Progress Report

Investigating artificial neural networks training methodologies for optimal generalisation

Solveig Leblanc
Supervisor: Dr. Richard Neville

A progress report submitted to the University of Manchester as a requirement for COMP60990 – Research Methods and Professional Skills.

09/05/2014
Table of contents

List of tables .................................................................................................................. 3
List of figures ................................................................................................................... 3
Abstract ............................................................................................................................ 4
1. Introduction .................................................................................................................. 5
  1.1. Aims and objectives ............................................................................................... 5
    1.1.1. Aims ............................................................................................................... 5
    1.1.2. Objectives ...................................................................................................... 5
  1.2. Organisation of the report ..................................................................................... 6
Chapter summary ............................................................................................................ 7
2. Background research ................................................................................................... 8
Overview .......................................................................................................................... 8
  2.1. Artificial Intelligence and the "brain-oriented" connectionist paradigm ................. 8
    2.1.1. An overview of artificial intelligence .............................................................. 8
    2.1.2. The connectionist paradigm .......................................................................... 9
    2.1.3. A brief history of artificial neural networks ................................................... 9
  2.2. Artificial Neural Networks .................................................................................... 10
    2.2.1. Tasks and applications .................................................................................. 10
    2.2.2. Network architectures and learning process .................................................. 11
    2.2.3. Taxonomy of artificial neurons .................................................................... 13
    2.2.4. Description of well-known ANN for function mapping ............................... 14
  2.3. Generalisation in Neural Networks ...................................................................... 17
    2.3.1. Evaluation of generalisation ........................................................................ 18
    2.3.2. Improve generalisation ............................................................................... 19
Chapter summary ............................................................................................................ 19
3. Research methods ...................................................................................................... 20
Overview .......................................................................................................................... 20
  3.1. Literature survey methodology ........................................................................... 20
  3.2. Methods for evaluating generalisation ................................................................ 20
    3.2.1. Cross-validation ............................................................................................ 20
    3.2.2. Data pre-processing ..................................................................................... 21
  3.3. Research hypotheses about techniques to improve generalisation ....................... 22
    3.3.1. Selection of the training technique ............................................................... 22
    3.3.2. Application of the cross-validated early stopping .......................................... 23
    3.3.3. Regulation of the complexity of the network ............................................... 23
    3.3.4. Adjustment of the learning parameters ....................................................... 25
  3.4. Evaluation strategy ............................................................................................... 25
Chapter summary ............................................................................................................ 26
4. Project progress .......................................................................................................... 27
Overview .......................................................................................................................... 27
  4.1. Progress of experiments ....................................................................................... 27
  4.2. Remaining tasks ................................................................................................... 31
  4.3. Project plan .......................................................................................................... 32
Chapter summary ............................................................................................................ 32
References ....................................................................................................................... 33
Appendix .......................................................................................................................... 35
Partial Research Relevance Table (RRT) .................................................................... 35
Gantt chart of the project ............................................................................................... 38
Progress Report

Solveig Leblanc

List of tables

Table 2.1: Characteristics of well-known ANN for function mapping........................................... 14
Table 2.2: Full cross-validation procedure............................................................................................ 18
Table 3.1: Hypotheses table.................................................................................................................. 26
Table 4.1: AND and OR logical functions datasets .............................................................................. 28
Table 4.2: XOR logical function dataset................................................................................................ 29
Table 4.3: Prototypes table .................................................................................................................... 32

List of figures

Figure 2.1: An example of a feedforward network (a) and a recurrent network (b) ......................... 12
Figure 2.2: TLU node structure ............................................................................................................. 13
Figure 2.3: Semilinear node structure .................................................................................................. 14
Figure 2.4: Underfitting and overfitting ............................................................................................... 17
Figure 3.1: Finding the optimal number of training cycles................................................................. 23
Figure 3.2: Effect of hidden layer size on network generalisation ...................................................... 24
Figure 4.1: AND and OR logical functions ........................................................................................... 28
Figure 4.2: XOR logical function ......................................................................................................... 29
Abstract

Artificial Neural Networks (ANN), refer to computational models inspired by biological brains. These connectionist systems are able to solve a variety of tasks that were hardly solved by other computational methods. Artificial Neural Networks are used in various fields such as computer science, statistics, biology, medicine, cognition science or engineering. Currently, researchers and industries work together to improve neural networks ability to generalise in order to use them in innovative devices.

This study aims to investigate artificial neural networks training methodologies for optimal generalisation. Because of the multitude of different training techniques and learning rules (of which taxonomy is provided in §2. Background research), this investigation is limited to the supervised learning techniques for mapping mathematical functions.

Assumptions are made about the generalization ability of different training techniques. Prototypes are built to test these training techniques. The corresponding algorithms are then implemented and used during the experimentation phase. Each experimental result should validate the hypotheses but also enrich the initial mathematical model. Machine learning techniques are used to analyse results. Some experiments still need to be carried out and will be described in detail in the final dissertation.
1. Introduction

While the earliest model of an artificial neuron has been introduced by McCulloch & Pitts in 1943 [1], interest for artificial neural networks has significantly regained interest since the 1990's when many researchers focused on neural networks in order to use them as an alternative to standard nonlinear regression or cluster analysis techniques [2]. Since, fields of application of artificial neural networks have been multiplied. Currently, researchers and industries are interested in ANN for solving real-life issues or for innovation. For instance, neural networks are designed for medical diagnosing or to interpret nucleotide sequences, forecasting weather or financial time series, just to name a few applications.

Thus, many researchers were interested in defining some efficient artificial neural networks techniques from an engineering perspective [3], and how to improve their results in accuracy and in generalization to suit the requirements of the industry. Important IT companies such as Facebook, Google or Microsoft recently claimed they work hand in hand with researchers to improve capacities of neural networks to use them for finding innovative solutions in face recognition, speech recognition and translation and many other concrete applications [4].

The way artificial neural networks are trained influences their efficiency for solving a given task. This study aims to investigate on factors and training techniques able to improve the generalisation capacity of neural networks. This project seeks to answer some interesting research questions:

**RQ#1** – Are neural networks able to approximate mathematical functions?

**RQ#2** – Are neural networks able to generalise in all circumstances?

**RQ#3** – What factors can affect the generalisation ability of a neural network?

**RQ#4** – If it were an artificial neural network training technique for optimal generalisation, what would it be, what would be the characteristics of such a neural network?

1.1. Aims and objectives

1.1.1. Aims

The aim of this project is to evaluate different artificial neural networks training techniques affecting generalisation.

1.1.2. Objectives

Consequently, the objectives are:

- determine what type of neural networks can deal with function approximation;
- carry out a literature survey to study previous works and findings in the field;
• elaborate an efficient way to evaluate the generalisation of a model;
• formulate hypotheses on factors that may affect generalisation;
• design different experimental models to confirm or invalidate hypotheses;
• gather observations, interpretations and conclusions to suggest a neural network training technique for optimal generalisation; and
• apply this technique for approximating a real mathematical function such as financial time series, if time permits.

In summary, this hypothesis-driven study aims to analyse and experiment some prototypes built to investigate different artificial neural network training methodologies for optimal generalisation, and ideally, suggest an evolved training technique for optimal generalisation based on the conclusions of this investigation.

1.2. Organisation of the report

This report is organised in five parts.

• **Chapter 1: Introduction**
  This part introduces the reader to the project. A brief overview of the aims and objectives is presented. Research questions to be answered are introduced. Finally, the structure of the project is detailed.

• **Chapter 2: Background research**
  This second part presents the previous work in the field of AI and connectionist machines. It describes artificial neural networks (ANN) in a complete and detailed way. It also provides a description of findings on generalisation that are useful for the reader to know.

• **Chapter 3: Research methods**
  This chapter discusses the research methodology used for the project. It also explains how the hypotheses are formulated, based on an important literature survey on the generalisation ability of artificial neural networks.

• **Chapter 4: Progress**
  This part intends to show the progress of the project. This chapter describes in detail the experiments conducted so far. It also gives an overview of the work to be done in the next period so that the aims and objectives are achieved. Finally, it shows the project plan and the milestones of the project until the dissertation submission in September. The Gantt chart has been updated to illustrate this plan.
Chapter summary

This chapter introduces the reader to the project. It presents the research questions to be answered, the aims and the objectives. The overview of the organisation of the project closes this chapter. The next chapter aims to restore the knowledge learnt from the background research.
2. **Background research**

**Overview**

Background research aims to set the project in its wider context and to study the previous work that has been done in the field. For this purpose, many research papers, journals and books have been read, studied, and summarized.

2.1. **Artificial Intelligence and the “brain-oriented” connectionist paradigm**

2.1.1. An overview of artificial intelligence

Artificial intelligence (AI) was founded in 1956 by a group of scientists who aim to show that machines may “be capable of doing any work a man can do”, as Herbert Simon claimed at the Dartmouth conference in 1956. In 2005, Ray Kurzweil predicted that machines with human-level intelligence may appear by 2029 [5]. AI aims to create intelligent agents, that is, systems that perceive their environment and take actions the same way a human, using its human-like intelligence, would do. Thus, AI attempts to study capabilities humans specifically have, that is, intelligent actions. For this purpose, an intelligent agent must be able to [6]:

- operate in real-time;
- deal with important amounts of knowledge;
- be able to manage error, unexpected and previously unseen input;
- use symbols and abstractions;
- communicate using natural language;
- learn from the environment; and
- adapt behaviour to goals.

Achievements in the field and a recent state of the art in AI are presented in [7]. In [8], it is highlighted that systems elaborated “in areas such as game playing, logical inference and theorem proving, planning, and medical diagnosis” can now perform as well as, or better than, human experts. While “in other areas, such as learning, vision, robotics, and natural language understanding” efficient research enables a rapid improvement of capabilities. Some successes in the field have already started to improve the human living conditions by offering real-life solutions to engineering or medicine for instance.
AI research is separated into two different lines of research, the symbolic-algorithmic paradigm challenged by the connectionist paradigm [9]. The symbolic paradigm has represented the main part of AI research for a long time using tools such as:

- Search and optimisation;
- Logic;
- Probabilistic methods; and
- Classifiers and statistical learning methods.

However, the initial objectives targeted by the originators of AI are far from being reached using these tools based on excessive assumptions that all information humans deal with can be formalised in a logical way [10]. This is the reason why the connectionist paradigm with the introduction of the connectionist machines, has become more significant. These are the topics addressed in the following parts.

2.1.2. The connectionist paradigm

In the connectionist view, the idea is to differentiate the knowledge which can be formalised in a logical way and that which cannot in order to move away from too simplistic an idea [11]. Connectionist research provides a bridge between engineering challenges and AI, and enables to address the AI research objectives from another point of view [12]. Thus, a connectionist machine, also called a neural network, is:

“an interconnected assembly of simple processing elements [...] whose functionality is loosely based on the animal neuron,” [13].

The processing ability of this system resides in the connection strengths (weights) between processors (artificial neurons) in the same way as for biological neurons. These weights are continually altered to respond to presented training vectors: the connectionist system is adaptive, rather than pre-programmed as symbolic systems used to be.

2.1.3. A brief history of artificial neural networks

The first formal model of an artificial neural network has been proposed by McCulloch and Pitts in 1957 [1]. They first modelled the neuronal input-output relation in an artificial neural network and therefore showed that networks can learn from training.

At the same time, biological discoveries brought useful information on the way animal neural networks operate. It has been found that synaptic strengths adapt to reinforce any correlation of activity between pre-synaptic and post-synaptic neurons. These facts have been translated into
artificial networks for them to suit better the reality: weights should be adapted considering the correlation between the input and the output of a unit.

Then, Rosenblatt presented the perceptron in 1957 and also proved that a simple learning algorithm converges if a solution exists (perceptron convergence theorem) [14].

A few years after, Minsky and Papert highlighted the limits of the perceptron [15]: the perceptron cannot solve non-linearly separable problem such as the simple XOR logical function. The perceptron convergence theorem must be corrected. But they faced a problem when they build a multilayer perceptron so as to get around these limits. This is the first time the credit assignment problem is raised: how much does each hidden unit contribute to the error the network has made in processing the current training vector? This problem is solved when the back-propagation algorithm is presented in 1974 by Werbos [16], and then later by Rumelhart, Hinton and Williams [17].

In 1982, Hopfield investigated the recurrent networks, a highly inter-connected type of neural networks built on an energy-based description [18].

These powerful new learning methods for artificial neural networks have been made possible by advances in technology and improvements in computer sciences that enabled complex computer simulations to be performed.

2.2. Artificial Neural Networks

Tasks and applications that networks are used for are detailed in the first part. Then, the networks structures and the types of learning are explained in a second part. Finally, taxonomy of artificial neurons and classification of popular artificial neural networks are proposed in parts three and four.

2.2.1. Tasks and applications

Currently, artificial neural networks are mainly applied to challenging tasks that belong to the following broad categories, detailed and illustrated in [19] and [20]:

- Pattern classification: supervised learning technique that aims to assign an input pattern to one of the target classes;
- Clustering: unsupervised learning technique that forms clusters by exploring the similarities between the input patterns;
- Function approximation: train an ANN on input-output data so as to find an estimate of the underlying function;
- Forecasting: train an ANN on some time series and use it for other scenarios to predict the behaviour of this sequence at some future time [21];
- Optimization: find a solution that optimises an objective function under a set of constraints; and
- Control: generate a control input so that the system follows the desired trajectory determined by the reference model [20].

Artificial neural networks have been used in many several fields of real-world applications since the 1990's. Indeed, they have been successfully applied to energy engineering problems: they are employed to solve some hydrologic problems [22], materials engineering problems [23], electrical engineering issues [24] and some nuclear and drainage engineering applications. Artificial neural networks have also been importantly used in finance [25], and more specifically in pricing applications [26], stock market predictions [27], credit ratings [28] or enterprise finance evaluations [29] just to name a few. The field of microbiology is also interested in artificial neural networks, as it is explained in [19]. The classification task of ANN is also very useful in medicine [30] [31] [32] [33]. Finally, artificial neural networks are efficient in computer vision applications such as face detection [34] or speech recognition.

Currently, artificial neural networks are applied in numerous scientific fields and enable solving a large number of different types of problem. The fields of application and the number of real-world problems solved by ANN are increasing year by year.

2.2.2. Network architectures and learning process

There are many different ways to describe an artificial neural network (neuron model, learning algorithm, network structure...). This part aims to define the different network architectures and the learning process of a neural network, and to introduce some terminology. There are two categories of neural networks architecture [20]:

- feedforward networks (Figure 2.1.a); and
- recurrent networks (Figure 2.1.b).
The learning process refers to the way the network updates weights to perform a specific task [20]. It is characterised by the learning algorithm, the learning paradigm and the learning rule, which are notions that are defined in the next paragraphs.

A learning algorithm is the process of adjusting the weights of the network by applying a learning rule. A classification of the most popular learning algorithms is given in §2.2.4. The learning paradigm specifies the information available to the network. There are three learning paradigms:

- supervised learning: the training patterns are provided with an input and an output (target) and the network is trained to make the smallest error compared to the target;
- unsupervised learning: the network must identify the underlying properties of the training data and generate categories to classify the input patterns; and
- hybrid learning: this type of learning combines supervised and unsupervised learning.

Finally, the learning rule depicts the way weights are updated. There are four main categories of learning rules:

- error-correction rules: this rule uses the discrepancy between the desired target and the network output to update the weights, so as to reduce the error. This is the type of learning rules that is used for this project;
- Boltzmann rules: this rule aims to update weights so that the state of visible units of a Boltzmann machine (“+1” units) satisfies a desired probability distribution;
- Hebbian rules: this rule updates weights locally, a change in weights of a neuron depends on the activity of the neurons connected to it; and
- competitive rules: this rules is based on the “winner-take-all” principle.
2.2.3. Taxonomy of artificial neurons

Pitts and McCulloch proposed the first model of an artificial neuron in 1943. The structure of the threshold logic unit (TLU), inspired from the “all-or-none character of nervous activity” [1] is defined in Figure 2.2 below:

\[ a = \sum_{i=1}^{n} w_i x_i \]

The threshold term \( \theta \) is included in the weight vector \( w \). This trick simplifies mathematic writings and guarantees that the weights and the threshold are trained uniformly.

The output signal takes the values “0” or “1” characterising the states “off” and “on”, and it is given by the following threshold relation, also called “step function”:

\[ y = \begin{cases} 
1 & \text{if } a \geq \theta \\
0 & \text{if } a < \theta 
\end{cases} \]

The node structure is improved to encode non-binary signal values, as real animal neurons are believed to do. Usually a sigmoid function (also called logistic function) is used to calculate the output so that the output \( y \) depends smoothly on the activation. The resulting artificial neurons are defined as semilinear. The output is therefore depicted as follows:

\[ y = \sigma(a) \equiv \frac{1}{1+e^{-(a-\theta)/\rho}} \]

where \( \rho \) is the inverse-width and defines the shape of the curve (when \( \rho \to 0 \) the sigmoid tends to behave like the step function).
2.2.4. Description of well-known ANN for function mapping

Each type of artificial neural network, and each learning algorithm, is designed for performing a few tasks very well. Table 2.1, adapted from [20], classifies the characteristics of the type of networks able to perform function approximation:

<table>
<thead>
<tr>
<th>Network</th>
<th>Paradigm</th>
<th>Learning Rule</th>
<th>Learning Algorithm</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single or multilayer</td>
<td>Supervised</td>
<td>Error-correction</td>
<td>-Perceptron</td>
<td>-Pattern Classification</td>
</tr>
<tr>
<td>perceptron</td>
<td></td>
<td></td>
<td>-Delta rule</td>
<td>-Function approximation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-Back-propagation</td>
<td>-Prediction, control.</td>
</tr>
<tr>
<td>Radial Basis Function</td>
<td>Hybrid</td>
<td>Error-correction</td>
<td>-RBF</td>
<td>-Pattern Classification</td>
</tr>
<tr>
<td>(RBF)</td>
<td></td>
<td>and competitive</td>
<td></td>
<td>-Function approximation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-Prediction, control.</td>
</tr>
</tbody>
</table>

Table 2.1: Characteristics of well-known ANN for function mapping

Two types of networks are highlighted in this table: the single or multilayer perceptron and the radial basis function (RBF). The second column gives the learning paradigm (supervised, unsupervised or hybrid). The third column gives the type of learning rule usually employed by these networks. Then, the fourth column lists the names of the learning algorithms that are used by these categories of networks. Finally, the last column attests that function approximation is one of the tasks that can be performed by these networks.

The role of the neural network for function approximation is to identify the underlying properties of the function. The training process must adapt the weights of the network so that it replicates the data correctly and can generalise to a previously unseen dataset. The most popular feedforward networks for function mapping are the single or multilayer perceptrons.
The perceptron learning algorithm is an iterative process that presents each training example and makes small changes to weights and threshold following a simple error-correction learning rule to bring them more in line with the desired classification. According to the perceptron convergence theorem, this algorithm is able to solve any linearly-separable problem in a finite number of iterations.

The delta rule learning algorithm is a slightly adapted version of the perceptron learning algorithm. This algorithm is based on a learning rule that calculates a delta with the gradient descent of the error. A sigmoid transfer function is used to calculate the output of each unit.

A back-propagation network is a multilayer perceptron. The back-propagation learning algorithm relies on propagating errors backwards and assigning “blame” for the error made by the network, to each hidden unit.

RBF networks are a special case of a three-layer back-propagation network. The sigmoid transfer function used for back-propagation networks is replaced by a radial basis function such as a Gaussian kernel for instance, which parameters need to be learnt from training.

### 2.2.4.1. Perceptron learning algorithm

The first learning algorithm is the perceptron learning algorithm, created to train the TLU [1]. This learning algorithm is based on the perceptron convergence theorem [14]:

**Theorem 1:** Perceptron convergence theorem

"If there is a set of weights that correctly classify the training patterns, that is, if the classification is linearly separable, then the learning algorithm will find one such weight set w* in a finite number of iterations."

**Algorithm 1:** Perceptron learning algorithm

```plaintext
initialise weights w with some small random values
initialise the learning rate \( \alpha \) with a sufficiently small value
repeat
    for each training example \((x, t)\)
        evaluate the output \( y \) that the TLU gives with \( x \) in input
        if \( y \neq t \) then
            apply the learning rule to the weight vector \( w \): \( w' = w + \alpha(t - y)x \)
        end if
    end for
until convergence criterion \((y = t\) for all examples)"
```
2.2.4.2. Delta rule learning algorithm

The delta rule learning algorithm is a variant of the perceptron learning algorithm adapted to a semilinear-type neuron [13]. It is based on the use of a smooth activation function (usually the sigmoid $\sigma$) rather than a step function. The learning rule, called the delta rule, takes in continuous values and calculates the discrepancy to a perfect classification. The convergence criterion concerns the rate of change of the error from iteration to another.

**Algorithm 2: Delta rule learning algorithm**

| initialise weights $w$ with some small random values |
| initialise the learning rate $\alpha$ with a sufficiently small value |
| repeat |
| for each training example $(x, t)$ |
| evaluate the output $y$ that the TLU gives with $x$ in input |
| calculate the delta considering the target: $\delta = \sigma'(u)(t - y)$ |
| adjust the weight vector $w$: $w' = w + \alpha \delta x$ |
| end for |
| until convergence criterion (the rate of change of the error is sufficiently small) |

2.2.4.3. Back-propagation learning algorithm

Finally, the back-propagation algorithm is a multilayer perceptron network of semilinear nodes which uses the same loop structure as the perceptron or the delta rule learning algorithm.

**Algorithm 3: Back-propagation learning algorithm**

| initialise weights $w$ with some small random values |
| initialise the learning rate $\alpha$ with a sufficiently small value |
| repeat |
| for each training example $(x, t)$ |
| train on that training example: |
| o let the hidden units evaluate their own output using the training pattern |
| o let the output units evaluate their own output using the results of hidden units |
| o calculate the delta on the output nodes considering the target pattern: $\delta_k = \sigma'(a_k)(t_k - y_k)$ |
| o apply the gradient descent learning rule to each output node: $w_{ki} = w_{ki} + \alpha \delta_k x_{ki}$ |
| o calculate the delta on the hidden nodes considering the updated weights: $\delta_k = \sigma'(a_k) \sum_j \delta_j w_{jk}$ |
apply the gradient descent learning rule to each hidden node:

\[ w_{ki} = w_{ki} + \alpha \delta^{(k)} x_{ki} \]

end for

until convergence criterion (the error is sufficiently low)

2.3. Generalisation in Neural Networks

An artificial neural network is said to have learnt correctly if it can handle imperfect training samples without affecting on response quality, and if it can generalise well to unknown datasets [19]. Generalisation is one of the main challenges in developing a neural network. The critical question to answer is: How well will this network make predictions for inputs that are not in the training set?

"In any real world application, the performance of artificial neural networks mostly depends on its generalisation ability." [35].

Artificial neural networks can suffer from two phenomena to avoid for optimal generalisation:

- Underfitting: the model fails to detect fully the signal; and
- Overfitting: the model fits the signal but also the noise.

Both lead to an increase in generalisation error, as it is shown and explained in the three graphs presented in Figure 2.4 below.

**Figure 2.4:** Underfitting and overfitting. **Graph a** shows a model that underfits the training data. The mapping is general and doesn't fit properly to particularities of the training data. Offset points A and B don't have been taken into account for the mapping. **Graph b** shows a correct fit, all points of the training data are close to the curve, and offset points are moderately influent. **Graph c** shows a model that overfits. The mapping is highly precise, each and every point sticks perfectly to the curve. The point B influences importantly the curve. The green cross is an extrapolation example. It is close to the model result in graph a, it perfectly fits to the model b but it is distant from the model of graph c.
Underfitting is usually observed when the network trained is not sufficiently complex and the dataset is complicated. In the contrary, overfitting is obtained when the system is too complex relatively to the dataset. Thus, for the network to generalise well, the inputs must contain enough information relatively to the complexity of the network. But finding the appropriate complexity is one of:

*“the most important unresolved problem in practical applications of feedforward neural networks.”* [36]

### 2.3.1. Evaluation of generalisation

Generalisation error of a network can be estimated by testing it on previously unseen data. However, precautions must be taken concerning the datasets on which tests are done. In fact, if the same dataset is always used, the estimation is not relevant because the network parameters may be tuned to this sample particularly.

Thus, some machine learning model validation techniques need to be employed here. The cross-validation process used to estimate the generalisation error of a network consists of three samples. The network is fed with a training sample. Then, the model obtained is tested on a validation set. This model validation step gives an insight on how the model can generalise. Finally, generalisation is evaluated once by applying a selected model to the test set. [37] For optimal generalisation, the selected model is the one that performs the best on the validation set.

![Diagram of cross-validation procedure](image)

**Table 2.2:** Full cross-validation procedure

The proportion of examples contained in these sets must be determined by experimentation so that the network is unlikely to suffer from much underfitting or overfitting.

Also, datasets must be standardised, that is, input samples must be normalised to zero-mean and unit-variance so that the generalisation results can be legitimately compared whatever input sample is used for experiments.
2.3.2. Improve generalisation

There are several approaches to avoid underfitting or overfitting and ensure that the network generalises well. The first way to control underfitting and overfitting is to find an appropriate complexity of the network relatively to the characteristics of the inputs. For this purpose, a model selection must be carried out to find the best parameters in terms of number of hidden units and hidden layers, number of weights or learning rate. In fact, it has been proved that the generalisation error depends on the number of effective parameters [38].

Another way to improve generalisation is to practice cross-validated early stopping [36]. This process aims to compute the validation error during training until convergence, and to pick the lowest value of validation error and choose to stop training the network at this moment before the validation error starts to go up. The advantage of this technique is that it is fast and easy to achieve. It can be applied to all problems, even if there are not many examples but many features.

Moreover, other techniques such as jittering or regularisation may be applied to the network in order to try and improve its generalisation ability. Jittering depicts the process of adding noise into the inputs during training so as to smooth functions even when the training set is small. This solution is closely related to the regularisation method called the weight decay. This method consists in adding a penalty term to the error function which causes the weights to converge to smaller absolute values than they otherwise would. It has been shown that generalisation ability importantly depends on the decay constant, especially with small training sets.

Chapter summary

This chapter investigates the wider context of the project. Previous works in the field of artificial neural networks are presented. Finally, the more specific topic of factors that can affect generalisation of ANN is covered at the end of the chapter. Next chapter deals with the research methods employed to achieve research objectives.
3. **Research methods**

**Overview**

This study aims to investigate factors that may affect artificial neural networks generalisation ability when performing function approximation. This chapter exposes the research methodology used for the project. It also shows how this literature survey leads to the identification of the experimental context and to the formulation of the research hypotheses.

### 3.1. Literature survey methodology

This research project can be separated into two main parts: the literature survey and the experiments part. The literature survey aims to build a background to the project and to place the problem in its wider context, but it also aims to highlight the issues of such a project, and helps defining the important research questions [39]. For this purpose, many research papers, journals and books have been read, studied, and summarized. Interesting papers are listed in a research relevance table (RRT) similar to that given in appendix. This table is an easy way to classify readings into subjects they address or not, and to allocate a level of relevance and any useful comment to a document. Using this method assists the organisation of the research and makes any subsequent consultation of previously studied document a lot easier.

### 3.2. Methods for evaluating generalisation

This part explains how generalisation is evaluated in experiments and choices that have to be made before any network is trained and tested.

#### 3.2.1. Cross-validation

Cross-validation is applied to the fitting process so that the generalisation error is estimated when the model is applied to unseen data. The initial dataset is separated in three subsets. One of them is kept as the testing set; the others are used as training set and evaluation set. Cross-validation is a way to evaluate generalisation, but it is also a tool for improving generalisation, as it is described in §3.3.2. To determine the number of folds to use, the data size must also be considered, as explained in the next paragraph.

The results obtained concerning generalisation will then have to be analysed and compared to the initial hypotheses.
3.2.2. Data pre-processing

As neural networks are applied to solve a function approximation task, inputs are logical or mathematical functions. For neural networks to be efficient, and for a faithful analysis of results, it is essential that the different datasets given in input are of optimal and similar quality. This data processing problem needs to be addressed before initiating any training.

3.2.2.1. Data size and partitioning

Applying cross-validation involves that the data is partitioned in three parts: the training set, the validation set and the testing set.

There is no mathematical rule for determining the optimal database size to give in input. However, several rules derived from experience estimate that the generalisation ability of a network is related to the ratio of the training sample size to the number of weights. If this ratio is low, that is, if the number of examples is small relative to the number of weights, it is unlikely that the training set is sufficiently large for the network to learn correctly: the network overfits.

Experiments carried out by Amari et al., show that overfitting is reduced when a ratio of 30 is used. W. Duch et al. carried out experiments on three different pre-processed datasets in 2005 and they came to the conclusion that overfitting is likely to disappear when the ratio is around 50 [40]. For this project, datasets with a minimum ratio of 30 are used so that optimal experiments can be carried out, and thus optimal results are obtained.

Moreover, it has been shown that a large test subset may highlight the generalization capability better but that the resulting smaller training set may not be adequate to train the network correctly. Thus, proportions need to be determined between the sizes of the training, validation and test sets [40]. Usually, proportions around 65% of the initial dataset for training, 25% for testing and 10% for validation are adopted.

Finally, if the database is too small to partition into fairly-sized subsets as needed for evaluating generalization, a solution is to interject some random noise in the available examples to generate new ones. This technique, called jittering, may enable artificial neural networks to generalise correctly with small datasets in inputs. It may be interesting to investigate such a training technique on particularly small datasets if time permits. This optional hypothesis (#H0) will be approached if all other hypotheses and their corresponding experiments concerning standard datasets are achieved.

| #H0 | Applying the jittering technique (adding superficial noise) to small datasets may enable artificial neural networks to generalise correctly |
3.2.2.2. **Data normalisation**

For accelerating convergence and for error results to be meaningful, it is necessary that data is normalised. The mean value should be adjusted to 0 and the variance to 1. This technique also prevents larger values of the input set from overriding smaller ones during learning.

3.3. **Research hypotheses about techniques to improve generalisation**

3.3.1. **Selection of the training technique**

There are two training modes for the back-propagation algorithm:

- sequential: the gradient descent error is estimated based on the information presented by each pattern and the error rate used for convergence criterion is based on the average of this estimation (as in Algorithm 3).
- batch: the gradient descent error is calculated based on the information of all patterns and weights are therefore updated from a common error.

For this project, the sequential mode of training is used for experiments. This type of training is slightly less fast than the batch training, but it enables the network to generalise better if parameters are well defined.

By now, two transfer functions have been detailed in the background research part:

- step function, associated to a threshold-logic unit (TLU) and used in the perceptron learning algorithm; and
- sigmoid function, associated to a semilinear node, and used in the delta rule learning algorithm.

The properties of these transfer functions may be experimented so that the most appropriate node is chosen for training back-propagation.

Three hypotheses may be formulated in relation to these transfer functions, and considering the perceptron convergence theorem:

| #H1 | Training utilising the perceptron learning algorithm on a single threshold logic unit (TLU) can solve a linearly separable problem. |
| #H2 | Training utilising the perceptron learning algorithm on a single threshold logic unit (TLU) can’t solve a non-linearly separable problem. |
| #H3 | Training utilising the delta rule learning algorithm on a single semilinear unit can solve a non-linearly separable problem. |
3.3.2. Application of the cross-validated early stopping

The number of training cycles may affect the generalisation ability of the network. If the number of training cycles is too high, then the ANN is able to memorise the training data, but is not capable of generalisation to unseen data.

Cross-validation may be used as a tool for finding the optimal number of training iterations for generalisation. This process is called cross-validated early stopping and consists in plotting the validation error graph during training, until convergence, against the number of iterations. Then the model with minimum validation error is selected as illustrated in the Figure 3.1, adapted from [19]. This optimal model is applied to the testing set and gives an estimate of the generalisation error with the optimal number of training cycles.

#H4 Cross-validated early stopping improves the generalisation ability of the network by determining the optimal number of training cycles.

![Figure 3.1: Finding the optimal number of training cycles](image)

In this figure, the validation error and the training error are plotted against the number of training cycles. The optimal number of training cycles is determined by selecting the network that gives the lowest validation error.

3.3.3. Regulation of the complexity of the network

“If the number of example patterns is small relative to the complexity of the system, generalization error is found to be high,” [35].
The complexity of a multilayer network needs to be regulated relative to the size of the training dataset. It concerns not only the number of hidden layers but also the number of hidden nodes (HN). Some rules derived from experience show that a continuous function can be correctly learnt by a single-hidden-layer network (a network consisting of the input layer, one hidden layer and the output layer). A discontinuous function may be correctly learnt by a network composed of two hidden layers, and more generally, all mathematical functions should be correctly learnt by a network with three hidden layers.

Moreover, the training error increases with an increasing number of hidden nodes: the training ability of the network is amplified. However, from a certain number of hidden nodes, the network is no longer able to generalise to unseen data as its complexity makes it learn and fit the data too precisely. **Figure 3.2** shows the various situations that can be obtained when the hidden layer size varies.

The most popular way to regulate the complexity of the network is to use the validation technique that is used for determining the optimal number of training cycles, and illustrated in **Figure 3.1**. But some scientists also suggested inequalities that link the number of input units, the number of output units and the number of hidden units.

---

**Figure 3.2**: Effect of hidden layer size on network generalisation

In this figure, three networks with a different number of hidden nodes created three approximations of same mathematical functions. It is noticed that the number of hidden nodes affects the way the network is able to approximate unknown testing points.


3.3.4. Adjustment of the learning parameters

The back-propagation learning algorithm depends on several learning parameters. A parameter analysis is needed to study how they can affect the network.

3.3.4.1. Learning rate $\alpha$

Using an appropriate learning rate is essential for training a neural network. If the learning rate is too large the network may oscillate around the solution without reaching it, while if it is too small, the training can be very slow. Usually, a learning rate around 0.1 is adopted. This is the value that is used in experiments for this project.

3.3.4.2. Weight initialisation and momentum coefficient $\mu$

Usually, weights and threshold are initialized randomly in a range of very small values, preferably with zero mean. This initialisation process is a way to avoid premature neurons saturation during training. When the back-propagation learning algorithm is used, a momentum term $\mu$ is often used. This additional term helps optimising the weights updates during training in order to speed up learning but also to avoid the oscillation symptom when learning becomes unstable. The momentum term consists of the momentum constant $\mu$ multiplied by the value of the previous weight change. This term is added to the learning rule.

|H6| An efficient weights initialisation and an appropriate momentum coefficient for weight decay improve the generalisation ability of a neural network.

3.4. Evaluation strategy

Research hypotheses are now formulated. An evaluation strategy is worked out and applied for each hypothesis. This strategy can be defined by the following steps:

- A prototype is elaborated to test the training technique that is addressed in the hypothesis;
- The prototype is experimented so that results can be recovered;
- Results are analysed and interpreted to draw conclusions on them;
- If the results are not satisfactory, other experiments may be considered; and
- Else, the results are used for validating the hypothesis.

In this way, each successful prototype brings a new piece of information and assists in the investigation of different artificial training techniques for optimal generalisation.
Chapter summary

This chapter provides the research methodology adopted for carrying out this hypothesis-driven study. The resulting hypotheses table is provided below. The next part shows how the project is progressing and what are the major advances and the remaining tasks.

| #H1          | Training utilising the perceptron learning algorithm on a single threshold logic unit (TLU) can solve a linearly separable problem. |
| #H2          | Training utilising the perceptron learning algorithm on a single threshold logic unit (TLU) can’t solve a non-linearly separable problem. |
| #H3          | Training utilising the delta rule learning algorithm on a single semilinear unit can solve a non-linearly separable problem. |
| #H4          | Cross-validated early stopping improves the generalisation ability of the network by determining the optimal number of training cycles. |
| #H5          | The generalisation ability of a neural network is related to its complexity (number and size of hidden layers) and an optimal complexity improves generalisation. |
| #H6          | An efficient weights initialisation and an appropriate momentum coefficient for weight decay improve the generalisation ability of a neural network. |
| #OH1         | Applying the jittering technique (adding superficial noise) to small datasets may enable artificial neural networks to generalise correctly. |

Table 3.1: Hypotheses table
4. Project progress

Overview

This section describes the extent of work completed so far, and the remaining tasks to achieve for reaching the aims and objectives detailed in introduction. At the time of writing this progress report, the background research has been done and the research methodology has been established. In fact, research hypotheses have been formulated and the implementation design has been set. Concerning the experimental part, some prototypes have already been successfully completed and are used to validate some hypotheses. These prototypes are detailed in §4.1. The remaining experiments to achieve are exposed in §4.2. For these prototypes, the adopted experimental procedure is presented.

4.1. Progress of experiments

For this hypothesis-driven study to be realised, different prototypes of ANN have to be implemented and experimented. Prototypes are built to validate the basic set of hypotheses given in the hypotheses table, but some additional hypotheses may be formulated if more precision is needed or if an interesting aspect of an experiment worth to be studied in more depth.

All the developments and the experiments are done using Matlab. A prototypes table is provided at the end of this chapter and summarises all the developed prototypes and provides some useful information about them.

Experiments deal with analysing different neural networks training techniques to answer the research questions, and more particularly, to validate the research hypotheses. The three first research hypotheses aim to investigate the properties of the different learning algorithms.

- #H1: Training utilising the perceptron learning algorithm on a single threshold logic unit (TLU) can solve a linearly separable problem.

  This hypothesis aims to validate the perceptron convergence theorem, that is, verify that if the classification problem is linearly separable, then the perceptron learning algorithm will find a solution in a finite number of iterations. For this purpose, simplistic logical functions are considered as training datasets. AND and OR logical functions are linearly separable classification problems, as it is easily noticeable in graphs Figure 4.1. They are used for validating hypothesis #H1. In the first prototype, the perceptron learning algorithm, as
described in §2.2.4. (Algorithm 1) is applied to the AND logical function dataset (or, the OR logical function dataset). In average, the perceptron learning algorithm achieves a perfect classification in two iterations. As weights are initialised with small random values, the randomness may eventually cause a variation in the number of iterations. The success of this experiment attests that hypothesis $\text{#H1}$ is validated.

![Figure 4.1: AND and OR logical functions](image)

This figure shows that the data points of the AND and OR logical functions can easily be separated into classes 0 and 1 drawing a linear boundary.

<table>
<thead>
<tr>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$\text{Table 4.1: AND and OR logical functions datasets}$

- $\text{#H2}$: Training utilising the perceptron learning algorithm on a single threshold logic unit (TLU) can’t solve a non-linearly separable problem.

The perceptron learning algorithm prototype has been applied to a non-linearly separable classification problem in order to see if convergence can still be reached with the perceptron. The XOR logical function dataset ($\text{Table 4.2}$) is given in input. The algorithm iterates infinitely, it is not able to execute any classification for a non-linearly separable problem. Validating the hypothesis $\text{#H2}$ with this experiment highlights the necessity to improve the learning process.
This figure shows that it is not possible to draw a linear boundary that would separate the data points of class 0 from the data points of class 1: the XOR logical function is a non-linearly separable problem.

![XOR logical function diagram](image)

**Figure 4.2**: XOR logical function

<table>
<thead>
<tr>
<th>XOR</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>t</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 4.2**: XOR logical function dataset

- **#H3**: Training utilising the delta rule learning algorithm on a single semilinear unit can solve a non-linearly separable problem.

Another prototype is built for the purpose of this hypothesis. A semilinear node based on a sigmoid function is trained with the sequential delta rule learning algorithm given in §2.2.4. (Algorithm 2). A non-linearly separable classification problem is given in input. The XOR logical function dataset (Table 4.2 above) feeds the neural network. The non-linearly separable classification is successfully learnt by the algorithm, and the classification finally executed by the network is always correct, that is, the training error is always zero. Thus, the hypothesis **#H3** is validated.

- **#H4**: Cross-validated early stopping improves the generalisation ability of the network by determining the optimal number of training cycles.

This prototype is being implemented. Even if consistent results have not been analysed yet, the experiment procedure is detailed below.
This new prototype has to feed a multilayer neural network with a simple mathematical function and to train it with back-propagation. Contrary to the three previous experiments, the dataset used for this prototype needs to be generated but also pre-processed following the rules expressed in the Research methods part. The mathematical function used in input will be the square function: \( f(x) = x^2 \).

A hundred of data points are generated randomly with values between -1 and +1. These data points are normalised to ensure an efficient comparison of values of different datasets. The target values are calculated taking the square value of the inputs. As it is explained before, the dataset is split in three parts, the training set, the validation set, and the testing set, in order to perform a standard cross-validation as explained and illustrated in §3.2.1. The multilayer network is trained with the back-propagation learning algorithm described in §2.2.4 (Algorithm 3). In this prototype, the network has one hidden layer. As the experiments on the complexity of the network have not been completed yet, the number of hidden nodes is fixed at 5. The process is launched successively with a number of iterations varying from 1 to 20. The simplified algorithm of the prototype \#P3 is given below:

**Algorithm 4: Prototype \#P3 simplified algorithm**

- generate the input dataset
- split the data into a training set, a validation set and a testing set
- for number_of_iterations varying from 1 to 20
  - initialize \( i = 0 \)
  - while \( i \leq \) number_of_iterations
    - train back-propagation on the training set and memorize the training error for iteration \( i \)
    - test the model on the validation set and memorize the validation error for iteration \( i \)
    - increment \( i \)
  - end while
- calculate the average training error and validation error over the number_of_iterations
- end for
- plot the training error and the validation error rates against the number_of_iterations

The training error and the validation error are plotted against the number of iterations. The appropriate number of iterations for optimal generalisation is the number of iterations that gives the lowest validation error. Implementation of this prototype is in completion and some results may be obtained by the submission of this progress report.
4.2. Remaining tasks

- **#H4:** Cross-validated early stopping improves the generalisation ability of the network by determining the optimal number of training cycles.

This experiment aims to perform early stopping on an artificial neural network. Comparing results in generalisation obtained with early stopping and without stopping will provide some useful information on whether the hypothesis can be validated or not. This experiment has to be completed.

- **#H5:** The generalisation ability of a neural network is related to its complexity (number and size of hidden layers) and an optimal complexity improves generalisation.

A new prototype must be built to train successively this artificial neural network with a changing complexity. The cross-validation process is used for model selection. That is, the prototype performs a standard cross-validation on a multilayer neural network with the same learning algorithm and the same inputs, but each time with a different number of hidden layers or hidden units. The graph of the validation error against the complexity of each network is plotted. The point of the curve with minimum validation error corresponds to the model that is the most likely to generalise well. The characteristics of this model are selected to be tested on the previously unseen testing set in order to estimate the generalisation error obtained with this model. On the other hand, the generalisation ability of a model randomly chosen, with no specific complexity, is also tested to estimate its generalisation ability. Finally, a comparison of the results in generalisation should enable the hypothesis to be validated or invalidated.

- **#H6:** An efficient weights initialisation and an appropriate momentum coefficient for weight decay improve the generalisation ability of a neural network.

The first prototype needs to show the difference between training a network which weights have been efficiently initialised and another one which weights have been initialised without following any rule. This experiment should provide a conclusion on how the weights initialisation may affect training artificial neural networks.

Moreover, a model selection must be performed with networks whose momentum coefficient varies. As it is explained for the other parameter analysis processes (**#H4** and **#H5**), the optimal model is selected regarding its validation error and this model is tested in order to estimate its generalisation ability. This generalisation error is compared the same network without any momentum term in order to show the effect that such a term may have on generalisation.
4.3. Project plan

The project plan of the project is updated and illustrated in the Gantt chart given in appendix for clarity. The different steps to achieve before the submission of the dissertation are detailed in this graph. There are three milestones, displayed as red dots, which are the submissions of the three reports. The project lifecycle is separated in three periods. The first period deals with preparation and planning of the project. This period has ended with the submission of the initial report. The second period is devoted to the hypothesis-driven research and terminates with the submission of the progress report. The third and last period consists in experiments and evaluation and ends with the submission of the final dissertation.

Chapter summary

This chapter summarizes the progress made so far. It exposes the experiments carried out before submission of the progress report, and describes the remaining tasks to be completed to achieve the aims and objectives of the project. Some of the hypotheses have already been validated by successful experiments. Results are collected at the end of successful experiments and the established evaluation strategy is carried out to conclude on the validity of hypotheses but also to enrich the conclusions concerning the investigation for optimal generalisation.

<table>
<thead>
<tr>
<th>#P</th>
<th>#H</th>
<th>Learning algorithm</th>
<th>Dataset</th>
<th>Status</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>#P1 #H1</td>
<td>Perceptron</td>
<td>AND or OR logical function</td>
<td>Complete</td>
<td>#H1 validated</td>
<td></td>
</tr>
<tr>
<td>#P2 #H2</td>
<td>Perceptron</td>
<td>XOR logic function</td>
<td>Complete</td>
<td>#H2 validated</td>
<td></td>
</tr>
<tr>
<td>#P3 #H3</td>
<td>Delta rule</td>
<td>XOR logic function</td>
<td>Complete</td>
<td>#H3 validated</td>
<td></td>
</tr>
<tr>
<td>#P4 #H4</td>
<td>Back-propagation</td>
<td>$f(x) = x^2$</td>
<td>In completion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Prototypes table

This prototypes table is an efficient way to synthesise the progress of experiments. This table will be updated as more prototypes are implemented.
References


## Appendix

### Partial Research Relevance Table (RRT)

This RRT is built for organising research. This table is used to classify the literature survey into meaningful categories.

<table>
<thead>
<tr>
<th>Idx</th>
<th>Authors</th>
<th>Year</th>
<th>Title</th>
<th>Artificial neuron</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R. Turetsky</td>
<td>2000</td>
<td>Training Neural Networks</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>Harvard (Lecture)</td>
<td></td>
<td>Evaluating Generalization</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>J. Bullinaria (Lecture)</td>
<td>2004</td>
<td>Improving Generalization</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Steve Renals</td>
<td>1998</td>
<td>Learning and Generalization in NN</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Kevin Gurney</td>
<td>1997</td>
<td>An introduction to Neural Networks</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>Jain, Mao, Mohiuddin</td>
<td>1996</td>
<td>Artificial Neural Networks: A tutorial</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>Wang, Van Gelder, Vrijling</td>
<td></td>
<td>Some Issues About the Generalization of NN for Time Series Prediction</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Siddhaling Urolagin, Prema K.V., and N.V. Subba Reddy</td>
<td>2012</td>
<td>Generalization Capability of Artificial Neural Network Incorporated with Pruning Method</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>I.A. Basheer and M. Hajmeer</td>
<td>2000</td>
<td>Artificial Neural Networks: Fundamentals, Computing, Design and Application</td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td>Kevin Gurney</td>
<td>1989</td>
<td>Learning in networks of structured hypercubes</td>
<td>X</td>
</tr>
<tr>
<td>12</td>
<td>Loraine Blaxter</td>
<td></td>
<td>How to research?</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>2014</td>
<td>Extremetech.com</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Warren S. McCulloch and Walter Pitts</td>
<td>1943</td>
<td>A Logical Calculus of the Ideas Immanent in Nervous Activity</td>
<td>X</td>
</tr>
<tr>
<td>15</td>
<td>Cheng B. &amp; Titterington D.M.</td>
<td>1994</td>
<td>Neural Networks: A Review from a Statistical Perspective</td>
<td>X</td>
</tr>
<tr>
<td>16</td>
<td>Haykin S.</td>
<td>1994</td>
<td>Neural Networks: A Comprehensive Foundation</td>
<td>X</td>
</tr>
<tr>
<td>17</td>
<td>F. Rosenblatt</td>
<td>1962</td>
<td>Principles of neurodynamics</td>
<td>X</td>
</tr>
<tr>
<td>18</td>
<td>Sarle</td>
<td>1994</td>
<td>Stopped training and other remedies for overfitting</td>
<td></td>
</tr>
<tr>
<td>Training algorithms</td>
<td>Tasks</td>
<td>Generalisation</td>
<td>Evaluation technique</td>
<td>Design parameters</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Improving Generalisation Techniques

<table>
<thead>
<tr>
<th>Regularization</th>
<th>Cross Validated</th>
<th>Others (jittering)</th>
<th>Relevance level (0-4)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight decay</td>
<td>early stopping</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td>2</td>
<td>useful lectures to understand what are artificial neural networks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>book used as a basis for reflexion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>very relevant tutorial on ANN (training modes, learning algo, tasks)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|                | X               | X                  | 2                     |          |
|                |                 |                    |                        |          |
|                |                 |                    | 2                     | importance of generalisation in real world application |
|                |                 |                    | 4                     | includes a very relevant survey of techniques for improving generalisation, microbio applications |
|                |                 |                    | 3                     | intro AI symbolic + connectionist paradigm (ANN) - phd thesis |
|                |                 |                    | 3                     | Research methodology |
|                |                 |                    | 2                     | real world applications |
|                |                 |                    | 3                     | first artificial neuron TLU (-fuzzy to me) |
|                |                 |                    | 4                     |          |
|                |                 |                    | 4                     | perceptron thm + learning algo (details) |
|                |                 |                    | 4                     | perceptron thm + learning algo (details) |
|                | X               | X                  | 3                     | useful for early stopping experiments |
Gantt chart of the project
This Gantt chart displays the project plan and the timeframe of each task to be performed for achieving the aims and objectives.