line connects the mean values of each generation and the black target line represents the response value of the optimal solution. The green lines are used for the ENN-Design approach whereas the blue lines are used for the genetic algorithm.
We can notice that both the approaches reach the optimal solutions within 10 generations, but ENN-Design algorithm reaches quickly the best combination of domains and the genetic algorithms needs more time to converge toward the optimum.
Figure 4.6 shows the comparison of the ENN-Design with the genetic algorithm in terms of the behavior of the average values of each generation in 10 Monte Carlo simulations. We can highlight that the new proposed algorithm identifies most of the local optimal solutions since the behavior of the average responses always outperforms the genetic algorithm. The descending behavior of the ENN-Design boxplots in Figures 4.5 and 4.6 is due to the fact that tested experiments cannot be present in the new generations; therefore, once the algorithm reaches the best experiment the maximum value cannot be reached anymore. In this research, we are interested in localizing the global optimum and good solutions but the ENN-Design algorithm allows in addition to determine a precise model of the system. In order to assess the neural network model after the first generation of points, we perform a comparison of three models (Neural Network, Elastic Net, and LASSO models) in terms of generalization capability and of discovering the non zero coefficients.
Figures 4.7 and 4.8 presents the coefficient profile plots of the coefficient paths for fitted generalized linear model via penalized maximum likelihood (respectively the LASSO and Elastic Net penalized models). The regulariza-

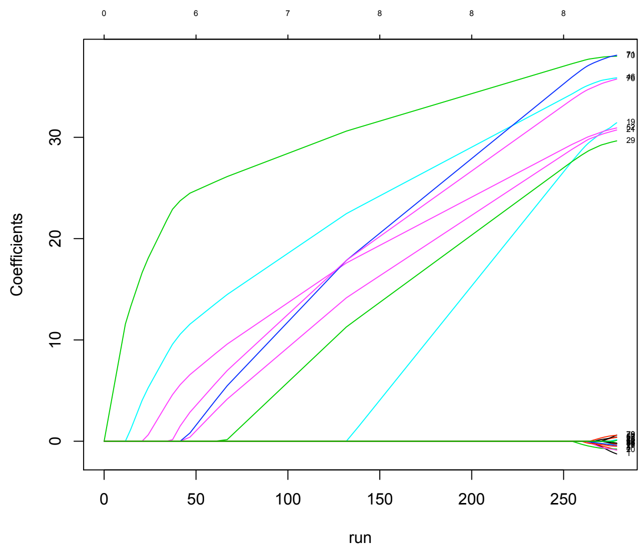


Figure 4.7: The regularization path of the LASSO model.

tion path is computed for the LASSO and Elastic Net penalty at a grid of values for the regularization parameter $\lambda$ that is set in these simulation to be descending. The values obtained at the end of the regularization path are the estimate of the variables coefficients of the models: when the values are shrunk toward 0, the variables in the models are deleted.
The neural network model, developed inside the Evolutionary procedure, is able to find the non-zero domains in the right positions since it finds out, in almost all the Monte Carlo simulation, the optimal solution in the second generation (see Figure 4.5). Moreover, both the Elastic Net and LASSO models are able to identify the non-zero domains and shrink to 0 all the others domains as shown in the Figures 4.7 and 4.8.

All the three models are efficient in finding the non zero-domains, but comparing their generalization performances on the same test set, we can immediately see a huge difference. The neural network model presents a very low root predictive error respect to the other models, as presented in Table 4.4.

Even in a simpler example than the real experimentation we can immediately see the goodness of the proposed approach in optimizing and modeling high dimensional combinatorial systems. In order to highlight the robustness of

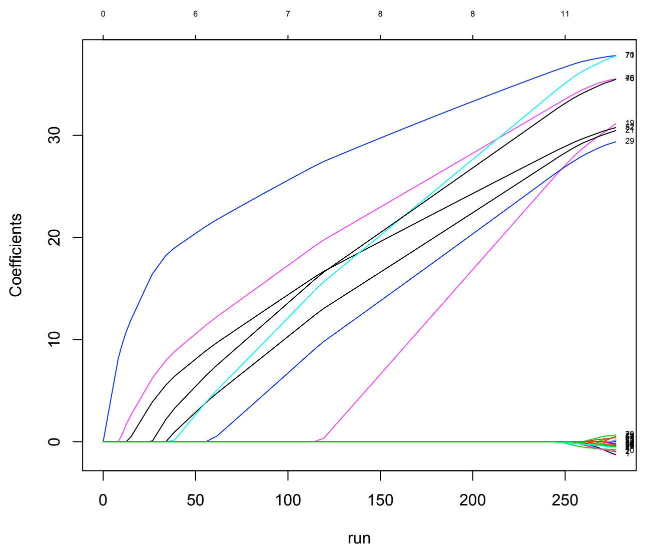


Figure 4.8: The regularization path of the Elastic Net model.

| | LASSO | Elastic Net | ENN |
|---|---|---|---|
| RPE | 5.219 | 4.218 | 0.228 |

Table 4.4: Root predictive errors of three models on the same test set in the sparse regression model simulation.

the new approach more simulative studies are developed.

## 4.2 Non-linear model

The sparse regression model to generate the response of the search space, presented in the previous simulation, is not always adequate to model complex biological systems. Therefore, same other comparative simulations are run for more complex functions and settings. The Rosenbrock function (Rosenbrock 1960) is a non-convex function, well-known as a benchmark for numerical optimization problems. The global minimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial. To converge to the global minimum, however, is difficult. The algorithms are tested on the generalized discrete Rosenbrock function

$$f(x_i) = \sum_{q=1}^{3}(100(Pos_{q+1} - Pos_q^2)^2 + (1 - Pos_q)^2). \tag{4.9}$$

This function has an optimal value equal to 0 when all the $Pos_q$s assume values equal to 1. In the bivariate case the shape of the Rosenbrock's function is represented by the Figure 4.9 where it is evident the local optimal valley.

As in the previous simulation, we compare the ENN-Design approach with the genetic algorithm, performing 10 Monte Carlo simulations each of which is run for 10 generations. Initially we test the approaches in a situations with $k = 20$ possible domains for each of the $q = 4$ positions; the domains are set to be equally spaced between -5 and 5. Even in this simulative study case, the ENN-Design outperforms the genetic algorithm both in reaching the optimal solution and in finding many good experiments of the valley. Figure 4.10 shows that the global optimum is achieved by ENN-Design within the fifth generation whereas the genetic algorithm only approaches the optimal value without hitting it in ten generations. Moreover, the performance of ENN-Design algorithm is much better than the genetic algorithm in terms of identifying the set of locally optimal solutions; in fact, Figure 4.11 shows that the average value within each generation is much lower in the ENN-Design approach than in the genetic algorithm.

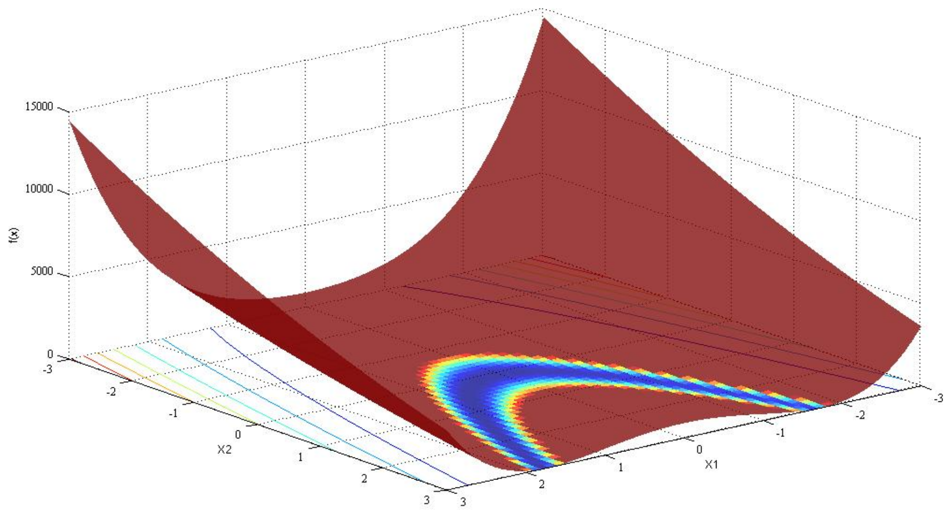


Figure 4.9: The Rosenbrock function

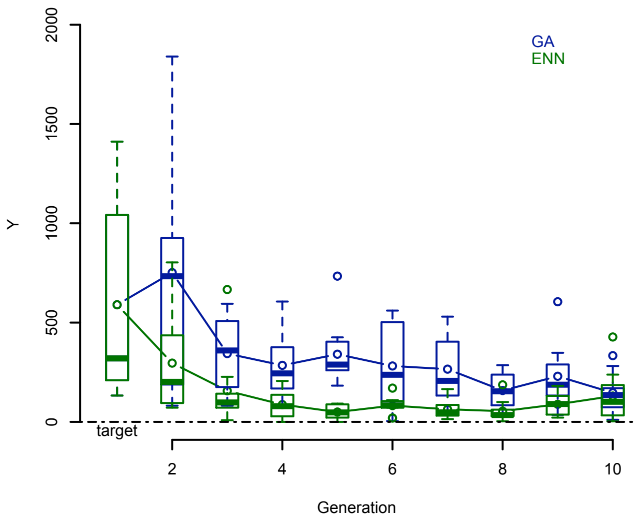


Figure 4.10: The Rosenbrock function with $k = 20$ and $q = 4$: behavior of the best solutions achieved in each of the 10 generations in 10 Monte Carlo simulations.

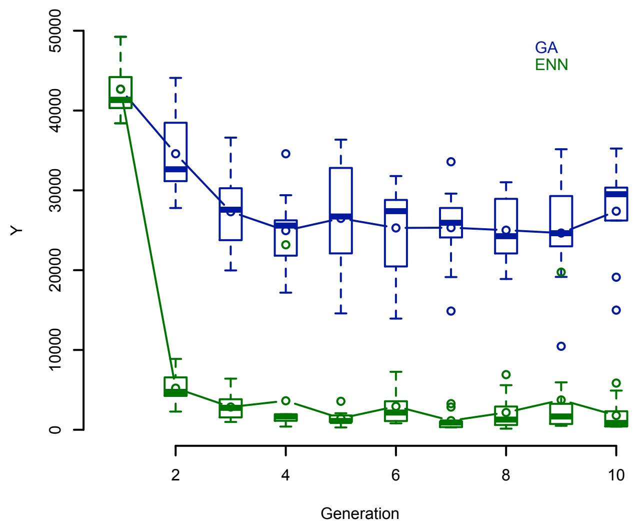


Figure 4.11: The Rosenbrock function with $k = 20$ and $q = 4$: behavior of the average responses of the 10 generations in 10 Monte Carlo simulations.

| | LASSO | Elastic Net | ENN |
|---|---|---|---|
| RPE | 43,590.56 | 32,811.34 | 11,113 |

Table 4.5: Root predictive mean squared errors of three models on the same test set in the Rosenbrock function simulation.

In this more complex scenario, the advantages of using the proposed algorithm instead of the more popular genetic algorithm is evident. In order to evaluate the goodness of generalization of the models, a comparison among the neural network model, the generalized LASSO model and the generalized Elastic Net models is performed. After the first generation, the best neural network, obtained within the ENN-Design optimization, has a much lower prediction error than the LASSO and Elastic Net models (calculated on the same test set) as showed in Table 4.5.

These simulation studies show a very good performance of our method compared to the high-performing GA and to the popular LASSO and Elastic Net models. This is the case for both reaching the optimal solution and fitting good predictive statistical models. Combining variable selection procedure to identify subsamples of the search space with higher probability to find the optimal solution and the neural network model, which represents a robust model in different scenarios, helps the improvement of the population of points, generation by generation, toward the global optimum and the suboptimal areas of the search space. The ENN-Design seems to be suited to address the model assessment and the optimization problems in high dimensional settings. Since the complexity of the Rosenbrock function could be indicate to represent the real experimentation that will be performed in laboratory, we create a new simulation with the same parameterization of the real setting. Therefore, we tested the proposed approach in the situation where the number of possible domains for each of the $q = 4$ positions are assumed to be 95 equally spaced between -5 and 5, generating an entire experimental space $\Omega$ of $N \simeq 8.1 \times 10^7$ of possible candidate solutions.

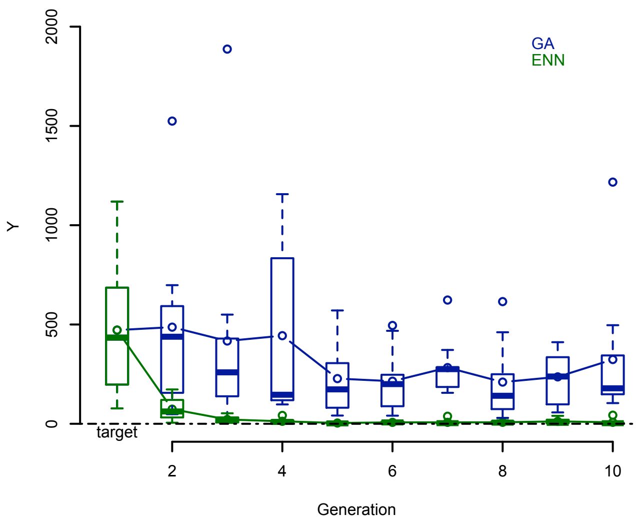


Figure 4.12: The Rosenbrock function with $k = 95$ and $q = 4$: behavior of the best solutions achieved in each of the 10 generations in 10 Monte Carlo simulations

As in previous examples, we can immediately highlight in Figures 4.12 and 4.13 how the behavior of the best solution and the behavior of the average

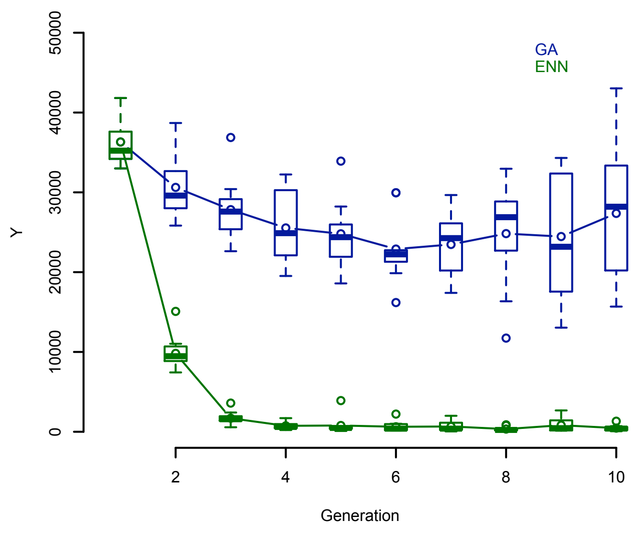


Figure 4.13: The Rosenbrock function with $k = 95$ and $q = 4$: behavior of the average responses of the 10 generations in 10 Monte Carlo simulations

response of the ENN-Design outperform the genetic algorithm. Moreover, the genetic algorithm in this huge search space is able to identify very few solutions with low response values. On the contrary, the best solutions identified by the ENN-Design approach converges since the second generation toward optimal values. In this more complex simulation the proposed approach seems to be very effective to identify the optimum and to exploit points in the valley.
Significant results are obtained in these simulation studies and they encourage to test the proposed approach in the real experimentation.

# 5 A real application: the synthetic protein discovery

The real experimentation aims to engineer novel "synthetic" proteins able to enhance catalysis within a cell. We apply the ENN-Design in order to identify some "synthetic" proteins among a huge experimental space that present a catalytic functionality comparable with the natural protein, the Serine esterase (cutinase) of Fusarium Solani. The ENN-Design approach is used to address the optimization of this high dimensional problem and the evolution of the generations of experimental points is presented. The aim of the research is to find new proteins which maximize the response of the biological system. The first generation of points is randomly selected in the huge combinatorial experimental space $\Omega$ formed by about $8.1\times10^7$ candidate "synthetic" proteins. The response values of the first generation is described in Figure 4.14.

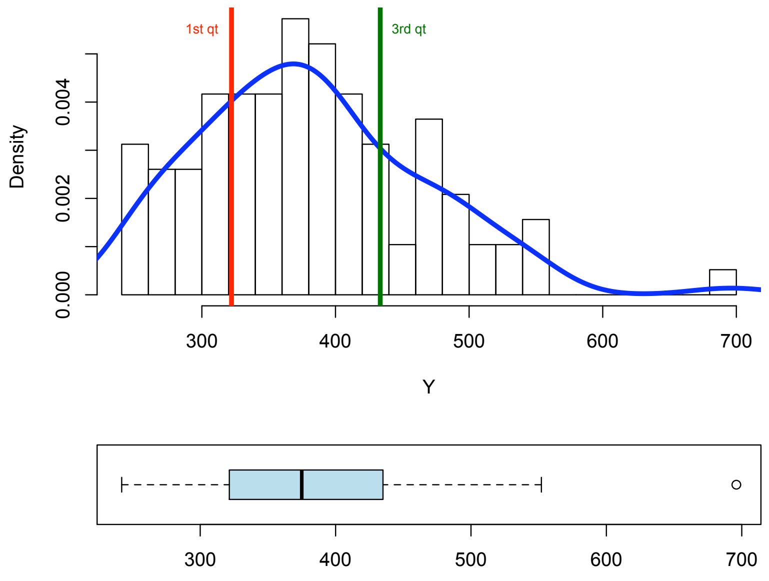


Figure 4.14: Descriptive analysis of the first generation: density distribution of the response values and some important statistics

This figure shows the distribution of the response $Y$ of the first generation of experimental points, summarizing the main statistics. We can note that the distribution of the $Y$ is almost symmetrical around the median value, except for an outlier whose response value is very high.

Since we are interested to maximize the response, we can identify which domains in each position are present in proteins with high system response values. These domains are part of "synthetic" proteins with higher system response values and, combined together, they can produce new biological entities that are more likely to be good. For example, Figure 4.15 presents all the domains appeared in position 1 of the "synthetic" proteins of the first generation. The domains appeared in experiments whose response values exceed the $3^{rd}$ quartile threshold are marked by a green vertical line; we can highlight that some of them are present only in experiments with high response values (e.g. domains labelled as 24, 39, 88) and others appear both in experiments with high and low values of the system response (e.g. domains labelled as 8, 75).

More information about the experimental space can be achieved comparing which domains in the first generation appear in experiments with the highest and the lowest values of $Y$; in fact, we need to avoid to test domains that can form new experiments that are more likely to have low response values. Figure 4.16 shows the domains in position 1 that generates the highest and the lowest response values. This information is very important in the ENN-Design approach when we calculate the probability of the domains to be candidate in forming the new set of experimental points. Tables 4.6 and 4.7 represent respectively the conditional probability distribution of the $k$ domains for each position given high values of system response and the marginal distribution of the domains for each position. This tables embody the information presented in Figure 4.16 and are applied to create the subset of variables that are used to form the next generation (as presented in Section 3).

Accordingly with the information derived with this preliminary analysis, it is reasonable that the domains in each position cannot be sufficient alone to give an higher response value of the system. Only interacting with other domains in different positions, it is possible to identify proteins with much higher response values. Figure 4.17 shows the interaction between domains of the first and second position of the first generation of "synthetic" proteins. The green bar is associated to interactions which lead to higher values of the system response (over the $3^{rd}$ quartile of the distribution) and the red bar refers to interactions leading to lower values of the response (below the $1^{st}$ quartile). We can observed that some domains in position 1 (e.g 25 and 80)

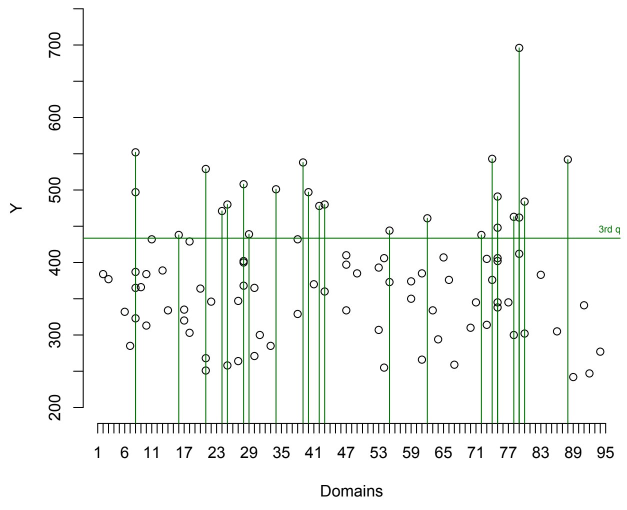


Figure 4.15: Scatterplot of the domains in position 1 of the "synthetic" proteins of the first generation.

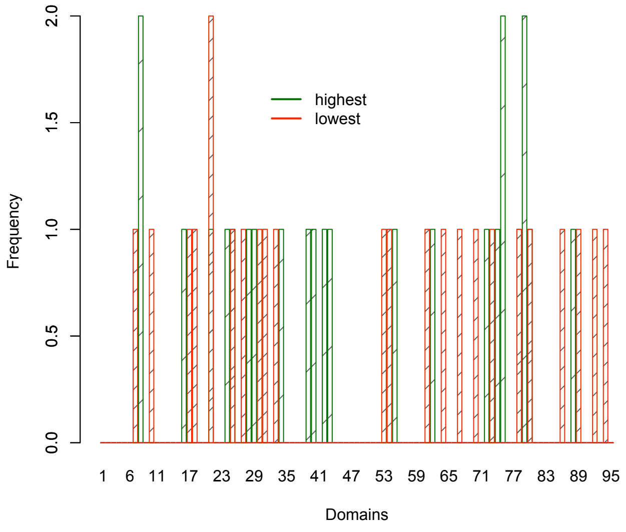


Figure 4.16: Presence of domains in position 1 in the lowest and highest experiments of the first generation.

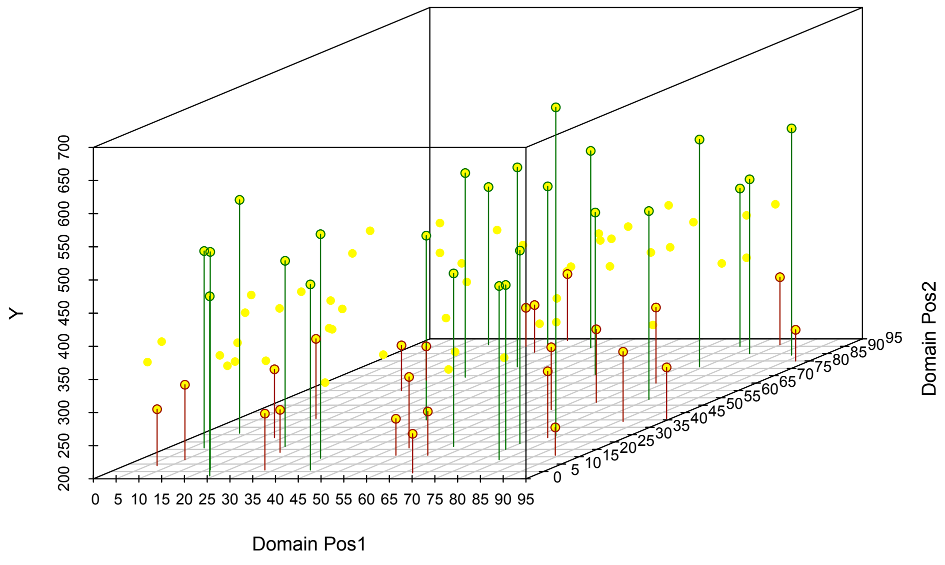
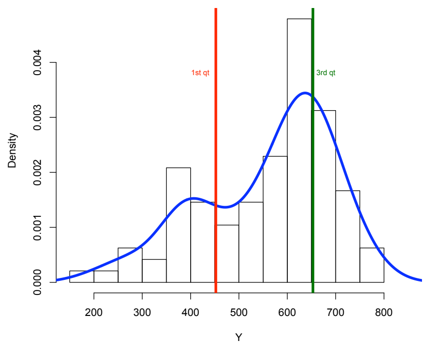


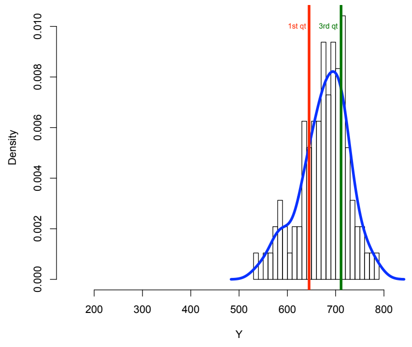
Figure 4.17: Interaction between domains in position 1 and domains in position 2 at the first generation: green bar represents higher values of the response and red bar lower values.

interacting with different domains in position 2 generate opposite results; this implies that some domains are not important to reach higher values of the system response by themselves, but they need to interact with other domains to form building block with higher probability to produce good proteins. This kind of analysis can be performed for each position and for all the possible interactions, giving the biologist interesting information about the bonds among strings of amino acids.
The ENN-Design algorithm is then run on the first generation of points, and a new set of $m$ "synthetic" proteins (not yet tested) is formed. The procedure is then iterated until the maximum number of experimental generations is reached. Figures 4.18 and 4.19 show the evolution of the initial population of random experimental points within generations.

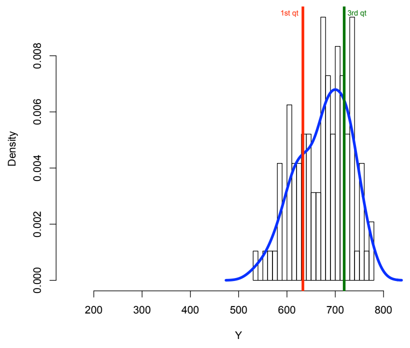
We can note that the initial exploration of the search space, due to the introduction of $m_{new}$ completely random experiments, brings to lower values of the response introducing more variability within the second generation. This exploration of the search space is balanced by the exploitation due to the cooperation of variable selection (identification of good domains with the entropy measure) and modeling procedure (prediction of the best points). In fact, since the second generation, we identify very good "synthetic" proteins, achieving much higher response values than in the first generation. The shift toward higher values is more evident in the third generation, where the ENN-Design approach identifies a pot of many good solutions. Moreover, it is evident a strong reduction of the variability in the third and forth generations, with an exploitation around the subsets of the $\Omega$ where good proteins are more like to be present.
Iterating the procedure since the fifth generation we can observed a strong improvement of the results; in fact, the last generation enables to identify a pot of good solutions with a remarkable shift of the response distribution toward optimal values. Moreover, an evident reduction of the variability is achieved, forming a very informative set of "synthetic" proteins and eventually reaching the maximum value. The evolutive path is constrained by the experimental requests to be five generations, but the response behavior of the experimental application suggests that iterating the procedure for other generations we can achieved even better solutions. We can conclude that



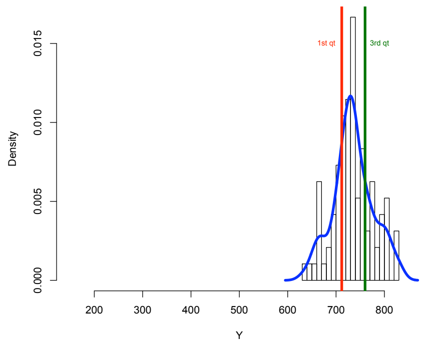
(a) Second generation



(b) Third generation



(c) Forth generation



(d) Fifth generation

Figure 4.18: Density distribution of the response values of the second (a), third (b), forth (c) and fifth (d) generation of experimental points.

the initial set of random “synthetic” proteins, therefore, has been evolved, forming new biological entities which are supposed to enhance catalysis.

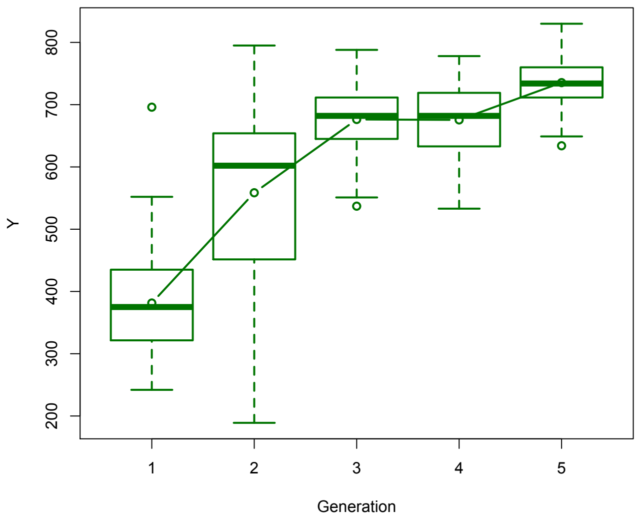


Figure 4.19: Boxplot of the response values of each generation of “synthetic” proteins.

| Domains | $\hat{P}(X_1\|Y)$ | $\hat{P}(X_2\|Y)$ | $\hat{P}(X_3\|Y)$ | $\hat{P}(X_4\|Y)$ |
|---|---|---|---|---|
| 1 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 2 | 0.0000 | 0.0442 | 0.0435 | 0.0000 |
| 3 | 0.0000 | 0.0000 | 0.0000 | 0.0469 |
| 4 | 0.0000 | 0.0000 | 0.0435 | 0.0000 |
| 5 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 6 | 0.0000 | 0.0732 | 0.0721 | 0.0000 |
| 7 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 8 | 0.0721 | 0.0000 | 0.0000 | 0.0000 |
| 9 | 0.0000 | 0.0000 | 0.0000 | 0.0469 |
| 10 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 11 | 0.0000 | 0.0000 | 0.0000 | 0.0777 |
| 12 | 0.0000 | 0.0000 | 0.0435 | 0.0469 |
| 13 | 0.0000 | 0.0442 | 0.0000 | 0.0469 |
| 14 | 0.0000 | 0.0442 | 0.0000 | 0.0000 |
| 15 | 0.0000 | 0.0000 | 0.0000 | 0.0469 |

| | | | | |
|---|---|---|---|---|
| 16 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 17 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 18 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 19 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 20 | 0.0000 | 0.0442 | 0.0435 | 0.0777 |
| 21 | 0.0435 | 0.0442 | 0.0000 | 0.0000 |
| 22 | 0.0000 | 0.0732 | 0.0000 | 0.0469 |
| 23 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 24 | 0.0435 | 0.0442 | 0.0000 | 0.0000 |
| 25 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 26 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 27 | 0.0000 | 0.0000 | 0.0435 | 0.0000 |
| 28 | 0.0435 | 0.0000 | 0.0435 | 0.0000 |
| 29 | 0.0435 | 0.0442 | 0.0435 | 0.0469 |
| 30 | 0.0000 | 0.0000 | 0.0435 | 0.0000 |
| 31 | 0.0000 | 0.0442 | 0.0000 | 0.0000 |
| 32 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 33 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 34 | 0.0435 | 0.0000 | 0.0435 | 0.0000 |
| 35 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 36 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 37 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 38 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 39 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 40 | 0.0435 | 0.0442 | 0.0000 | 0.0000 |
| 41 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 42 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 43 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 44 | 0.0000 | 0.0000 | 0.0435 | 0.0000 |
| 45 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 46 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 47 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 48 | 0.0000 | 0.0000 | 0.0000 | 0.0469 |
| 49 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |

| | | | | |
|---|---|---|---|---|
| 50 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 51 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 52 | 0.0000 | 0.0000 | 0.0435 | 0.0000 |
| 53 | 0.0000 | 0.0000 | 0.0435 | 0.0000 |
| 54 | 0.0000 | 0.0442 | 0.0000 | 0.0777 |
| 55 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 56 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 57 | 0.0000 | 0.0000 | 0.0721 | 0.0000 |
| 58 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 59 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 60 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 61 | 0.0000 | 0.0000 | 0.0435 | 0.0469 |
| 62 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 63 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 64 | 0.0000 | 0.0000 | 0.0435 | 0.0000 |
| 65 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 66 | 0.0000 | 0.0000 | 0.0000 | 0.0469 |
| 67 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 68 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 69 | 0.0000 | 0.0442 | 0.0000 | 0.0000 |
| 70 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 71 | 0.0000 | 0.0442 | 0.0000 | 0.0000 |
| 72 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 73 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 74 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 75 | 0.0721 | 0.0000 | 0.0721 | 0.0000 |
| 76 | 0.0000 | 0.0732 | 0.0000 | 0.0000 |
| 77 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 78 | 0.0435 | 0.0000 | 0.0435 | 0.0000 |
| 79 | 0.0721 | 0.0000 | 0.0000 | 0.0000 |
| 80 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 81 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 82 | 0.0000 | 0.0000 | 0.0435 | 0.0469 |
| 83 | 0.0000 | 0.0000 | 0.0000 | 0.1020 |

| | | | | |
|---|---|---|---|---|
| 84 | 0.0000 | 0.0442 | 0.0000 | 0.0469 |
| 85 | 0.0000 | 0.0442 | 0.0000 | 0.0000 |
| 86 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 87 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 88 | 0.0435 | 0.0000 | 0.0000 | 0.0000 |
| 89 | 0.0000 | 0.0442 | 0.0000 | 0.0000 |
| 90 | 0.0000 | 0.0442 | 0.0000 | 0.0000 |
| 91 | 0.0000 | 0.0732 | 0.0435 | 0.0000 |
| 92 | 0.0000 | 0.0000 | 0.0435 | 0.0000 |
| 93 | 0.0000 | 0.0000 | 0.0000 | 0.1020 |
| 94 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 95 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |

Table 4.6: Conditional probability distribution of the $k$ domains for each position, given high values of the system response.

| Domain | $\hat{P}(X_1)$ | $\hat{P}(X_2)$ | $\hat{P}(X_3)$ | $\hat{P}(X_4)$ |
|---|---|---|---|---|
| 1 | 0.0000 | 0.0000 | 0.0000 | 0.0116 |
| 2 | 0.0117 | 0.0120 | 0.0118 | 0.0116 |
| 3 | 0.0117 | 0.0000 | 0.0000 | 0.0279 |
| 4 | 0.0000 | 0.0210 | 0.0206 | 0.0116 |
| 5 | 0.0000 | 0.0210 | 0.0000 | 0.0000 |
| 6 | 0.0117 | 0.0359 | 0.0352 | 0.0116 |
| 7 | 0.0117 | 0.0210 | 0.0206 | 0.0116 |
| 8 | 0.0413 | 0.0000 | 0.0000 | 0.0203 |
| 9 | 0.0117 | 0.0288 | 0.0283 | 0.0203 |
| 10 | 0.0205 | 0.0120 | 0.0000 | 0.0116 |
| 11 | 0.0117 | 0.0000 | 0.0000 | 0.0203 |
| 12 | 0.0000 | 0.0000 | 0.0283 | 0.0116 |
| 13 | 0.0117 | 0.0210 | 0.0206 | 0.0203 |
| 14 | 0.0117 | 0.0210 | 0.0000 | 0.0203 |
| 15 | 0.0000 | 0.0000 | 0.0000 | 0.0203 |
| 16 | 0.0117 | 0.0424 | 0.0118 | 0.0000 |
| 17 | 0.0205 | 0.0120 | 0.0118 | 0.0000 |
| 18 | 0.0205 | 0.0120 | 0.0000 | 0.0000 |
| 19 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 20 | 0.0117 | 0.0120 | 0.0206 | 0.0279 |
| 21 | 0.0281 | 0.0210 | 0.0283 | 0.0203 |
| 22 | 0.0117 | 0.0210 | 0.0118 | 0.0116 |
| 23 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 24 | 0.0117 | 0.0210 | 0.0206 | 0.0116 |
| 25 | 0.0205 | 0.0000 | 0.0000 | 0.0000 |
| 26 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 27 | 0.0205 | 0.0000 | 0.0118 | 0.0116 |
| 28 | 0.0350 | 0.0288 | 0.0118 | 0.0116 |
| 29 | 0.0117 | 0.0120 | 0.0206 | 0.0116 |
| 30 | 0.0205 | 0.0120 | 0.0206 | 0.0203 |
| 31 | 0.0117 | 0.0210 | 0.0118 | 0.0203 |
| 32 | 0.0000 | 0.0000 | 0.0118 | 0.0203 |

| | | | | |
|---|---|---|---|---|
| 33 | 0.0117 | 0.0120 | 0.0000 | 0.0000 |
| 34 | 0.0117 | 0.0000 | 0.0118 | 0.0203 |
| 35 | 0.0000 | 0.0000 | 0.0000 | 0.0203 |
| 36 | 0.0000 | 0.0000 | 0.0118 | 0.0000 |
| 37 | 0.0000 | 0.0000 | 0.0118 | 0.0000 |
| 38 | 0.0205 | 0.0000 | 0.0000 | 0.0203 |
| 39 | 0.0117 | 0.0120 | 0.0118 | 0.0000 |
| 40 | 0.0117 | 0.0120 | 0.0000 | 0.0116 |
| 41 | 0.0117 | 0.0359 | 0.0000 | 0.0116 |
| 42 | 0.0117 | 0.0210 | 0.0000 | 0.0000 |
| 43 | 0.0205 | 0.0000 | 0.0000 | 0.0203 |
| 44 | 0.0000 | 0.0000 | 0.0206 | 0.0000 |
| 45 | 0.0000 | 0.0210 | 0.0000 | 0.0203 |
| 46 | 0.0000 | 0.0000 | 0.0118 | 0.0116 |
| 47 | 0.0281 | 0.0120 | 0.0206 | 0.0000 |
| 48 | 0.0000 | 0.0000 | 0.0118 | 0.0203 |
| 49 | 0.0117 | 0.0120 | 0.0206 | 0.0116 |
| 50 | 0.0000 | 0.0120 | 0.0000 | 0.0116 |
| 51 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 52 | 0.0000 | 0.0210 | 0.0118 | 0.0116 |
| 53 | 0.0205 | 0.0000 | 0.0206 | 0.0203 |
| 54 | 0.0205 | 0.0120 | 0.0118 | 0.0279 |
| 55 | 0.0205 | 0.0120 | 0.0206 | 0.0116 |
| 56 | 0.0000 | 0.0000 | 0.0118 | 0.0203 |
| 57 | 0.0000 | 0.0000 | 0.0283 | 0.0000 |
| 58 | 0.0000 | 0.0000 | 0.0118 | 0.0000 |
| 59 | 0.0205 | 0.0000 | 0.0118 | 0.0000 |
| 60 | 0.0000 | 0.0210 | 0.0000 | 0.0203 |
| 61 | 0.0205 | 0.0120 | 0.0118 | 0.0203 |
| 62 | 0.0117 | 0.0000 | 0.0206 | 0.0000 |
| 63 | 0.0117 | 0.0120 | 0.0000 | 0.0116 |
| 64 | 0.0117 | 0.0120 | 0.0118 | 0.0116 |
| 65 | 0.0117 | 0.0120 | 0.0000 | 0.0000 |
| 66 | 0.0117 | 0.0000 | 0.0000 | 0.0279 |

| | | | | |
|---|---|---|---|---|
| 67 | 0.0117 | 0.0120 | 0.0118 | 0.0116 |
| 68 | 0.0000 | 0.0000 | 0.0206 | 0.0116 |
| 69 | 0.0000 | 0.0120 | 0.0206 | 0.0000 |
| 70 | 0.0117 | 0.0000 | 0.0000 | 0.0116 |
| 71 | 0.0117 | 0.0120 | 0.0000 | 0.0000 |
| 72 | 0.0117 | 0.0000 | 0.0000 | 0.0000 |
| 73 | 0.0205 | 0.0000 | 0.0118 | 0.0116 |
| 74 | 0.0205 | 0.0000 | 0.0000 | 0.0000 |
| 75 | 0.0472 | 0.0120 | 0.0283 | 0.0000 |
| 76 | 0.0000 | 0.0210 | 0.0206 | 0.0000 |
| 77 | 0.0117 | 0.0000 | 0.0206 | 0.0000 |
| 78 | 0.0205 | 0.0210 | 0.0118 | 0.0000 |
| 79 | 0.0281 | 0.0000 | 0.0118 | 0.0116 |
| 80 | 0.0205 | 0.0120 | 0.0206 | 0.0116 |
| 81 | 0.0000 | 0.0120 | 0.0000 | 0.0000 |
| 82 | 0.0000 | 0.0000 | 0.0118 | 0.0116 |
| 83 | 0.0117 | 0.0000 | 0.0000 | 0.0279 |
| 84 | 0.0000 | 0.0120 | 0.0352 | 0.0347 |
| 85 | 0.0000 | 0.0120 | 0.0000 | 0.0000 |
| 86 | 0.0117 | 0.0288 | 0.0000 | 0.0000 |
| 87 | 0.0000 | 0.0000 | 0.0206 | 0.0116 |
| 88 | 0.0117 | 0.0288 | 0.0000 | 0.0000 |
| 89 | 0.0117 | 0.0120 | 0.0118 | 0.0116 |
| 90 | 0.0000 | 0.0288 | 0.0206 | 0.0000 |
| 91 | 0.0117 | 0.0485 | 0.0352 | 0.0000 |
| 92 | 0.0117 | 0.0000 | 0.0118 | 0.0116 |
| 93 | 0.0000 | 0.0210 | 0.0000 | 0.0279 |
| 94 | 0.0117 | 0.0210 | 0.0000 | 0.0203 |
| 95 | 0.0000 | 0.0000 | 0.0118 | 0.0116 |

Table 4.7: Marginal probability distribution of the $k$ domains in the first generation of random points.

# Conclusion

The issue of high dimensionality represents a challenging topic for statisticians and data miners with many problems that are still unsolved. Classical statistical methods cannot cope with the increasing availability of massive data. The large number of variables and the scarcity of observations characterize the current state of real applications, therefore, new methods and theories need to be developed.

The thesis is an attempt to overcome classical statistical limitations in modeling and optimizing high dimensional systems, giving a new flexible tool for identifying important variables and modeling complex scenarios. The introduction of statistical model and variable selection procedures in the paradigm of evolution is still an innovation in the optimization fields. Moreover, the attempt to solve the optimization problem in very few generations of experimental trials can be fundamental to address costly experimentation.

This attempt might be very competitive in high dimensional optimization problems both in reaching optimal solutions and in producing robust predictive models. It has been proven in the simulative examples and eventually in the real experimentation to be more flexible and robust than the classical approaches in comparison with benchmarks as genetic algorithm for optimization and LASSO and Elastic Net for modeling.

The Evolutionary Neural Network Design algorithm, developed in this thesis, is capable of handling problems with very few observations compared with the number of variables. It merges some of the stronger features of traditional approaches in order to derive a new evolutive procedure, optimizing complex scenarios. The approach combines a model selection procedure, used to assess accurate predictive neural network models, with a ranking variable selection strategy. The procedure shifts the classical problem of variable and model

selection in the domain of the evolutionary experimental design of experiments.
In fact, the model and variable selection procedures drive the evolution of an initial random generation of points toward subsets of the search space where the optimal solutions are more likely to be found. At each generation of points, both the model and the variable ranking procedure evolve. This increases the efficiency of the algorithm in identifying new optimal solutions. The results of some simulation studies show an important improvement of ENN-Design over the traditional and widely applied optimization techniques. Moreover, the application of the ENN-Design method for identifying new "synthetic" proteins among a huge search space of competitive candidates shows a remarkable shift of the initial population toward higher response values areas of the search space. As a result it generates "synthetic" proteins which are more likely to enhance a chemical reaction.
A first effort has been made in this thesis to solve this high combinatorial optimization problem, but further efforts are needed in order to create a more efficient computational procedure. Embedding the variable selection directly in the neural network model selection procedure could be, for example, an important step toward creating network topologies with fewer input neurons and connections and, consequently, a more efficient algorithm.

# Bibliography

Aita, T. & Husimi, M. I. Y. (2001), 'A cross-section of the fitness landscape of dihydrofolate reductase.', *Protein Engineering Design and Selection* **14**(9), 633–638.

Akaike, H. (1973), 'Maximum likelihood identification of gaussian autoregressive moving average models', *Biometrika* **60**(2), 255–265.

Altschul, S., Madden, T., Schäffer, A., Zhang, J., Zhang, Z., Miller, W. & Lipman, D. (1997), 'Gapped blast and psi-blast: a new generation of protein database search programs', *Nucleic Acids Research* **25**(17), 3389–3402.

Arlot, S. & Celisse, A. (2010), 'A survey of cross-validation procedures for model selection', *Statistics Surveys* **4**, 40–79.

Bäck, T. (1996), *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*, Oxford University Press, Dortmund, Germany.

Back, T., Hammel, U. & Schwefel, H. (1997), 'Evolutionary computation: comments on the history and current state', *IEEE Transactions on Evolutionary Computation* **1**(1), 3–17.

Baragona, R., Battaglia, F. & Poli, I. (2011), *Evolutionary Statistical Procedures*, Springer-Verlag.

Battiti, R. (1992), 'First and second-order methods for learning: between steepest descent and newton's method', *Neural Computation* **4**, 141–166.

Beasley, D., Bull, D. R. & Martin, R. R. (1993), 'An overview of genetic algorithms: Part 1, fundamentals', *University Computing* **15**(2), 58–69.

Bekkerman, R., Yaniv, R., Tishby, N. & Winter, Y. (2003), ‘Distributional word clusters vs. words for text categorization’, *Journal of Machine Learning Research* **3**, 1183–1208.

Bellman (1961), *Adaptive Control Processes*, Princeton University Press.

Bishop, C. (1996), *Neural Networks for Pattern Recognition*, 1 edn, Oxford University Press, USA.

Blum, A. & Langley, P. (1997), ‘Selection of relevant features and examples in machine learning’, *Artificial Intelligence* **97**(1-2), 245–271.

Boser, B. E., Guyon, I. M. & Vapnik, V. N. (1992), A training algorithm for optimal margin classifiers, *in* ‘COLT ’92: Proceedings of the fifth annual workshop on Computational learning theory’, ACM, New York, NY, USA, pp. 144–152.
**URL:** *http://dx.doi.org/10.1145/130385.130401*

Breiman, L. (2001), ‘Random forests’, *Machine Learning* **45**, 5–32.

Breiman, L., Friedman, J., Stone, C. & Olshen, R. (1984), *Classification and Regression Trees*, 1 edn, Chapman and Hall/CRC.

Broyden, C. (1970), ‘The convergence of a class of double-rank minimization algorithms: 2. the new algorithm’, *IMA J Appl Math* **6**(3), 222–231.

Caruana, R., Sa, V. R. D., Guyon, I. & Elisseeff, A. (2003), ‘Benefitting from the variables that variable selection discards’, *Journal of Machine Learning* **3**, 1245–1264.

Clomburg, J. & Gonzalez, R. (2010), ‘Biofuel production in *Escherichia coli*: the role of metabolic engineering and synthetic biology’, *Applied Microbiology and Biotechnology* **86**(2), 419–434–434.

D. De March, D. Slanzi, I. P. (2009), Evolutionary algorithms for complex experimental designs, *in* A. P. S.M. Ermakov, V.B. Melas, ed., ‘Proceedings of the 6th St. Petersburg Workshop on Simulation’, Vol. I, St. Petersburg VVM com. Ltd, St. Petersburg, Russia, pp. 547–551.

D. Slanzi, D. De March, I. P. (2009), Probabilistic graphical models in high dimensional systems, *in* A. P. S.M. Ermakov, V.B. Melas, ed., 'Proceedings of the 6th St. Petersburg Workshop on Simulation', Vol. I, St. Petersburg VVM com. Ltd, St. Petersburg, Russia, pp. 557–560.

Danino, T., Mondragón-Palomino, O., Tsimring, L. & Hasty, J. (2010), 'A synchronized quorum of genetic clocks.', *Nature* **463**(7279), 326–330.

Darwin, C. (1872), *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, sixth edn, J. Murray.

De Lucrezia, D., Minervini, G. & Poli, I. (2009), A novel similarity metric to evaluate secondary structure, Technical report, ECLT European Centre for Living Technology.

De March, D., Forlin, M., Slanzi, D. & Poli, I. (2008), An evolutionary predictive approach to design high dimensional experiments, *in* R. Serra, M. Villani & I. Poli, eds, 'Artificial Life and Evolutionary Computation:Proceedings of WIVACE 2008', World Scientific, pp. 81–88.

De March, D., Slanzi, D. & Poli, I. (2009), Evolutionary algorithm for complex experimental designs, *in* P. A. Ermakov S.M., Melas V.B., ed., 'Proceedings of 6th St. Petersburg Workshop on Simulation', Vol. 2, pp. 547–551.

Demuth, H., Beale, M. & Hagan, M. (2009), 'Matlab neural network toolbox user's guide version 6. the mathworks inc'.

Donoho, D. L. (2000), Aide-memoire. high-dimensional data analysis: The curses and blessings of dimensionality, *in* 'American Mathematical Society Conf. Math Challenges of the 21st Century'.

Dreyfuss, G. (2005), *Neural Networks: Methodology and Applications*, Springer.

Efron, B. (1979), 'Bootstrap methods: Another look at the jackknife', *The Annals of Statistics* **7**(1), 1–26.

Efron, B. (1983), 'Estimating the error rate of a prediction rule: Improvement on cross-validation', *Journal of the American Statistical Association* **78**(382), 316–331.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004), 'Least angle regression', *Annals of Statistics* **32**(2), 407–451.

Efron, B. & Tibshirani, R. (1994), *An Introduction to the Bootstrap (Monographs on Statistics and Applied Probability)*, Chapman and Hall-CRC.

Efron, B. & Tibshirani, R. (1997), 'Improvements on cross-validation: The .632+ bootstrap method', *Journal of the American Statistical Association* **92**(438), 548–560.

Eiben, A. & Smith, J. (2008), *Introduction to Evolutionary Computing*, Springer.

Engelbrecht, A. P. (2007), *Computational Intelligence: An Introduction*, Wiley-IEEE Press.

Fan, J. & Li, R. (2001), 'Variable selection via nonconcave penalized likelihood and its oracle properties', *Journal of the American Statistical Association* **96**(456), 1348–1360.

Fan, J. & Li, R. (2006), Statistical challenges with high dimensionality: feature selection in knowledge discovery, *in* J. L. M.Sanz-Sole, J.Soria & J.Verdera, eds, 'International Congress of Mathematicians', Vol. 3, European Mathematical Society, Freiburg, pp. 595–622.

Fletcher, R. (1970), 'A new approach to variable metric algorithms', *The Computer Journal* **13**(3), 317–322.

Fogel, D. (2005), *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, Wiley-IEEE Press.

Fogel, G. & Corne, D. (2002), *Evolutionary computation in bioinformatics*, Morgan Kaufmann.

Forlin, M., Poli, I., De March, D., Packard, N., Gazzola, G. & Serra, R. (2008), 'Evolutionary design of experiments for self-assembling amphiphilic systems', *Chemometrics and Intelligent Laboratory Systems* **90**(2), 153–160.

Forman, G. (2003), 'An extensive empirical study of feature selection metrics for text classification', *Journal of Machine Learning Research* **3**, 1289–1305.

Fouskakis, D. & Draper, D. (2002), 'Stochastic optimization: a review', *International Statistical Review* **70**(3), 315–349.

Fouskakis, D. & Draper, D. (2008), 'Comparing stochastic optimization methods for variable selection in binary outcome prediction, with application to health policy', *Journal of the American Statistical Association* **103**(484), 1367–1381.

Frank, I. & Friedman, J. (1993), 'A statistical view of some chemometrics regression tools', *Technometrics* **35**(2), 109–135.

Fraser, A. S. (1962), 'Simulation of genetic systems', *Journal of Theoretical Biology* **2**(3), 329 – 346.

Friedman, J., Hastie, T., Rosset, S., Tibshirani, R. & Zhu, J. (2004), 'Discussion of boosting papers', *The Annals of Statistics* **32**, 102–107.

Friedman, J., Hastie, T. & Tibshirani, R. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**(1), 1–22.

Gilli, M. & Winker, P. (2008), Heuristic optimization methods in econometrics, *in* D. A. Belsley & E. Kontoghiorghes, eds, 'Handbook of Computational Econometrics', Wiley, Chichester, pp. 81–119.

Glover, F. (1977), 'Heuristics for integer programming using surrogate constraints', *Decision Sciences;* **8**(1), 156–166.

Glover, F. W. & Kochenberger, G. A. (2003), *Handbook of Metaheuristics (International Series in Operations Research & Management Science)*, Springer.

Goldfarb, D. (1970), 'A family of variable-metric methods derived by variational means', *Mathematics of Computation* **24**(109), 23–26.

Gopalakrishnan, K. (2010), 'Effect of training algorithms on neural networks aided pavement diagnosis', *International Journal of Engineering, Science and Technology* **2**(2), 83–92.

Gray, F. (1953), 'Pulse code communication', Patent n 2733432.

Guyon, I. & Elisseeff, A. (2003), 'An introduction to variable and feature selection', *Journal of Machine Learning Research: Special Issue on Variable and Feature Selection* **3**, 1157–1182.

Hagan, M., Demuth, H. & Beale, M. (1995), 'Neural network design'.

Haldane, J. B. S. (1928), *Possible Worlds*, Hugh and Bros., New York.

Hastie, T., Tibshirani, R. & Friedman, J. (2003), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, corrected edn, Springer.

Hoerl, A. & Kennard, R. (1970), 'Ridge regression: Biased estimation for nonorthogonal problems', *Technometrics* **12**(1), 55–67.

Holland, J. (1975), *Adaptation in natural and artificial systems*, University of Michigan Press.

Hong, D., Chen, H.-X., Ge, R. & Li, J.-C. (2010), 'Genetically engineered mesenchymal stem cells: The ongoing research for bone tissue engineering', *The Anatomical Record: Advances in Integrative Anatomy and Evolutionary Biology* **293**, 531–537.

Izenman, A. (2008), *Modern Multivariate Statistical Techniques : Regression, Classification, and Manifold Learning*, Springer New York.

John, G. H., Kohavi, R. & Pfleger, K. (1994), Irrelevant features and the subset selection problem, *in* 'International Conference on Machine Learning', pp. 121–129.

Johnstone, I. M. & Titterington, D. M. (2009), 'Statistical challenges of high-dimensional data', *Philos Transact A Math Phys Eng Sci* **367**(1906), 4237–53.

Kallel, R., Cottrell, M. & Vigneron, V. (2002), ‘Bootstrap for neural model selection’, *Neurocomputing* **48**, 175–183.

Kardaun, O. J. W. F. (2005), *Classical Methods of Statistics*, Springer-Verlag.

Kelly, J. P., Lagunat, T. M. & Glover, F. (1994), ‘A study of diversification strategies for the quadratic assignment problem’, *Computers and Operations Research* **21**, 885–893.

Khalili, A. & Chen, J. (2007), ‘Variable selection in finite mixture of regression models’, *Journal of the American Statistical Association* **102**(479), 1025–1038.

Kirkpatrick, S., Gelatt, C. & Vecchi, M. (1983), ‘Optimization by simulated annealing’, *Science* **220**(4598), 671–680.

Kohavi, R. & John, G. (1997), ‘Wrappers for feature subset selection’, *Artificial Intelligence* **97**(1-2), 273–324.

Lal, T. N., Chapelle, O., Weston, J. & Elisseeff, A. (2006), *Feature Extraction, Foundations and Application*, Springer-Verlag, chapter Embedded Methods.

Larson, S. (1931), ‘The shrinkage of the coefficient of multiple correlation’, *Journal of Educational Psychology* **22**(1), 45 – 55.

Lashkia, G. V. & Anthony, L. (2004), ‘Relevant, irredundant feature selection and noisy example elimination’, *IEEE Trans Syst Man Cybern B Cybern* **34**(2), 888–97.

Lee, K. Y. & El-Sharkawi, M. A., eds (2008), *Modern Heuristic Optimization Techniques - Theory and application to Power Systems*, Wiley-IEEE Press.

Levenberg, K. (1944), ‘A method for the solution of certain non-linear problems in least squares’, *Quarterly Journal of Applied Mathmatics* **2**(2), 164–168.

Levine, D. (1997), ‘Genetic algorithms: A practitioner’s view’, *INFORMS Journal on Computing* **9**(3), 256–259.

Mallows, C. (1973), 'Some comments on cp', *Technometrics* **15**(1), 661–675.

Marquardt, D. (1963), 'An algorithm for least-squares estimation of nonlinear parameters', *SIAM Journal on Applied Mathematics* **11**(2), 431–441.

MATLAB (2008), *version 7.6.0 (R2008a)*, The MathWorks Inc., Natick, Massachusetts.
**URL:** *http://www.mathworks.com/*

McCulloch, W. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biology* **5**(4), 115–133.

Mcguffin, L., Bryson, K. & Jones, D. (2000), 'The psipred protein structure prediction server.', *Bioinformatics* **16**(4), 404–405.

Meier, L., van de Geer, S. & Buhlmann, P. (2008), 'The group lasso for logistic regression', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70**(1), 53–71.

Meinshausen, N. (2007), 'Relaxed lasso', *Computational Statistics & Data Analysis* **52**(1), 374 – 393.

Miller, S. L. (1953), 'A production of amino acids under possible primitive earth conditions', *Science* **117**, 528–529.

Minsky, M. & Papert, S. (1969), *Perceptrons: An Introduction to Computational Geometry*, The MIT Press.

Mitra, P., Murthy, C. & Pal, S. (2002), 'Ieee transactions on pattern analysis and machine intelligence; unsupervised feature selection using feature similarity', **24**(4).

Mosteller, F. & Tukey, J. W. (1968), Data analysis, including statistics, *in* 'In G Lindzey and E Aronson, Eds., Handbook of Social Psychology, 2nd edition', Addison-Wesley, pp. 80–203.

Oparin, A. I. (1924), *The Origin of Life*, Moscow Worker publisher, Moscow.

Polya, G. (1971), *How to Solve It*, Princeton University Press.

R Development Core Team (2010), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.
**URL:** *http://www.R-project.org/*

Radcliffe, N. J. & Surry, P. D. (1995), 'Fitness variance of formae and performance predictions'.

Rechenberg, I. (1973), *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*, Frommann-Holzboog.

Reeves, C. R. (1995), 'A genetic algorithm for flowshop sequencing', *Computers and Operations Research* **22**(1), 5 – 13. Genetic Algorithms.

Rosenblatt, F. (1962), *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*, Spartan Books Washington,.

Rosenbrock, H. (1960), 'An automatic method for finding the greatest or least value of a function', *The Computer Journal* **3**(3), 175–184.

Roth, V. & Fischer, B. (2008), The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms, *in* 'Proceedings of the 25th international conference on Machine learning', ICML '08, ACM, New York, NY, USA, pp. 848–855.

Rudolph, G. (1996), Convergence of evolutionary algorithms in general search spaces, *in* 'in Proc. of the third Congress on Evolutionary Computation', IEEE Computer Society Press, pp. 50–54.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), *Learning internal representations by error propagation*, MIT Press, Cambridge, MA, USA, pp. 318–362.

Saeys, Y., Inza, I. & Larrañaga, P. (2007), 'A review of feature selection techniques in bioinformatics', *Bioinformatics* **23**(19), 2507–17.

Schwarz, G. (1978), 'Estimating the dimension of a model', *The Annals of Statistics* **6**(2), 461–464.

Shanno, D. (1970), 'Conditioning of quasi-newton methods for function minimization', *Mathematics of Computation* **24**(111), 647–656.

Shannon, C. E. (1948), 'A mathematical theory of communication', *Bell system technical journal* **27**.

Shao, J. (1996), 'Bootstrap model selection', *Journal of the American Statistical Association* **91**(434), 655–665.

Stone, M. (1974), 'Cross-validatory choice and assessment of statistical predictions', *Journal of the Royal Statistical Society. Series B (Methodological)* **36**(2), 111–147.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society Series B-Methodological* **58**(1), 267–288.

van Rossum, G. (1995), Python tutorial, Technical report, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, NL.
**URL:** *http://www.python.org/*

Voß, S. (2001), Meta-heuristics: The state of the art, *in* 'Proceedings of the Workshop on Local Search for Planning and Scheduling-Revised Papers', Springer-Verlag, London, UK, pp. 1–23.

Voß, S., Martello, S., Osman, I. & Roucairol, C. (1999), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, Boston.

Whitley, D. (2001), 'Information and software technology; an overview of evolutionary algorithms: practical issues and common pitfalls', **43**(14), 817–831.

Winker, P. (2000), 'Optimized multivariate lag structure selection', *Computational Economics* **16**, 87–103.

Winker, P. & Maringer, D. (2007), The threshold accepting optimisation algorithm in economics and statistics, *in* E. J. Kontoghiorghes & C. Gatu, eds, 'Optimisation, Econometric and Financial Analysis', Vol. 9 of *Advances in Computational Management Science*, Springer, Berlin, pp. 107–125.

Yu-Fen, W. & Xiao-Juan, G. (2009), Real-coded genetic algorithm and application in the automatic composing the test paper, *in* 'Intelligent Information Technology Application'.

Yu, L. & Liu, H. (2004), 'Efficient feature selection via analysis of relevance and redundancy', *Journal of Machine Learning Research* **5**, 1205–1224.

Zhao, H. (2007), 'Directed evolution of novel protein functions.', *Biotechnology and bioengineering* **98**(2), 313–317.

Zou, H. & Hastie, T. (2005), 'Regularization and variable selection via the elastic net', *Journal of the Royal Statistical Society Series B-Statistical Methodology* **67**, 301–320.

Zou & Hui (2006), 'The adaptive lasso and its oracle properties', *Journal of the American Statistical Association* **101**(476), 1418–1429.

# Acknowledgement

This thesis is the result of a long collaboration with very special people of the European Center for Living Technology. The supervision and the suggestions from Prof. Irene Poli, director of this International center, were and are source of inspiration in developing new researches and tackling exciting challenges. Without her trust and sacrifices this thesis would have never been written. I would acknowledge my supervisor Prof. Andrea Giommi for encouraging me to embark on this innovative research and for participating actively in the research. A special thanks to my old schoolmates and colleagues Matteo, Michele and Tommy with whom I embarked this enterprise to achieve our shared goal; I have the most extraordinary adventures with you since we met at the university and i hope they will continue. I'm grateful to the entire research group (in particular Debbi , DDL, Giovanni, Ale, DF, Alex, Elena, Laura) with whom I will be sharing a long research career; they are not just simple colleagues, they are close friends that are and will be part of my experiences and life. A special acknowledgment goes to Prof. Phil J. Brown for welcoming me at the University of Kent. His scientific advice and clever suggestions were extremely important to the development of the idea presented in this thesis. I would like to thank Teresa and Roger Jones who invited me to join the "QForma" company where I met a fantastic teamwork and great people and to give me the chance to explore new social contexts. The most important thanks are devoted to people who trusted, encouraged me during my everyday life. Thanks Giovanna and Franco, my parents, who always support my decision and the choices of my life. Finally I can give them back a rewarding result and satisfaction for this success. A lovely thanks to Fanny, who stood by my side in every moment of my last years, inspiring me and breathing life into my activities. Thanks to all my friends, in particular

Toz, Ozz, Ale, Bon, my “brother” Gab and his girlfriend Marta, Giovanni, Linda, Stacey, Tizi, Chiara and Isi, Vale, Jacopo, Lucio and many others, for their always interesting and active discussion and arguments, they are wonderful relaxing people when the stress of the life reaches its peaks. The last thanks to Kitty, Luisa, Hyda, Antonio, Eleanna, Panagiota, Katerina, Rodica, Stefania, Beth, Clay, Elena, Carlos, friends I met in my periods abroad. When I needed someone to make me feel at home, you did it guys.

Thank you all.